3.0 VU Formale Modellierung

Gernot Salzer

Forschungsbereich Theory and Logic
Institut für Logic and Computation

14.5.2019

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
   - 7.1. Motivation
   - 7.2. Terme
   - 7.3. Syntax
   - 7.4. Semantik
   - 7.5. Modellierung

Prädikatenlogik – Syntax

$\mathcal{V}$ ... Individuenvariablensymbole
$\mathcal{F}$, $\mathcal{P}$ ... Funktions- bzw. Prädikatensymbole mit Stelligkeiten
$(\mathcal{F}, \mathcal{P})$ ... „Signatur“

Syntax prädikatenlogischer Formeln

Die Menge $\mathcal{PF}$ der prädikatenlogischen Formeln über der Signatur $(\mathcal{F}, \mathcal{P})$ ist die kleinste Menge, für die gilt:

(p1) $P \in \mathcal{PF}$, wenn $P/0 \in \mathcal{P}$;

(p2) $P(t_1, \ldots, t_n) \in \mathcal{PF}$, wenn $P/n \in \mathcal{P}$ und $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$;

(p3) $\top, \bot \in \mathcal{PF}$;

(p4) $\neg F \in \mathcal{PF}$, wenn $F \in \mathcal{PF}$;

(p5) $(F * G) \in \mathcal{PF}$, wenn $F, G \in \mathcal{PF}$ und $* \in \{\wedge, \uparrow, \vee, \downarrow, \equiv, \not\equiv, \supset, \subset\}$.

(p6) $\forall x\, F \in \mathcal{PF}$, wenn $x \in \mathcal{V}$ und $F \in \mathcal{PF}$.

(p7) $\exists x\, F \in \mathcal{PF}$, wenn $x \in \mathcal{V}$ und $F \in \mathcal{PF}$.

P, $P(t_1, \ldots, t_n)$ ... „Atomformeln“

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
   - 7.1. Motivation
   - 7.2. Terme
   - 7.3. Syntax
   - 7.4. Semantik
   - 7.5. Modellierung

Prädikatenlogik – Semantik

Prädikatensymbole: repräsentieren Relationen bzw. Elementaraussagen

- Vordefinierte Symbole wie $\top$ und $\bot$: feste, unveränderliche Bedeutung (fix eingebaut in Formelsemantik)
- Freie Symbole: Interpretation durch Relationen bzw. Wahrheitswerte

(Prädikatenlogische) Interpretation

Eine Interpretation I über einem Wertebereich $\mathcal{U}$ ordnet jedem Symbol der Signatur $(\mathcal{F}, \mathcal{P})$ eine Funktion bzw. Relation wie folgt zu:

$I(f) \in \mathcal{U}$	für $f/0 \in \mathcal{F}$
$I(f) : \mathcal{U}^n \mapsto \mathcal{U}$	für $f/n \in \mathcal{F}$, $n > 0$
$I(P) \in \{0, 1\}$	für $P/0 \in \mathcal{P}$
$I(P) \subseteq \mathcal{U}^n$	für $P/n \in \mathcal{P}$, $n > 0$
$I(P) : \mathcal{U}^n \mapsto \{0, 1\}$	(alternative Sichtweise)

Semantik prädikatenlogischer Formeln

Der Wert einer Formel in einer Interpretation I mit Variablenbelegung σ wird festgelegt durch die Funktion $\mathrm{val}_{I,\sigma}$, definiert als:

(v1) $\mathrm{val}_{I,\sigma}(P) = I(P)$ für $P/0 \in \mathcal{P}$;

(v2) $\mathrm{val}_{I,\sigma}(P(t_1, \ldots, t_n)) = I(P)(\mathrm{val}_{I,\sigma}(t_1), \ldots, \mathrm{val}_{I,\sigma}(t_n))$ für $P/n \in \mathcal{P}$;

(v3) $\mathrm{val}_{I,\sigma}(\top) = 1$ und $\mathrm{val}_{I,\sigma}(\bot) = 0$;

(v4) $\mathrm{val}_{I,\sigma}(\neg F) = \mathrm{not}\,\mathrm{val}_{I,\sigma}(F)$;

(v5) $\mathrm{val}_{I,\sigma}(\,(F * G)\,) = \mathrm{val}_{I,\sigma}(F) \circledast \mathrm{val}_{I,\sigma}(G)$,
wobei $\circledast$ die logische Funktion zum Operator $*$ ist;

(v6) $$\mathrm{val}_{I,\sigma}(\,\forall x\, F\,) = \begin{cases} 1 & \text{falls } \mathrm{val}_{I,\sigma'}(F) = 1 \text{ für alle } \sigma' \overset{x}{\sim} \sigma \\ 0 & \text{sonst} \end{cases}$$

(v7) $$\mathrm{val}_{I,\sigma}(\,\exists x\, F\,) = \begin{cases} 1 & \text{falls } \mathrm{val}_{I,\sigma'}(F) = 1 \text{ für mind. ein } \sigma' \overset{x}{\sim} \sigma \\ 0 & \text{sonst} \end{cases}$$

$\sigma \overset{x}{\sim} \sigma' \ldots\ \sigma(v) = \sigma'(v)$ für alle $v \in \mathcal{V}$ mit $v \neq x$

(σ und σ' sind identisch, nur $\sigma(x)$ und $\sigma'(x)$ können verschieden sein.)

Eine Formel F heißt

- erfüllbar, wenn $\mathrm{val}_{I,\sigma}(F) = 1$ für mindestens ein I und ein σ (I und σ nennt man Modell von F);
- widerlegbar, wenn $\mathrm{val}_{I,\sigma}(F) = 0$ für mindestens ein I und ein σ (I und σ nennt man Gegenbeispiel oder Gegenmodell zu F);
- unerfüllbar, wenn $\mathrm{val}_{I,\sigma}(F) = 0$ für alle I und alle σ;
- (allgemein)gültig, wenn $\mathrm{val}_{I,\sigma}(F) = 1$ für alle I und alle σ (F nennt man in diesem Fall auch Tautologie).

Äquivalenz, Konsequenz und Gültigkeit

Semantische Äquivalenz

Zwei Formeln F und G heißen äquivalent, geschrieben $F = G$,
wenn $\mathrm{val}_{I,\sigma}(F) = \mathrm{val}_{I,\sigma}(G)$ für alle I und alle σ gilt.

$F_1, \ldots, F_n \models_{I,\sigma} G$:
„Aus $\mathrm{val}_{I,\sigma}(F_1) = \cdots = \mathrm{val}_{I,\sigma}(F_n) = 1$ folgt $\mathrm{val}_{I,\sigma}(G) = 1$.“

Logische Konsequenz

$F_1, \ldots, F_n \models G$: $F_1, \ldots, F_n \models_{I,\sigma} G$ gilt für alle I und σ.

Die Formeln F und G sind äquivalent ($F = G$) genau dann,
wenn $F \equiv G$ eine gültige Formel ist.

$F_1, \ldots, F_n \models G$ genau dann, wenn $(F_1 \wedge \cdots \wedge F_n) \supset G$ gültig.

Wichtige Äquivalenzen

Neben den aussagenlogischen Äquivalenzen gelten auch noch folgende:

$\neg\forall x\, F = \exists x\, \neg F$ $\neg\exists x\, F = \forall x\, \neg F$	Dualität von $\forall$ und $\exists$
$\forall x\, F[x] = \forall y\, F[y]$ $\exists x\, F[x] = \exists y\, F[y]$	Umbenennung gebundener Variablen (sofern y nicht bereits in F vorkommt)
$\forall x\, \forall y\, F = \forall y\, \forall x\, F$ $\exists x\, \exists y\, F = \exists y\, \exists x\, F$	Vertauschung gleichartiger Quantoren
$\forall x\, (F \wedge G) = \forall x\, F \wedge \forall x\, G$	Distributivität $\forall/\wedge$
$\exists x\, (F \vee G) = \exists x\, F \vee \exists x\, G$	Distributivität $\exists/\vee$

Falls x nicht frei in F vorkommt, gilt außerdem:

$\forall x\, F = F \qquad \exists x\, F = F$	
$\forall x\, (F \vee G) = F \vee \forall x\, G$	Distributivität $\forall/\vee$
$\exists x\, (F \wedge G) = F \wedge \exists x\, G$	Distributivität $\exists/\wedge$

Was Sie letztes Mal hörten

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
   - 7.1. Motivation
   - 7.2. Terme
   - 7.3. Syntax
   - 7.4. Semantik
   - 7.5. Modellierung

Wahl der Prädikate

Wähle ein eigenes Symbol für Satzteile, die öfter auftreten (können).

Max liest Zeitung.
Max_liest_Zeitung
Liest_Zeitung(*max*)
Liest(*max*, *zeitung*)

Mache Zeitwörter bzw. Eigenschaften zu Prädikatensymbole und Hauptwörter zu ihren Argumenten.

Sokrates ist sterblich.
Ist_sterblich(*sokrates*)

Fürwörter (Pronomen)

Ersetze Fürwörter durch Namen. Wiederhole Satzteile, die durch Fürwörter ersetzt wurden.
Führe Variablen für unbestimmte Fürwörter ein.
Mache Negationen explizit.

Mia liebt Max, der wiederum sie liebt.
$Liebt(mia, max) \wedge Liebt(max, mia)$

Nichts währt ewig.
$\neg\exists x\, Währt\_ewig(x)$

Ich kann nichts sehen.
$\neg\exists x\, Kann\_sehen(gernot, x)$

Vor mir ist etwas Großes, und es ist hungrig.
$\exists x\,(Befindet\_sich\_vor(x,gernot) \wedge Groß(x) \wedge Hungrig(x))$

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
   - 7.1. Motivation
   - 7.2. Terme
   - 7.3. Syntax
   - 7.4. Semantik
   - 7.5. Modellierung
8. Petri-Netze

Quantoren, Typen und Beziehungen

Jeder Student ist jünger als irgendein Professor.

Verwende $\forall$-Quantoren für „alle“/„jede“ und $\exists$-Quantoren für „es gibt“/„jemand“/„irgendein“.

$\forall x\,(\mathit{Stud}(x) \supset \exists y\,(\mathit{Prof}(y) \wedge \mathit{Jünger}(x, y)))$

Für alle Individuen x gilt:
Falls x ein Student ist, gibt es mindestens ein Individuum y,
sodass y Professor ist und x jünger als y ist.

Funktionssymbole

Jedes Kind ist jünger als seine Mutter.

„x ist ein Kind“ $\Rightarrow \mathit{Kind}(x)$
„y ist Mutter von x“ $\Rightarrow \mathit{Mutter}(y, x)$

$\forall x \, \forall y ((\mathit{Kind}(x) \wedge \mathit{Mutter}(y, x)) \supset \mathit{Jünger}(x, y))$

Diese Formalisierung berücksichtigt nicht, dass jedes Kind genau eine genetische Mutter hat.

Besser: Fasse „Mutter“ als Funktion auf, die jedem Kind seine eindeutig bestimmte Mutter zuordnet.

„Mutter von x“ $\Rightarrow \mathit{mutter}(x)$

$\forall x \, (\mathit{Kind}(x) \supset \mathit{Jünger}(x, \mathit{mutter}(x))$

Alternierende Quantoren

Vergleichen Sie:

$\forall x \exists y\, \mathit{Mutter}(y, x)$	$\exists y \forall x\, \mathit{Mutter}(y, x)$
„Jeder hat eine Mutter.“	„Jemand ist die Mutter von allen.“
y hängt vom gewählten x ab.	y ist unabhängig vom gewählten x.

Vertauschung unterschiedlicher Quantoren ändert die Bedeutung!

Quantoren und Eigenschaften

Alle vernünftigen Leute verabscheuen Gewalt.

$\forall$ vernünftige x (x verabscheut Gewalt) (Zwischenschritt, ist keine Formel!)

$\forall x\,(\mathit{Vernünftig}(x) \supset \mathit{Verabscheut}(x, \mathit{gewalt}))$

Eigenschaften $\forall$-quantifizierter Variablen werden zu Prämissen einer Implikation.

Manche vernünftige Leute verabscheuen Gewalt.
Es gibt vernünftige Leute, die Gewalt verabscheuen.

$\exists$ vernünftige x (x verabscheut Gewalt) (Zwischenschritt, ist keine Formel!)

$\exists x\,(\mathit{Vernünftig}(x) \wedge \mathit{Verabscheut}(x, \mathit{gewalt}))$

Eigenschaften $\exists$-quantifizierter Variablen werden zu Teilen einer Konjunktion.

Beispiel: Sport

$\mathcal{P} = \{\mathit{Mensch}/1, \mathit{Sportart}/1, \mathit{Anstrengend}/1, \mathit{Betreibt}/2\}$
$\mathcal{F} = \{\mathit{handball}/0, \mathit{laufen}/0\}$

Alle Menschen betreiben eine anstrengende Sportart.

Zu jedem Menschen x gibt es eine Sportart,
die anstrengend ist und von x betrieben wird.

Zu jedem Menschen x gibt es eine Sportart y,
sodass y anstrengend ist und x y betreibt.

