

OE-Vorlesung 2023

Einführung in die Theorie der Petrinetze

Prof. Dr. Peter Kling

Wintersemester 2023/24

Universität Hamburg

Peter Kling

- Büro: G-229
- E-Mail: peter.kling@uni-hamburg.de
- Leitung des ABs Theorie Effizienter Algorithmen



Peter Kling

- Büro: G-229
- E-Mail: peter.kling@uni-hamburg.de
- Leitung des ABs Theorie Effizienter Algorithmen



Sprechstunden

- Wann immer meine Bürotür offen steht oder...
- ...nach Absprache per Email

Wer spricht denn da?

Peter Kling

- Büro: G-229
- E-Mail: peter.kling@uni-hamburg.de
- Leitung des ABs Theorie Effizienter Algorithmen

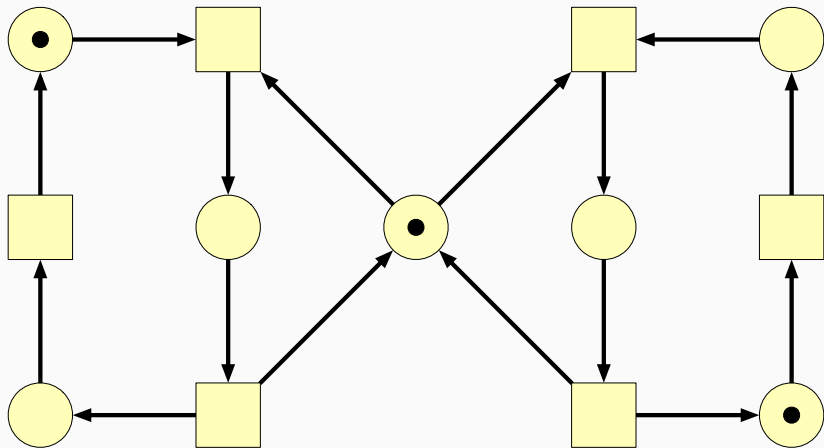


Sprechstunden

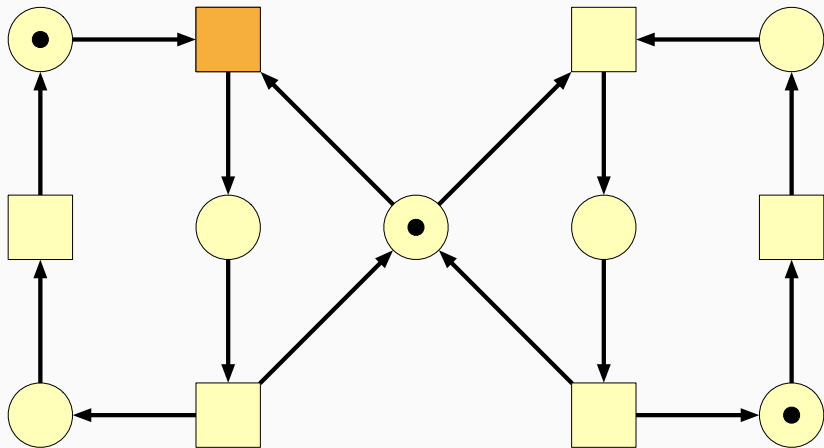
- Wann immer meine Bürotür offen steht oder...
- ...nach Absprache per Email

Seid nicht schüchtern!
Stellt Fragen!

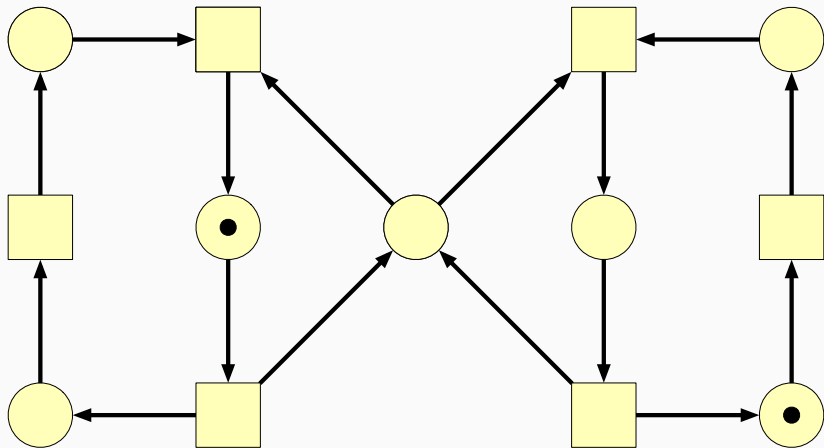
Petrinetze: Was soll das sein?



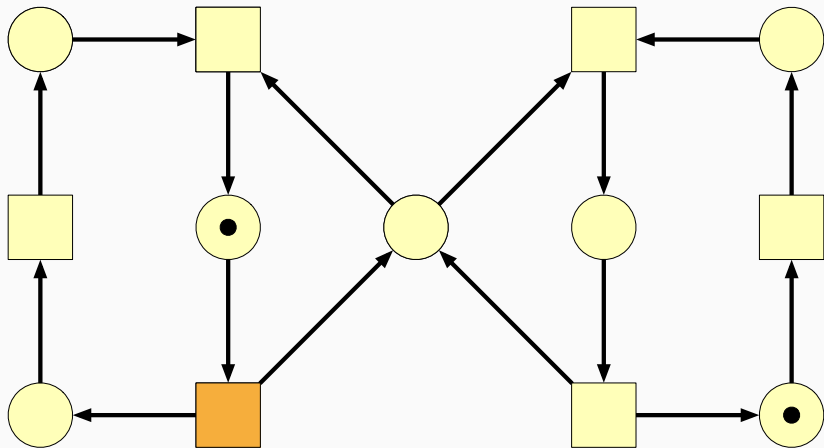
Petrinetze: Was soll das sein?



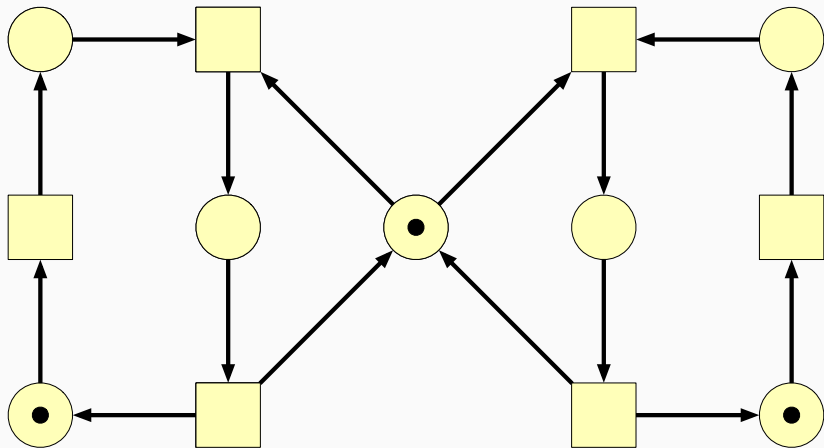
Petrinetze: Was soll das sein?



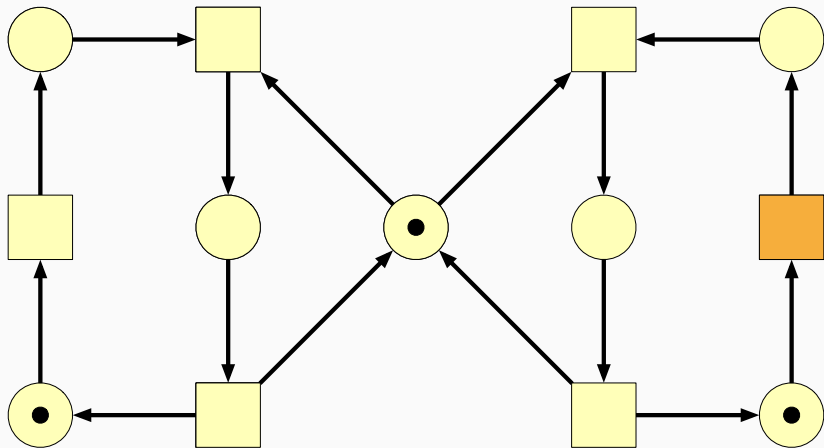
Petrinetze: Was soll das sein?



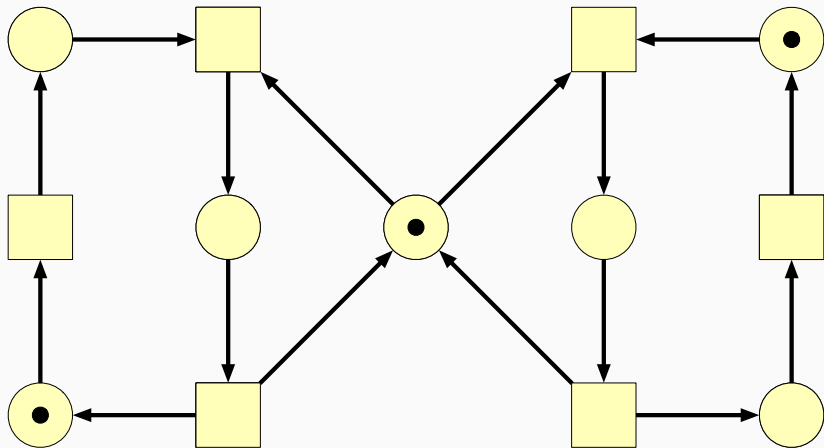
Petrinetze: Was soll das sein?



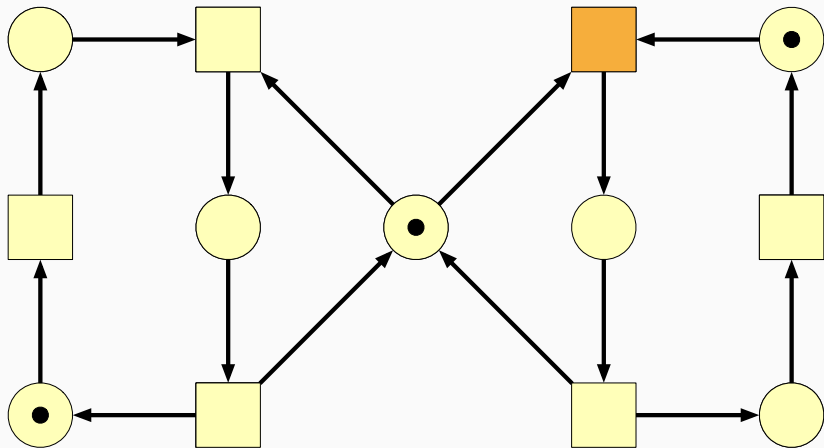
Petrinetze: Was soll das sein?



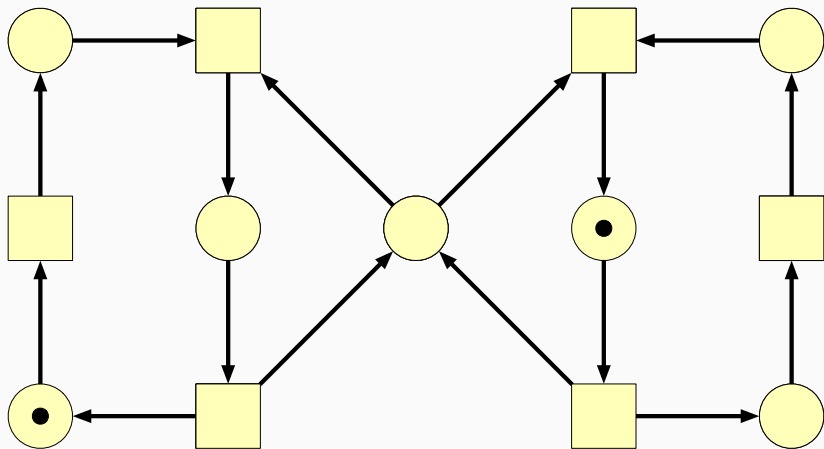
Petrinetze: Was soll das sein?



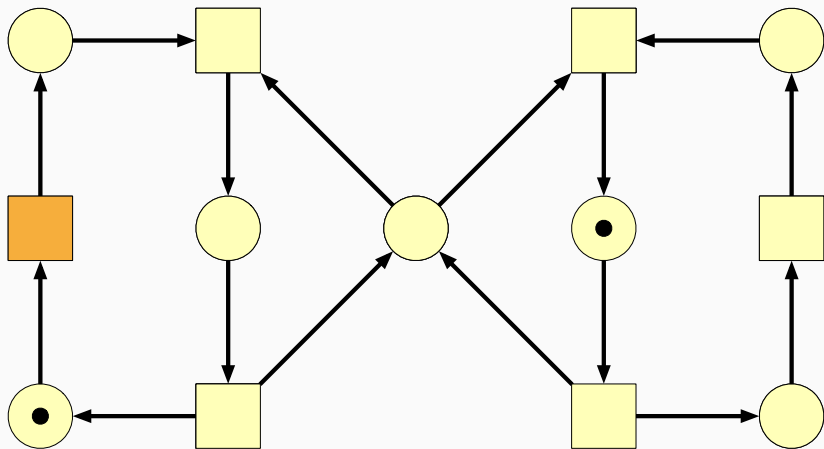
Petrinetze: Was soll das sein?



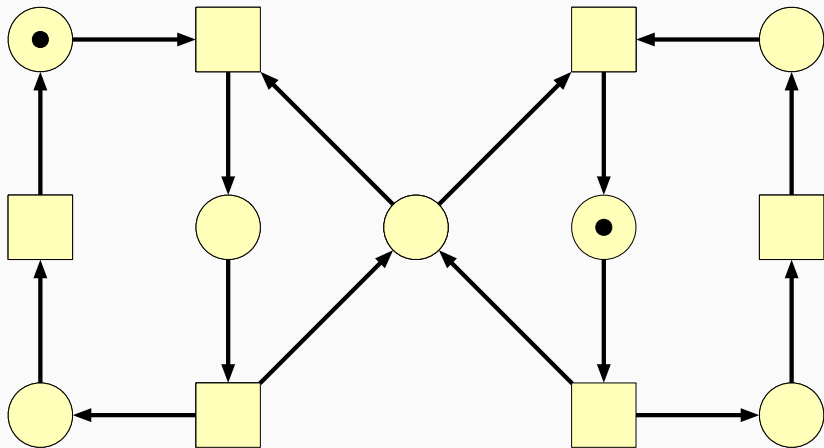
Petrinetze: Was soll das sein?



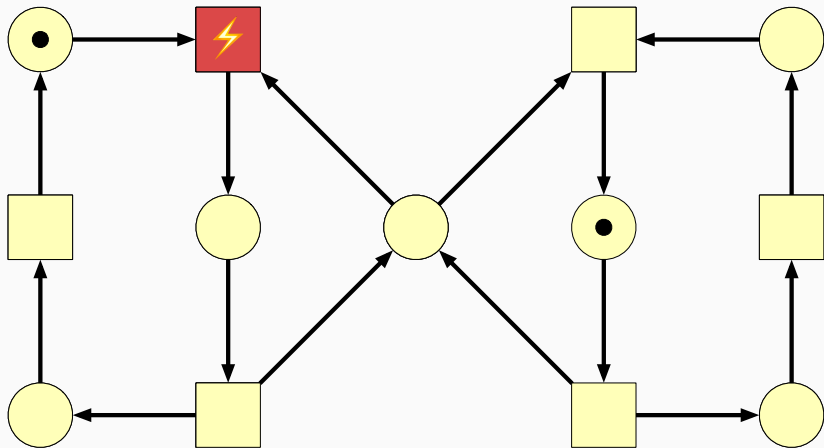
Petrinetze: Was soll das sein?



