Frage: Welche Arten von Sprachen können durch reguläre Ausdrücke beschrieben werden?

Antwort: Genau dieselben Sprachen, die durch (deterministische oder nicht-deterministische) endliche Automaten akzeptiert werden können.

— Diese Sprachen werden reguläre Sprachen genannt.

Details: In der Veranstaltung GL-2.

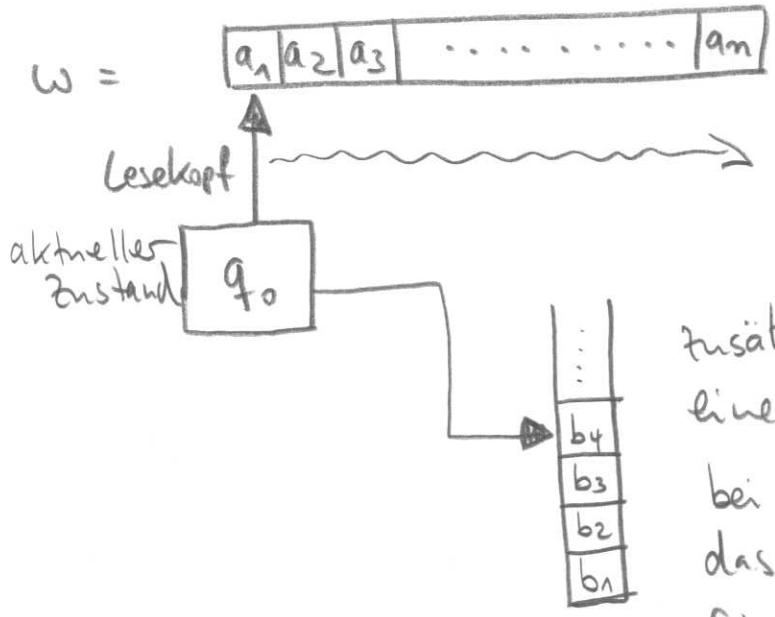
Ausblick:

Abgesehen von DFAs, NFAs und regulären Ausdrücken kann man die regulären Sprachen auch durch bestimmte Grammatiken erzeugen: so genannte reguläre Grammatiken — das sind kontextfreie Grammatiken, die von einer besonders einfachen Form sind. Generell gilt: Für jede reguläre Sprache L gibt es eine kontextfreie Grammatik, die die Sprache L erzeugt. Aber es gibt kontextfreie Grammatiken, die nicht-reguläre Sprachen erzeugen.

Analog zu DFAs und NFAs gibt es auch ein erweitertes Automatenmodell, das genau diejenigen Sprachen akzeptiert, die von kontextfreien Grammatiken erzeugt werden: so genannte

Kellerautomaten

Schematische Darstellung der Verarbeitung eines Eingabeworts durch einen Kellerautomaten:



zusätzlicher Speicher in Form eines Kellers ("Stapel"), bei dem immer nur auf das oberste Element des Stapels zugegriffen werden kann.

Details: In der Vorlesung GL-2. Dort werden auch allgemeinere Arten von Grammatiken betrachtet, z.B. so genannte kontext-sensitive Grammatiken.

7.2 Petri-Netze

Petri-Netze:

- eingeführt von C.A. Petri, 1962
- formaler Kalkül zur Modellierung von Abläufen, an denen mehrere Prozesse beteiligt sind
- modelliert werden die Interaktionen zwischen Prozessen und die Effekte, die sich daraus ergeben, dass Operationen prinzipiell gleichzeitig ausgeführt werden können (Stichwort: "Nebenläufigkeit").

Typische Anwendungsbeispiele für Petri-Netze:

zur Modellierung von

- realen oder abstrakten Automaten und Maschinen
- kommunizierenden Prozessen (z.B. in Rechnern)
- Verhalten von Software- oder Hardware-Komponenten
- Geschäftsabläufe
- Spiele (Spielregeln)

Der Kalkül der Petri-Netze basiert auf bipartiten gerichteten Graphen:

- 2 Sorten von Knoten, die
  - Bedingungen oder Zustände (sog. "Stellen") bzw.
  - Aktivitäten (sog. "Transitionen")

  repräsentieren
- Kanten verbinden "Aktivitäten" mit ihren "Vorbedingungen" und ihren "Nachbedingungen"
- Knotenmarkierungen repräsentieren den veränderlichen "Zustand" des Systems.

