

# Beispielarchitekturen II

Eigenschaften mobiler und  
eingebetteter Systeme:

Dipl.-Inf. Jan Richling  
Wintersemester 2002/2003

# Message Scheduled System (MSS)

- Entwickelt seit 1999 am Lehrstuhl für Rechnerorganisation und Kommunikation der HU Berlin

- Ziele:

- Architektur für eingebettete Echtzeitsysteme
- Unterstützung von Komponierbarkeit in Bezug auf das zeitliche Verhalten
- Generalisierbarkeit für andere Eigenschaften
- „Nebenziele“:
  - \* Architekturbegriff
  - \* Komponierbarkeitsbegriff
  - \* Verifikationen
  - \* Simulationen

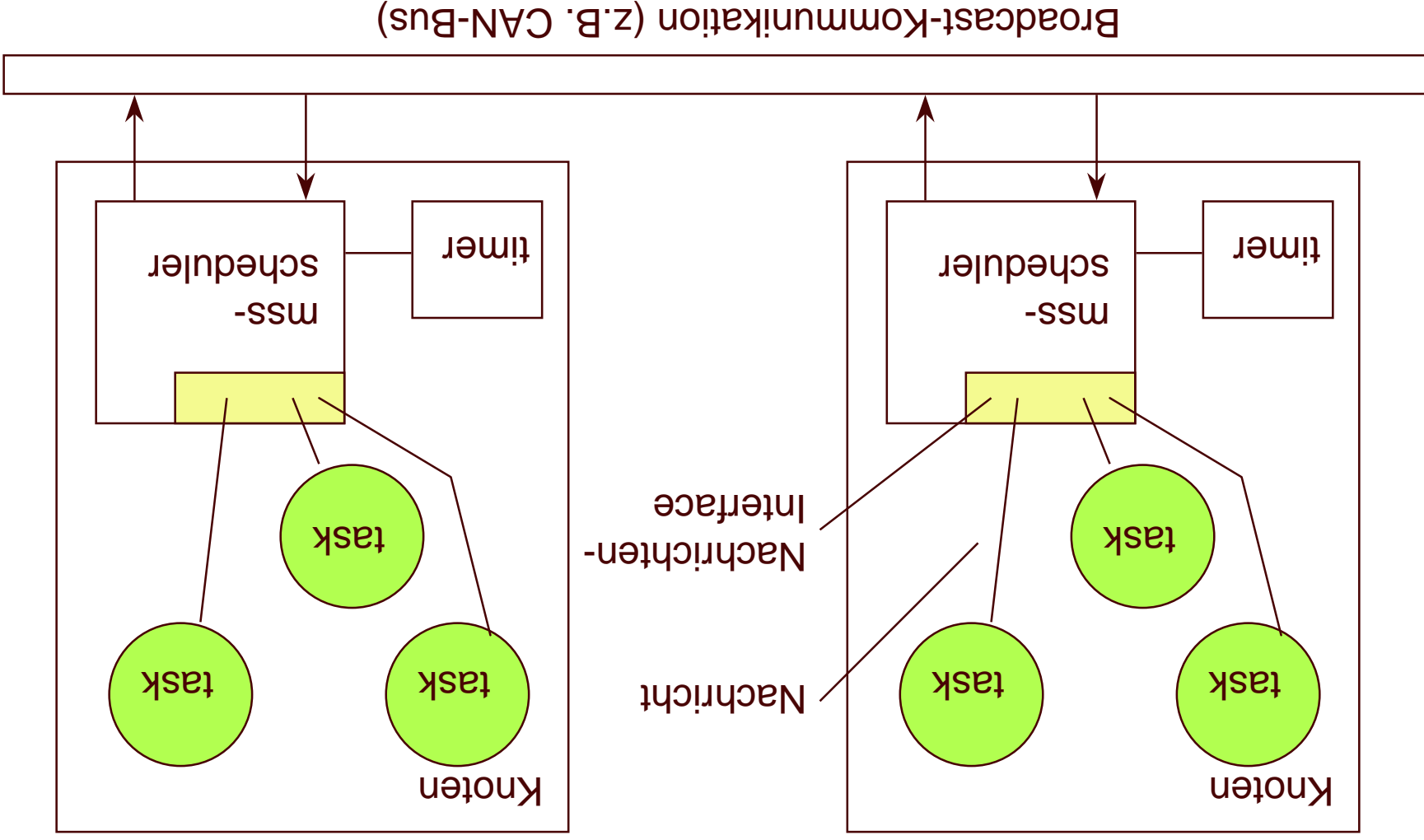
# Message Scheduled System (MSS)

- Ziel:
  - Eingebettete Echtzeitsysteme (Fahrzeuge, Flugzeuge, Automatisierung)
  - Entwicklung einer Architektur mit Unterstützung von Komponierbarkeit in Bezug auf das zeitliche Verhalten
- Idee:
  - globales Scheduling mit lokalem Wissen
  - Mehrstufige Abbildung von Komponierbarkeits- auf Schedulingentscheidungen
- Voraussetzungen:
  - Echtzeitfähige Knoten an einem Echtzeitkommunikationsmedium
  - Globale Prioritäten

## MSS — Arten von Komponenten

- **Tasks**  
erzeugen aus einem Satz von periodischen Eingangsnachrichten einen Satz von Ausgangsnachrichten (mit Deadline)
- **Knoten**  
führt Tasks aus; verfügt über "Nachrichtenmanager", der den Nachrichtentrffic der Tasks verwaltet und über die Nutzung des Kommunikationsmediums entscheidet
- **Kommunikationsmedium**  
echtzeitfähiger Bus, der die Knoten verbindet