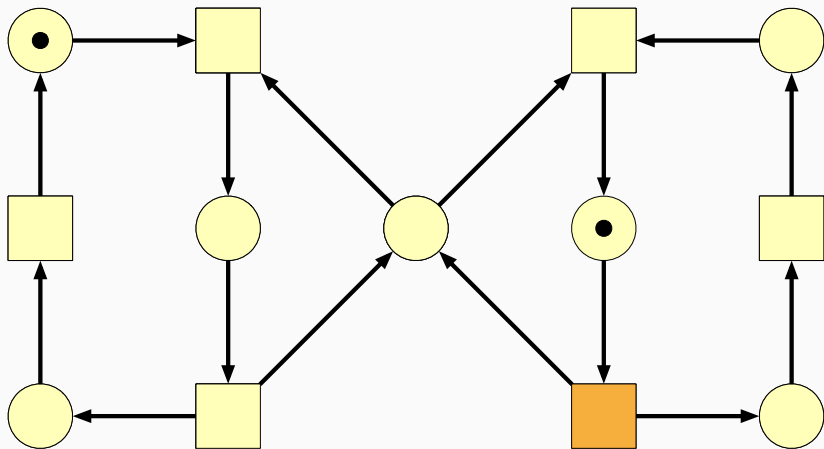
Petrinetze: Was soll das sein?



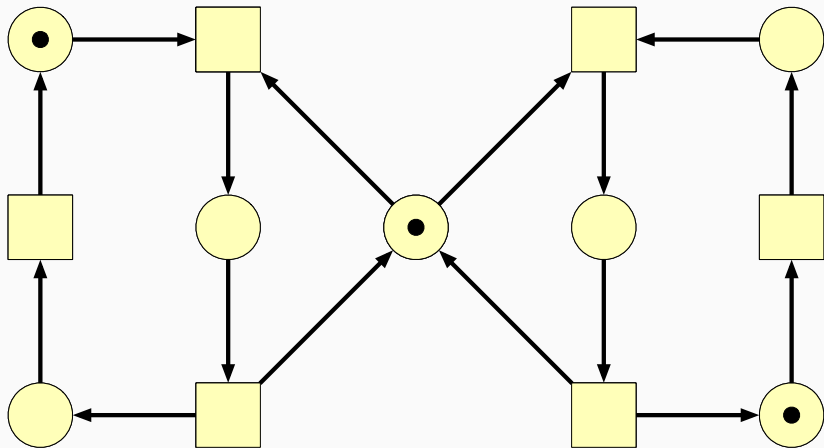
Petrinetze: Was soll das sein?



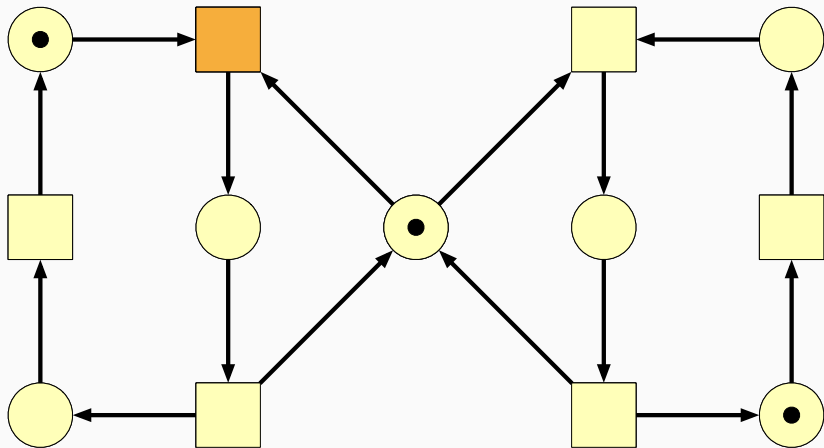
Petrinetze: Was soll das sein?



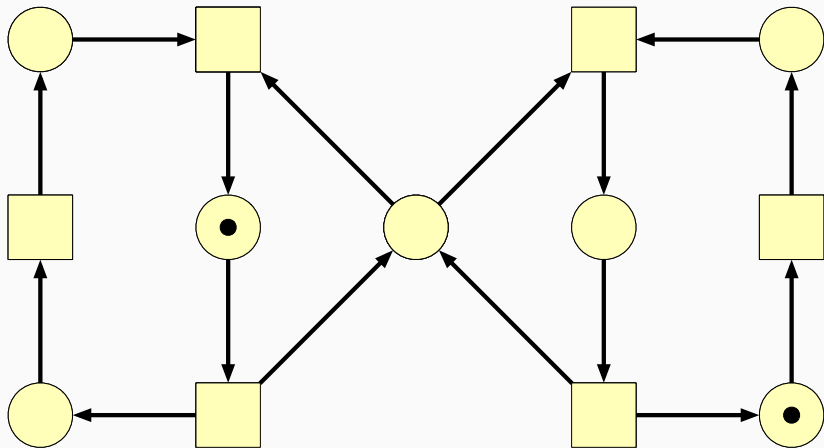
Petrinetze: Was soll das sein?



Petrinetze: Was soll das sein?



Petrinetze: Was soll das sein?



Übersicht

- 1 🖥️ Informatik, Modellierung & Mathematik
- 2 🧸 Graphische Einführung in Petrinetze
- 3 🎓 Formale Einführung in Petrinetze
- 4 🍇 Reaping the Fruits!
- 5 🍴 Let's go Dining!
- 6 📖 Zusammenfassung & 🔭 Ausblick



1) Informatik, Modellierung & Mathematik

Computer science is *not about machines* in the same way that astronomy is not about Telescopes. There is an *essential unity* of *mathematics* and *computer science*.

—*Michael Fellows*, Professor @ Universität Bergen



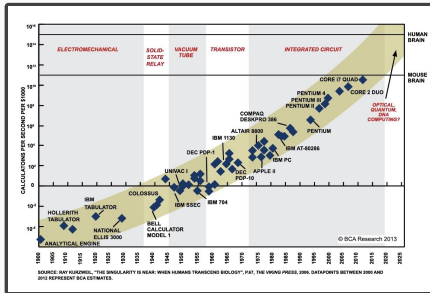
Technologischer Fortschritt = bessere Hardware?

Moore's Law

[Wikipedia Link](#)

Die Komplexität integrierter Schaltkreise verdoppelt sich etwa alle 2 Jahre.

oder
ähnlich



Technologischer Fortschritt = bessere Hardware?

Moore's Law

[Wikipedia Link](#)

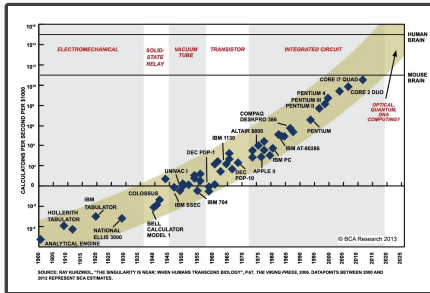
Die Komplexität integrierter Schaltkreise verdoppelt sich etwa alle 2 Jahre.

oder
ähnlich

Lösung von Optimierungsproblem

- 1988: 82 Jahre
- 2003: 1 Minute

⇒ Faktor $43 \cdot 10^6$



Technologischer Fortschritt = bessere Hardware?

Moore's Law

[Wikipedia Link](#)

Die Komplexität integrierter Schaltkreise verdoppelt sich etwa alle 2 Jahre.

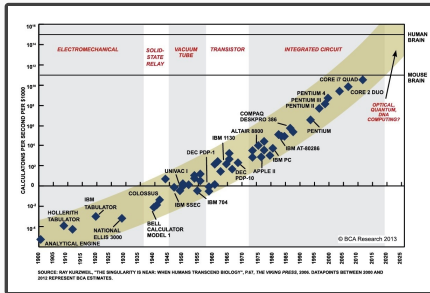
oder
ähnlich

Lösung von Optimierungsproblem

- 1988: 82 Jahre
- 2003: 1 Minute

⇒ Faktor $43 \cdot 10^6$

HW-Verbesserung: $\leq 10^3$



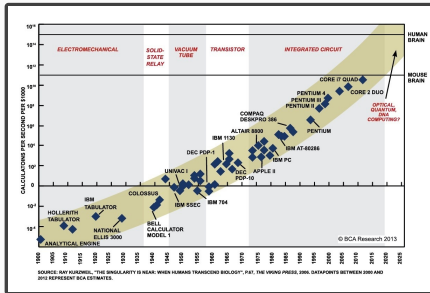
Technologischer Fortschritt = bessere Hardware?

Moore's Law

[Wikipedia Link](#)

Die Komplexität integrierter Schaltkreise verdoppelt sich etwa alle 2 Jahre.

oder
ähnlich



Lösung von Optimierungsproblem

- 1988: 82 Jahre
- 2003: 1 Minute

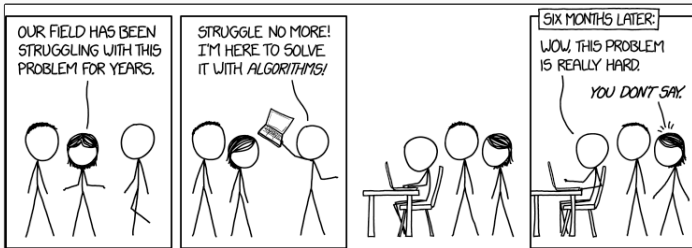
⇒ Faktor $43 \cdot 10^6$

HW-Verbesserung: $\leq 10^3$

Rest

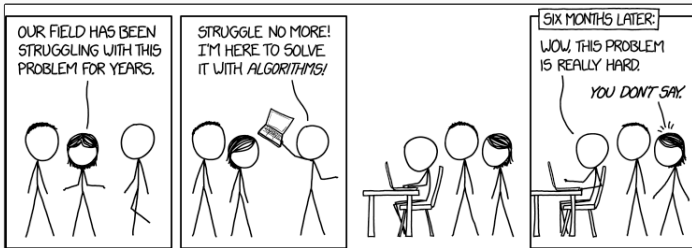
Algorithmen!

Gute Algorithmen finden: Wie schwer kann das schon sein?



[click for more xkcd comics](#)

Gute Algorithmen finden: Wie schwer kann das schon sein?

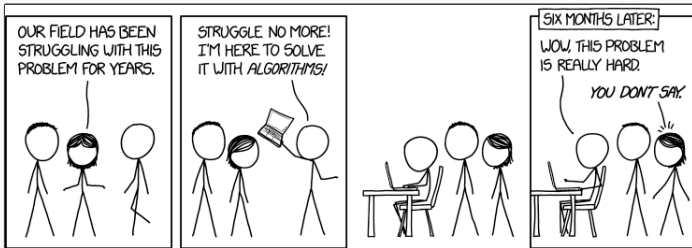


[click for more xkcd comics](#)

Problem-Lösung

- 1) Verstehen des Problems
- 2) Formalisierung & Modellierung
- 3) Entwicklung eines Algorithmus
- 4) Beweis der Korrektheit & Effizienz

Gute Algorithmen finden: Wie schwer kann das schon sein?



[click for more xkcd comics](#)

Problem-Lösung

- 1) Verstehen des Problems
- 2) Formalisierung & Modellierung
- 3) Entwicklung eines Algorithmus
- 4) Beweis der Korrektheit & Effizienz

Informatiker

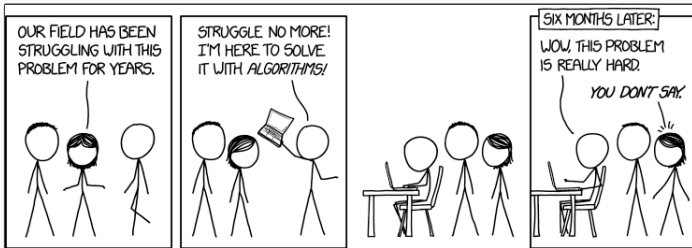
=

~~Programmierer~~

GPPS

General Purpose Problem Solver

Gute Algorithmen finden: Wie schwer kann das schon sein?



[click for more xkcd comics](#)

Problem-Lösung

- 1) Verstehen des Problems
- 2) **Formalisierung & Modellierung**
- 3) Entwicklung eines Algorithmus
- 4) Beweis der Korrektheit & Effizienz

Informatiker

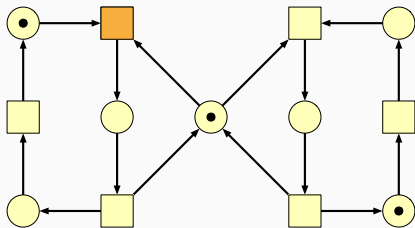
=

~~Programmierer~~

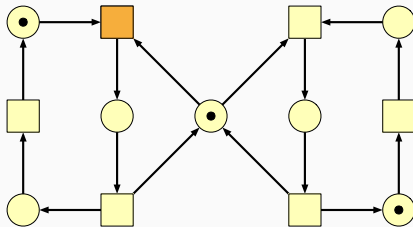
GPPS

General Purpose Problem Solver

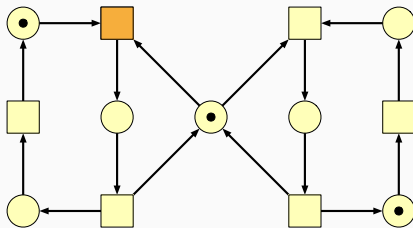
2) 🧸 Graphische Einführung in Petrinetze



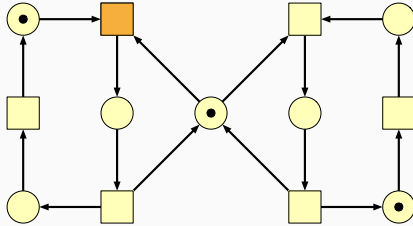
- eine (von vielen!) Modellierungstechnik



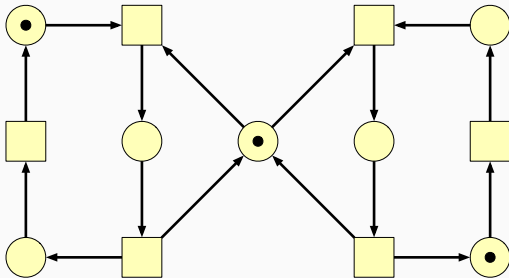
- eine (von vielen!) Modellierungstechnik
- gute graphische Visualisierung
 - ⇒ Grundidee intuitiv verständlich

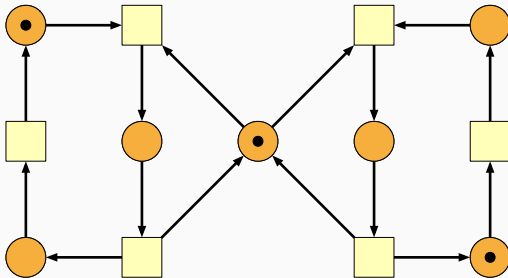


- eine (von vielen!) Modellierungstechnik
- gute graphische Visualisierung
 - ⇒ Grundidee intuitiv verständlich
- können Nebenläufigkeiten & Parallelität abbilden

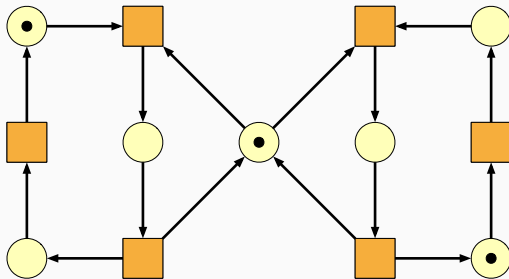


- eine (von vielen!) Modellierungstechnik
- gute graphische Visualisierung
 - ⇒ Grundidee intuitiv verständlich
- können Nebenläufigkeiten & Parallelität abbilden
- mathematische Definition
 - ⇒ präzise & exakt
 - ⇒ erlaubt formale (und ggfs. automatisierte) Überprüfung

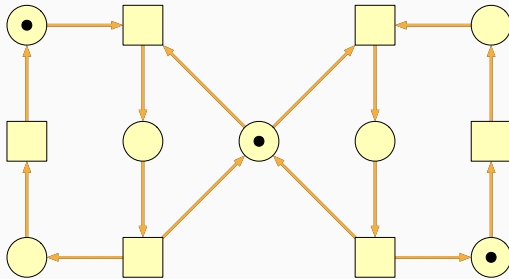




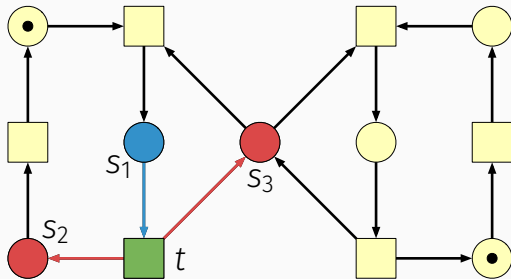
- Stellen: passive Komponenten



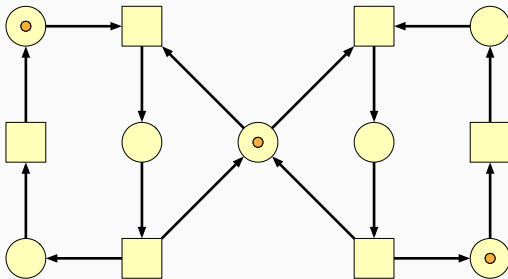
- Stellen: passive Komponenten
- Transitionen: aktive Komponenten