<u>Definition 7.21</u>: (Petri-Netz)

Ein <u>Petri-Netz</u> $P = (S, T, F)$ besteht aus

- einer endlichen Menge S, den sog. <u>Stellen</u> von P
- einer endlichen Menge T, den sog <u>Transitionen</u> von P
- einer Relation $F \subseteq (S \times T) \cup (T \times S)$, den sog. Kanten von P.

Die Mengen S und T sind disjunkt, d.h. $S \cap T = \emptyset$.

Ein Petri-Netz P bildet einen bipartiten gerichteten Graphen mit Knotenmenge $S \cup T$ und Kantenmenge F.

<u>Graphische Darstellung</u>:

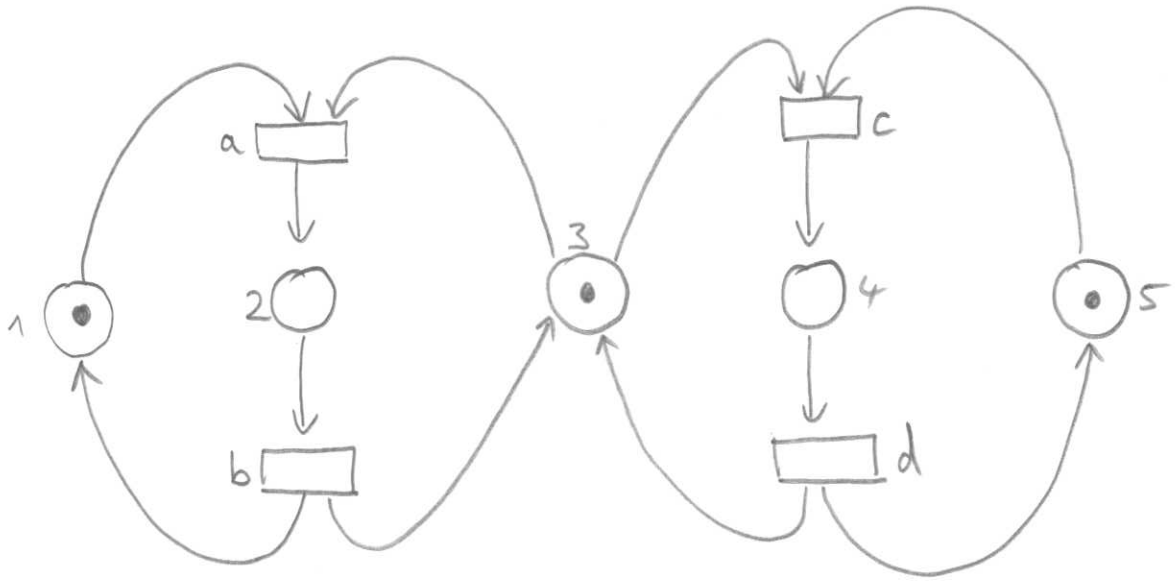
- "Stellen", d.h. Knoten in S, werden durch Kreise dargestellt
- "Transitionen", d.h. Knoten in T, werden durch Rechtecke dargestellt

<u>Definition 7.22</u>

Der "Zustand" eines Petri-Netzes $P = (S, T, F)$ wird durch eine <u>Markierungsfunktion</u> (kurz: <u>Markierung</u>) $M : S \to \mathbb{N}$, die jeder Stelle $s \in S$ eine Anzahl $M(s)$ von sog. <u>Marken</u> zuordnet, repräsentiert.

Beispiel 7.23:

Graphische Darstellung eines Petri-Netzes $P = (S, T, F)$ und einer Markierung M: die einzelnen "Marken", die M einer Stelle $s \in S$ zuordnet, werden durch Punkte • in dem die Stelle s repräsentierenden Knoten dargestellt.



$S = \{1, 2, 3, 4, 5\}$

$T = \{a, b, c, d\}$

$F = \{ (1,a), (3,a), (a,2),$
$(2,b), (b,1), (b,3),$
$(3,c), (5,c), (c,4),$
$(4,d), (d,3), (d,5) \}$

$M : S \to \mathbb{N}$ mit $M(1) = M(3) = M(5) = 1$ und $M(2) = M(4) = 0$.

Definition 7.24

Sei $P = (S, T, F)$ ein Petri-Netz und sei $t \in T$ eine Transition von P.