# MSS — Architektur



# MSS — Komponenteneigenschaften

Beispiele, nicht vollständig

- Task:
  - WCET (Worst Case Execution Time)
  - Periode (bzw. Minimal Interarrival Time) — entspricht Deadline
  - Eingangsnachrichten mit Perioden und Typen, „Wake-Up-Message“
  - Ausgangsnachrichten mit Perioden und Typen
- Knoten:
  - Zusammengefaßte Parameter der Tasks
- System:
  - Zusammengefaßte Parameter der Knoten

# MSS — Komponierbarkeit

MSS bildet Entscheidungen der Komponierbarkeit auf Schedulingentscheidungen ab:

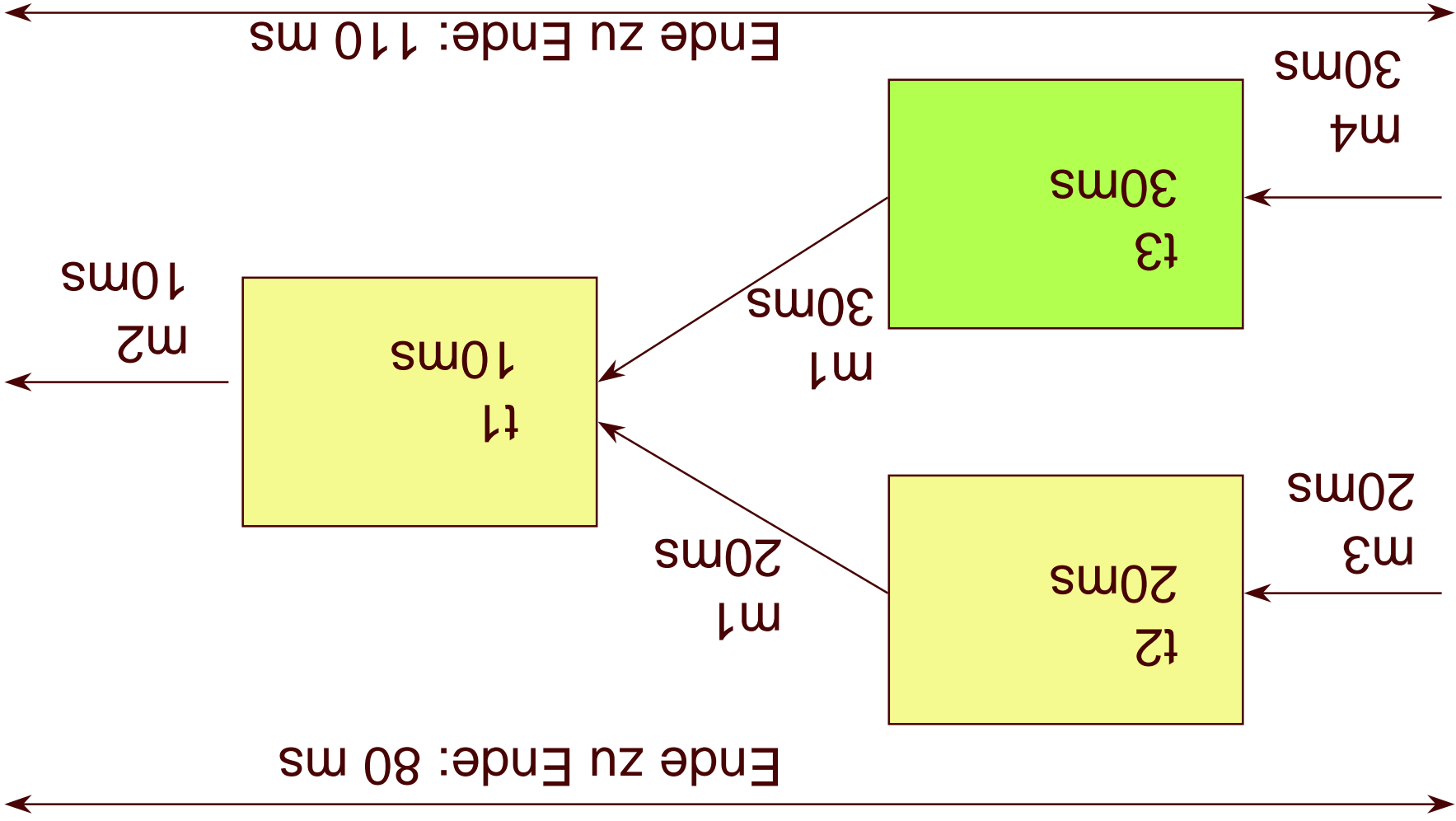
- Lokales Scheduling aller Tasks auf einem Knoten
  - betrachtet die lokalen Ressourcen eines Knoten wie CPU-Zyklen
- Globales Scheduling auf dem Echtzeit-Netzwerk

- zwischen allen Nachrichten verschiedener Typen
  - betrachtet die Nachrichten-Slots auf dem Kommunikationsmedium
- aller Nachrichten gleichen Typs an den gleichen Empfänger
  - betrachtet die Inanspruchnahme eines Empfängers, an den mehrere Sender Nachrichten senden

## MSS — Behauptungen

- Existenz von Schedules auf allen Ebenen:
  - Alle Anforderungen der Komponenten erfüllt
  - Die Berechnung der Schedules ist in konstanter Zeit und unter Nutzung bereits existierendem Wissens (Parameter wie Last) möglich
  - Die Berechnungen setzen auf eingeführten Verfahren der Echtzeit-Forschung auf (RMA, EDF)
  - Bandbreite und Last als Abstraktion
- Systemeigenschaften können aus den Komponenteneigenschaften berechnet werden (beispielsweise End-zu-End-Zeiten)