- Stellen: passive Komponenten
- Transitionen: aktive Komponenten
- Kanten: gerichtete Verbindung von Stellen und Transitionen

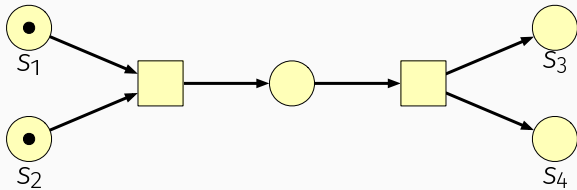


- Stellen: passive Komponenten
- Transitionen: aktive Komponenten
- Kanten: gerichtete Verbindung von Stellen und Transitionen
 - Eingangsstelle von Transition t : S_1
 - Ausgangsstelle von Transition t : S_2 und S_3

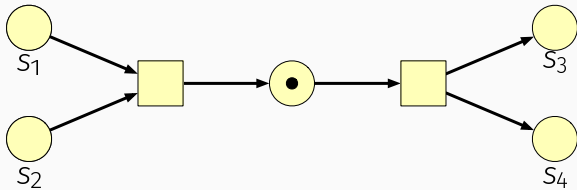


- Stellen: passive Komponenten
- Transitionen: aktive Komponenten
- Kanten: gerichtete Verbindung von Stellen und Transitionen
 - Eingangsstelle von Transition t : s_1
 - Ausgangsstelle von Transition t : s_2 und s_3
- Marken: „lokaler Zustand“ einer Stelle

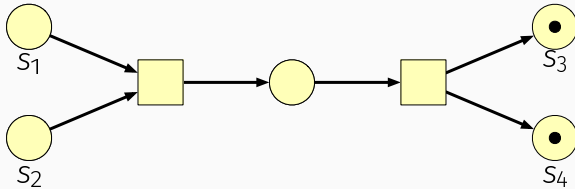
Dynamik von Petrinetzen



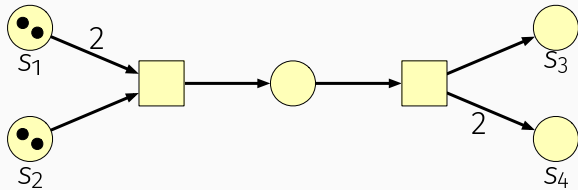
Dynamik von Petrinetzen



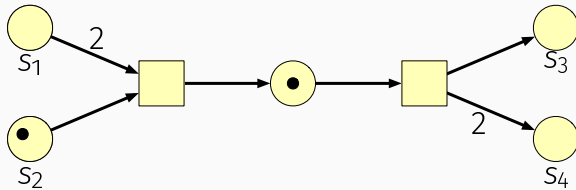
Dynamik von Petrinetzen



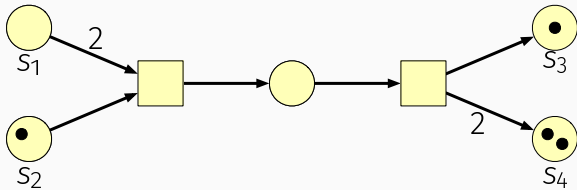
Dynamik von Petrinetzen



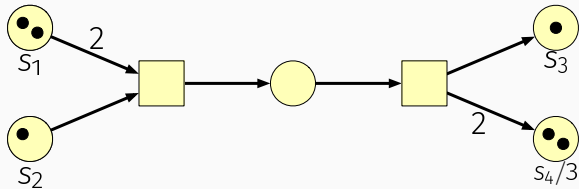
Dynamik von Petrinetzen



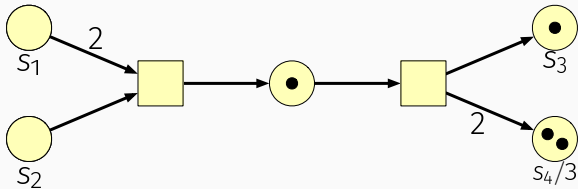
Dynamik von Petrinetzen



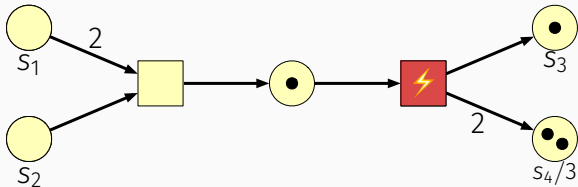
Dynamik von Petrinetzen

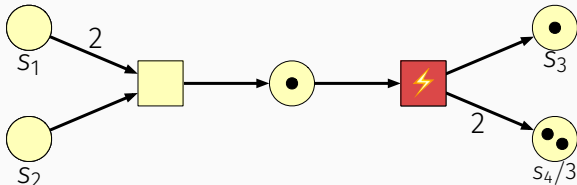


Dynamik von Petrinetzen



Dynamik von Petrinetzen

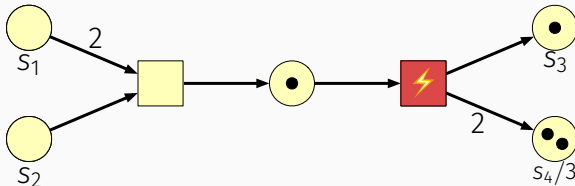




Definition 1: Aktiviertheit

Eine Transition ist **aktiviert**, wenn

- (a) alle Eingangsstellen ausreichend Marken beinhalten und
- (b) die Kapazität jeder Ausgangsstellen ausreicht, um entsprechend viele zusätzliche Marken aufzunehmen.



Definition 1: Aktiviertheit

Eine Transition ist **aktiviert**, wenn

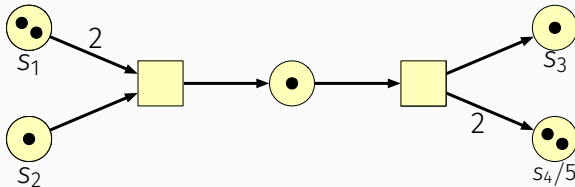
- (a) alle Eingangsstellen ausreichend Marken beinhalten und
- (b) die Kapazität jeder Ausgangsstellen ausreicht, um entsprechend viele zusätzliche Marken aufzunehmen.

Definition 2: Schalten

Ist eine Transition aktiviert, so **kann** sie **schalten**. Dabei

- (a) werden von allen Eingangsstellen Marken entfernt und
- (b) zu allen Ausgangsstellen Marken gelegt.

Dies geschieht entsprechend der Kantengewichtung.



Definition 1: Aktiviertheit

Eine Transition ist **aktiviert**, wenn

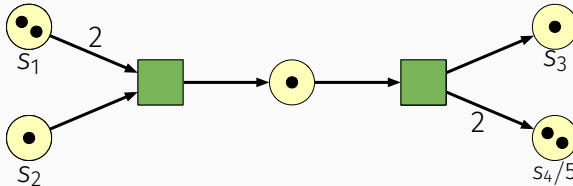
- (a) alle Eingangsstellen ausreichend Marken beinhalten und
- (b) die Kapazität jeder Ausgangsstellen ausreicht, um entsprechend viele zusätzliche Marken aufzunehmen.

Definition 2: Schalten

Ist eine Transition aktiviert, so **kann** sie **schalten**. Dabei

- (a) werden von allen Eingangsstellen Marken entfernt und
- (b) zu allen Ausgangsstellen Marken gelegt.

Dies geschieht entsprechend der Kantengewichtung.



Definition 1: Aktiviertheit

Eine Transition ist **aktiviert**, wenn

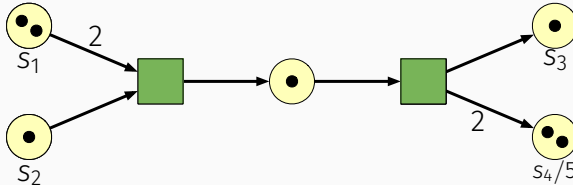
- (a) alle Eingangsstellen ausreichend Marken beinhalten und
- (b) die Kapazität jeder Ausgangsstellen ausreicht, um entsprechend viele zusätzliche Marken aufzunehmen.

Definition 2: Schalten

Ist eine Transition aktiviert, so **kann** sie **schalten**. Dabei

- (a) werden von allen Eingangsstellen Marken entfernt und
- (b) zu allen Ausgangsstellen Marken gelegt.

Dies geschieht entsprechend der Kantengewichtung.



Definition 1: Aktiviertheit

Eine Transition ist **aktiviert**, wenn

- (a) alle Eingangsstellen ausreichend Marken beinhalten und
- (b) die Kapazität jeder Ausgangsstellen ausreicht, um entsprechend viele zusätzliche Marken aufzunehmen.

Definition 2: Schalten

Ist eine Transition aktiviert, so **kann** sie **schalten**. Dabei

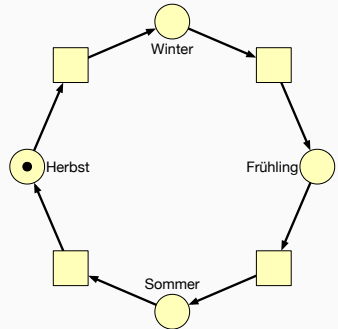
- (a) werden von allen Eingangsstellen Marken entfernt und
- (b) zu allen Ausgangsstellen Marken gelegt.

Dies geschieht entsprechend der Kantengewichtung.

Beispiel: Modell der vier Jahreszeiten

Beispiel: Modell der vier Jahreszeiten

- eine Stelle pro Jahreszeit
- Position der Marke beschreibt aktuelle Jahreszeit
- sich ändernde Jahreszeiten
 - ~ dynamisches System
 - ~ modelliert durch Transitionen

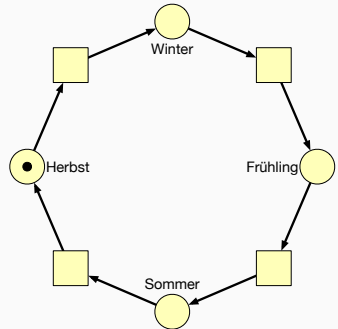


Beispiel: Modell der vier Jahreszeiten

- eine Stelle pro Jahreszeit
- Position der Marke beschreibt aktuelle Jahreszeit
- sich ändernde Jahreszeiten
 - ~ dynamisches System
 - ~ modelliert durch Transitionen

Eigenschaften dieses Systems

- genau eine Marke (**Invariante**)
- Netz stellt **Kausalitäten** dar
- Transitionen schalten **sequentiell**

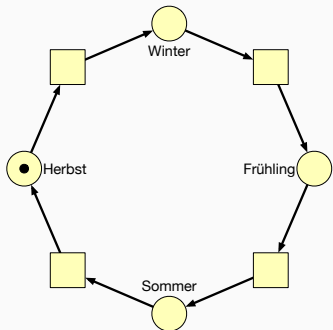


Beispiel: Modell der vier Jahreszeiten

- eine Stelle pro Jahreszeit
- Position der Marke beschreibt aktuelle Jahreszeit
- sich ändernde Jahreszeiten
 - ~ dynamisches System
 - ~ modelliert durch Transitionen

Eigenschaften dieses Systems

- genau eine Marke (**Invariante**)
- Netz stellt **Kausalitäten** dar
- Transitionen schalten **sequentiell**



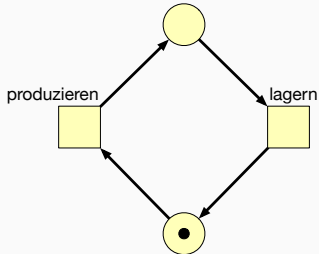
Nicht alle Systeme haben diese Eigenschaften!

Beispiel: Producer-Consumer

- Prozess 1: produziert eine Ressource
- Prozess 2: konsumiert eine Ressource

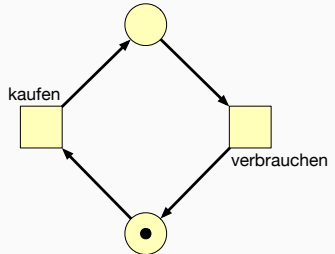
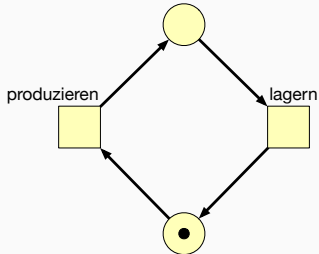
Beispiel: Producer-Consumer

- Prozess 1: produziert eine Ressource
- Prozess 2: konsumiert eine Ressource



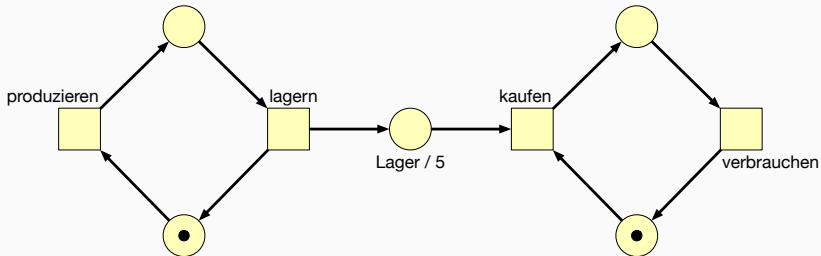
Beispiel: Producer-Consumer

- Prozess 1: produziert eine Ressource
- Prozess 2: konsumiert eine Ressource



Beispiel: Producer-Consumer

- Prozess 1: produziert eine Ressource
- Prozess 2: konsumiert eine Ressource



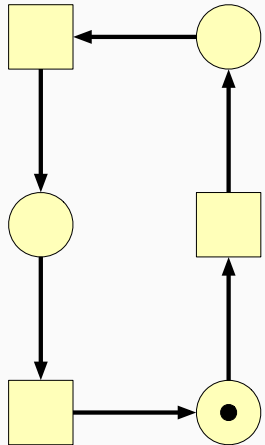
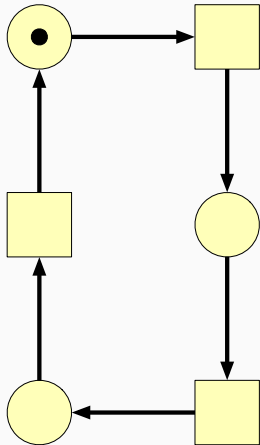
Zu Modellieren

Zwei Prozesse, die in einem kritischen Bereich eine gemeinsame Ressource alleine nutzen wollen.