Wir setzen

- $\text{Vorbereich}(t) := \{ s \in S : (s,t) \in F \}$

  = "Menge aller Stellen, von denen aus eine Kante in t hineinführt"

- $\text{Nachbereich}(t) := \{ s \in S : (t,s) \in F \}$

  = "Menge aller Stellen, zu denen eine von t ausgehende Kante hinführt"

Petri-Netze verändern ihren "Zustand", indem Transitionen "schalten" und dadurch die "Markierung" des Petri-Netzes ändern. Das wird folgendermaßen präzisiert:

Schaltregel:

Sei $P = (S, T, F)$ ein Petri-Netz und sei $M: S \to \mathbb{N}$ eine Markierung.

Das "Schalten einer Transition $t \in T$" überführt die Markierung M in eine Markierung $M': S \to \mathbb{N}$.

Die Transition <u>t kann schalten</u>, wenn gilt:

f.a. Stellen $s \in$ Vorbereich(t) ist $M(s) \geq 1$,

dh: Jede Stelle s in Vorbereich(t) hat mindestens eine Marke.

Wenn die Transition t schaltet, so gilt für die sog. <u>Nachfolgemarkierung M'</u>: f.a. $s \in S$:

$$M'(s) := \begin{cases} M(s) - 1 & \text{falls } s \in \text{Vorbereich}(t) \setminus \text{Nachbereich}(t) \\ M(s) + 1 & \text{falls } s \in \text{Nachbereich}(t) \setminus \text{Vorbereich}(t) \\ M(s) & \text{sonst} \end{cases}$$

D.h. Das "Schalten von Transition t" bewirkt, dass in jeder Stelle in Vorbereich(t) eine Marke entfernt wird und dass in jeder Stelle in Nachbereich(t) eine Marke hinzugefügt wird.

Wenn mehrere Transitionen schalten können, so wird eine davon "nicht-deterministisch ausgewählt".

In jedem Schritt schaltet genau eine Transition.

Durch schrittweises Schalten von Transitionen wird der Ablauf von Prozessen modelliert.

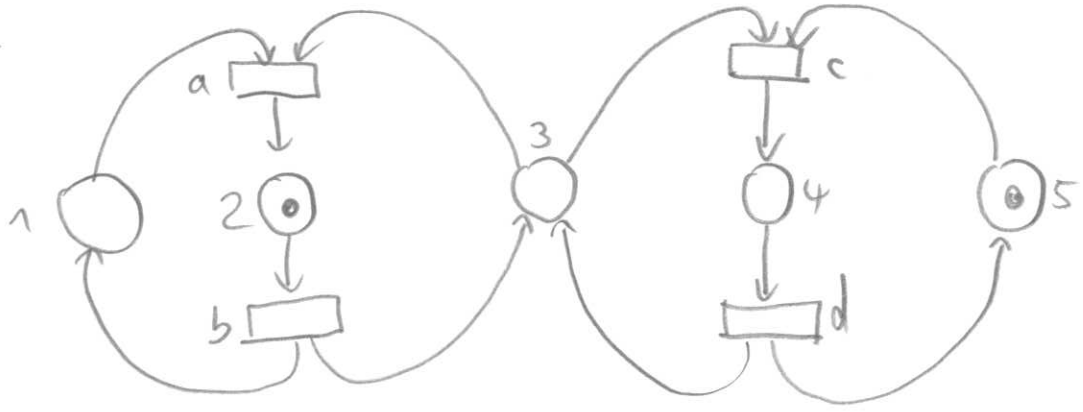
<u>Beispiel 7.25:</u>

Seien $P = (S, T, F)$ und $M: S \to \mathbb{N}$ das Petri-Netz und die Markierung aus Beispiel 7.23.

Bei der angegebenen Markierung M können die Transitionen a und c schalten.

Wenn wir a schalten lassen ergibt sich als Nachfolgemarkierung die Markierung $M': S \to \mathbb{N}$ mit $M'(1) = 0$, $M'=(3) = 0$, $M'(2) = 1$, $M'(4) = 0$, $M'(5) = 1$,

d.h.:



Als nächstes kann Transition c <u>nicht</u> schalten, da die Stelle 3 keine Marke trägt.

Die einzige Transition, die jetzt schalten kann, ist Transition b, deren Schalten bewirkt, dass die Marke bei 2 verschwindet und stattdessen Marken bei 1 und 3 erzeugt werden. Nach dem Schalten von b ist das System also wieder in seinem "ursprünglichen Zustand", d.h. es trägt die Markierung M.

Insgesamt gilt: Das Petri-Netz P aus Bsp. 7.23 (zusammen mit der Startmarkierung M) modelliert zwei zyklisch ablaufende Prozesse.