# MSS — Beispiel



# MSS — Spezifikation

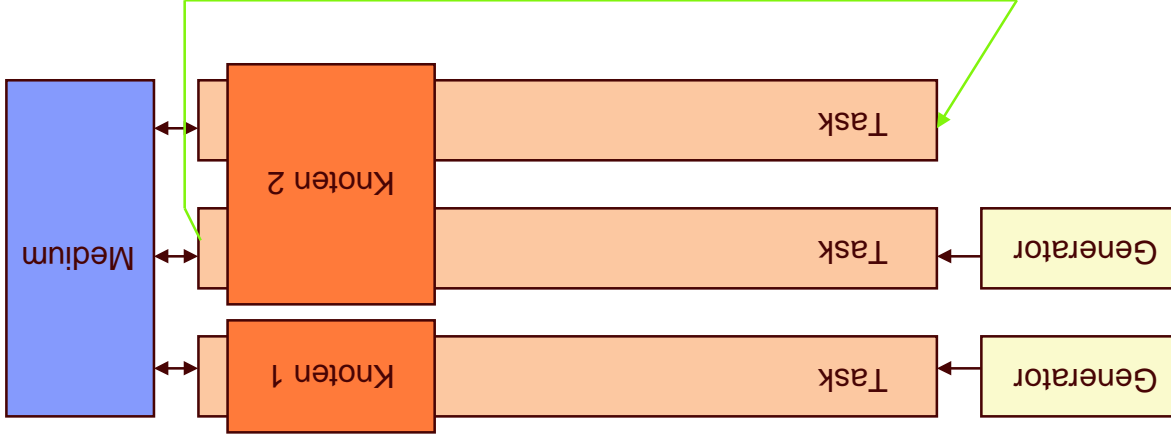
- Bisherige „Spezifikation“ von MSS ist informal:
  - Textuelle Beschreibungen
  - Komplex, aber nur „intuitiv“ vollständig
  - Unmöglichkeit von (automatischen) Analysen auf Basis der Spezifikation
- Ziel: Formale Spezifikation
  - Klar definierte Semantik
  - Automatische Analysen sowie Beweise/Modelchecks (Toolsupport)
  - Ausgangspunkt für Implementation

# MSS — Ansätze zur Spezifikation

- Logikbasierte Ansätze
  - RealTime Logic
  - Temporal Logic of Actions
- automatenbasierte Ansätze
  - Statecharts bzw. Modecharts (StateMate, Modechart-Toolset)
  - timed Automata (HyTech, UPPAAL)
- Petrietzubasierte Ansätze
  - timed Petrietze (INA)

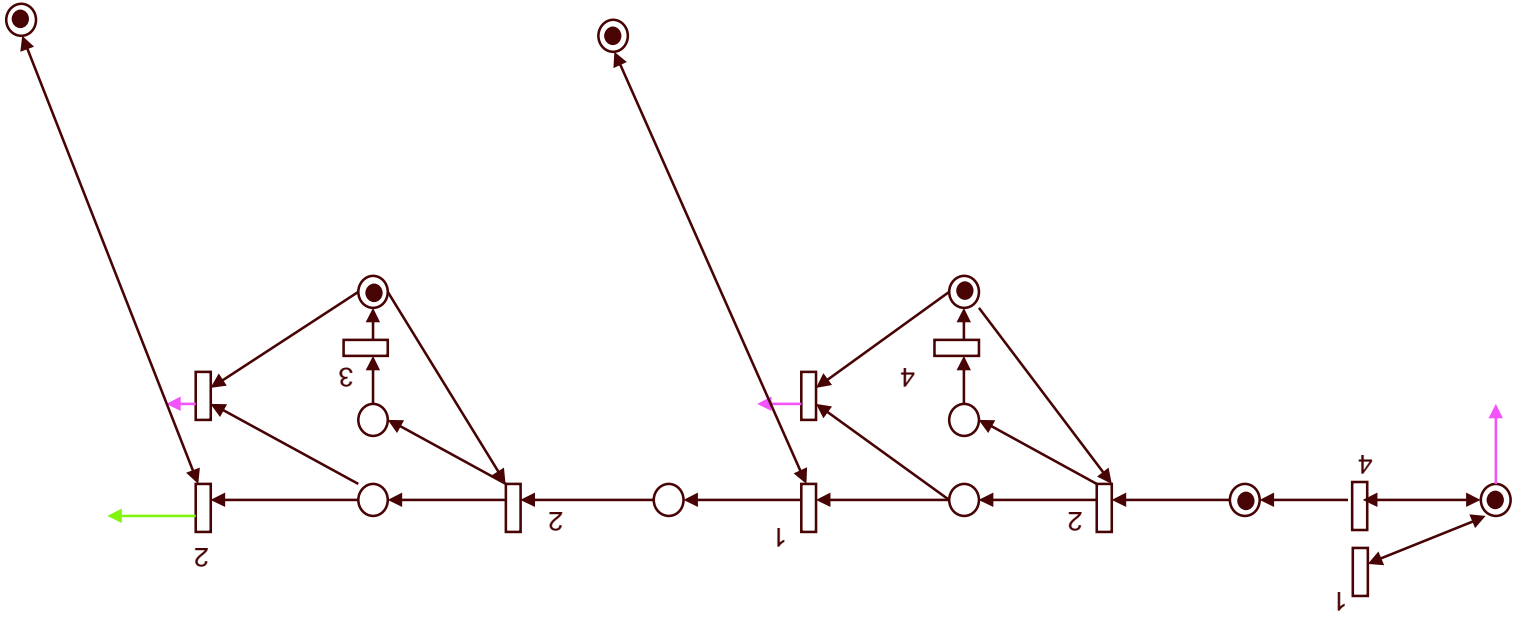
# MSS — Spezifikation mit timed petrinets