Beispiel: Kritischer Bereich

Zu Modellieren

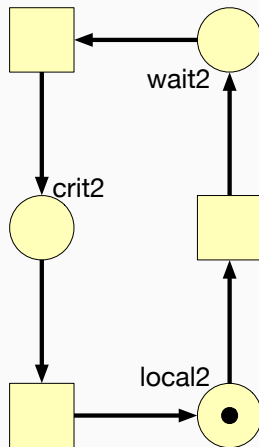
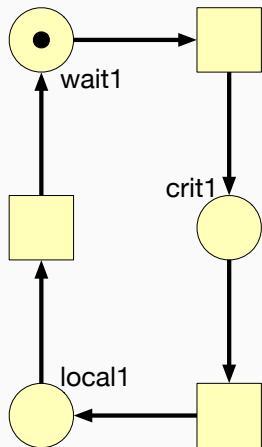
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

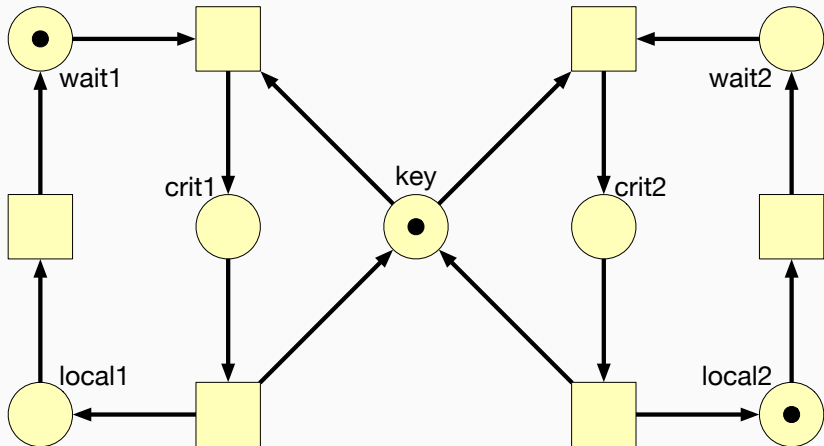
Zu Modellieren

Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Zu Modellieren

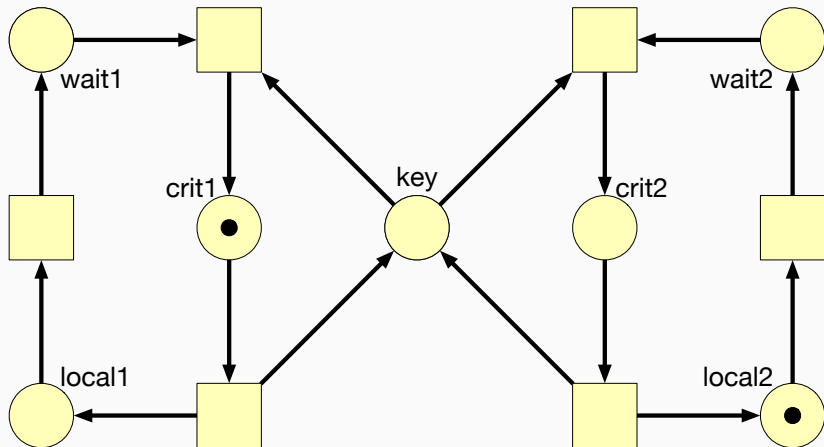
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

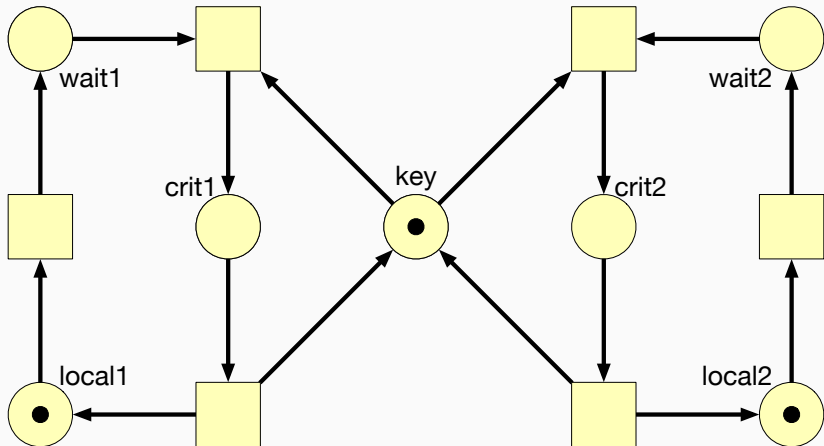
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

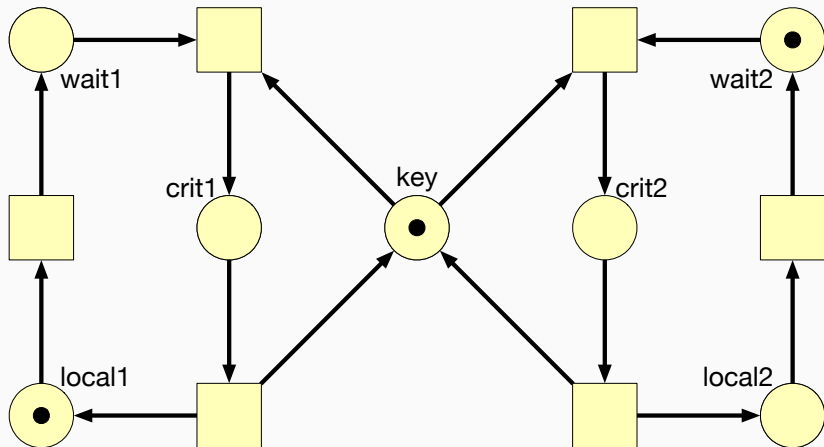
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

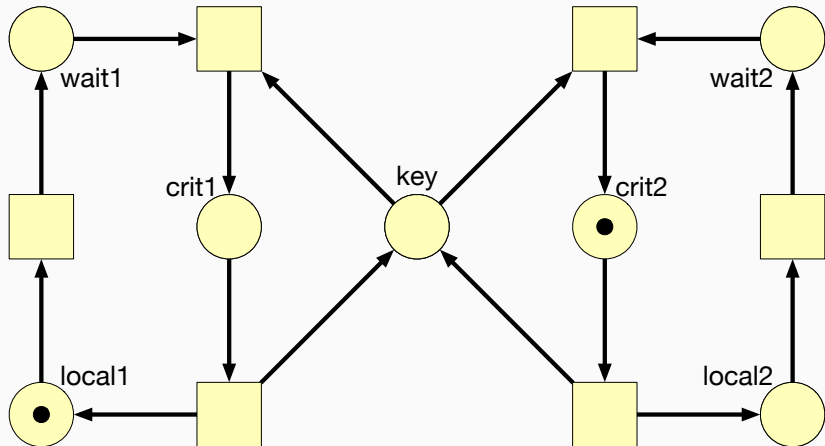
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

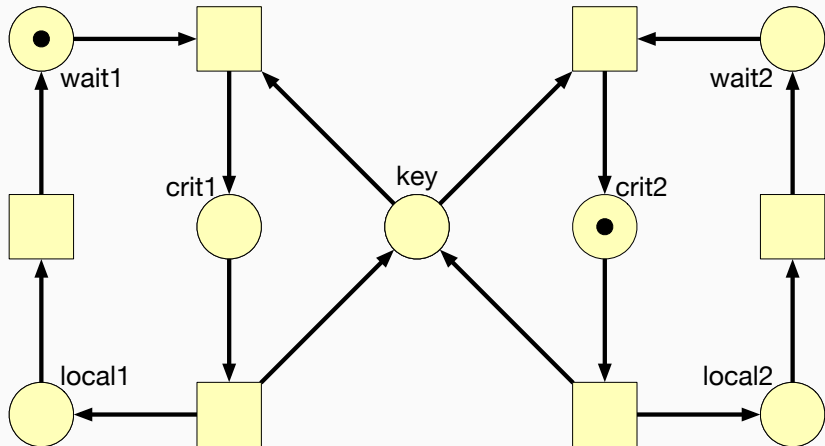
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

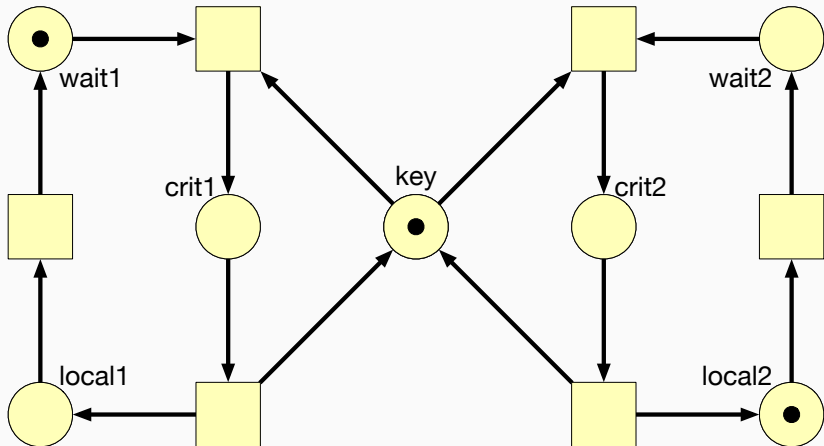
Zu Modellieren

Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Zu Modellieren

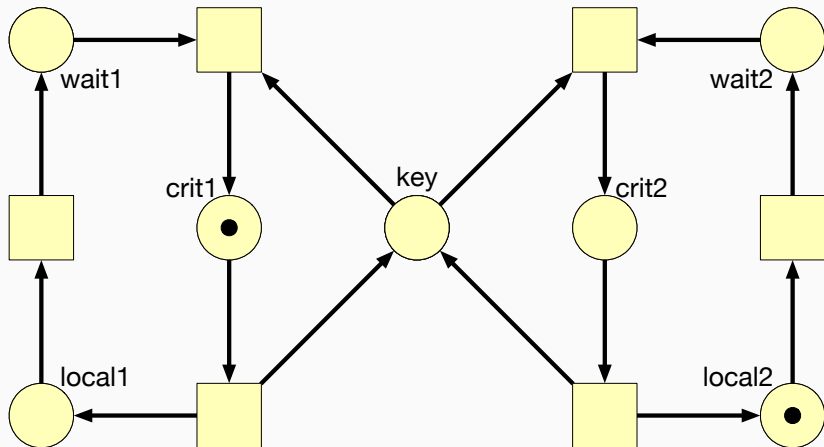
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

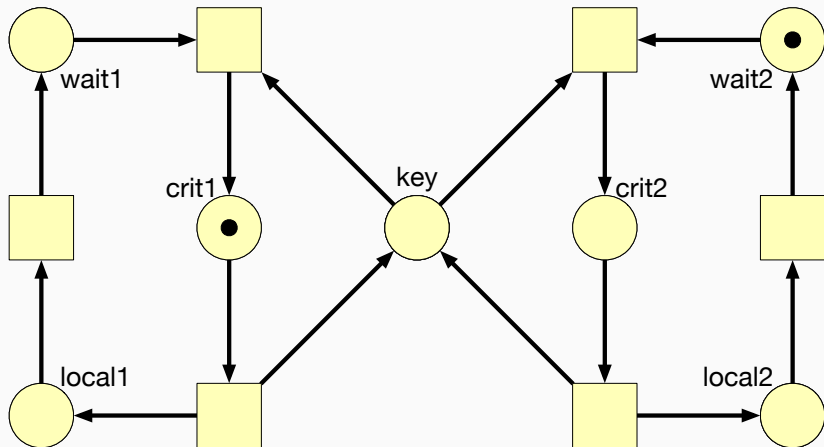
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

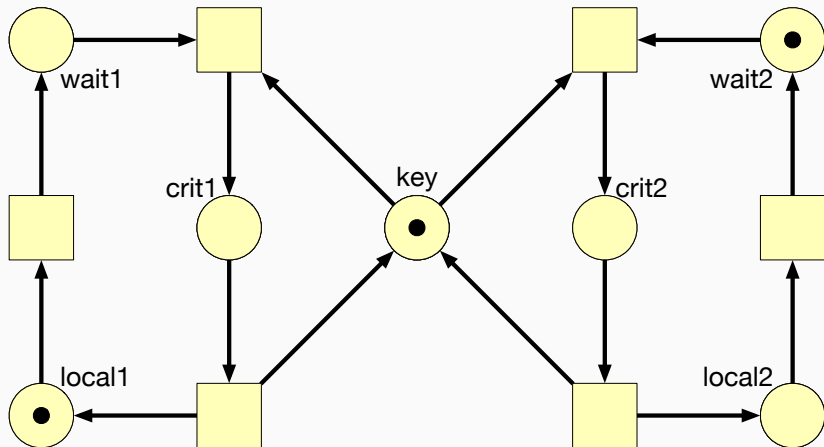
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

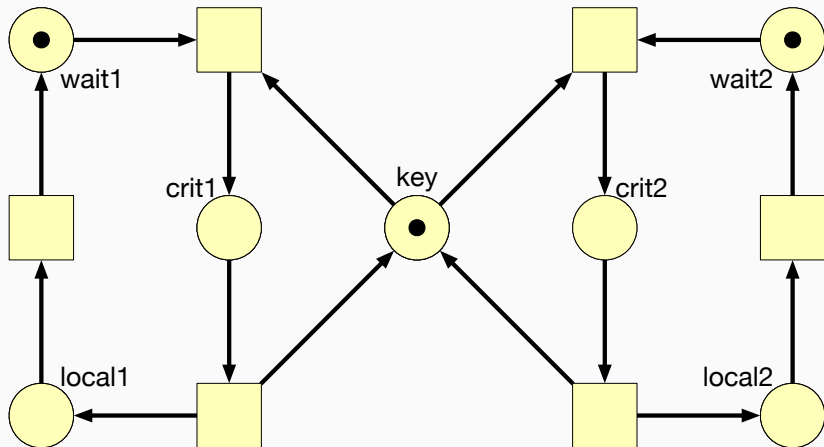
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

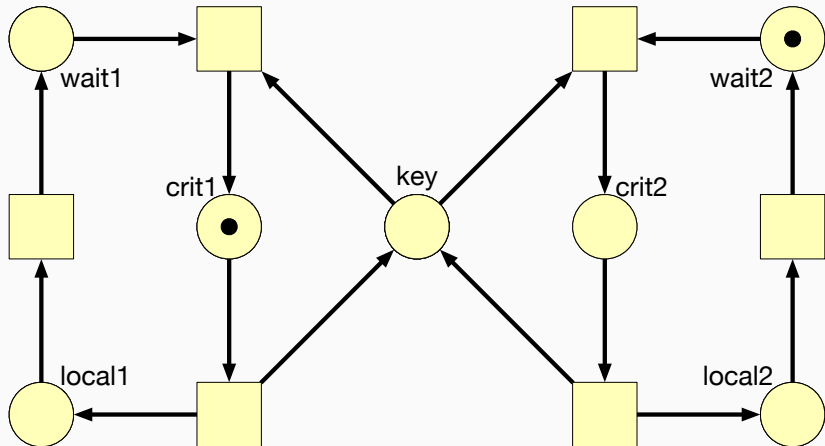
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

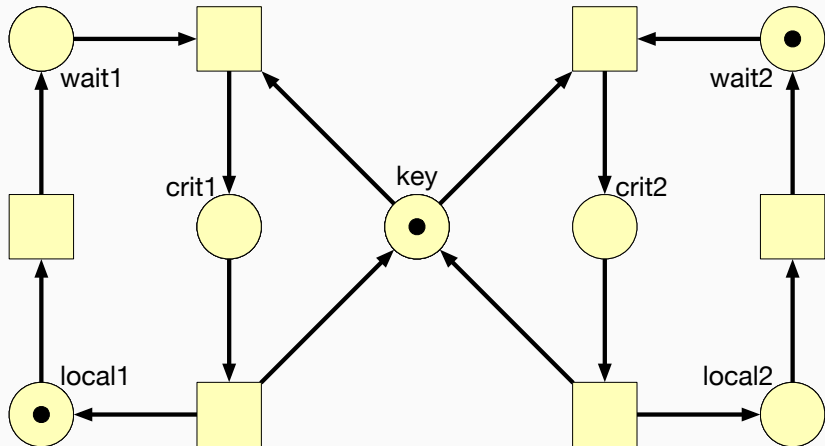
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

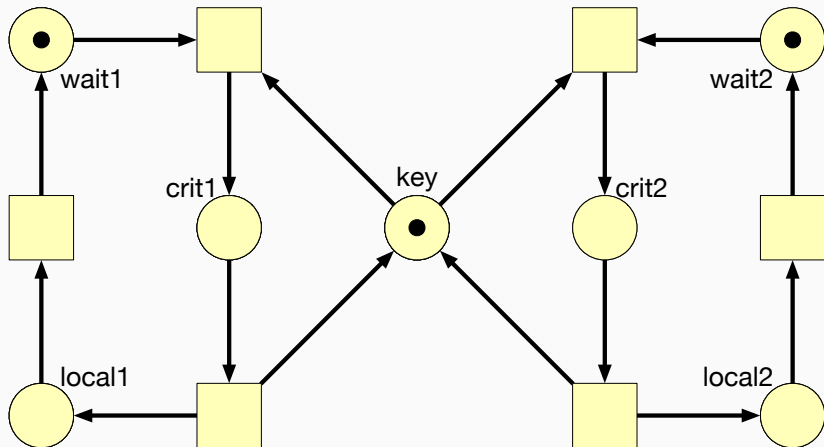
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