Die Stelle 3 "synchronisiert" die beiden Prozesse, so dass sich nie gleichzeitig in beiden Stellen 2 und 4 eine Marke befinden kann.

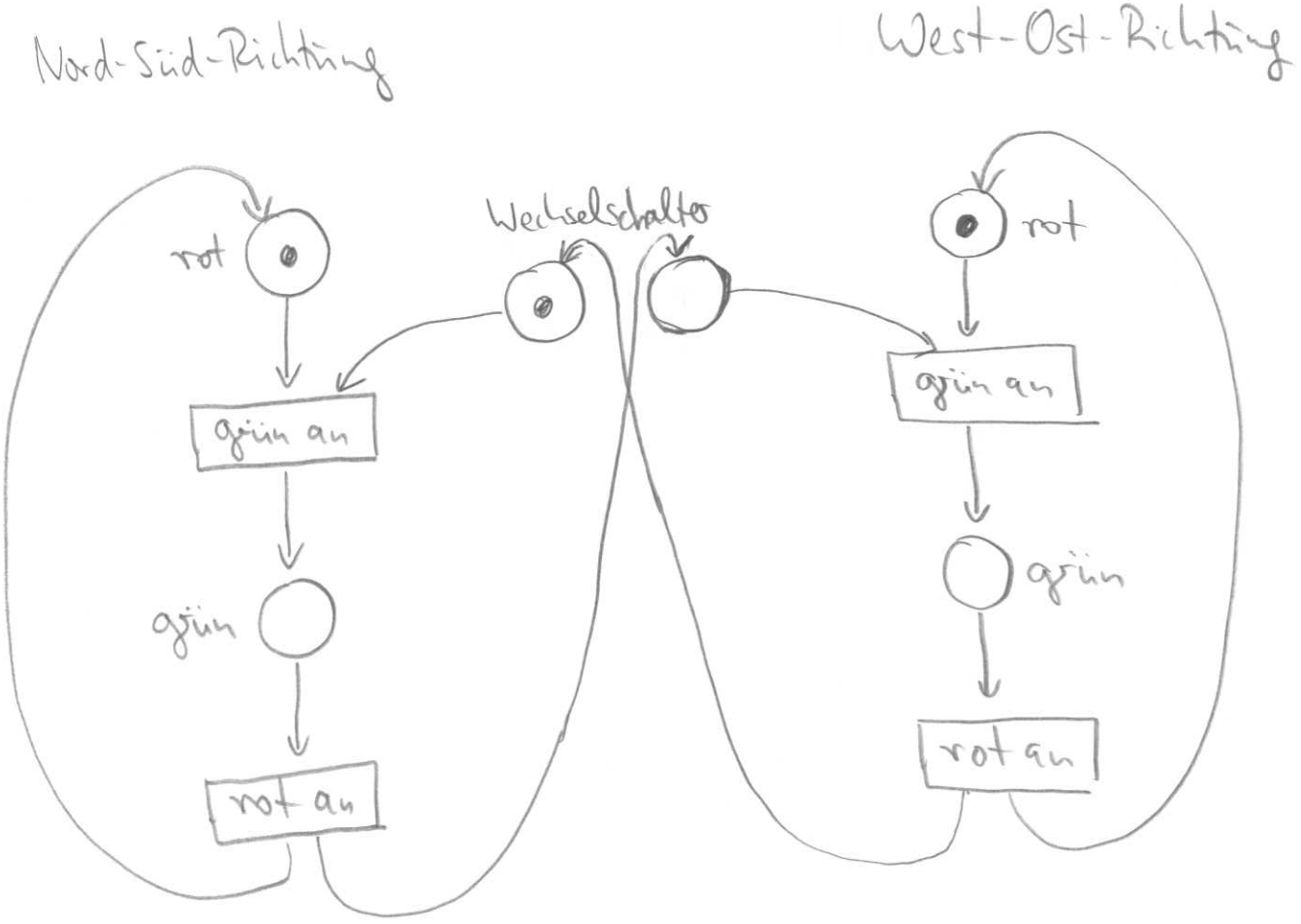
Auf diese Weise könnte man z.B. beschreiben, wie Autos eine 1-spurige Brücke von 2 Seiten überqueren, so dass sich immer nur 1 Auto auf der Brücke befindet.