- Spezifikation soll System in seinen Teilen „natürlich“ widerspiegeln (u.a. Zeiten, Prioritäten)
- Spezifikation soll „komponierbar“ sein
- Spezifikation für konkretes System soll generierbar sein
- Aussagen für allgemeine Systeme

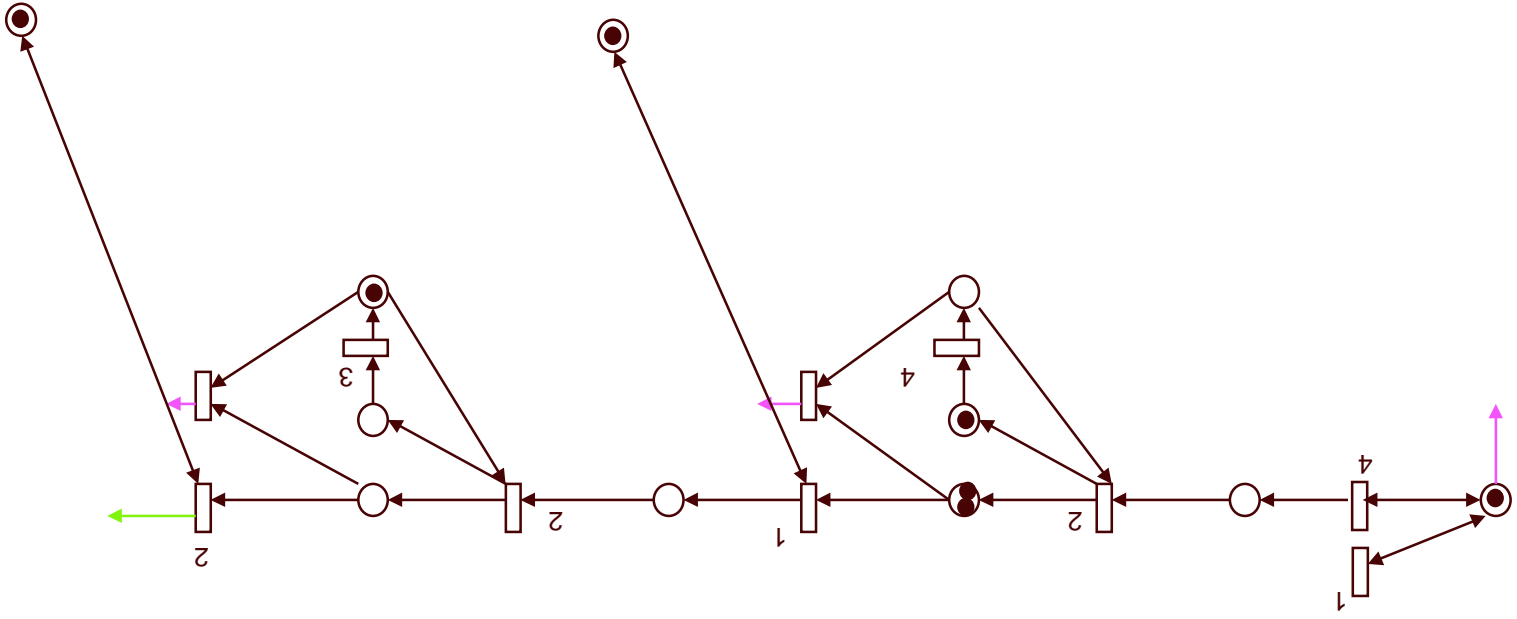




# MSS — Spezifikation mit timed petri nets



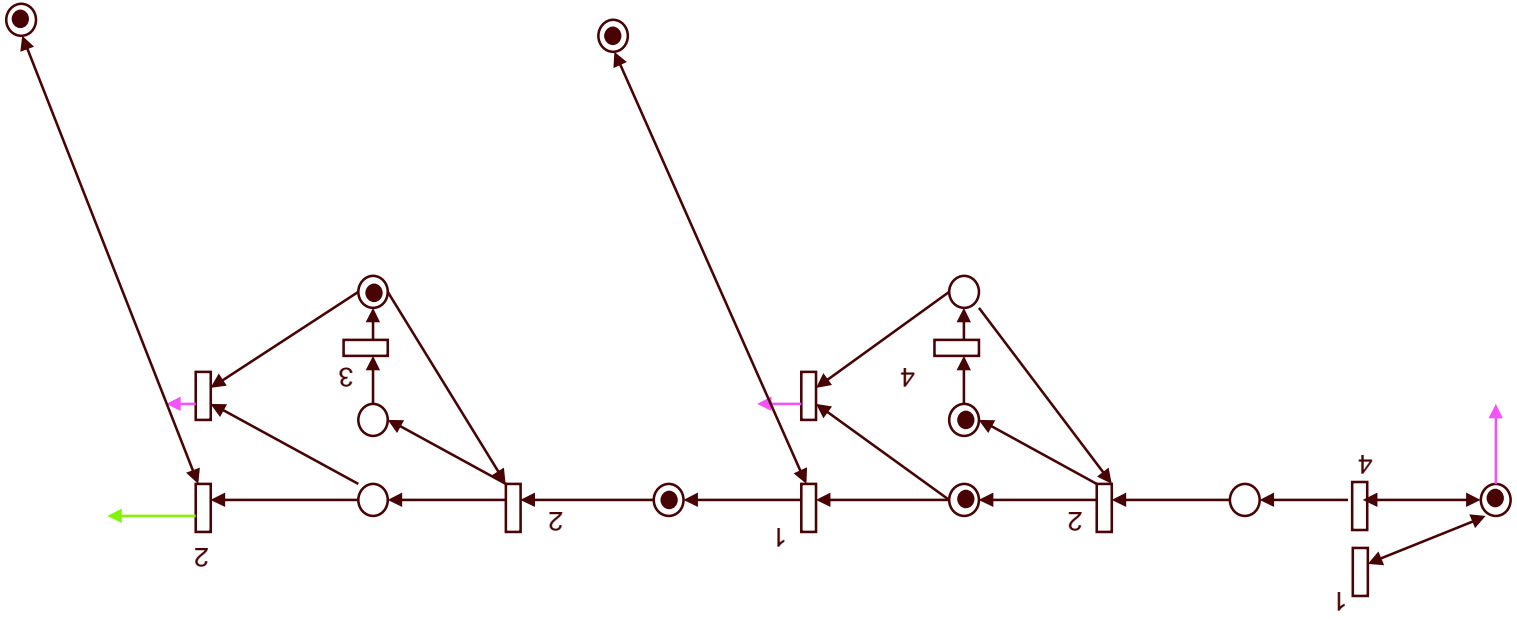
# MSS — Spezifikation mit timed petri nets