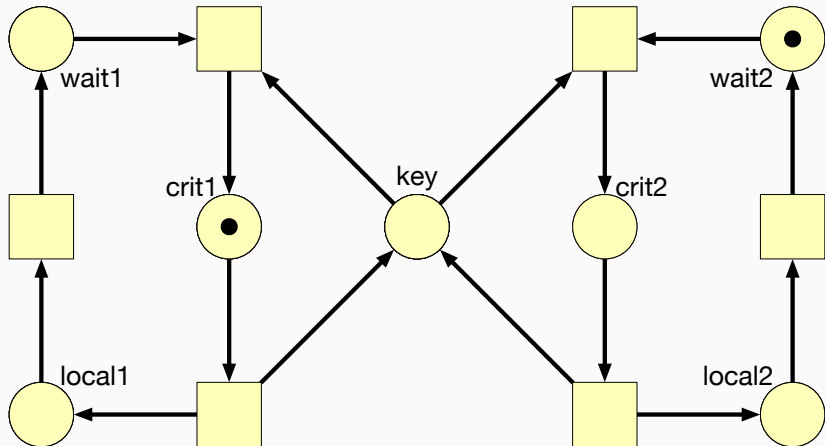
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

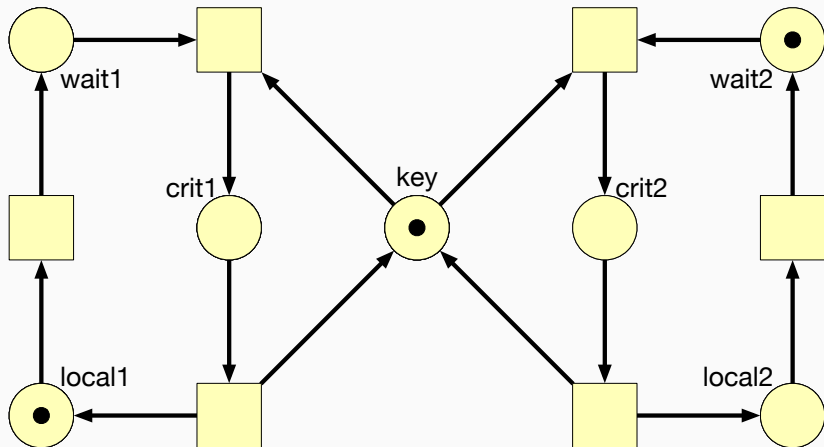
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

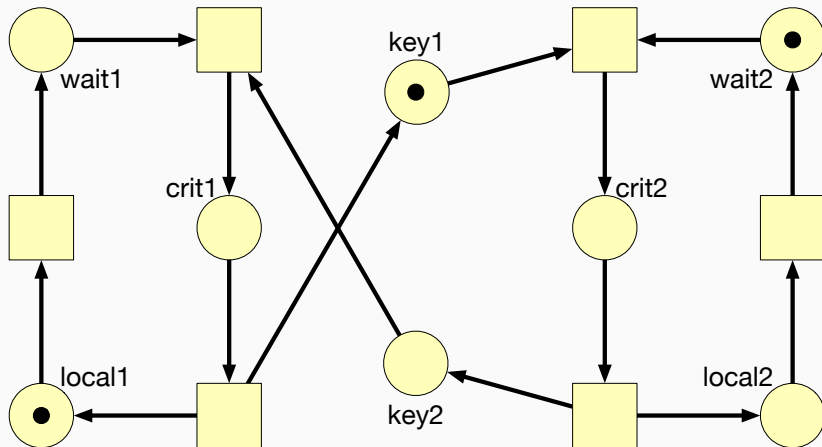
Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



Beispiel: Kritischer Bereich

Zu Modellieren

Zwei Prozesse, die in einem **kritischen Bereich** eine **gemeinsame Ressource** alleine nutzen wollen.



3) 🎓 Formale Einführung in Petrinetze

- Menge: Ansammlung von Elementen
 - z. B. $M_1 = \{1, 2, 3\}$ und $M_2 = \{2, \diamond\}$

- Menge: Ansammlung von Elementen
 - z. B. $M_1 = \{1, 2, 3\}$ und $M_2 = \{2, \diamond\}$
- Enthaltensein: Gehört ein Element zu einer Menge?
 - z. B. $1 \in M_1$ aber $1 \notin M_2$

- Menge: Ansammlung von Elementen
 - z. B. $M_1 = \{1, 2, 3\}$ und $M_2 = \{2, \diamond\}$
- Enthaltensein: Gehört ein Element zu einer Menge?
 - z. B. $1 \in M_1$ aber $1 \notin M_2$
- Teilmenge: Menge ist Teil einer anderen Menge
 - z. B. $\{1, 2\} \subseteq M_1$
 - aber $M_1 \not\subseteq M_2$ und $M_2 \not\subseteq M_1$

- Menge: Ansammlung von Elementen
 - z. B. $M_1 = \{1, 2, 3\}$ und $M_2 = \{2, \diamond\}$
- Enthaltensein: Gehört ein Element zu einer Menge?
 - z. B. $1 \in M_1$ aber $1 \notin M_2$
- Teilmenge: Menge ist Teil einer anderen Menge
 - z. B. $\{1, 2\} \subseteq M_1$
 - aber $M_1 \not\subseteq M_2$ und $M_2 \not\subseteq M_1$
- Vereinigung: Zusammenwerfen zweier Mengen
 - z. B. $M_1 \cup M_2 = \{1, 2, 3, \diamond\}$

- Menge: Ansammlung von Elementen
 - z. B. $M_1 = \{1, 2, 3\}$ und $M_2 = \{2, \diamond\}$
- Enthaltensein: Gehört ein Element zu einer Menge?
 - z. B. $1 \in M_1$ aber $1 \notin M_2$
- Teilmenge: Menge ist Teil einer anderen Menge
 - z. B. $\{1, 2\} \subseteq M_1$
 - aber $M_1 \not\subseteq M_2$ und $M_2 \not\subseteq M_1$
- Vereinigung: Zusammenwerfen zweier Mengen
 - z. B. $M_1 \cup M_2 = \{1, 2, 3, \diamond\}$
- Schnitt: gemeinsame Elemente
 - z. B. $M_1 \cap M_2 = \{2\}$

- **Menge**: Ansammlung von Elementen
 - z. B. $M_1 = \{1, 2, 3\}$ und $M_2 = \{2, \diamond\}$
- **Enthaltensein**: Gehört ein Element zu einer Menge?
 - z. B. $1 \in M_1$ aber $1 \notin M_2$
- **Teilmenge**: Menge ist Teil einer anderen Menge
 - z. B. $\{1, 2\} \subseteq M_1$
 - aber $M_1 \not\subseteq M_2$ und $M_2 \not\subseteq M_1$
- **Vereinigung**: Zusammenwerfen zweier Mengen
 - z. B. $M_1 \cup M_2 = \{1, 2, 3, \diamond\}$
- **Schnitt**: gemeinsame Elemente
 - z. B. $M_1 \cap M_2 = \{2\}$
- **Kartesisches Produkt**: alle möglichen Pärchen
 - z. B. $M_1 \times M_2 = \{(1, 2), (1, \diamond), (2, 2), (2, \diamond), (3, 2), (3, \diamond)\}$

Definition 3: Abbildung

Eine **Abbildung**, notiert als

$$f: A \rightarrow B,$$

bildet jedes Element $a \in A$ auf ein Element $f(a) \in B$ ab.

Definition 3: Abbildung

Eine **Abbildung**, notiert als

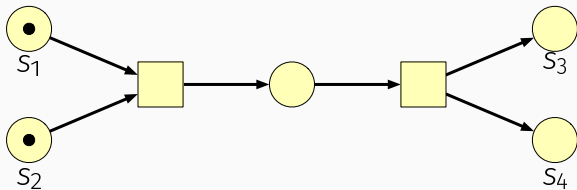
$$f: A \rightarrow B,$$

bildet jedes Element $a \in A$ auf ein Element $f(a) \in B$ ab.

Beispiele

- $f: \mathbb{N} \rightarrow \mathbb{N}$ mit $n \mapsto n^2$.
- $g: \mathbb{Z} \rightarrow \mathbb{N}$ mit $x \mapsto |x|$.
- $h: \{\diamond, \circ\} \rightarrow \{1, 2\}$ mit $h(\diamond) = h(\circ) = 1$.

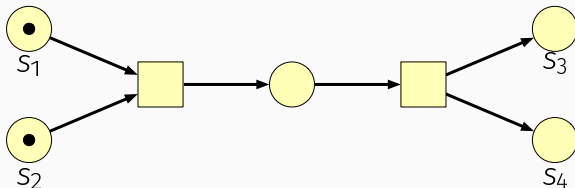
Formale Definition von Petri-Netzen



Definition 4: P/T-Netz

Ein **P/T-Netz** ist ein Tupel $N = (P, T, F, W, m_0)$ mit:

Formale Definition von Petri-Netzen

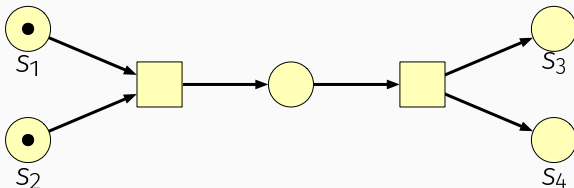


Definition 4: P/T-Netz

Ein **P/T-Netz** ist ein Tupel $N = (P, T, F, W, m_0)$ mit:

- einer endlichen Menge P von **Plätzen** (Stellen),

Formale Definition von Petri-Netzen

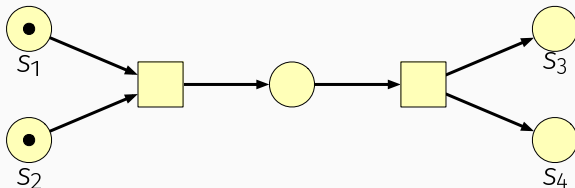


Definition 4: P/T-Netz

Ein **P/T-Netz** ist ein Tupel $N = (P, T, F, W, m_0)$ mit:

- einer endlichen Menge P von **Plätzen** (Stellen),
- einer endlichen Menge T von **Transitionen** mit $P \cap T = \emptyset$,

Formale Definition von Petri-Netzen

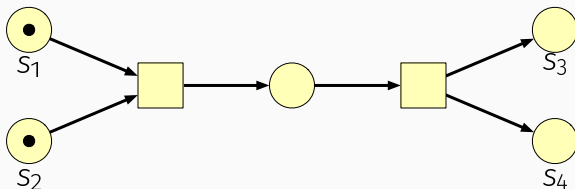


Definition 4: P/T-Netz

Ein **P/T-Netz** ist ein Tupel $N = (P, T, F, W, m_0)$ mit:

- einer endlichen Menge P von **Plätzen** (Stellen),
- einer endlichen Menge T von **Transitionen** mit $P \cap T = \emptyset$,
- einer **Flussrelation** $F \subseteq (P \times T) \cup (T \times P)$,

Formale Definition von Petri-Netzen

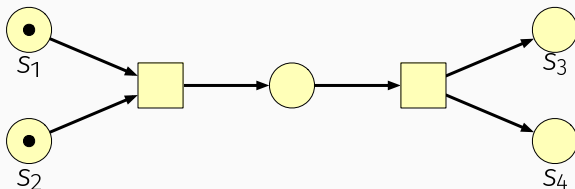


Definition 4: P/T-Netz

Ein **P/T-Netz** ist ein Tupel $N = (P, T, F, W, m_0)$ mit:

- einer endlichen Menge P von **Plätzen** (Stellen),
- einer endlichen Menge T von **Transitionen** mit $P \cap T = \emptyset$,
- einer **Flussrelation** $F \subseteq (P \times T) \cup (T \times P)$,
- einer **Kantenbewertung** $W: (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$
mit $W(x, y) = 0 \iff (x, y) \notin F$

Formale Definition von Petri-Netzen

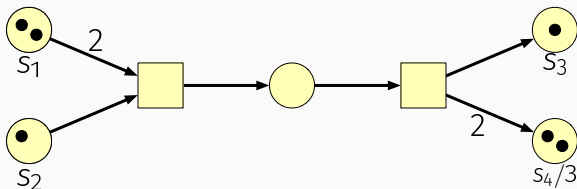


Definition 4: P/T-Netz

Ein **P/T-Netz** ist ein Tupel $N = (P, T, F, W, m_0)$ mit:

- einer endlichen Menge P von **Plätzen** (Stellen),
- einer endlichen Menge T von **Transitionen** mit $P \cap T = \emptyset$,
- einer **Flussrelation** $F \subseteq (P \times T) \cup (T \times P)$,
- einer **Kantenbewertung** $W: (P \times T) \cup (T \times P) \rightarrow \mathbb{N}$
mit $W(x, y) = 0 \iff (x, y) \notin F$
- und einer **Anfangsmarkierung** $m_0: P \rightarrow \mathbb{N}$.

Erweiterung der Definition für Kapazitäten

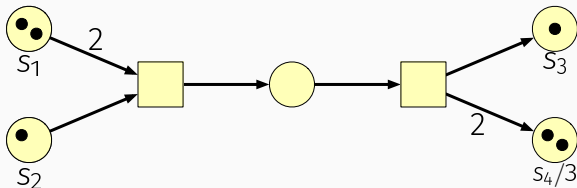


Definition 5: P/T-Netz mit Kapazitäten

Ein **P/T-Netz mit Kapazitäten** ist ein Tupel $N = (P, T, F, W, K, m_0)$ mit:

- einem P/T-Netz $N' = (P, T, F, W, m_0)$,
- einer **Kapazitätsfunktion** $K: P \rightarrow \mathbb{N} \cup \{\infty\}$
- und $m_0(p) \leq K(p)$ für alle $p \in P$.

Erweiterung der Definition für Kapazitäten



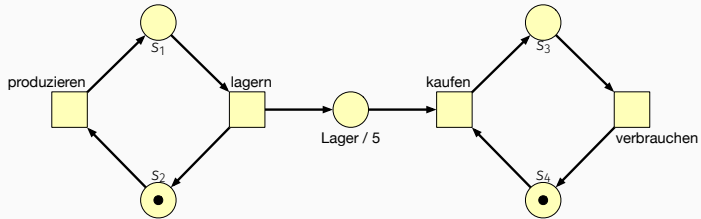
Definition 5: P/T-Netz mit Kapazitäten

Ein **P/T-Netz mit Kapazitäten** ist ein Tupel $N = (P, T, F, W, K, m_0)$ mit:

- einem P/T-Netz $N' = (P, T, F, W, m_0)$,
- einer **Kapazitätsfunktion** $K: P \rightarrow \mathbb{N} \cup \{\infty\}$
- und $m_0(p) \leq K(p)$ für alle $p \in P$.