„Zu jedem Menschen x ...“ $\forall x(\mathit{Mensch}(x) \supset \cdots)$
„Es gibt eine Sportart y ...“ $\exists y(\mathit{Sportart}(y) \wedge \cdots)$
„y ist anstrengend und x betreibt y“ $\mathit{Anstrengend}(y) \wedge \mathit{Betreibt}(x, y)$

$\forall x(\mathit{Mensch}(x) \supset \exists y(\mathit{Sportart}(y) \wedge \mathit{Anstrengend}(y) \wedge \mathit{Betreibt}(x, y)))$
$\forall x \exists y(\mathit{Mensch}(x) \supset (\mathit{Sportart}(y) \wedge \mathit{Anstrengend}(y) \wedge \mathit{Betreibt}(x, y)))$

Beispiel: Sport

$\mathcal{P} = \{\mathit{Mensch}/1, \mathit{Sportart}/1, \mathit{Anstrengend}/1, \mathit{Betreibt}/2\}$
$\mathcal{F} = \{\mathit{handball}/0, \mathit{laufen}/0\}$

Es gibt anstrengende Menschen, die Handball und Laufen betreiben.

Es gibt (mindestens) einen Menschen x,
der anstrengend ist, Handball betreibt und Laufen betreibt.

Es gibt (mindestens) einen Menschen x,
sodass x anstrengend ist, x Handball betreibt und x Laufen betreibt.

„Es gibt einen Menschen x“ $\exists x(\mathit{Mensch}(x) \wedge \cdots)$
„x ist anstrengend“ $\mathit{Anstrengend}(x)$
„x betreibt Handball“ $\mathit{Betreibt}(x, \mathit{handball})$
„x betreibt Laufen“ $\mathit{Betreibt}(x, \mathit{laufen})$

$$\exists x(\mathit{Mensch}(x) \wedge \mathit{Anstrengend}(x) \wedge \mathit{Betreibt}(x, \mathit{handball}) \wedge \mathit{Betreibt}(x, \mathit{laufen}))$$

Fooling people

„You can fool some of the people all of the time,
and all of the people some of the time,
but you cannot fool all of the people all of the time.“

(zugeschrieben Abraham Lincoln, 16. Präsident der USA, 1809–1865)

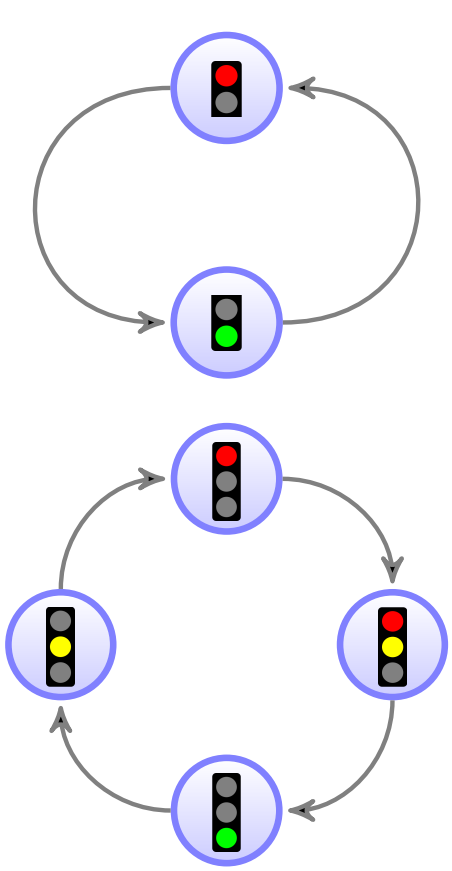
$\mathit{FoolAt}(x, y)$	... you can fool x at time y
$\mathit{Person}(x)$	... x is a person / x is one of the people
$\mathit{PointInTime}(y)$	... y is a point in time

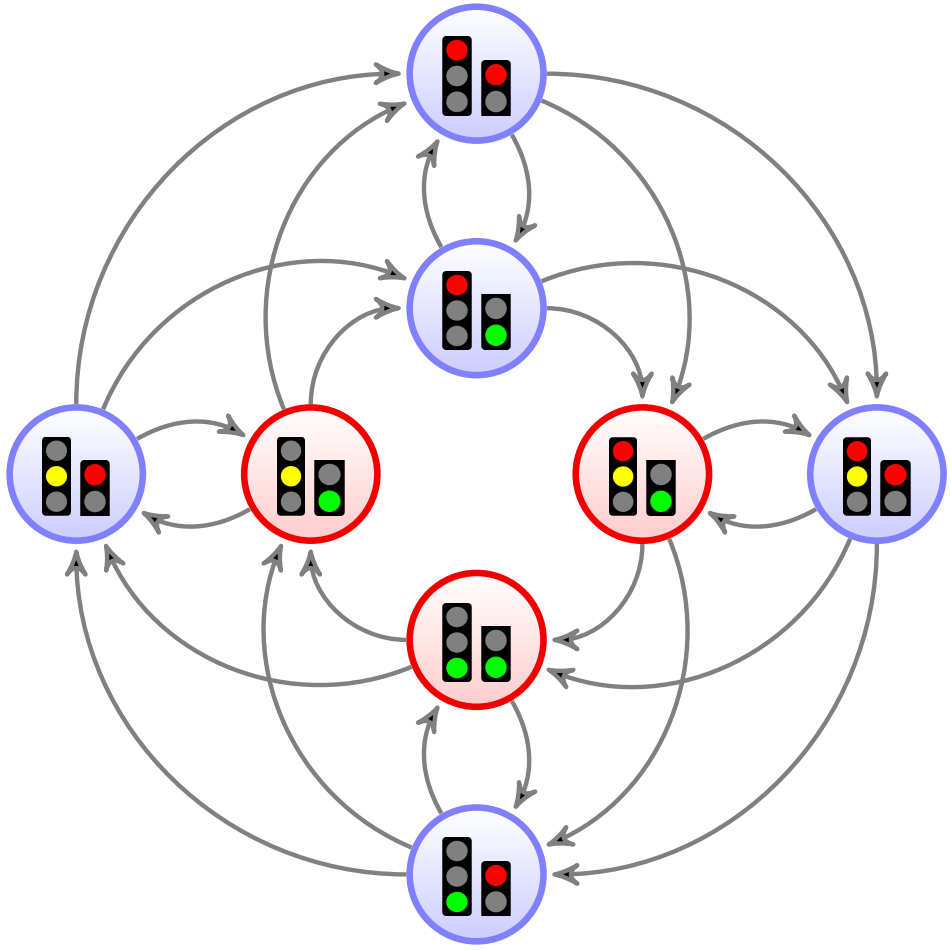
$$
\begin{aligned}
& \exists x(\mathit{Person}(x) \wedge \forall y(\mathit{PointInTime}(y) \supset \mathit{FoolAt}(x, y))) \\
\wedge \quad & \forall x(\mathit{Person}(x) \supset \exists y(\mathit{PointInTime}(y) \wedge \mathit{FoolAt}(x, y))) \\
\wedge \neg & \forall x(\mathit{Person}(x) \supset \forall y(\mathit{PointInTime}(y) \supset \mathit{FoolAt}(x, y)))
\end{aligned}
$$

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
8. Petri-Netze
   - 8.1. Motivation
   - 8.2. Definitionen
   - 8.3. Modellierung

Fußgängerkreuzung als endlicher Automat

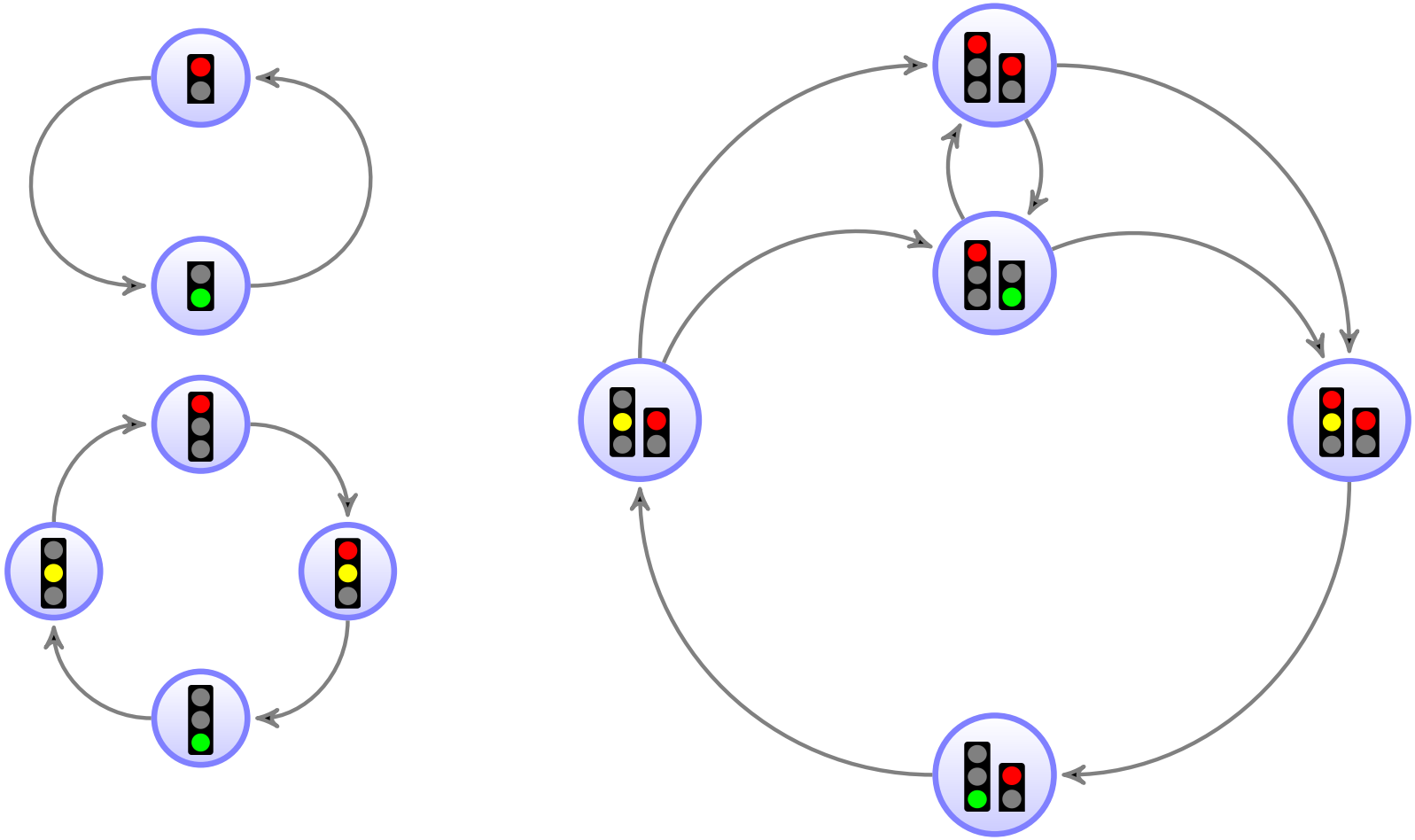




Modellierung des Gesamtsystems durch einen Produktautomaten:

- Bilde alle Kombinationszustände.
- Übergange dort, wo die ursprünglichen Automaten welche hatten.

Fußgängerkreuzung als endlicher Automat



Modellierung des Gesamtsystems durch einen Produktautomaten:

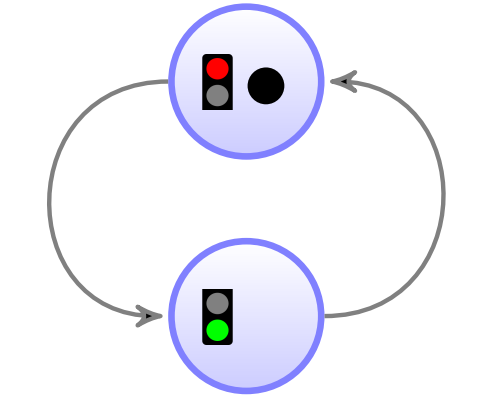
- Bilde alle Kombinationszustände.
- Übergange dort, wo die ursprünglichen Automaten welche hatten.
- Elimination unerwünschter Zustände und Übergänge.

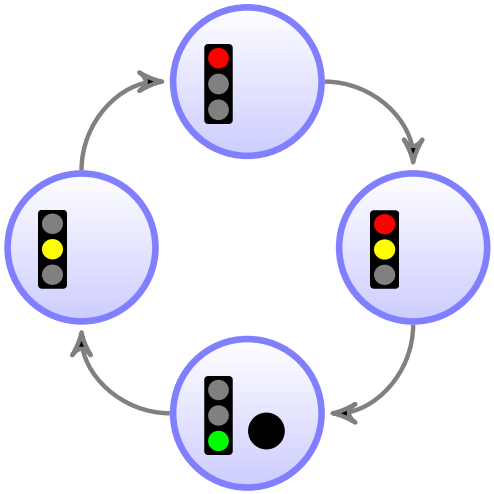
Endliche Automaten ungeeignet für Modellierung verteilter Systeme:

- Anzahl der Kombinationszustände steigt exponentiell mit der Zahl der Komponenten.
- Anzahl der Übergänge steigt exponentiell.
- Es ist schwierig, Änderungen einer Komponente in den Gesamtautomaten zu übertragen.
- Endliche Automaten mit einem aktiven Zustand entsprechen nicht der Idee verteilter Systeme mit vielen unabhängigen Abläufen nebeneinander, die nur bei Bedarf synchronisiert werden.