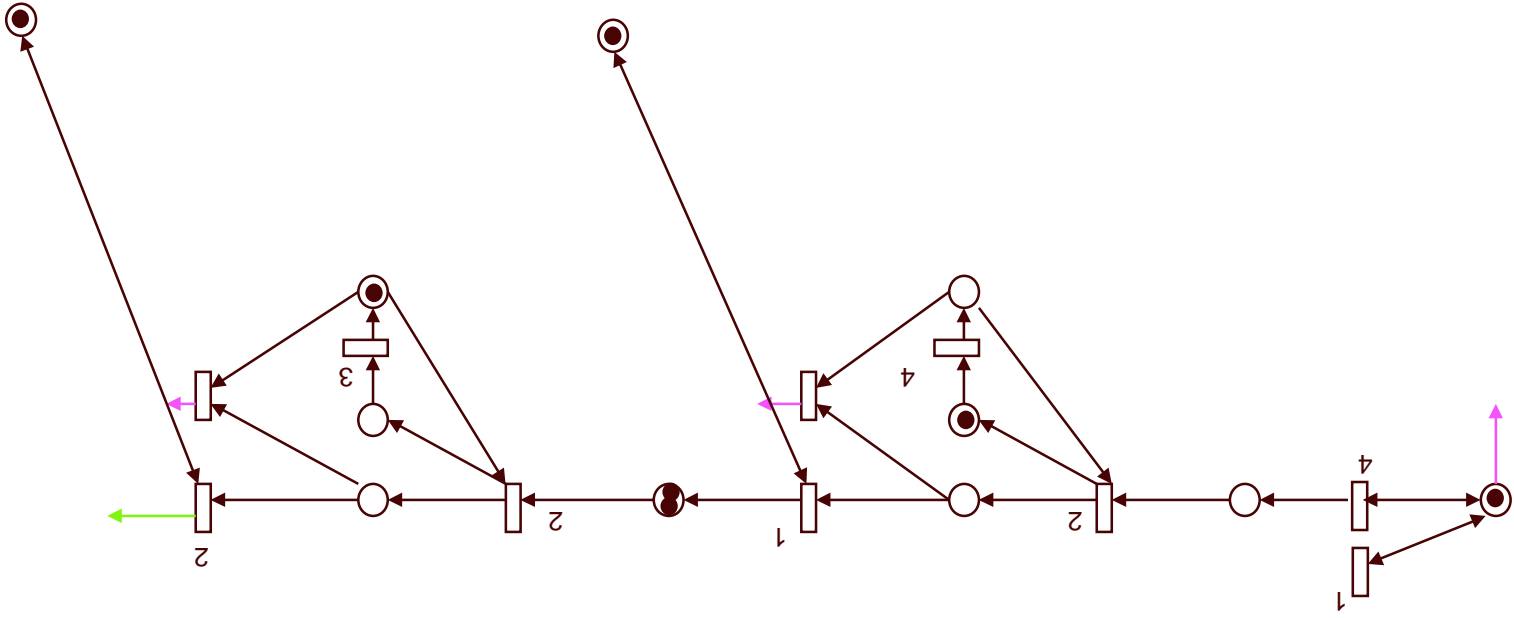




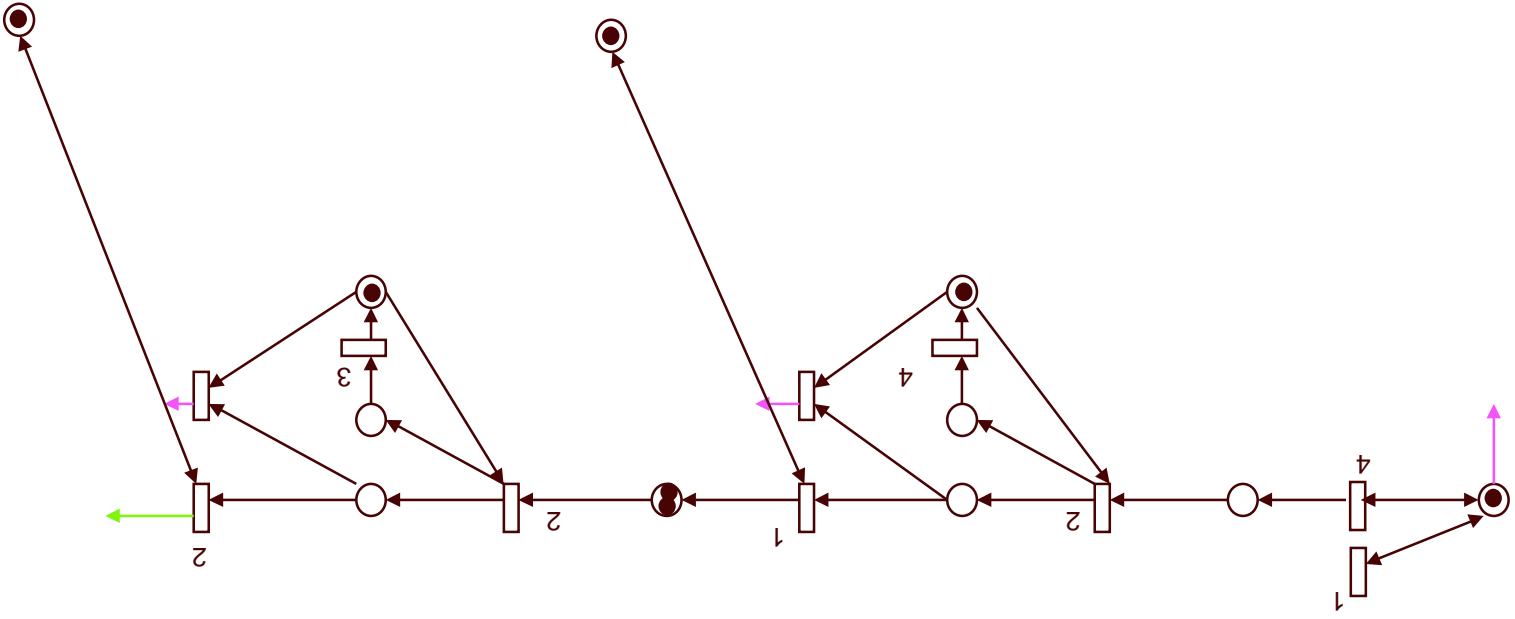
# MSS — Spezifikation mit timed petri nets