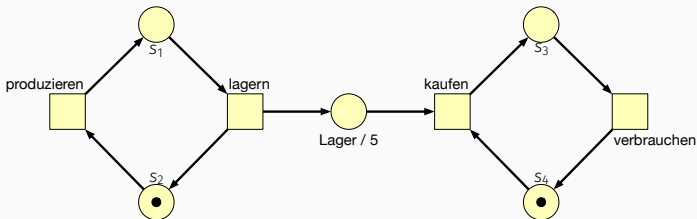
auch ω
statt ∞

Beispiel zur formalen Definition



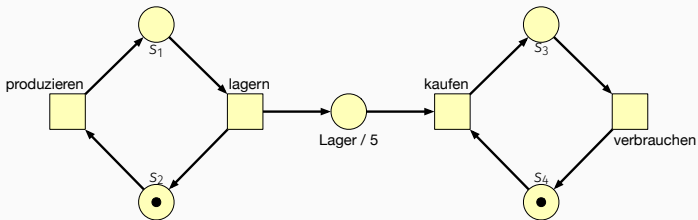
- $P =$
- $T =$
- $F =$

Beispiel zur formalen Definition



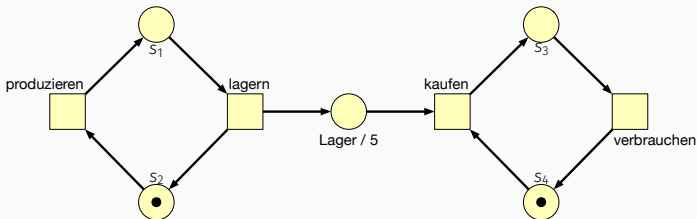
- $P = \{ S_1, S_2, S_3, S_4, \text{Lager} \}$
- $T = \{ \text{lagern}, \text{produzieren}, \text{kaufen}, \text{verbrauchen} \}$
- $F = \{ (\text{produzieren}, S_1), (S_1, \text{lagern}), (\text{lagern}, S_2), (S_2, \text{produzieren}), (\text{lagern}, \text{Lager}), (\text{Lager}, \text{kaufen}), (\text{kaufen}, S_3), (S_3, \text{verbrauchen}), (\text{verbrauchen}, S_4), (S_4, \text{kaufen}) \}$

Beispiel zur formalen Definition



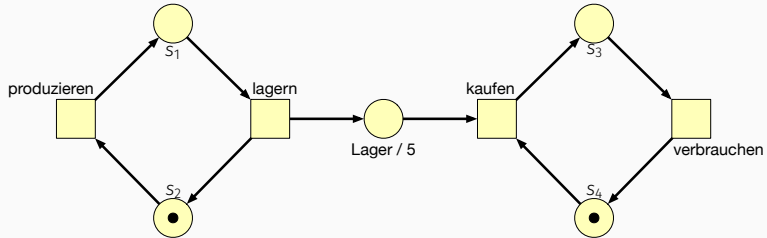
- W ist gegeben durch:
- m_0 ist gegeben durch:
- K ist gegeben durch:

Beispiel zur formalen Definition

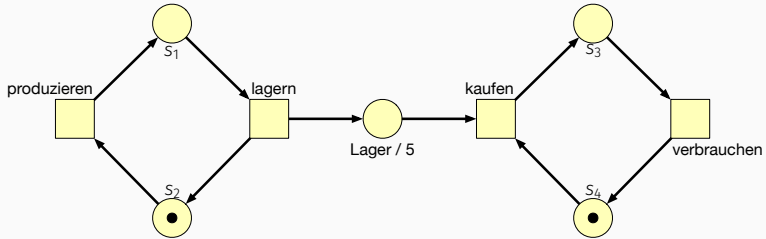


- W ist gegeben durch:
 $W(x, y) = 1$ für alle $(x, y) \in F$ und $W(x, y) = 0$ sonst
- m_0 ist gegeben durch:
 $m_0(s_2) = m_0(s_4) = 1$ und $m_0(s_1) = m_0(s_3) = m_0(\text{Lager}) = 0$
- K ist gegeben durch:
 $K(\text{Lager}) = 5$ und $K(s_1) = K(s_2) = K(s_3) = K(s_4) = \infty$

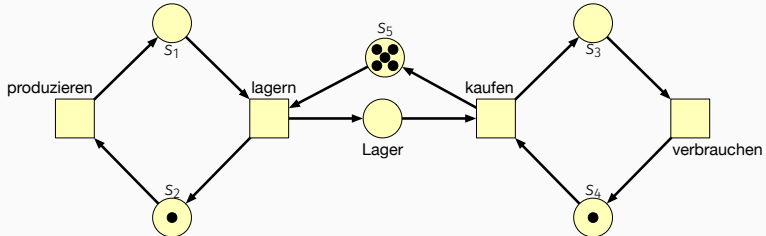
Modellierungsmächtigkeit: Sind Kapazitäten wirklich notwendig?



Modellierungsmächtigkeit: Sind Kapazitäten wirklich notwendig?



P/T-Netze mit und ohne Kapazitäten sind äquivalent!



Gegeben:

- ein P/T-Netz (mit Kapazitäten) $N = (P, T, F, W, K, m_0)$,
- eine Transition $t \in T$ und
- eine Markierung m_1 .

Gegeben:

- ein P/T-Netz (mit Kapazitäten) $N = (P, T, F, W, K, m_0)$,
- eine Transition $t \in T$ und
- eine Markierung m_1 .

Definition 6: Aktivierung

Die Transition t ist **aktiviert** in m_1 , falls für alle $p \in P$ folgendes gilt:

Gegeben:

- ein P/T-Netz (mit Kapazitäten) $N = (P, T, F, W, K, m_0)$,
- eine Transition $t \in T$ und
- eine Markierung m_1 .

Definition 6: Aktivierung

Die Transition t ist **aktiviert** in m_1 , falls für alle $p \in P$ folgendes gilt:

- (a) $m_1(p) \geq W(p, t)$ und
- (b) $m_1(p) - W(p, t) + W(t, p) \leq K(p)$.

Gegeben:

- ein P/T-Netz (mit Kapazitäten) $N = (P, T, F, W, K, m_0)$,
- eine Transition $t \in T$ und
- eine Markierung m_1 .

Definition 6: Aktivierung

Die Transition t ist **aktiviert** in m_1 , falls für alle $p \in P$ folgendes gilt:

- (a) $m_1(p) \geq W(p, t)$ und
- (b) $m_1(p) - W(p, t) + W(t, p) \leq K(p)$.

Wir schreiben $m_1 \xrightarrow{t}$.

Gegeben:

- ein P/T-Netz (mit Kapazitäten) $N = (P, T, F, W, K, m_0)$,
- eine Transition $t \in T$ und
- eine Markierung m_1 .

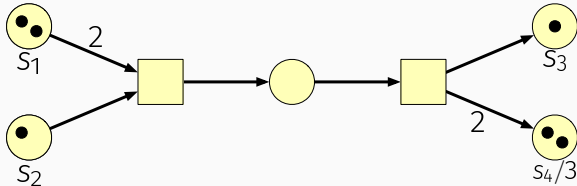
Definition 6: Aktivierung

Die Transition t ist **aktiviert** in m_1 , falls für alle $p \in P$ folgendes gilt:

- (a) $m_1(p) \geq W(p, t)$ und
- (b) $m_1(p) - W(p, t) + W(t, p) \leq K(p)$.

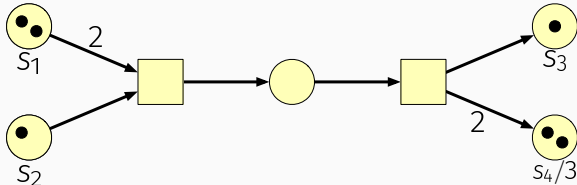
Wir schreiben $m_1 \xrightarrow{t}$.

P/T-Netz **ohne** Kapazitäten: nur erste Bedingung.



Definition 7: Schalten

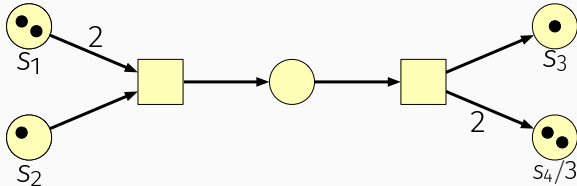
Sei $N = (P, T, F, W, K, m_0)$ ein P/T-Netz, $t \in T$ eine Transition und m_1, m_2 Markierungen. Die Transition t **schaltet** m_1 zu m_2 , falls



Definition 7: Schalten

Sei $N = (P, T, F, W, K, m_0)$ ein P/T-Netz, $t \in T$ eine Transition und m_1, m_2 Markierungen. Die Transition t **schaltet** m_1 zu m_2 , falls

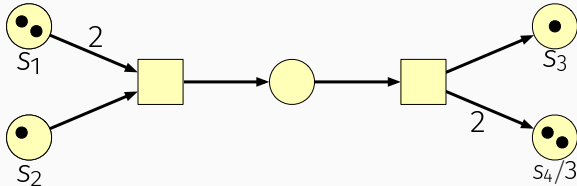
- (a) t in m_1 aktiviert ist und



Definition 7: Schalten

Sei $N = (P, T, F, W, K, m_0)$ ein P/T-Netz, $t \in T$ eine Transition und m_1, m_2 Markierungen. Die Transition t **schaltet** m_1 zu m_2 , falls

- (a) t in m_1 aktiviert ist und
- (b) $\forall p \in P: m_2(p) = m_1(p) - W(p, t) + W(t, p)$ gilt.



Definition 7: Schalten

Sei $N = (P, T, F, W, K, m_0)$ ein P/T-Netz, $t \in T$ eine Transition und m_1, m_2 Markierungen. Die Transition t **schaltet** m_1 zu m_2 , falls

- (a) t in m_1 aktiviert ist und
- (b) $\forall p \in P: m_2(p) = m_1(p) - W(p, t) + W(t, p)$ gilt.

Wir schreiben $m_1 \xrightarrow{t} m_2$ und nennen m_2 **Folgemarkierung** von m_1 .

Eine **Schaltfolge** ist ein endliches Wort

$$w = t_1 t_2 t_3 \dots t_n$$

mit $t_i \in T$ für alle $i \in \{1, \dots, n\}$ und $n \in \mathbb{N}$.

Eine **Schaltfolge** ist ein endliches Wort

$$w = t_1 t_2 t_3 \dots t_n$$

mit $t_i \in T$ für alle $i \in \{1, \dots, n\}$ und $n \in \mathbb{N}$.

Definition 8: Schalten einer Schaltfolge

Eine Schaltfolge w **schaltet** m zu m' , falls

Eine **Schaltfolge** ist ein endliches Wort

$$w = t_1 t_2 t_3 \dots t_n$$

mit $t_i \in T$ für alle $i \in \{1, \dots, n\}$ und $n \in \mathbb{N}$.

Definition 8: Schalten einer Schaltfolge

Eine **Schaltfolge** w **schaltet** m zu m' , falls

(a) entweder $w = \lambda$ (leeres Wort mit $n = 0$) und $m = m'$

Eine **Schaltfolge** ist ein endliches Wort

$$w = t_1 t_2 t_3 \dots t_n$$

mit $t_i \in T$ für alle $i \in \{1, \dots, n\}$ und $n \in \mathbb{N}$.

Definition 8: Schalten einer Schaltfolge

Eine **Schaltfolge** w **schaltet** m zu m' , falls

- (a) entweder $w = \lambda$ (leeres Wort mit $n = 0$) und $m = m'$
- (b) oder $w = (u \cdot t)$ für $u \in T^*$ und $t \in T$, so dass $m \xrightarrow{u} m_1$ und $m_1 \xrightarrow{t} m'$ für eine Markierung m_1 gilt.

Eine **Schaltfolge** ist ein endliches Wort

$$w = t_1 t_2 t_3 \dots t_n$$

mit $t_i \in T$ für alle $i \in \{1, \dots, n\}$ und $n \in \mathbb{N}$.

Definition 8: Schalten einer Schaltfolge

Eine **Schaltfolge** w **schaltet** m zu m' , falls

- (a) entweder $w = \lambda$ (leeres Wort mit $n = 0$) und $m = m'$
- (b) oder $w = (u \cdot t)$ für $u \in T^*$ und $t \in T$, so dass $m \xrightarrow{u} m_1$ und $m_1 \xrightarrow{t} m'$ für eine Markierung m_1 gilt.

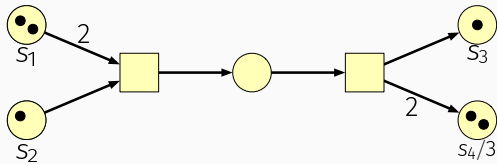
Wir schreiben $m \xrightarrow{w} m'$ oder $m \xrightarrow{*} m'$ (falls konkretes w unwichtig).

4) 🍇 Reaping the Fruits!



Definition 9

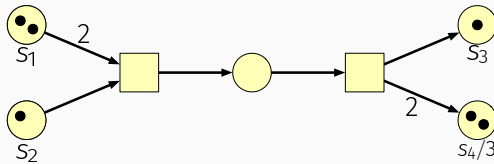
- m ist erreichbar, wenn





Definition 9

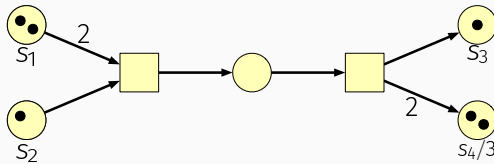
- m ist **erreichbar**, wenn es eine Schaltfolge w gibt mit $m_0 \xrightarrow{w} m$. Die Menge aller erreichbaren Markierungen eines Netzes N wird mit $R(N)$ bezeichnet.





Definition 9

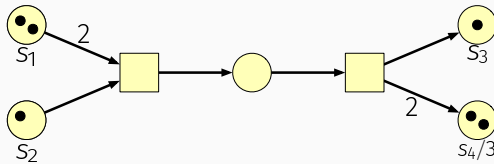
- m ist **erreichbar**, wenn es eine Schaltfolge w gibt mit $m_0 \xrightarrow{w} m$. Die Menge aller erreichbaren Markierungen eines Netzes N wird mit $R(N)$ bezeichnet.
- $t \in T$ ist **aktivierbar**, wenn





Definition 9

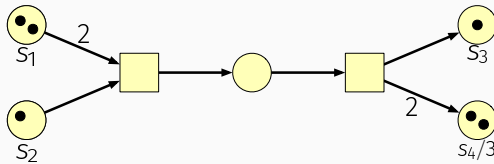
- m ist **erreichbar**, wenn es eine Schaltfolge w gibt mit $m_0 \xrightarrow{w} m$. Die Menge aller erreichbaren Markierungen eines Netzes N wird mit $R(N)$ bezeichnet.
- $t \in T$ ist **aktivierbar**, wenn es eine erreichbare Markierung m gibt mit $m \xrightarrow{t}$.





Definition 9

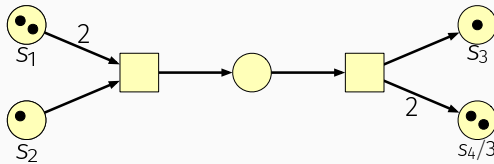
- m ist **erreichbar**, wenn es eine Schaltfolge w gibt mit $m_0 \xrightarrow{w} m$. Die Menge aller erreichbaren Markierungen eines Netzes N wird mit $R(N)$ bezeichnet.
- $t \in T$ ist **aktivierbar**, wenn es eine erreichbare Markierung m gibt mit $m \xrightarrow{t}$.
- $t \in T$ ist **tot**, wenn





Definition 9

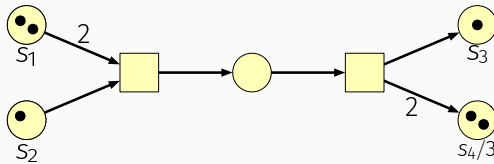
- m ist **erreichbar**, wenn es eine Schaltfolge w gibt mit $m_0 \xrightarrow{w} m$. Die Menge aller erreichbaren Markierungen eines Netzes N wird mit $R(N)$ bezeichnet.
- $t \in T$ ist **aktivierbar**, wenn es eine erreichbare Markierung m gibt mit $m \xrightarrow{t}$.
- $t \in T$ ist **tot**, wenn t nicht (mehr) aktivierbar ist.





Definition 9

- m ist **erreichbar**, wenn es eine Schaltfolge w gibt mit $m_0 \xrightarrow{w} m$. Die Menge aller erreichbaren Markierungen eines Netzes N wird mit $R(N)$ bezeichnet.
- $t \in T$ ist **aktivierbar**, wenn es eine erreichbare Markierung m gibt mit $m \xrightarrow{t}$.
- $t \in T$ ist **tot**, wenn t nicht (mehr) aktivierbar ist.
- $t \in T$ ist **lebendig**, wenn

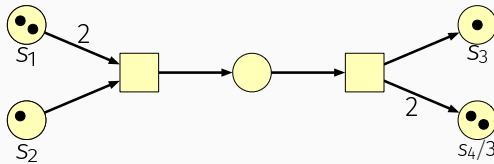




Eigenschaften von Markierungen & Transitionen

Definition 9

- m ist **erreichbar**, wenn es eine Schaltfolge w gibt mit $m_0 \xrightarrow{w} m$. Die Menge aller erreichbaren Markierungen eines Netzes N wird mit $R(N)$ bezeichnet.
- $t \in T$ ist **aktivierbar**, wenn es eine erreichbare Markierung m gibt mit $m \xrightarrow{t}$.
- $t \in T$ ist **tot**, wenn t nicht (mehr) aktivierbar ist.
- $t \in T$ ist **lebendig**, wenn t immer aktivierbar ist. D. h. $\forall$ erreichbare Markierung $m \exists$ Schaltfolge w mit $m \xrightarrow{wt}$.