Petri-Netz = Automat mit mehreren aktiven Stellen + Synchronisation

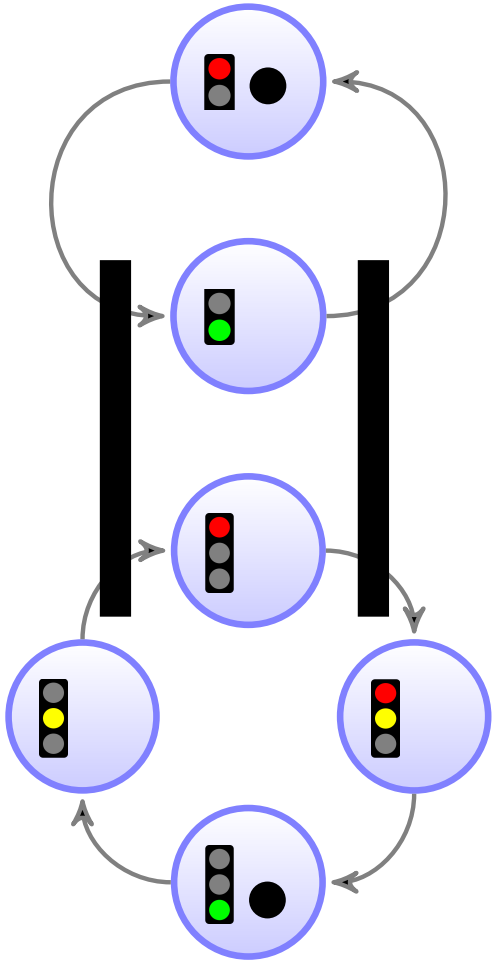
Fußgängerkreuzung als Petri-Netz





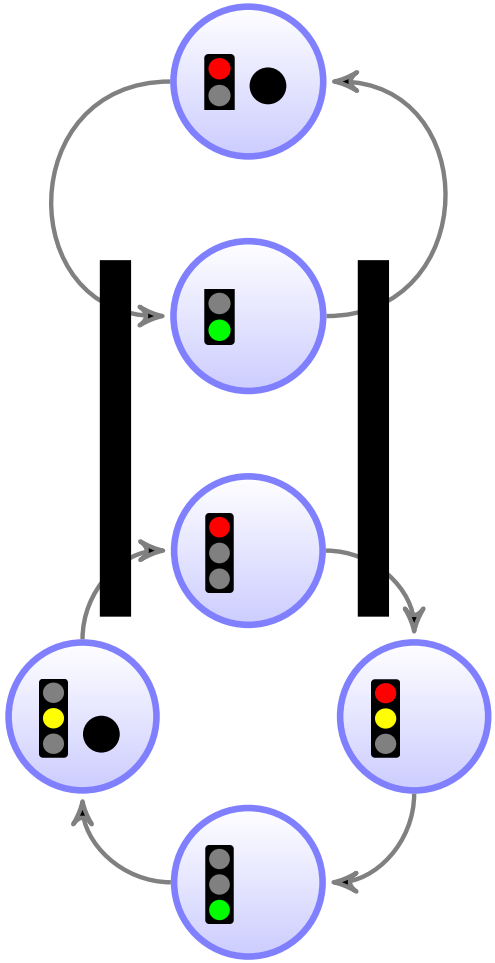
- Marken zeigen die aktiven Stellen an.

Fußgängerkreuzung als Petri-Netz



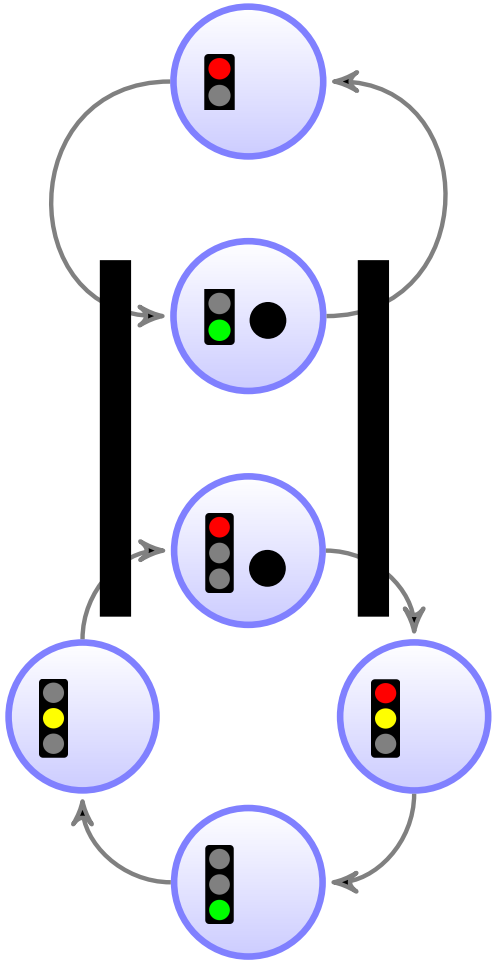
- Marken zeigen die aktiven Stellen an.
- Transitionen werden nur durchlässig („feuern“), wenn alle Eingangszustände Marken besitzen.
  Marken dürfen nur gleichzeitig weiterwandern.

Fußgängerkreuzung als Petri-Netz



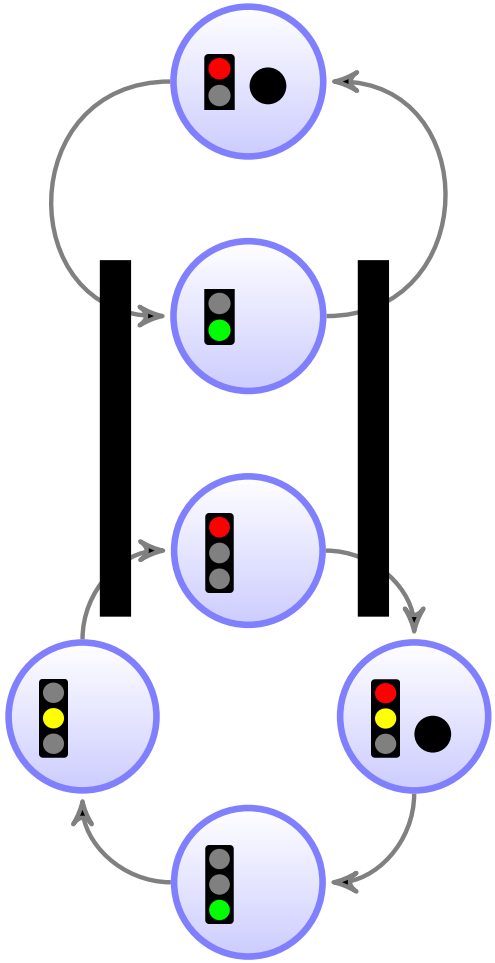
- Marken zeigen die aktiven Stellen an.
- Transitionen werden nur durchlässig („feuern“), wenn alle Eingangszustände Marken besitzen.
  Marken dürfen nur gleichzeitig weiterwandern.

Fußgängerkreuzung als Petri-Netz



- Marken zeigen die aktiven Stellen an.
- Transitionen werden nur durchlässig („feuern“), wenn alle Eingangszustände Marken besitzen.
  Marken dürfen nur gleichzeitig weiterwandern.

Fußgängerkreuzung als Petri-Netz



- Marken zeigen die aktiven Stellen an.
- Transitionen werden nur durchlässig („feuern“), wenn alle Eingangszustände Marken besitzen.
  Marken dürfen nur gleichzeitig weiterwandern.

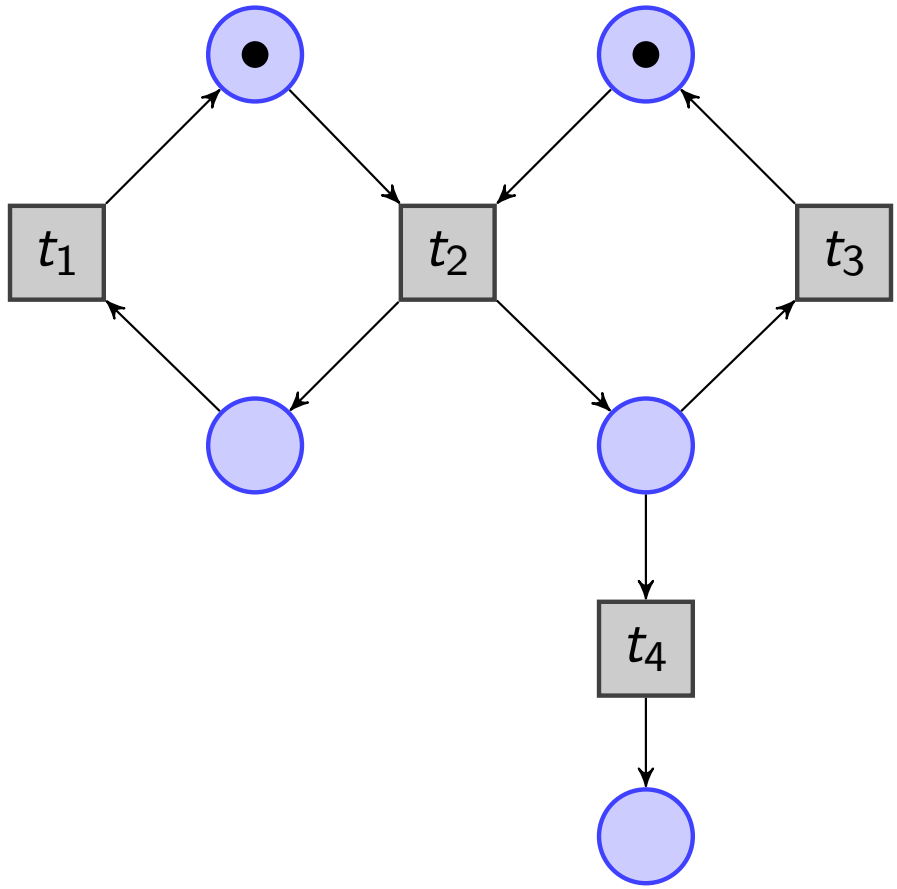
Petrinetze: Motivation

Formalismus zur Modellierung von nebenläufigen Systemen (concurrent/parallel systems).

- Bei Systemübergängen können Ressourcen konsumiert und neu erzeugt werden.
- Natürliche Modellierung der räumlichen Verteilung von Ressourcen, Nebenläufigkeit und (Zugriffs-)Konflikten.
- Intuitive graphische Darstellung.
- Weit verbreitet, z.B. Aktivitätsdiagramme (activity diagrams) in UML (Unified Modeling Language).

Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Motivation



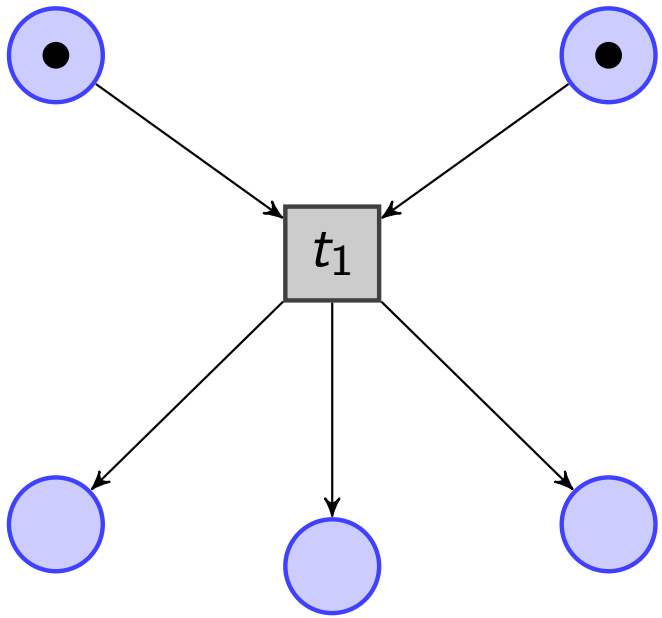
Notation:

- Stellen (dargestellt als Kreise): mögliche Plätze für Ressourcen
- Marken (dargestellt als kleine gefüllte Kreise): Ressourcen
- Transitionen (dargestellt als Rechtecke oder Balken): Systemübergänge

Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Motivation

Darstellung einer Transition:

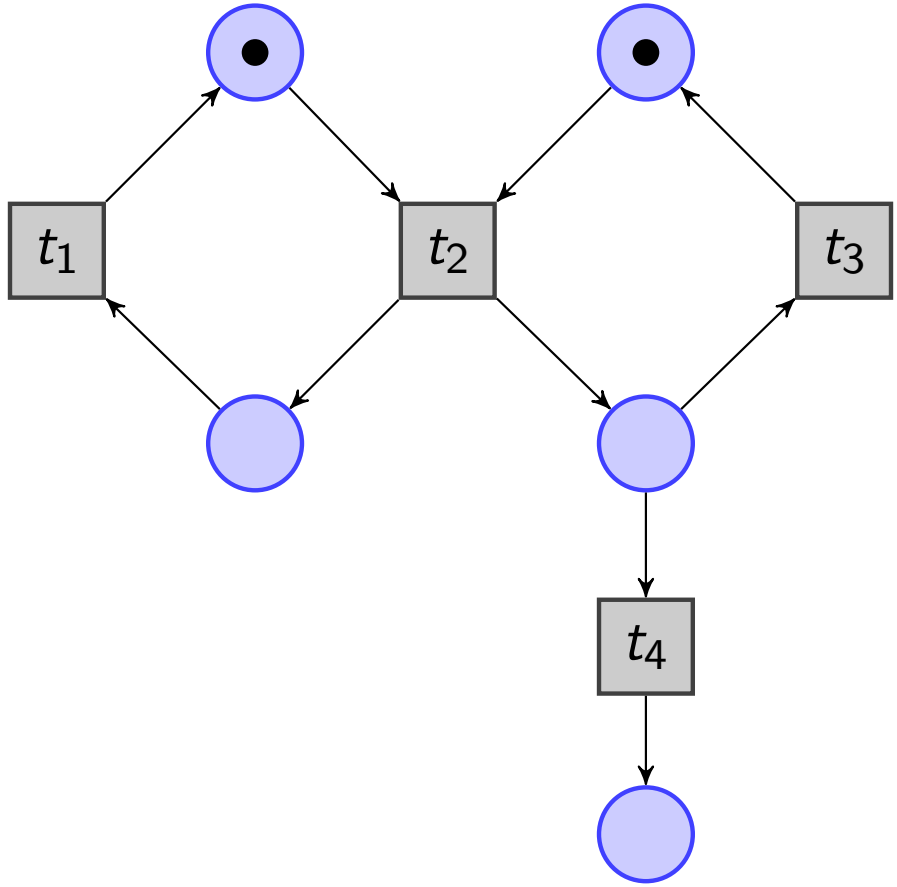


Vorbedingungen

Nachbedingungen

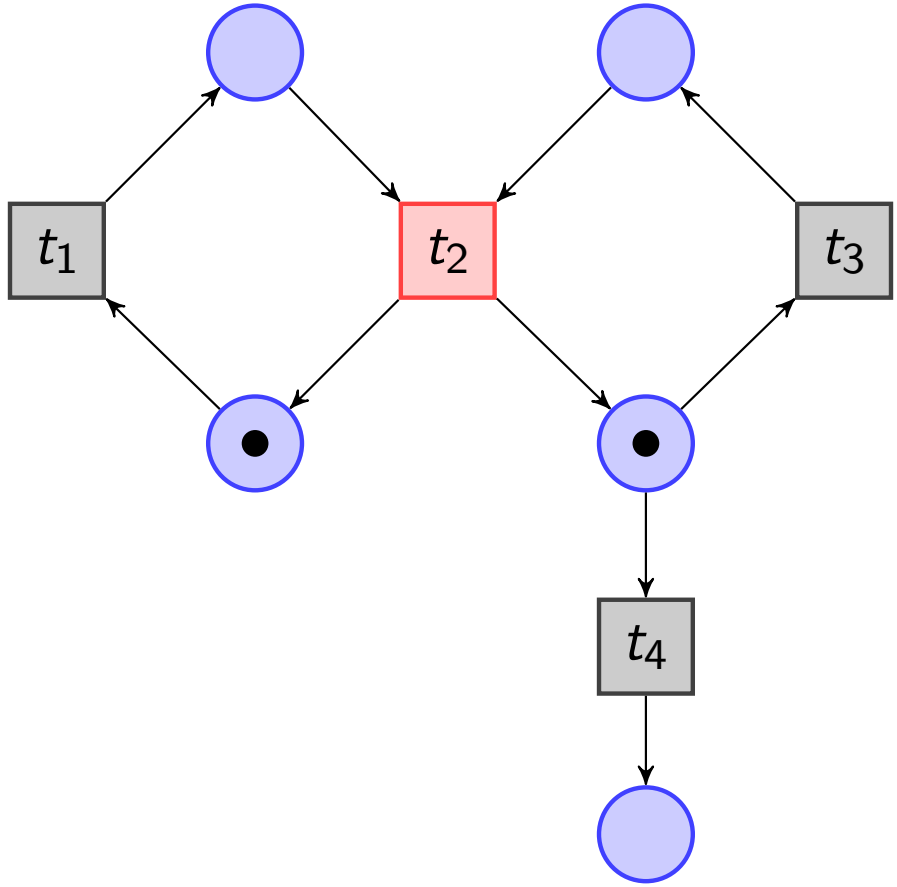
- Vorbedingungen sind die Marken, die konsumiert werden
- Nachbedingungen sind die Marken, die erzeugt werden
- Das Entfernen der Marken der Vorbedingungen und das Erzeugen der Marken der Nachbedingungen nennt man Schalten bzw. Feuern der Transition.

Petrinetze: Beispiel



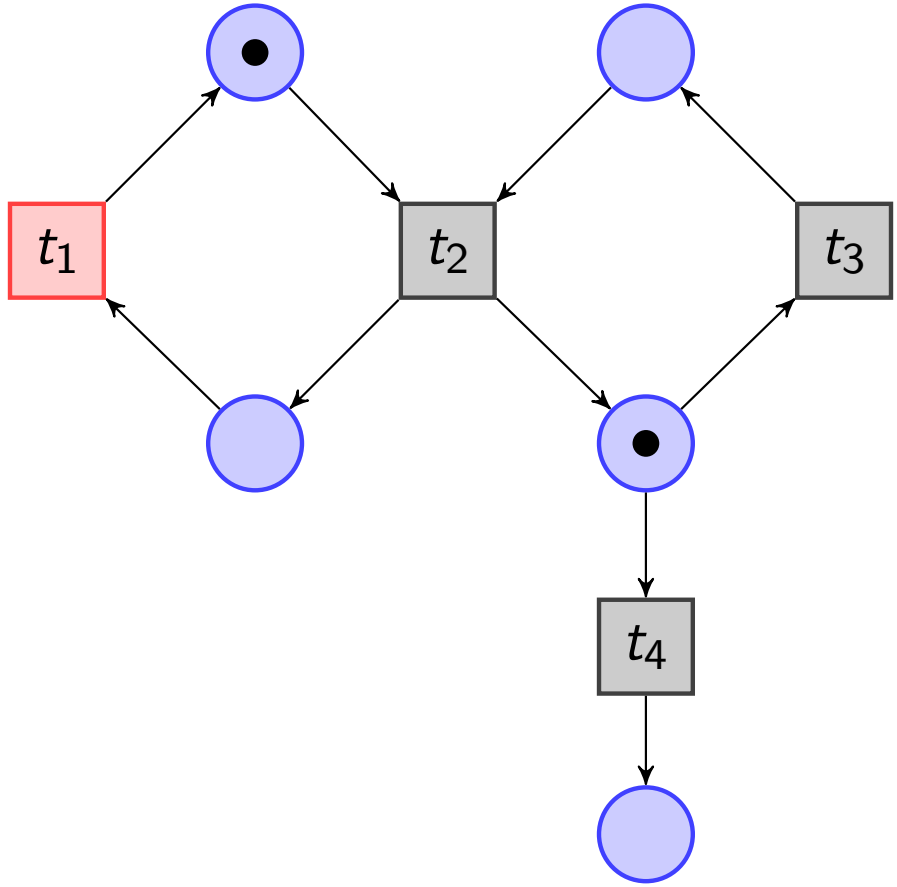
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



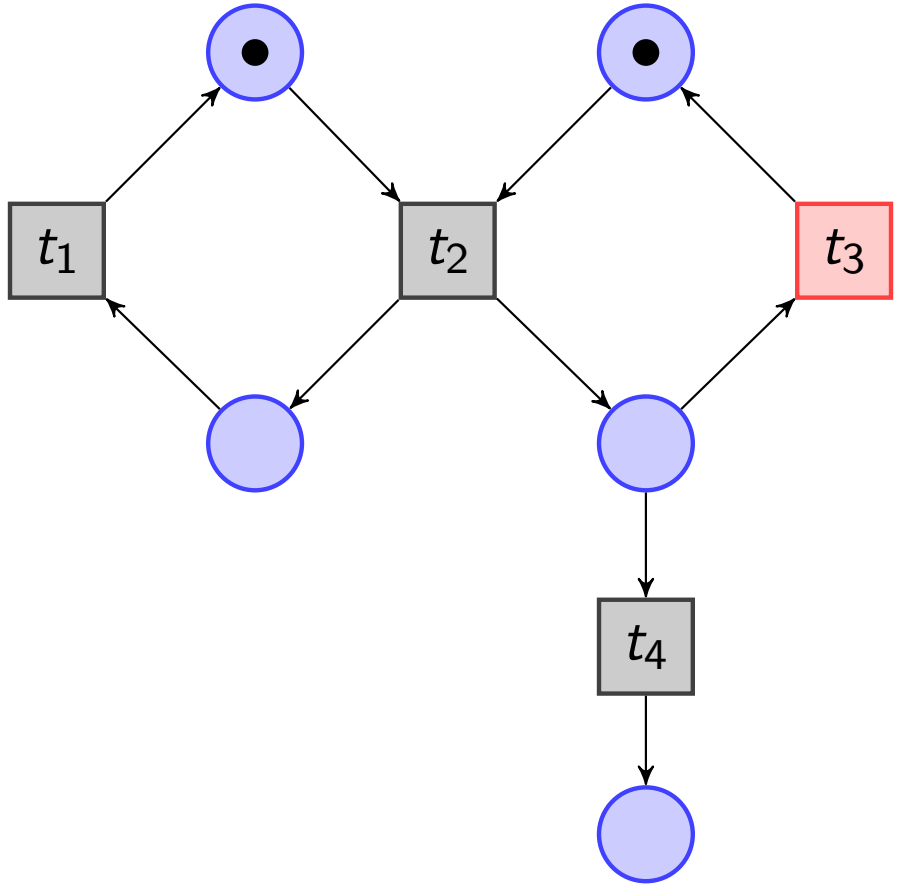
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



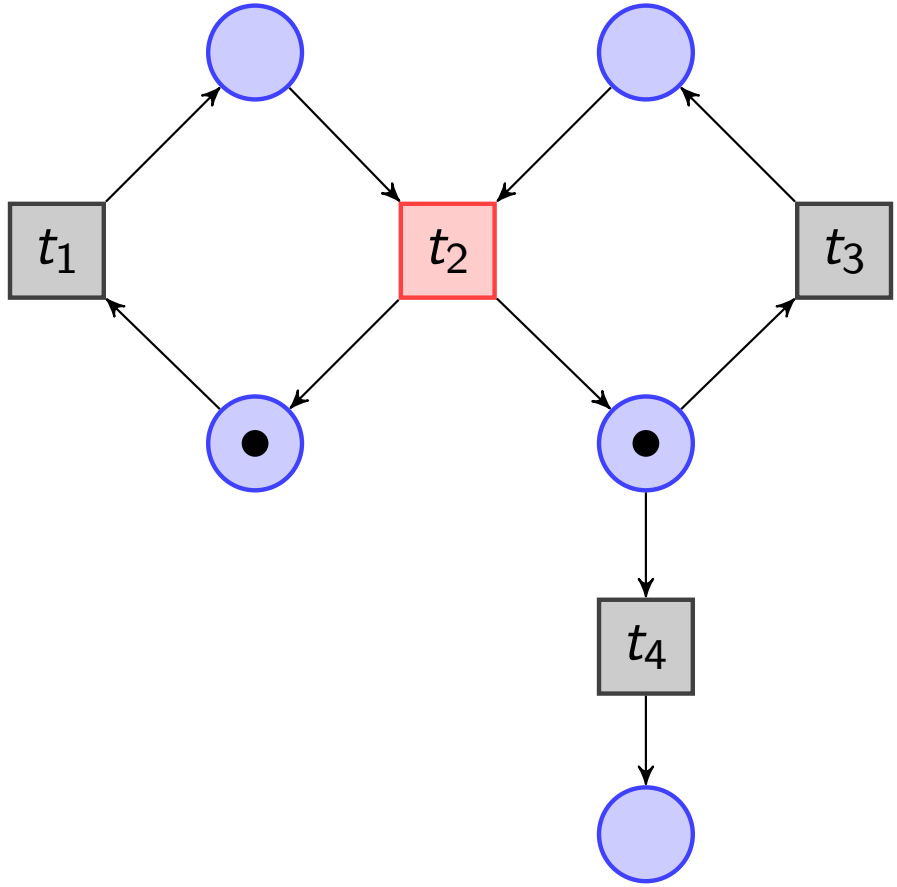
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



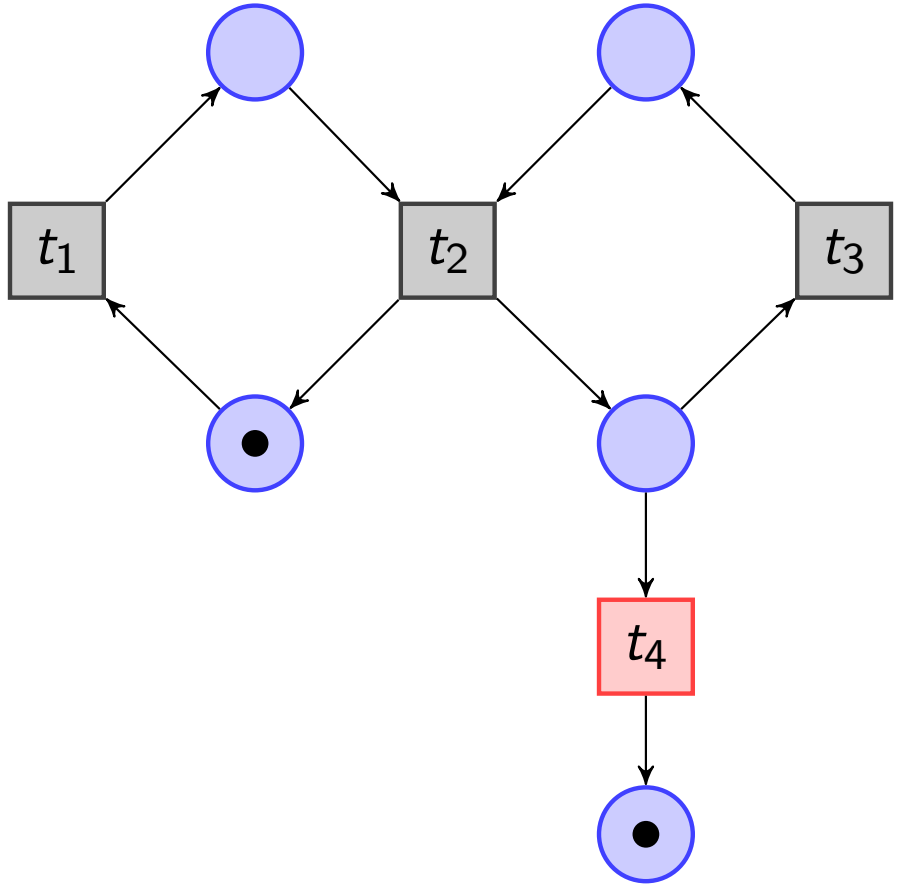
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