Beispiel 7.26

Modellierung einer Ampel an einer Kreuzung

- 2 sich zyklisch wiederholende Prozesse:
  - "grün" in Nord-Süd-Richtung
  - "grün" in West-Ost-Richtung
- die beiden Prozesse sollen sich immer abwechseln

Petri-Netz incl. Anfangs-Markierung:



Die beiden Stellen "Wechselschalter" koppeln die Prozesse, so dass abwechselnd in West-Ost- bzw in Nord-Süd-Richtung die Ampel "grün" ist.

8. Eine Fallstudie

In diesem Kapitel steht ein konkretes Anwendungsbeispiel im Vordergrund. Seine Strukturen, Eigenschaften etc. werden mit verschiedenen Kalkülen modelliert (die unterschiedlichen Kalküle werden eingesetzt, um unterschiedliche Aspekte des Anwendungsbeispiels zu beschreiben).

8.1 Aufgabenstellung: Autowerkstatt

Ziel: Modelliere die Auftragsabwicklung in einer Autowerkstatt.

Genauer:
- Datenbank entwerfen
- Abläufe analysieren und verbessern

8.2 Datenbank-Entwurf: Autowerkstatt

Kurzbeschreibung der "Informationsstruktur":

1) Kunde: hat einen Namen,
besitzt Kraftfahrzeug(e) (kurz: KFZ),
erteilt Aufträge

2) Auftrag: hat ein Eingangsdatum,
betrifft ein KFZ, wird von Mechaniker(n)
bearbeitet, benötigt Ersatzteile bestimmter
Arten und Mengen (i.S.v. "Anzahlen")

3) KFZ: hat Fahrgestellnummer und Baujahr,
ist entweder ein PKW oder ein Motorrad;
zu PKWs interessiert ihre Farbe, zu
Motorrädern der Tuningsatz

4) Typ: jedes KFZ hat einen Typ; jeder
Mechaniker ist für einige Typen ausgebildet;
Ersatzteile sind für bestimmte Typen
verwendbar.

"Informationsstruktur" als ER-Modell

Zentrale Entity-Typen:

Kunde, Auftrag, KFZ, KFZ-Typ

Relationen-Typen:

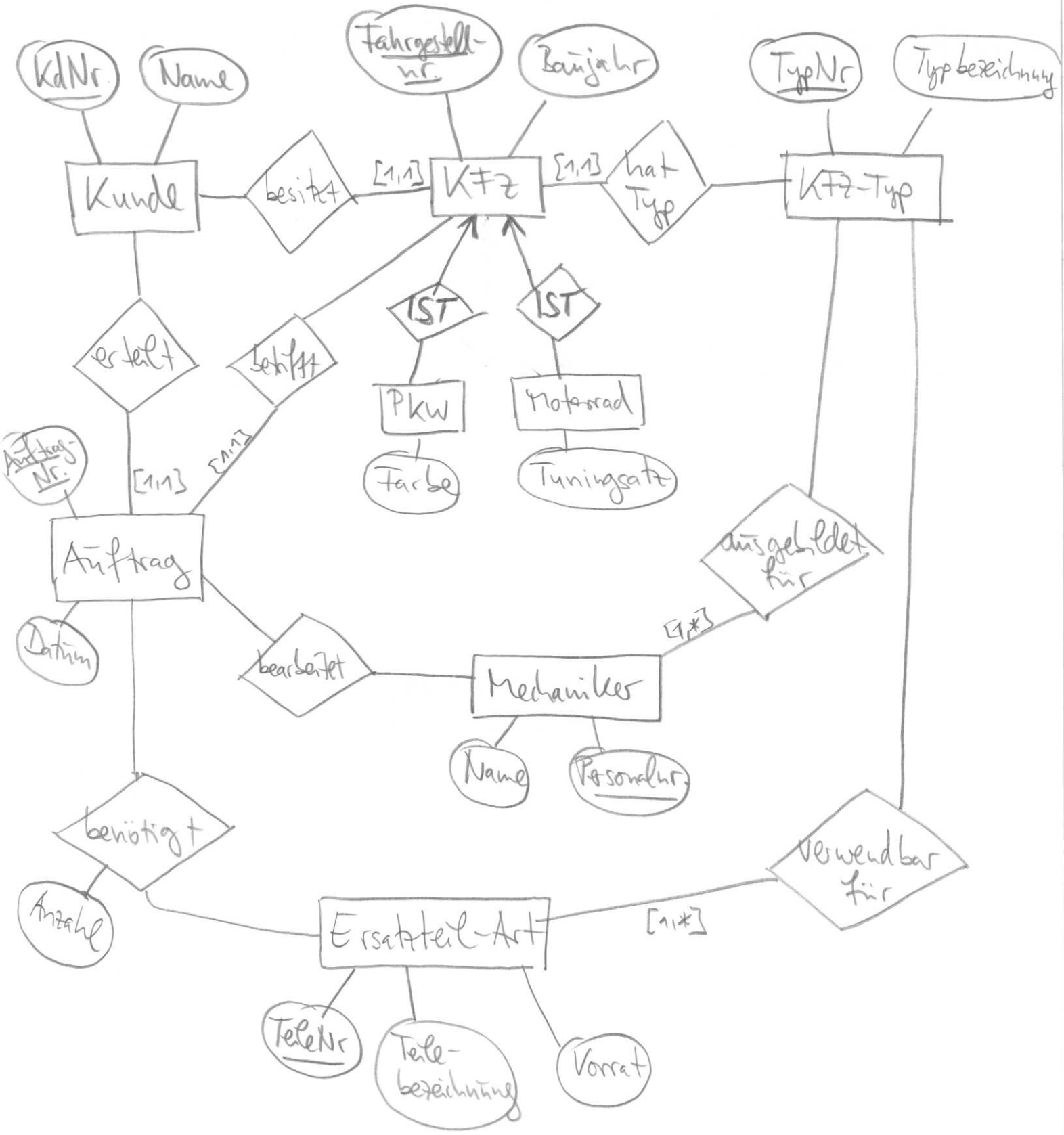
- besitzt (Kunde besitzt KFZ)
- erteilt (Kunde erteilt Auftrag)
- betrifft (Auftrag betrifft KFZ)
- hat Typ (KFZ hat KFZ-Typ)
- ausgebildet für (Mechaniker ist ausgebildet für KFZ-Typen)
  (⇒ Entity-Typ "Mechaniker" einführen)