# MSS — Spezifikation mit timed petri nets

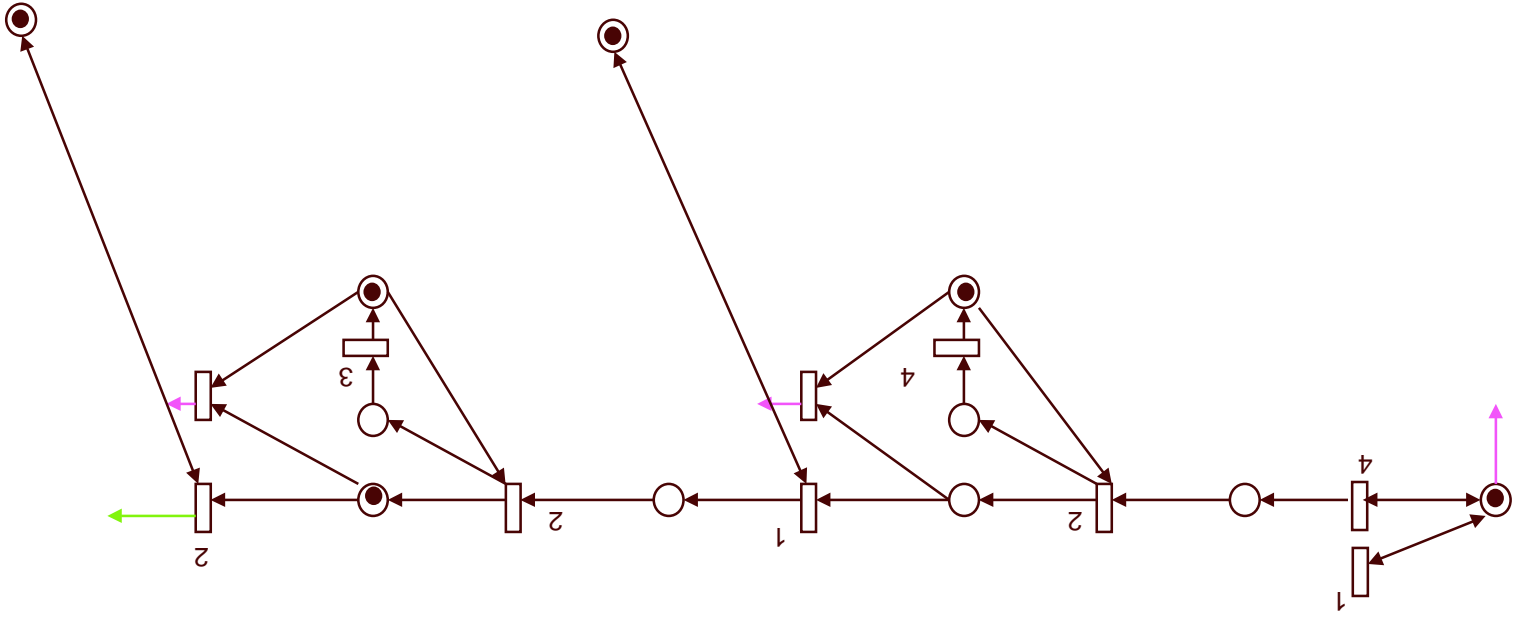


# MSS — Spezifikation mit timed petri nets