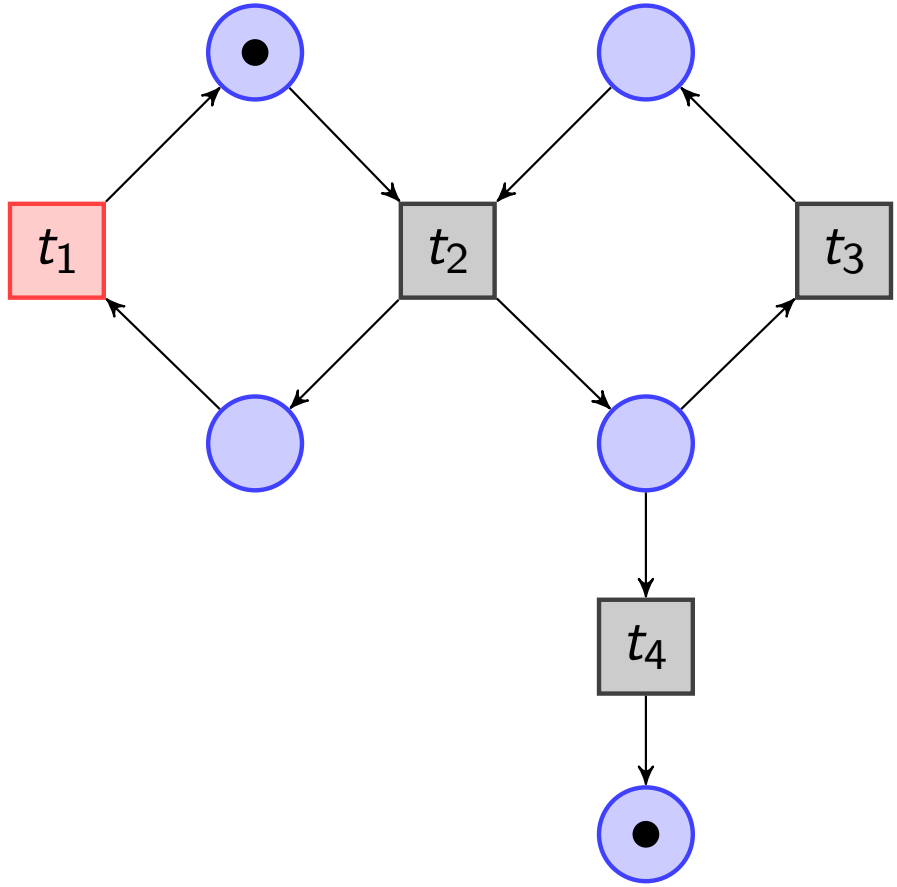
Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Petrinetze: Beispiel



Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

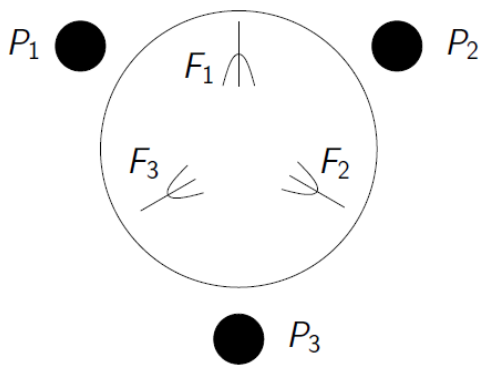
Petrinetze: Beispiel



Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Dining Philosophers

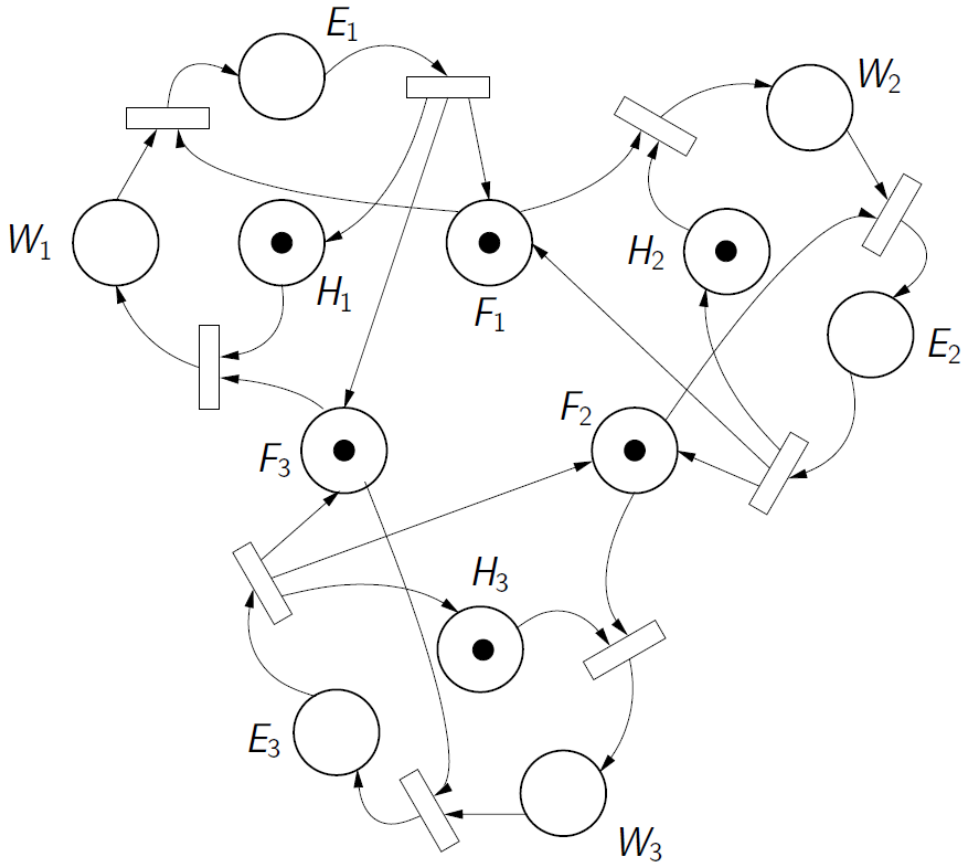
- Drei Philosophen sitzen um einen runden Tisch, zwischen je zwei Philosophen liegt eine Gabel.
- Wird ein Philosoph hungrig, hält er sich an folgenden Ablauf:
  1. Er nimmt die Gabel zu seiner Rechten.
  2. Er nimmt die Gabel zu seiner Linken.
  3. Er isst.
  4. Er legt beide Gabeln zurück.
- Sollte eine Gabel nicht an ihrem Platz liegen, wenn der Philosoph sie benötigt, so wartet er, bis die Gabel wieder verfügbar ist.



Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Dining Philosophers

Modellierung als Petrinetz:



Deadlock erreichbar, d.h., eine Markierung, bei der keine Transition mehr geschaltet werden kann.

Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
8. Petri-Netze
   - 8.1. Motivation
   - 8.2. Definitionen
   - 8.3. Modellierung

Definitionen

Petrinetz

... wird beschrieben durch ein 5-Tupel $N = \langle S, T, {}^\bullet(), ()^\bullet, m_0 \rangle$, wobei

- S ... endliche Menge von Stellen
- T ... endliche Menge von Transitionen
- ${}^\bullet(): T \mapsto M$ ... Vorbedingungen
- $()^\bullet: T \mapsto M$ ... Nachbedingungen
- $m_0 \in M$ ... Anfangsmarkierung

M ... Menge der Markierungen, d.h., aller Abbildungen $S \mapsto \mathbb{N}$

$m_0 \in M$ legt fest, wieviele Marken zu Beginn in jeder Stelle liegen.

${}^\bullet t \in M$ legt fest, wieviele Marken die Transition t aus jeder Stelle entfernt.

$t^\bullet \in M$ legt fest, wieviele Marken die Transition t zu jeder Stelle hinzufügt.

Schalten und Erreichbarkeit

$m, m' \in M$ ... Markierungen

Ordnung: $m \leq m'$ falls $m(s) \leq m'(s)$ für alle $s \in S$

Addition: $m \oplus m' = m''$ falls $m''(s) = m(s) + m'(s)$ für alle $s \in S$.

Subtraktion: $m \ominus m' = m''$ falls $m''(s) = m(s) - m'(s)$ für alle $s \in S$.

- Eine Transition t ist für eine Markierung m aktiviert, wenn ${}^\bullet t \leq m$ gilt, d.h., wenn genug Marken vorhanden sind, um die Transition zu schalten.
- Sei t eine Transition, die für die Markierung m aktiviert ist. Dann kann t schalten (oder feuern), was zu der Nachfolgemarkierung $m' = m \ominus {}^\bullet t \oplus t^\bullet$ führt. Symbolisch $m[t\rangle m'$.
- Eine Markierung m_n heißt erreichbar in einem Netz, falls es eine Folge von Transitionen $t_1, \ldots, t_n$ gibt mit $m_0[t_1\rangle m_1 \ldots m_{n-1}[t_n\rangle m_n$, wobei m_0 die Anfangsmarkierung ist.

Übernommen von Barbara König, Vorlesung „Modellierung nebenläufiger Systeme“, Uni Duisburg-Essen, 2011

Graphische Notation

In der graphischen Notation werden ${}^\bullet t$ und $t^\bullet$ folgendermaßen dargestellt:

- Kein Pfeil zwischen s und t, falls ${}^\bullet t(s) = 0$ (bzw. $t^\bullet(s) = 0$).
- Ein Pfeil zwischen s und t, falls ${}^\bullet t(s) = 1$ (bzw. $t^\bullet(s) = 1$).
- Ein Pfeil mit Beschriftung n zwischen s und t, falls ${}^\bullet t(s) = n > 1$ (bzw. $t^\bullet(s) = n > 1$).

Der Wert ${}^\bullet t(s)$ bzw. $t^\bullet(s)$ wird auch als Gewicht bezeichnet.

Was Sie heute erwartet

1. Organisatorisches
2. Was bedeutet Modellierung?
3. Aussagenlogik
4. Endliche Automaten
5. Reguläre Sprachen
6. Kontextfreie Grammatiken
7. Prädikatenlogik
8. Petri-Netze
   - 8.1. Motivation
   - 8.2. Definitionen
   - 8.3. Modellierung

Beispiel: Leser-Schreiber-Problem

Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

Beispiel: Leser-Schreiber-Problem

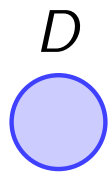
Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem

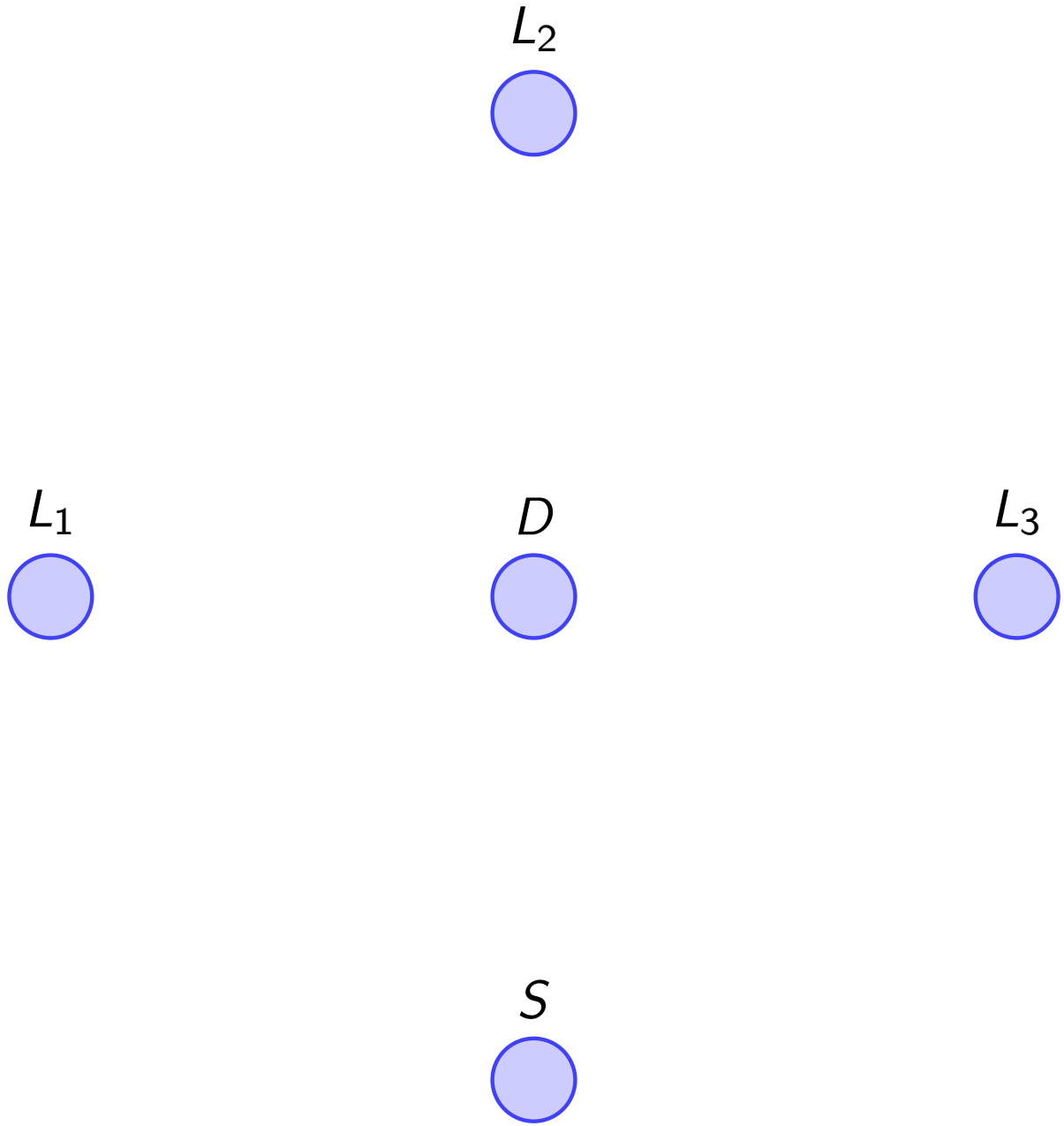
Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