Definition 10

- Ein Netz ist lebendig, wenn



Definition 10

- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.



Definition 10

- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn



Definition 10

- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.



Definition 10

- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.



Definition 10

- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**,



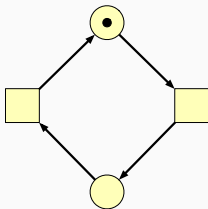
Definition 10

- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.



Definition 10

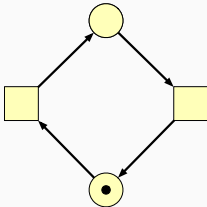
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

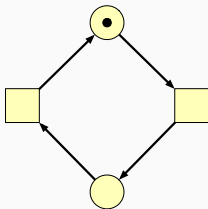
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

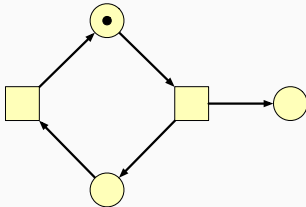
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

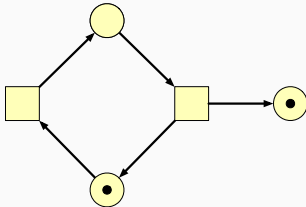
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

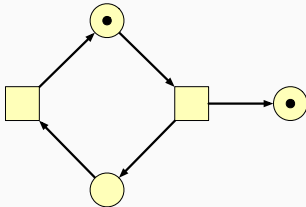
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

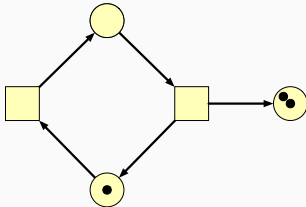
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

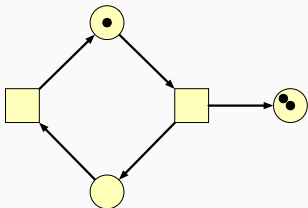
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

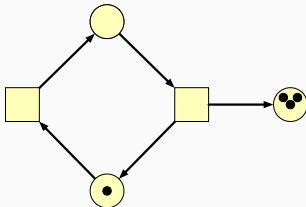
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

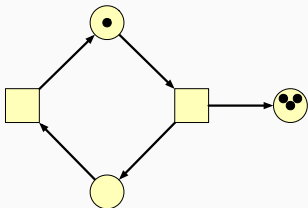
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

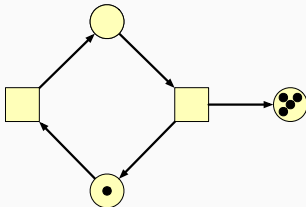
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

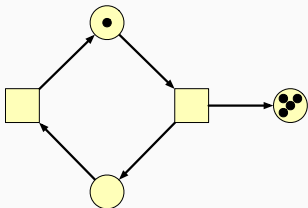
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

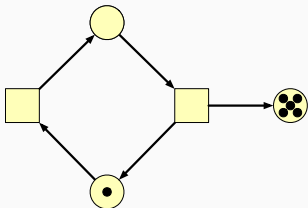
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

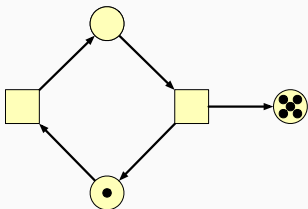
- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.





Definition 10

- Ein Netz ist **lebendig**, wenn alle Transitionen lebendig sind.
- Ein Netz ist **beschränkt**, wenn zu jedem Platz $p \in P$ eine natürliche Zahl n_p existiert, so dass in jeder erreichbaren Markierung nie mehr als n_p Marken auf p liegen.
 - Ein Netz ist **k -beschränkt** oder **k -sicher**, wenn $\forall p \in P: n_p = k$.
- Ein Netz ist **rücksetzbar**, wenn m_0 aus jeder erreichbaren Markierung heraus wieder erreichbar ist.



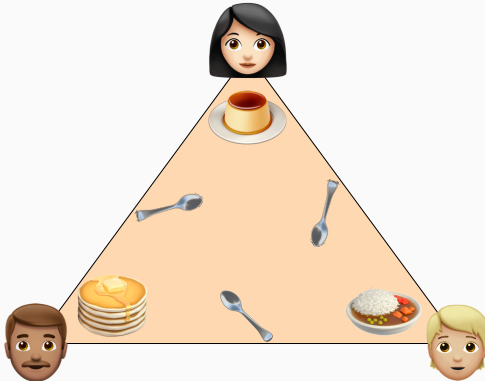
Übung

Sind diese Begriffe orthogonal?

5) 🍴 Let's go Dining!

Szenario: Dining Philosophers

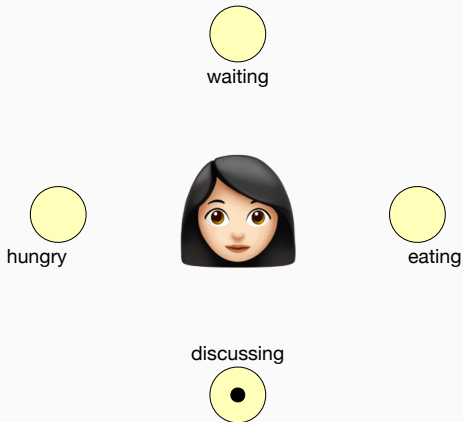
- es sitzen drei Philosoph*inn*en an einem Tisch
- zwischen je zwei Sitzplätzen liegt ein Löffel
- die drei Diskutierenden werden von Zeit zu Zeit hungrig
- zum Essen benötigt eine Person zwei Löffel
 - müssen **nacheinander** genommen werden
- nach dem Essen werden beide Löffel zurück gelegt



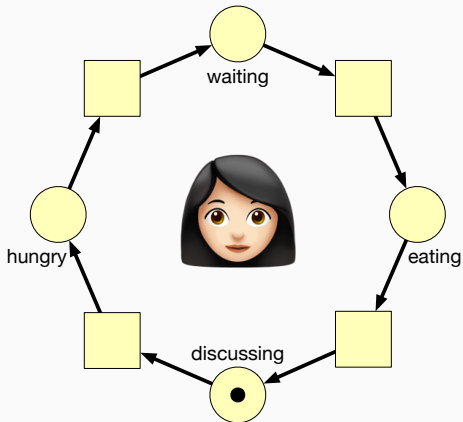
Wie modelliert man eine Philosophin?



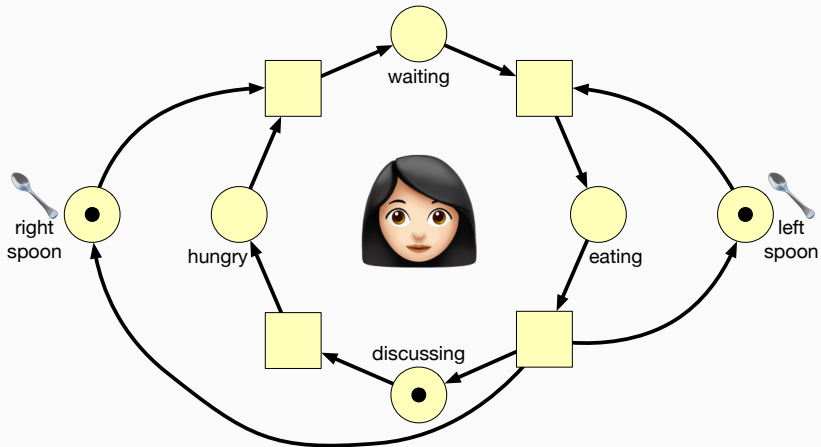
Wie modelliert man eine Philosophin?



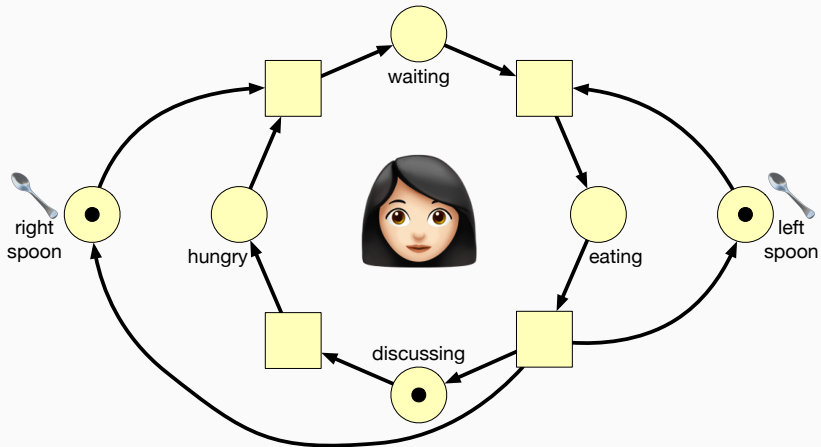
Wie modelliert man eine Philosophin?



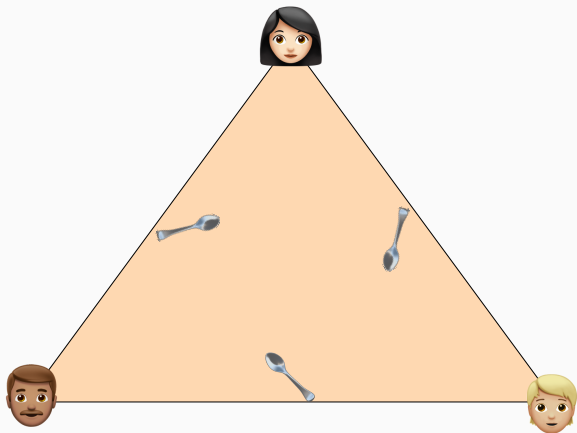
Wie modelliert man eine Philosophin?

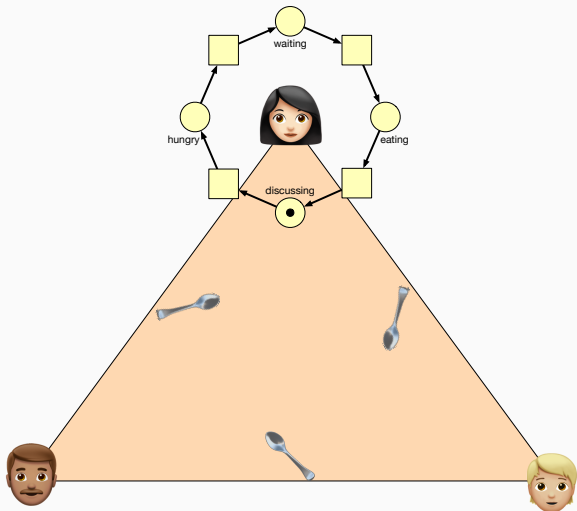


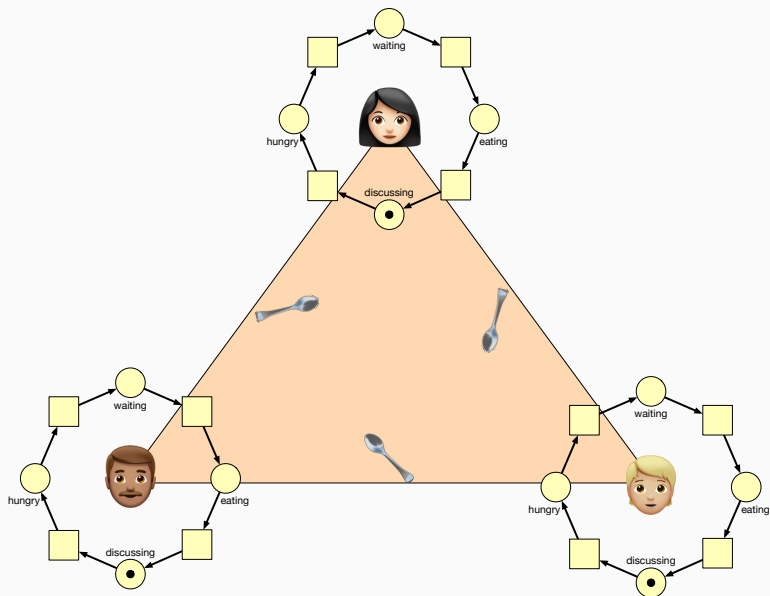
Wie modelliert man eine Philosophin?

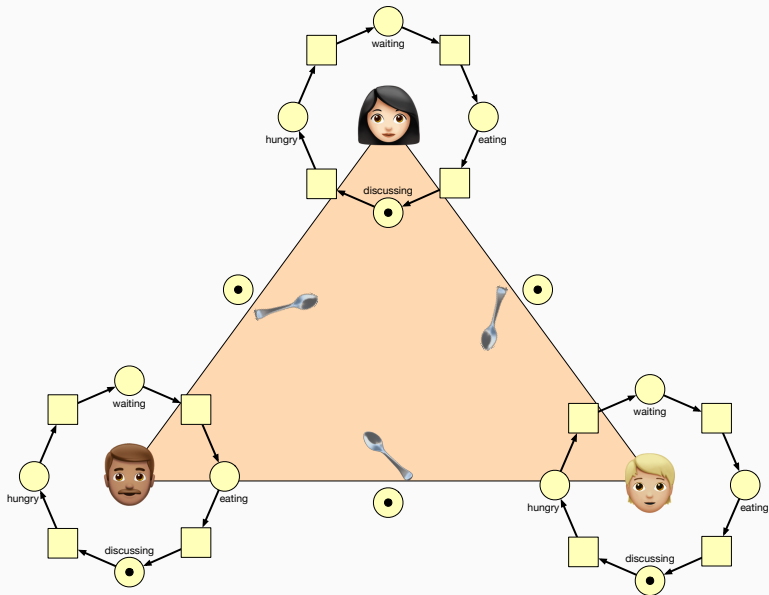


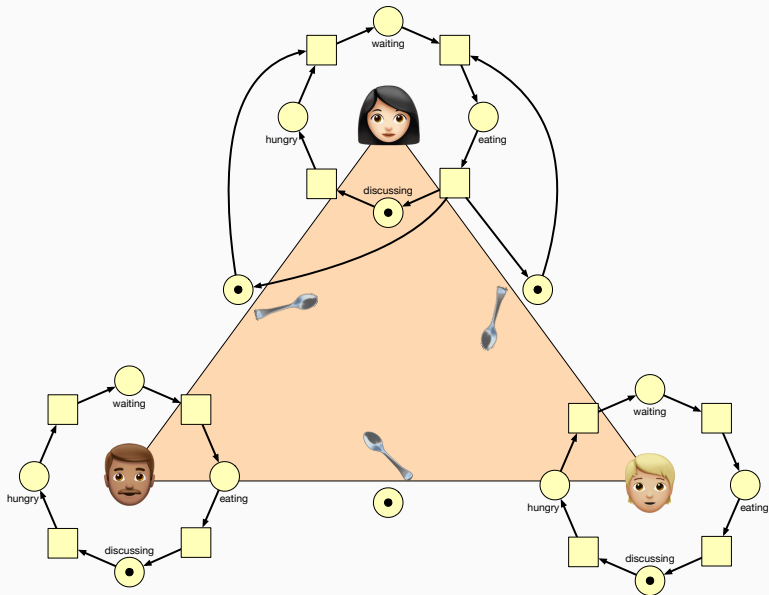
Und wie sieht nun das ganze
Szenario aus?