auch nötig: Relationen zur Modellierung davon

- welche Ersatzteile ein Auftrag benötigt
- welche Ersatzteile für welchen KFZ-Typ geeignet sind
- welcher Mechaniker welchen Auftrag bearbeitet

auch noch nötig: Unterscheidung von KFZ in PKW und Motorräder

E.R-Modell : Autowerkstatt



<u>Beachte:</u> Durch Angabe von Kardinalitäten haben wir einige Entscheidungen getroffen:

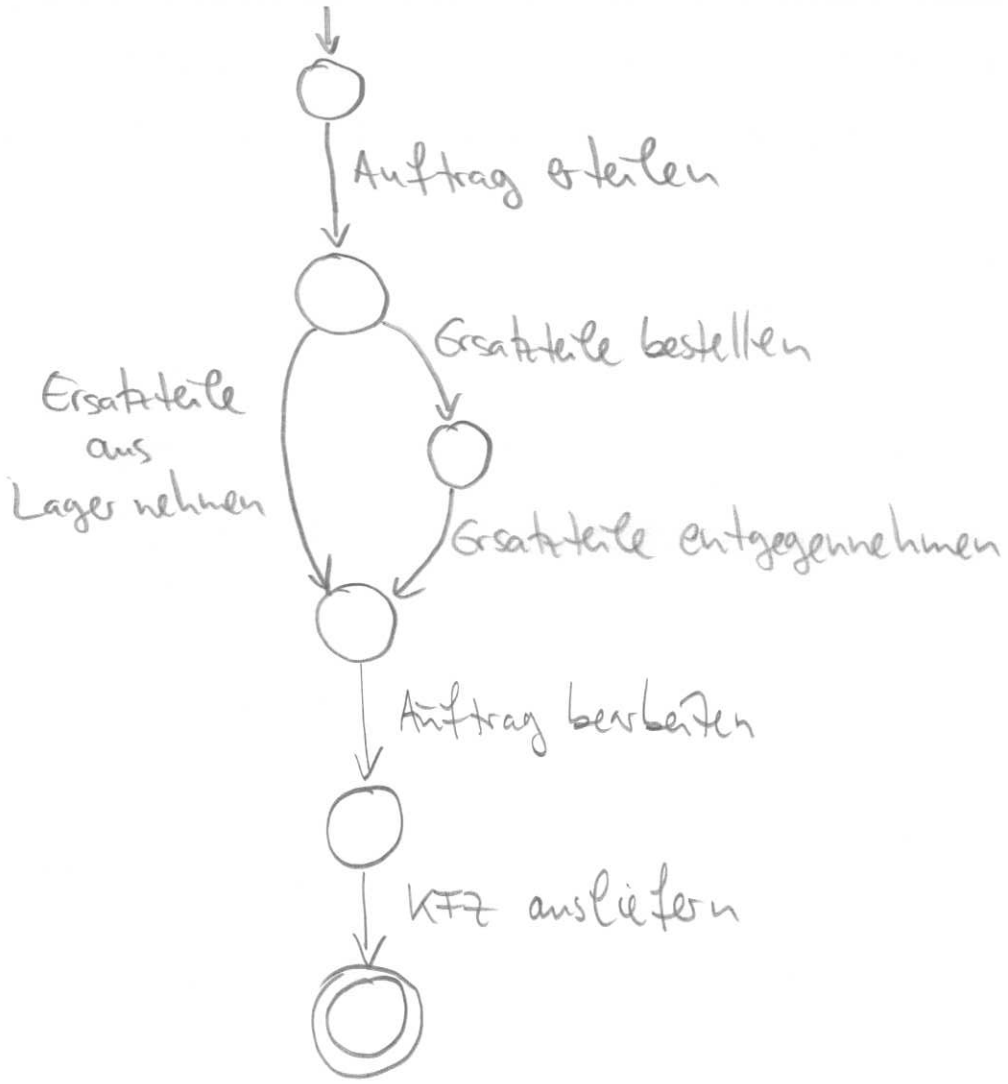
- jedes KFZ hat genau einen KFZ-Typ
- jedes KFZ hat genau einen Besitzer
- jeder Auftrag betrifft genau ein KFZ
- jeder Mechaniker ist für mindestens einen KFZ-Typ ausgebildet
- jede Ersatzteil-Art ist für mindestens einen KFZ-Typ verwendbar.