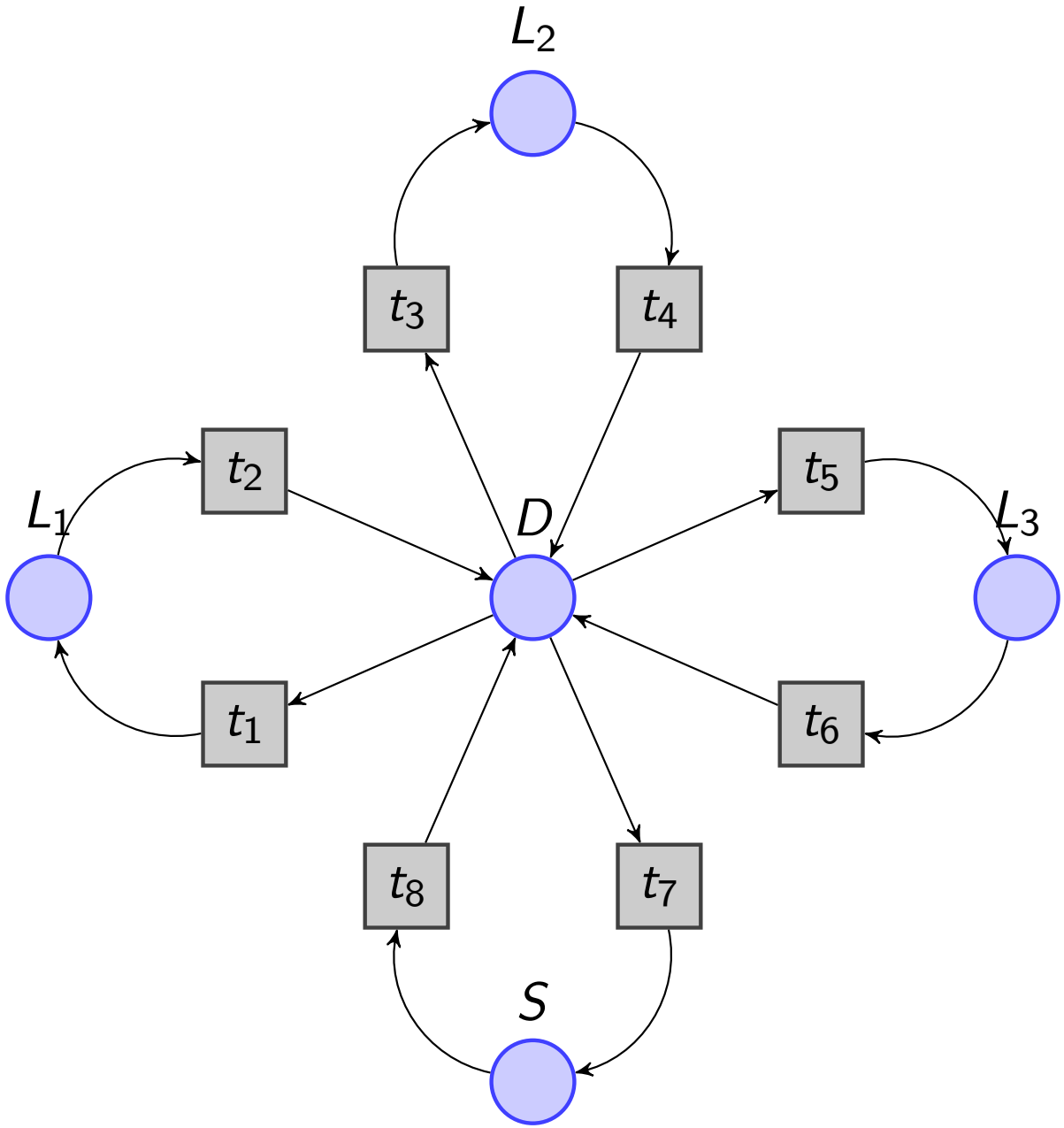
- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem

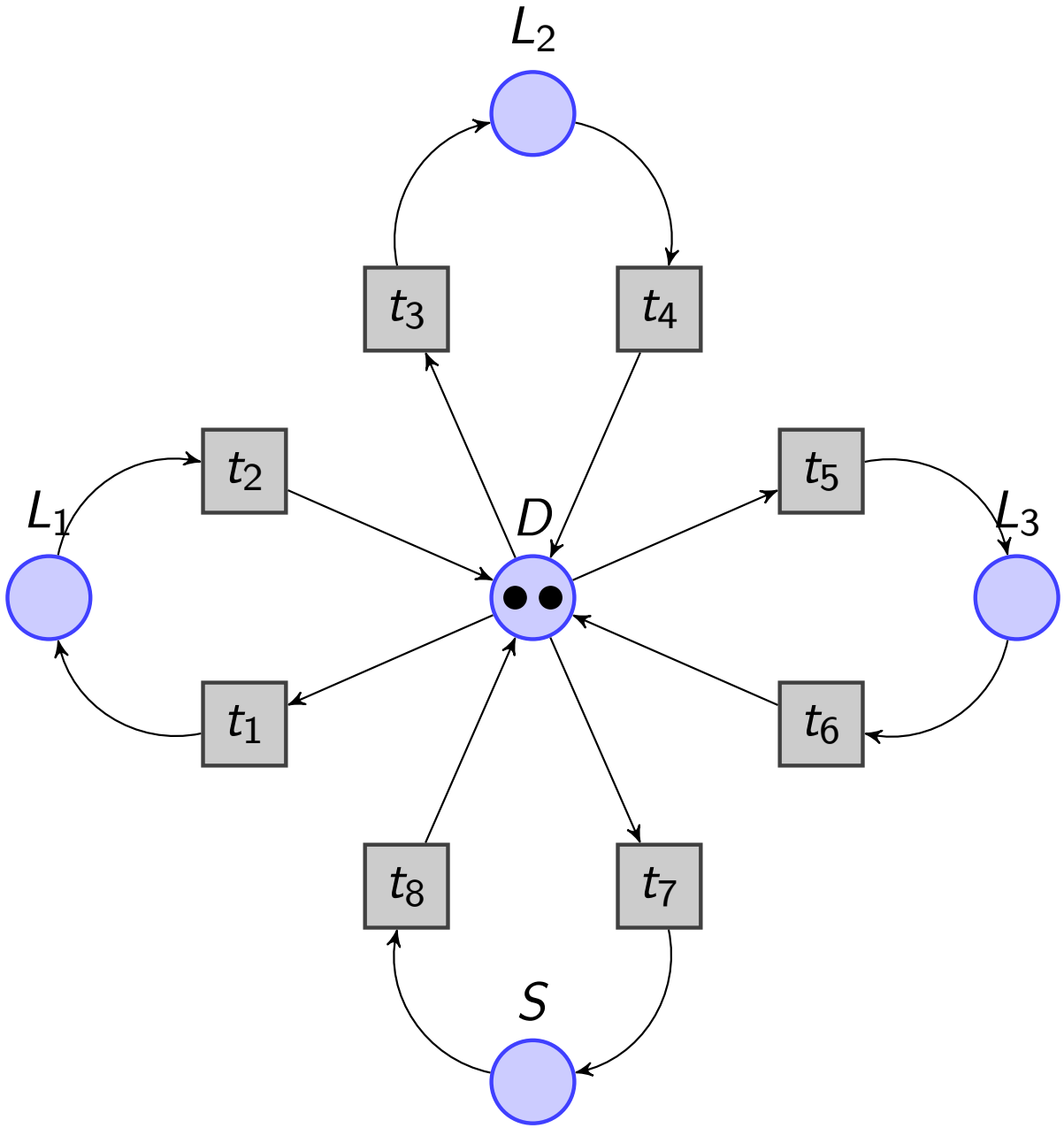
Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem

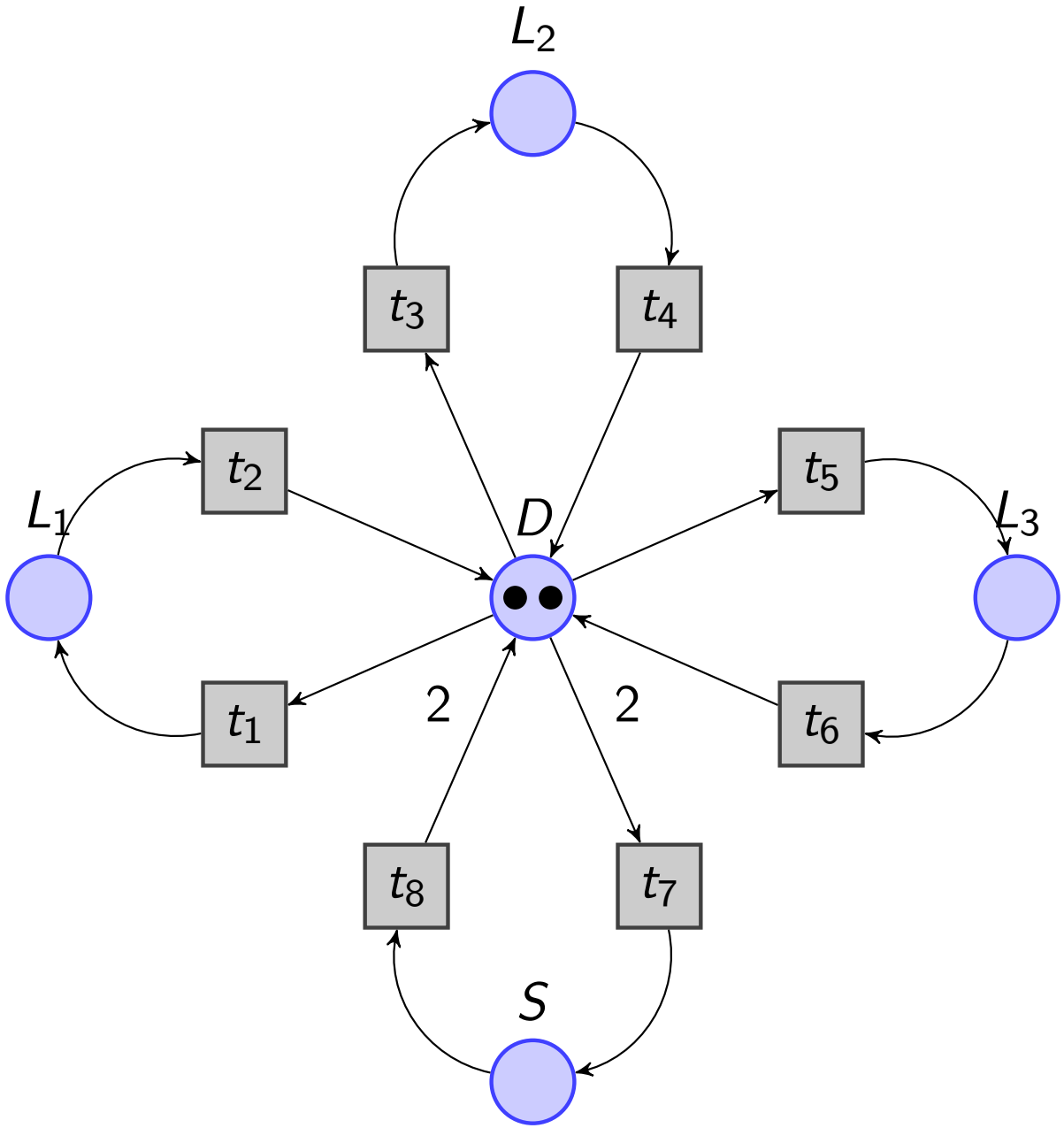
Leser-Schreiber-Problem

Beim Leser-Schreiber-Problem operieren n Leserprozesse und m Schreiberprozesse auf ein und derselben Datei. Damit die Dateiinhalte nicht inkonsistent werden, müssen die folgenden Bedingungen beachtet werden:

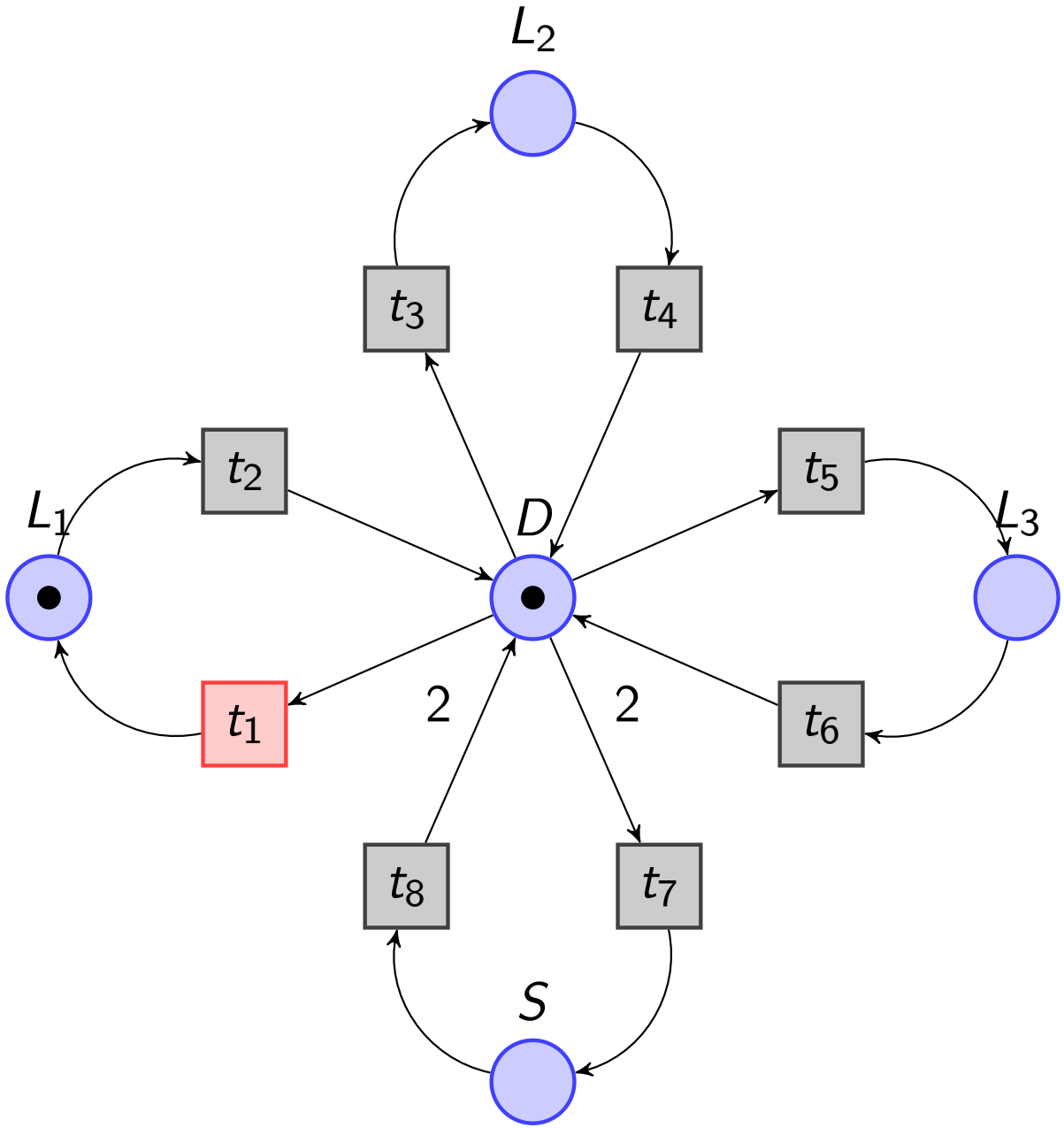
- Es können zur gleichen Zeit mehrere Leserprozesse auf die Datei zugreifen.
- Ein Schreiberprozess darf nur dann auf die Datei zugreifen, wenn gerade kein anderer Prozess (lesend oder schreibend) auf die Datei zugreift.

Modellieren Sie das Leser-Schreiber-Problem als Petrinetz mit $n = 3$ und $m = 1$. Es können maximal 2 Leserprozesse gleichzeitig die Datei lesen.