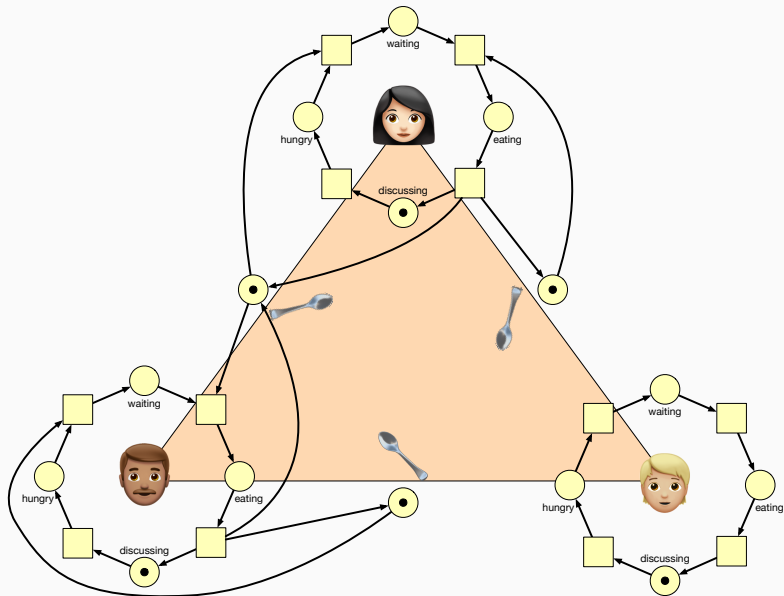


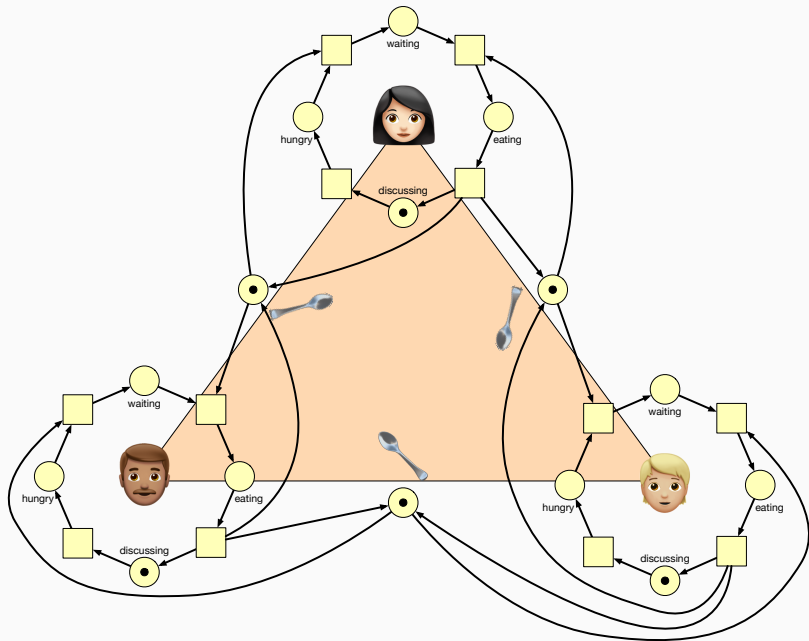


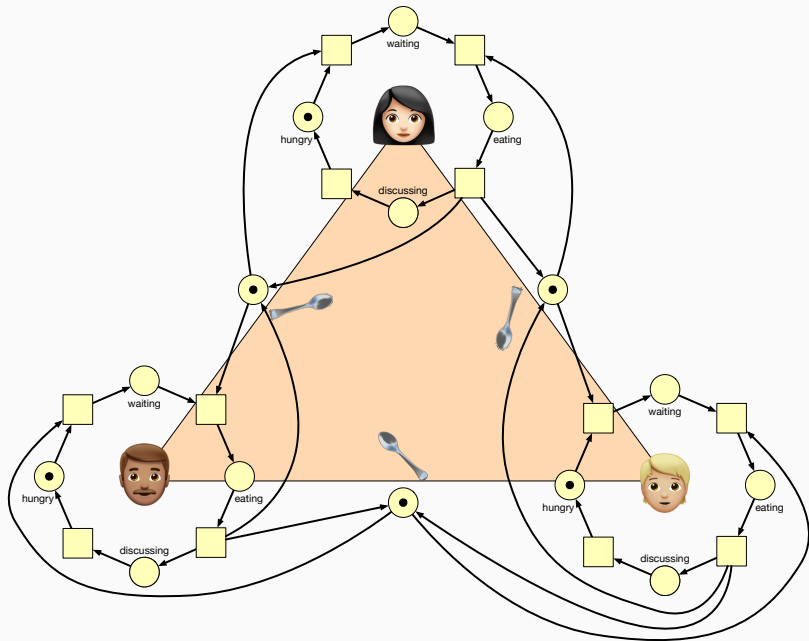


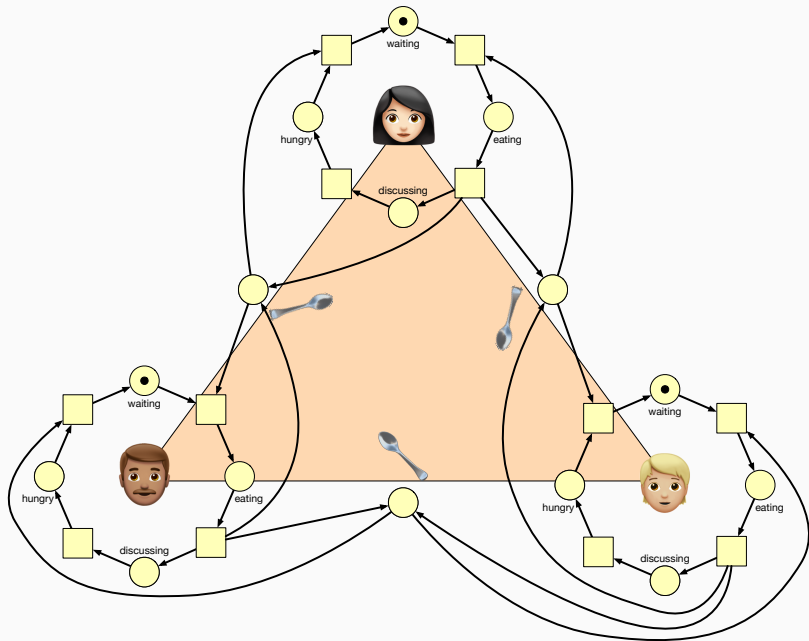


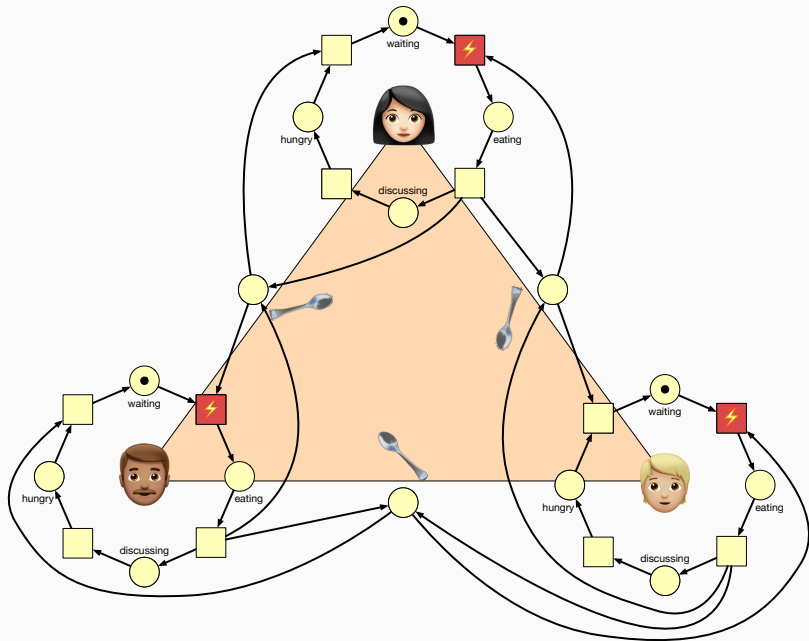


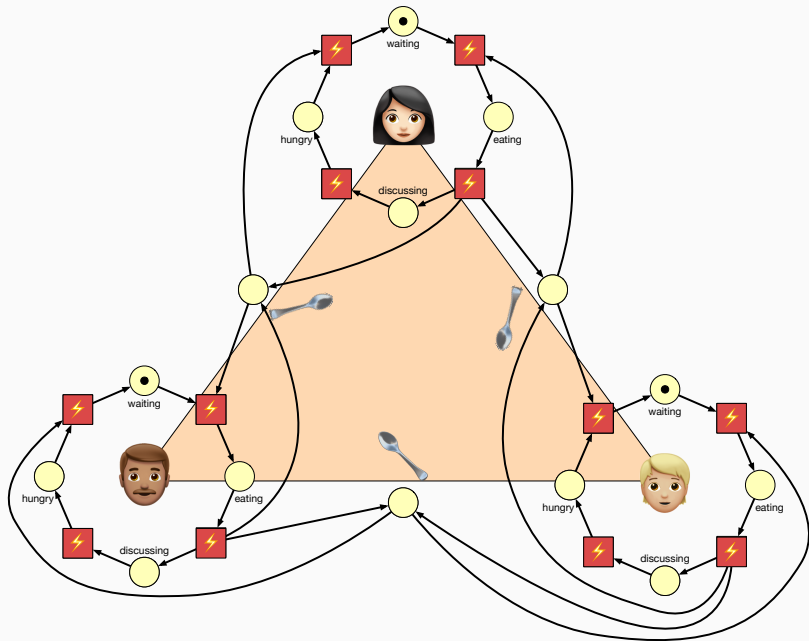


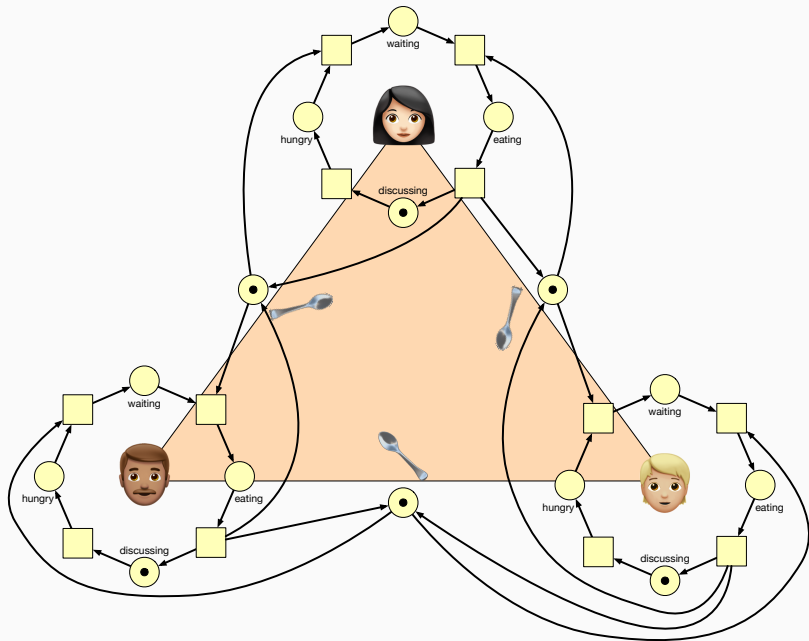


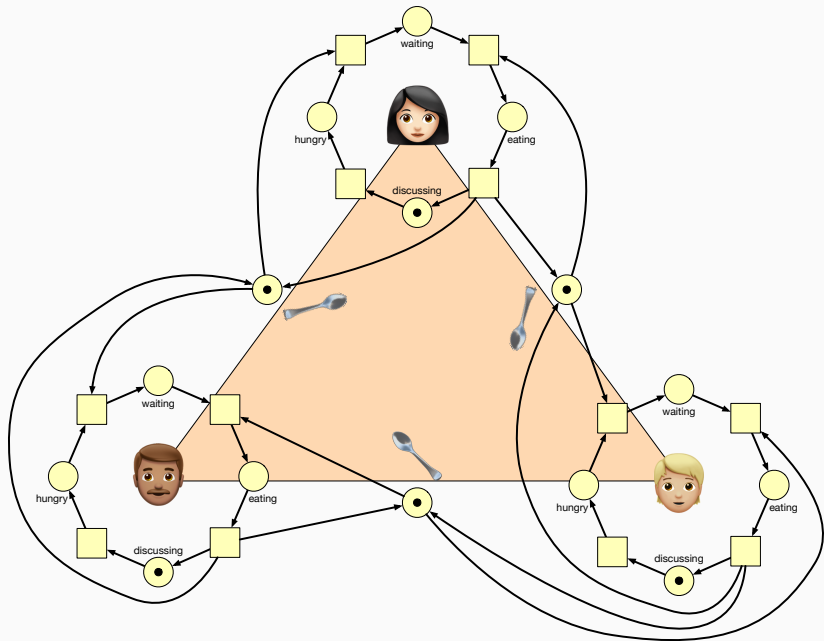












6) 📜 Zusammenfassung & 🔭 Ausblick



Petrinetze

- eine Modellierungstechnik
- Visualisierung + mathematische Formalisierung

von
vielen!



Petrinetze

- eine Modellierungstechnik
- Visualisierung + mathematische Formalisierung
- erlaubt **präzise Formulierung** von Eigenschaften...
- ...und deren **mathematischer Nachweis**

von
vielen!

oder
Widerle-
gung!



Petrinetze

- eine Modellierungstechnik
- Visualisierung + mathematische Formalisierung
- erlaubt **präzise Formulierung** von Eigenschaften...
- ...und deren **mathematischer Nachweis**

von
vielen!

oder
Widerlegung!

Mehr zu Petrinetzen

- z. B. in Reisig (2010): „Petrinetze“
- am FB entwickeltes Werkzeug: RENEW



Petrinetze

- eine Modellierungstechnik
- Visualisierung + mathematische Formalisierung
- erlaubt **präzise Formulierung** von Eigenschaften...
- ...und deren **mathematischer Nachweis**

von
vielen!

oder
Widerle-
gung!

Mehr zu Petrinetzen

- z. B. in Reisig (2010): „Petrinetze“
- am FB entwickeltes Werkzeug: RENEW



Takeaway

Informatiker $\neq$ Programmierer, HW-Designer, PC-Doktor, Word-Experte, ...



Petrinetze

- eine Modellierungstechnik
- Visualisierung + mathematische Formalisierung
- erlaubt **präzise Formulierung** von Eigenschaften...
- ...und deren **mathematischer Nachweis**

von
vielen!

oder
Widerle-
gung!

Mehr zu Petrinetzen

- z. B. in Reisig (2010): „Petrinetze“
- am FB entwickeltes Werkzeug: RENEW



Takeaway

Informatiker $\neq$ Programmierer, HW-Designer, PC-Doktor, Word-Experte, ...

Informatiker = **Problemlösekünstler!**



Wir sehen uns wieder...



- Grundlagenvorlesungen, Seminare, Abschlussarbeiten, ...
- ...und vielleicht auch irgendwann im Master Informatik!



Wir sehen uns wieder...



- Grundlagenvorlesungen, Seminare, Abschlussarbeiten, ...
- ...und vielleicht auch irgendwann im Master Informatik!

Zum Schluss ein paar Tipps

- seid engagiert und aktiv



Wir sehen uns wieder...



- Grundlagenvorlesungen, Seminare, Abschlussarbeiten, ...
- ...und vielleicht auch irgendwann im Master Informatik!



Zum Schluss ein paar Tipps

- seid engagiert und aktiv
- ein Studium ist ein Vollzeitjob
 - ⇒ Behandelt es so!



Wir sehen uns wieder...



- Grundlagenvorlesungen, Seminare, Abschlussarbeiten, ...
- ...und vielleicht auch irgendwann im Master Informatik!



Zum Schluss ein paar Tipps

- seid engagiert und aktiv
- ein Studium ist ein Vollzeitjob
 - ⇒ Behandelt es so!
- „It's dangerous to go alone!“
 - ⇒ Findet Lerngruppen!





Wir sehen uns wieder...



- Grundlagenvorlesungen, Seminare, Abschlussarbeiten, ...
- ...und vielleicht auch irgendwann im Master Informatik!



Zum Schluss ein paar Tipps

- seid engagiert und aktiv
- ein Studium ist ein Vollzeitjob
 - ⇒ Behandelt es so!
- „It's dangerous to go alone!“
 - ⇒ Findet Lerngruppen!



Aber vor allem:
Habt Spaß!

- [1] Wolfgang Reisig. ***Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien***. 1. Auflage. Leitfäden der Informatik. Wiesbaden: Vieweg+Teubner, 2010. ISBN: 9783834812902. (Besucht am 11.10.2023).