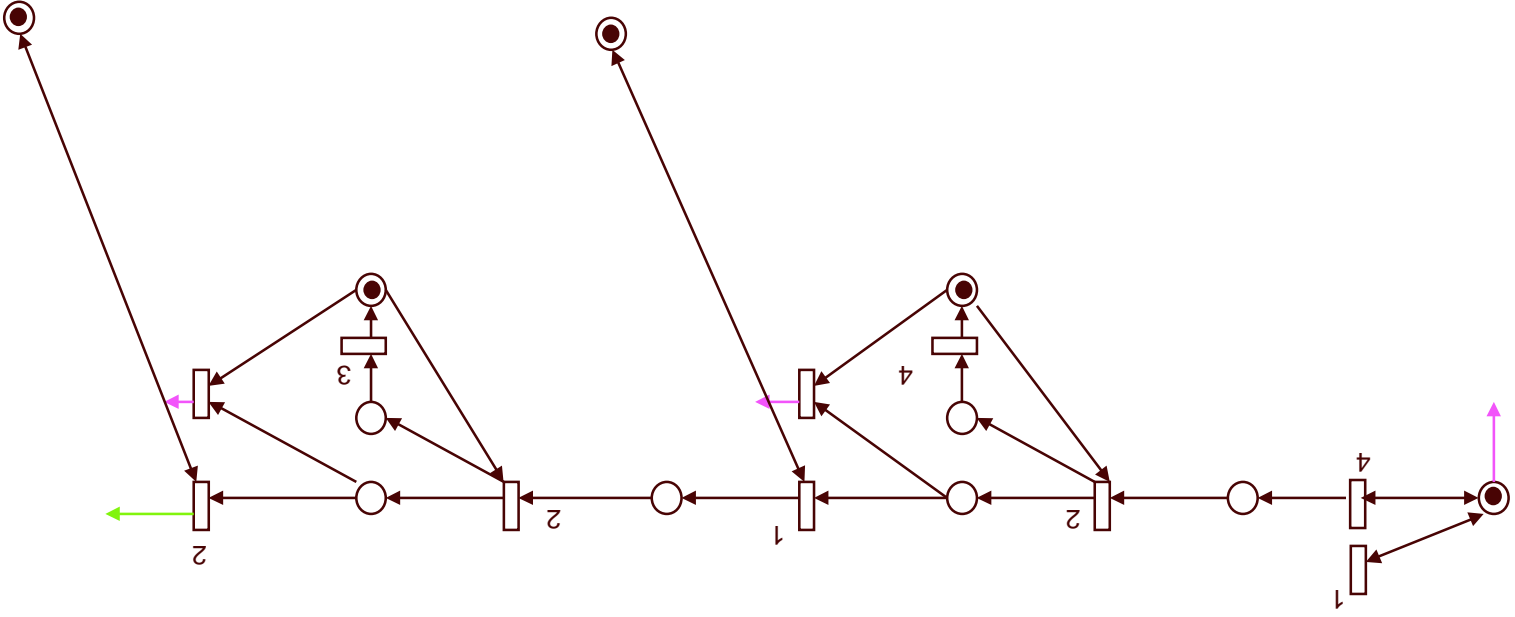




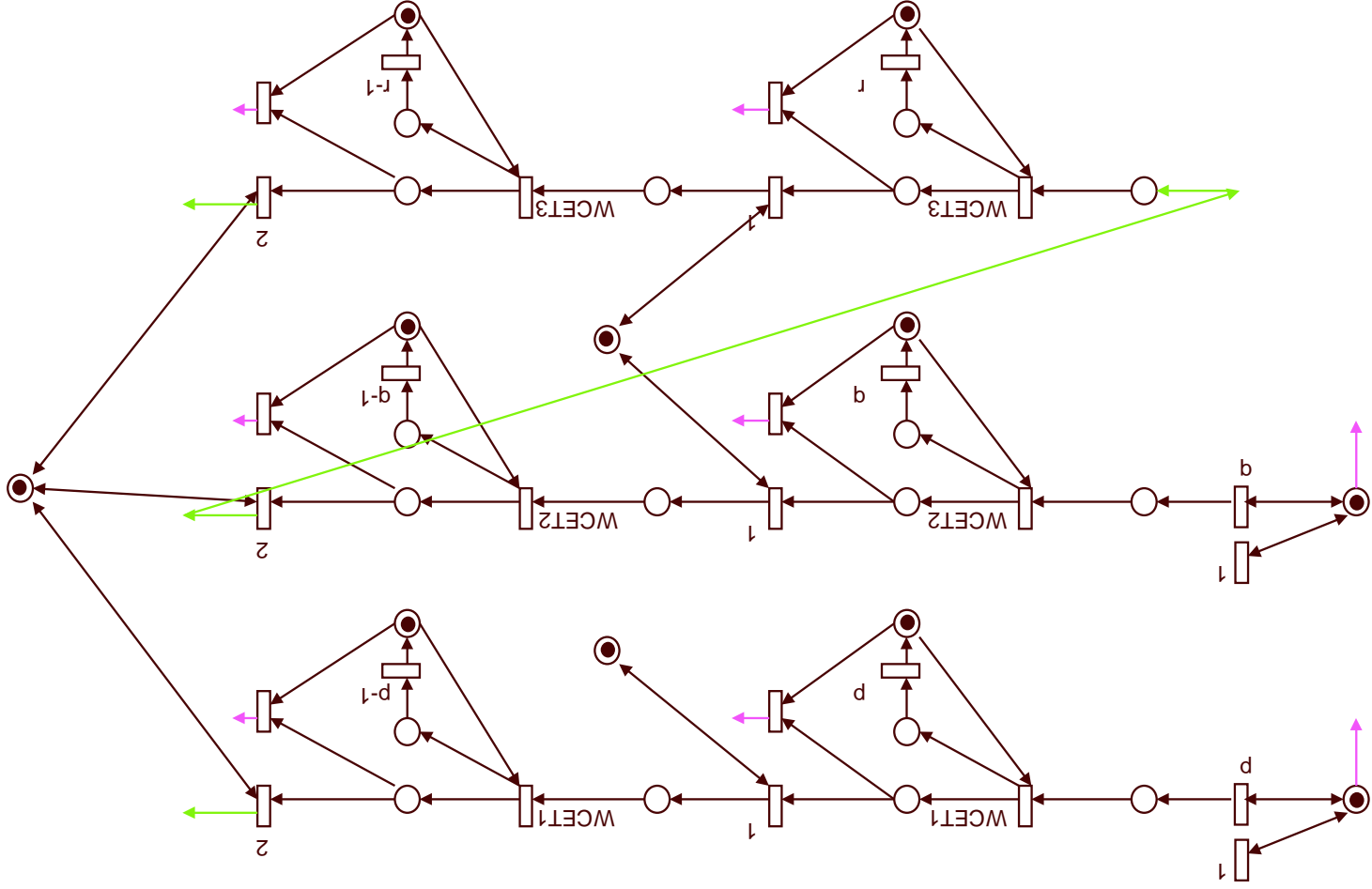
# MSS — Spezifikation mit timed petri nets