8.3 Abläufe bei der Auftragserteilung

Eine Untersuchung der Geschäftsabläufe in der Autowerkstatt ergibt, dass jeder Auftrag folgende Stationen durchläuft:

1. Der Auftrag wird erteilt
2. Fehlende Ersatzteile werden bestellt und nach dem Eintreffen entgegengenommen
3. Vorhandene Ersatzteile werden aus dem Lager genommen
4. Der Auftrag wird von einem Mechaniker bearbeitet
5. Das KFZ wird dem Kunden ausgeliefert.

Modellierung dieser Abläufe als Transitionssystem (endlicher Automat)



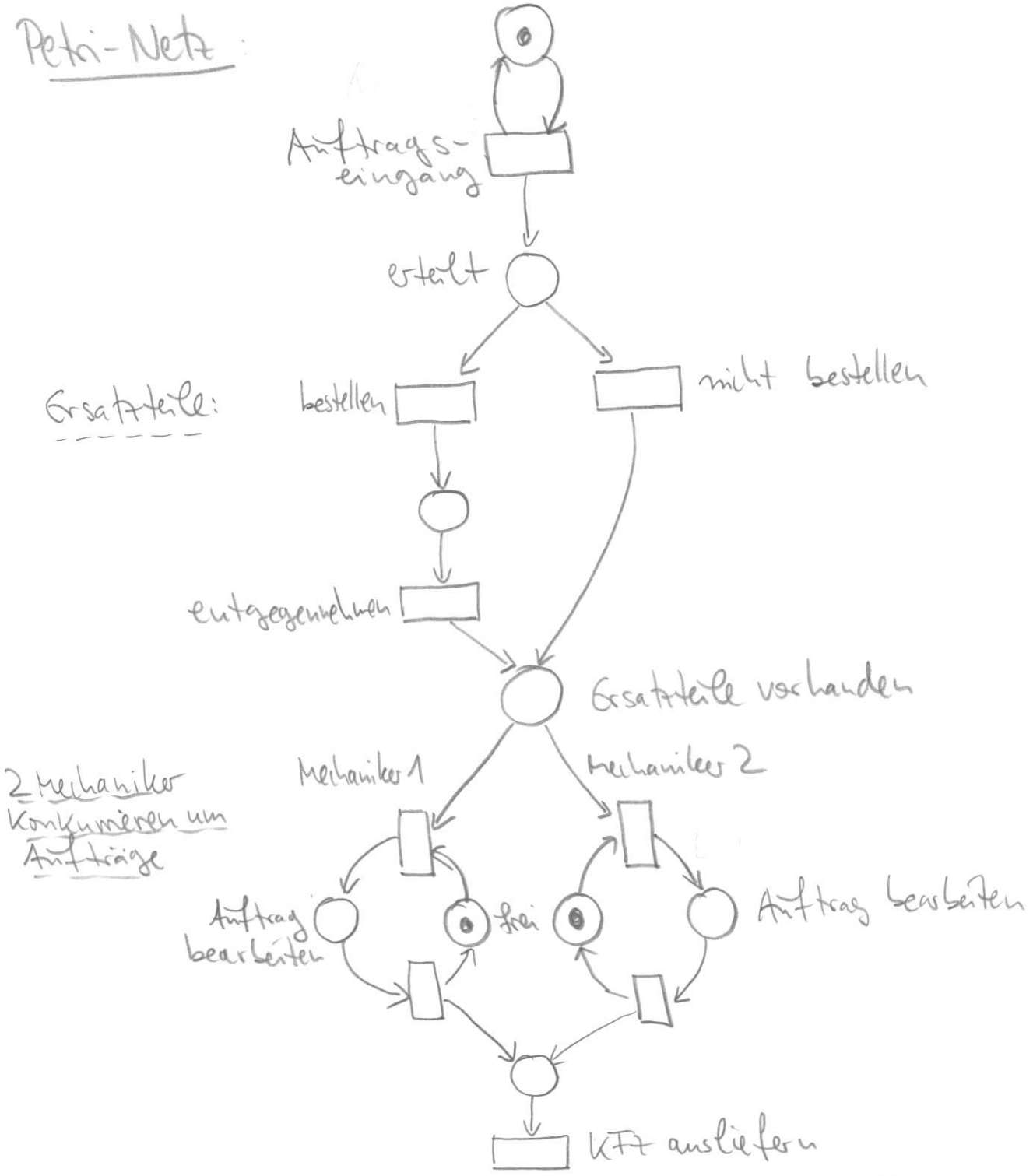
Einschränkungen dieses Modells:

- die Abläufe in der Werkstatt werden nur aus der Sicht eines einzelnen Auftrags beschrieben
- das Modell spricht nur über die "Ersatzteile insgesamt", aber nicht über ihre Art und Anzahl
- Aktionsfolgen, bei denen mehrere Aufträge von mehreren Mechanikern bearbeitet werden, können durch dieses Modell nicht beschrieben werden.

Modellierung der Auftragsbearbeitung durch ein Petri-Netz:

Ziel: modelliere, wie mehrere Aufträge nebenläufig von 2 miteinander um Aufträge konkurrierenden Mechanikern bearbeitet werden

Petri-Netz:



Organisatorisches

- keine Übungsstunden diesen Freitag!
  (heute (Mi) und morgen (Do) finden die Übungs-
  stunden noch statt).

- Beispiel-Lösung für Übungsblätter 11 + 13:
  auf Web-Seite

- korrigierte Aufgaben zu Blatt 13 können
  ab Mitte nächster Woche abgeholt werden
  (bei André Herrich oder Nicole Schweikardt)

- Ablauf der Klausur:
  - Details siehe Web-Seite der Vorlesung
  - WICHTIG:
    Studi-Ausweis + Lichtbild-Ausweis
    (Personalausweis od. Goethe-Card)
    mitbringen!!

- Keinerlei Hilfsmittel sind zugelassen (insbes: kein Skript, keine Notizen, kein Taschenrechner, <u>kein Handy</u>)

- Papier wird von uns bereitgestellt

- bringen Sie einen dokumentenechten Schreibstift mit (Kugelschreiber o.ä) !

- Die Sitzordnung wird von uns festgelegt.

- Termin für Klausureinsicht wird auf der Web-Seite bekanntgegeben

- <u>Bonuspunkte</u>: spätestens zum 14.2.08 wird eine Liste vorliegen (auch über's WWW) — bitte prüfen Sie, ob "ihre" Bonusplete korrekt eingetragen sind. Einspruch ist bis spätestens 22.2.08 möglich.

- Unterstützung zur Klausurvorbereitung

  - Do, 14.2., 14–18 h: Zusammenfassung + Fragestunde (Magnus-HS)

  - <u>Help-Desk:</u>

    - Di, 19.2., 10–12 h, R 117 (Sviatlana Danilava)
    - Mi, 20.2., 15–16 h, R 115 (ich)
    - Do, 21.2., 15–16 h, R 113 (André Hermich)
    - Fr, 22.2. 10–12 h, R 117 (William Blacoe)
    - Fr, 22.2., 14–16 h, R 117 (Martin Flockenhagen)

  + evtl noch mehr Termine (siehe Web-Seite der Vorlesung)