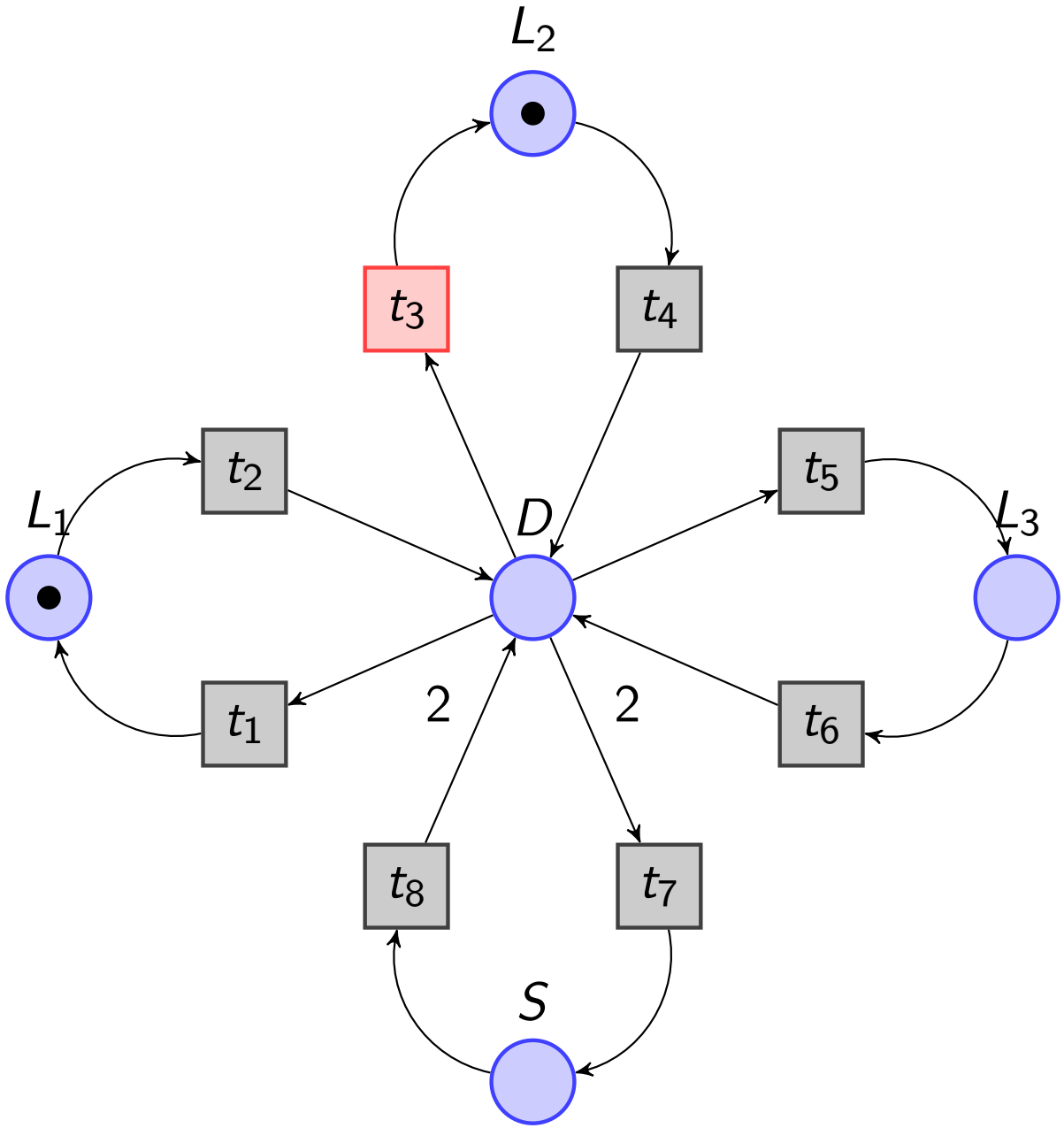
Beispiel: Leser-Schreiber-Problem



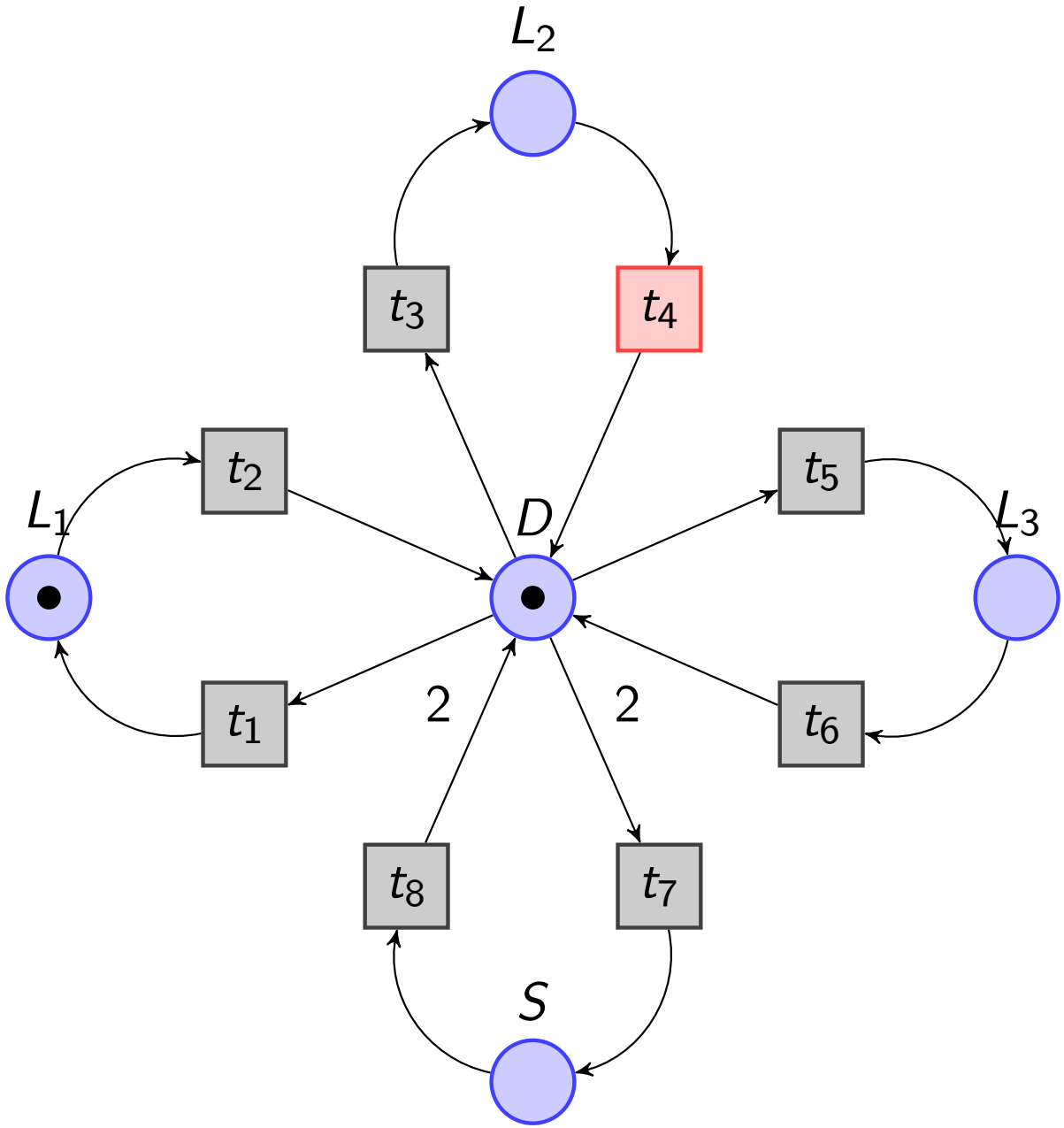
Beispiel: Leser-Schreiber-Problem



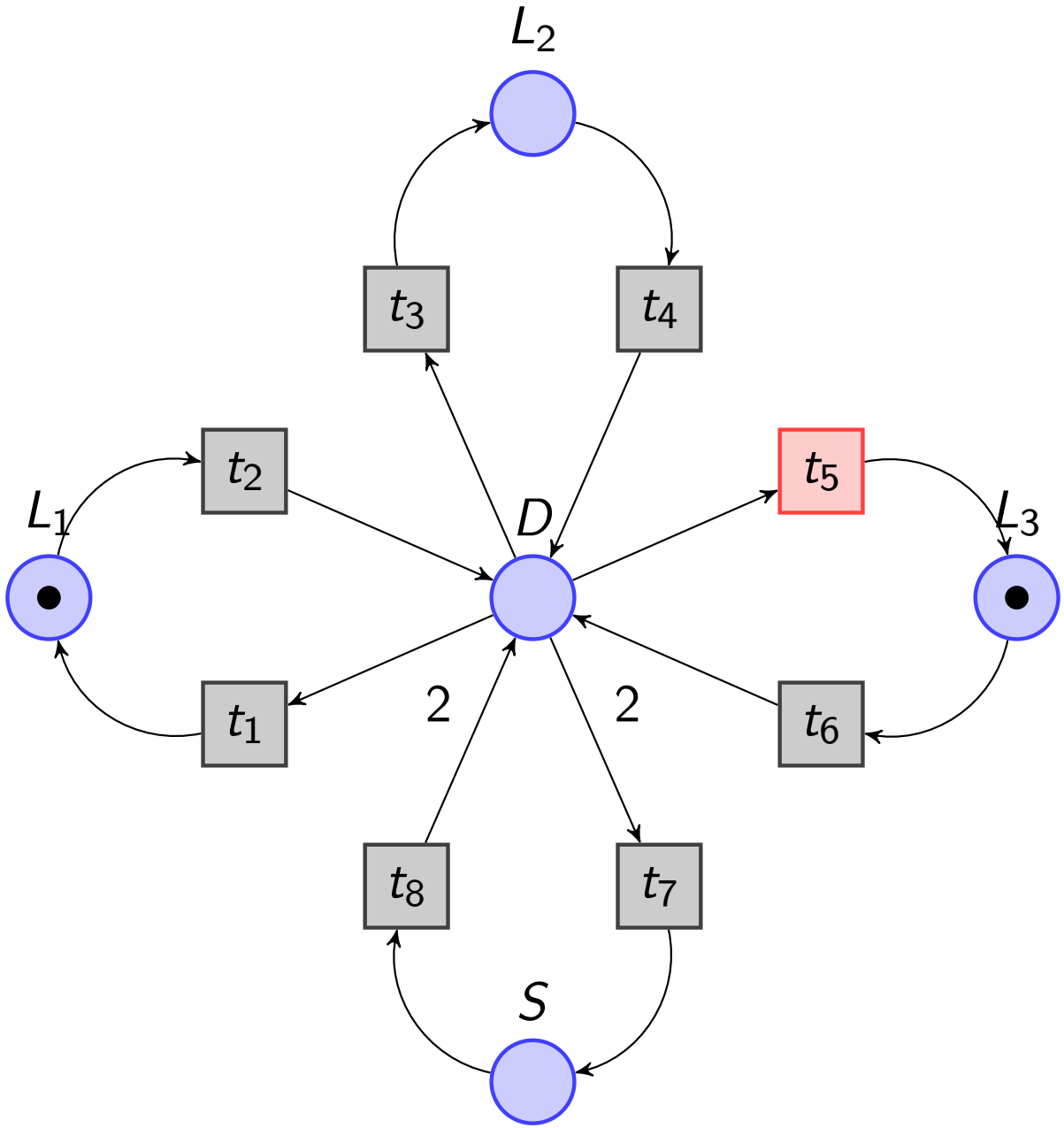
Beispiel: Leser-Schreiber-Problem



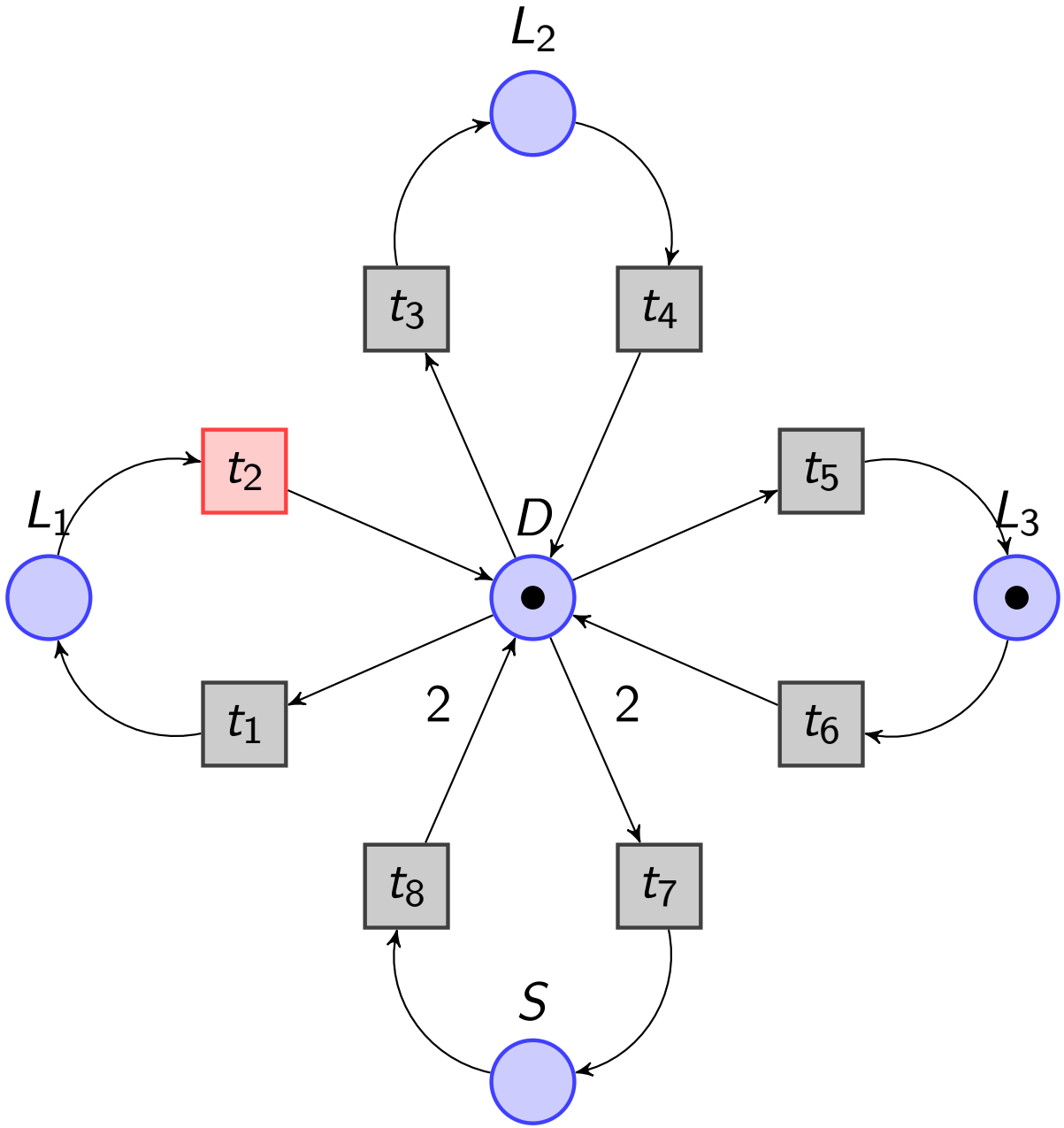
Beispiel: Leser-Schreiber-Problem



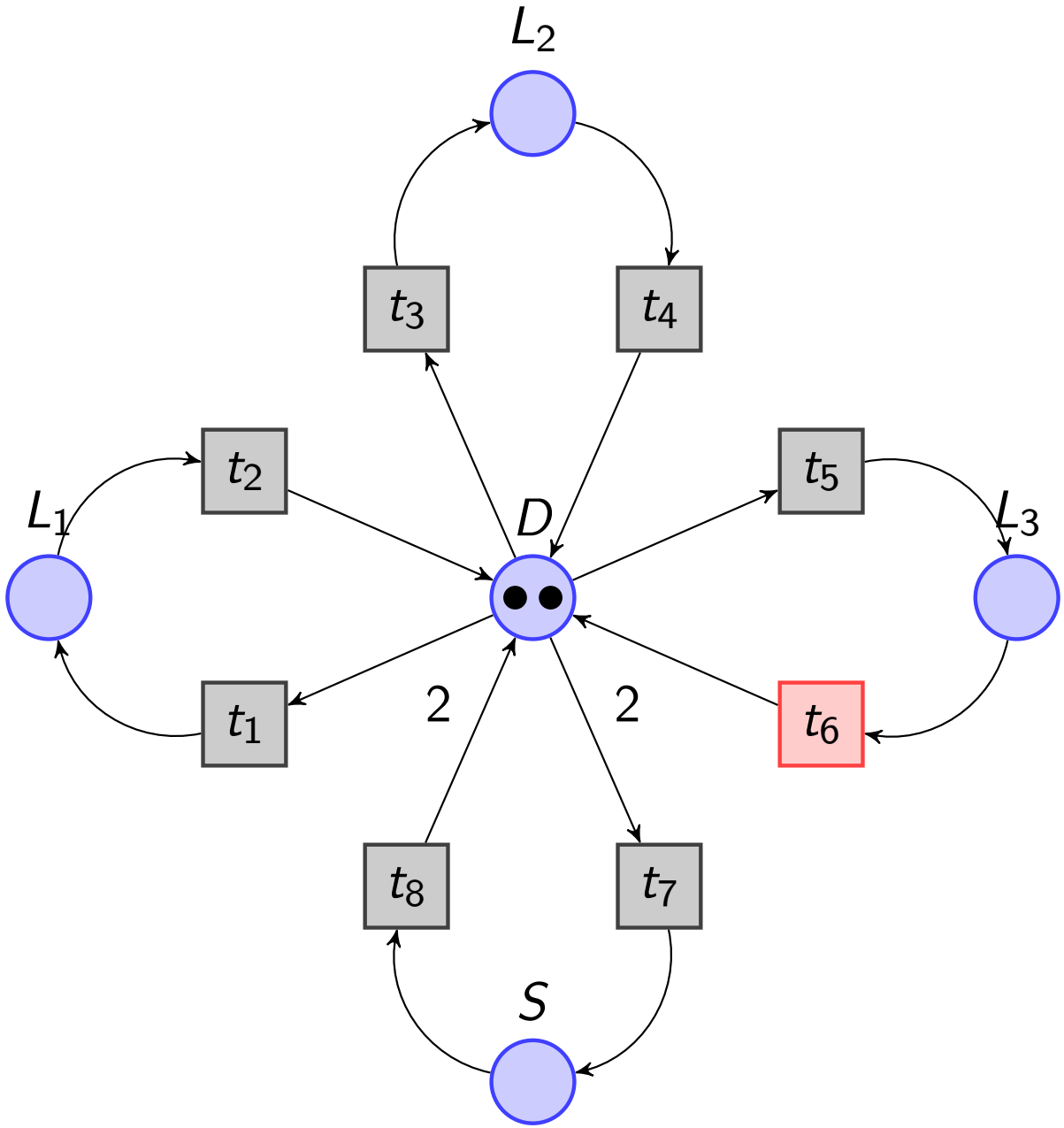
Beispiel: Leser-Schreiber-Problem



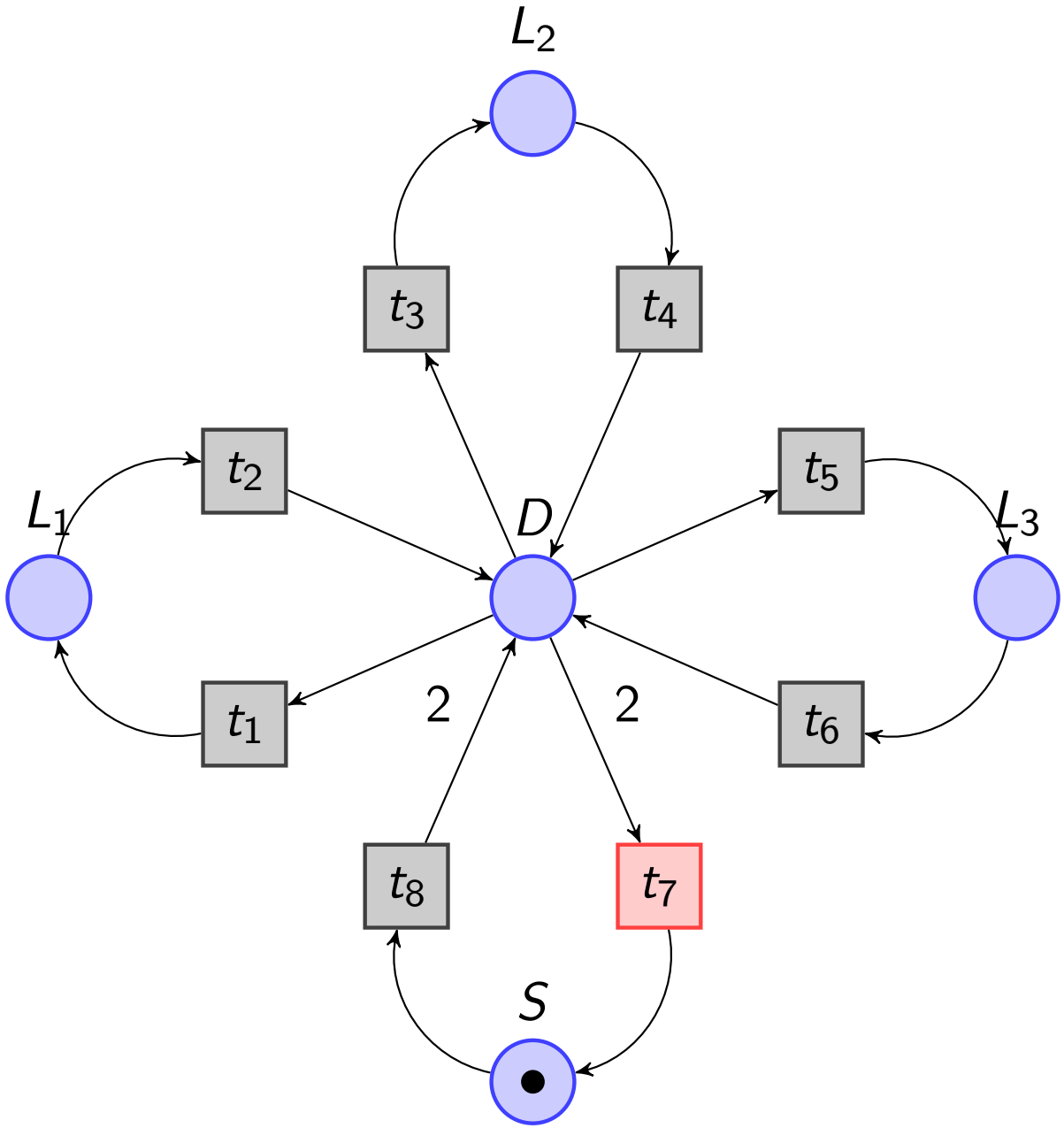
Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem



Beispiel: Leser-Schreiber-Problem