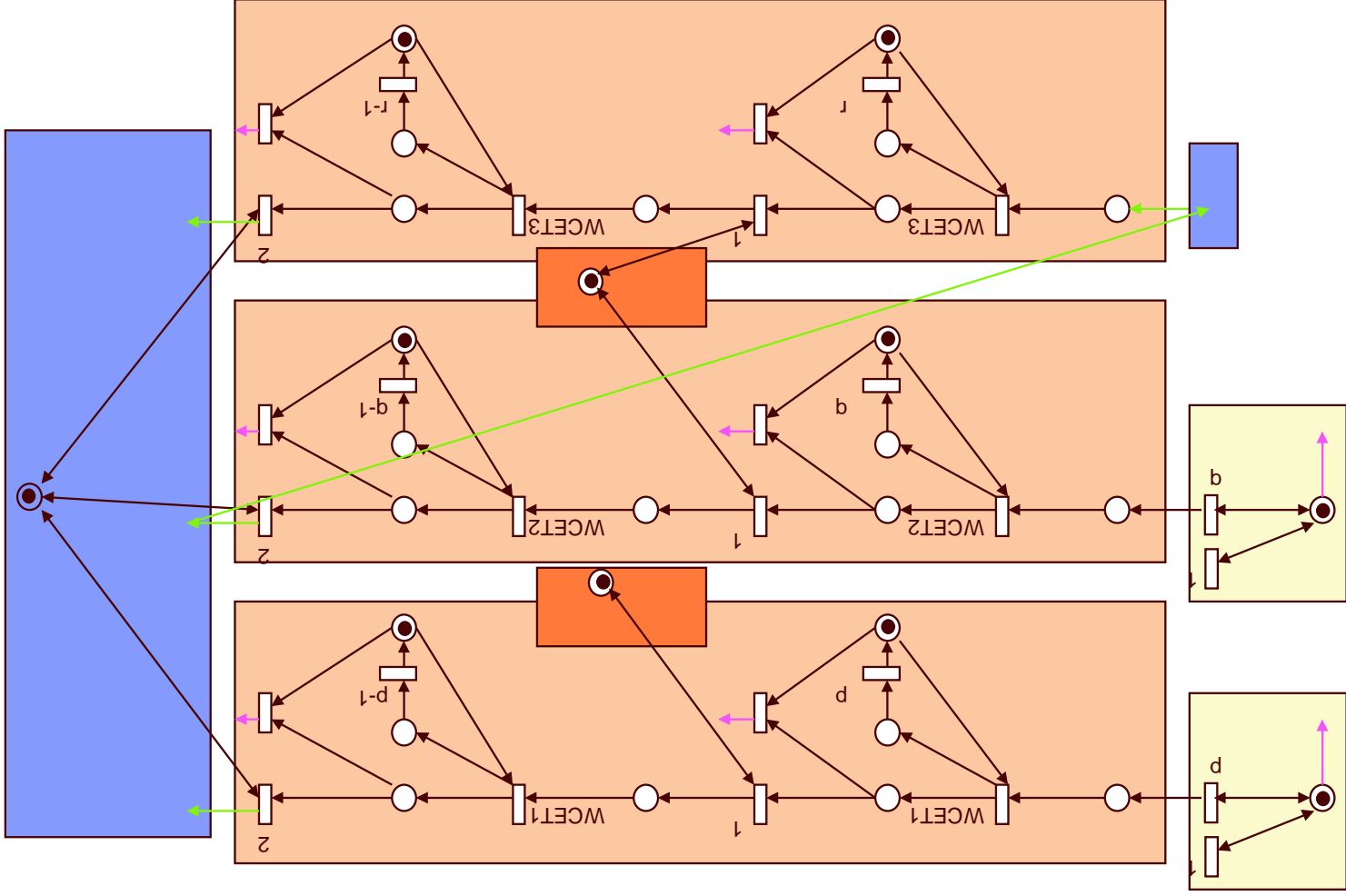
# MSS — Spezifikation mit timed petri nets



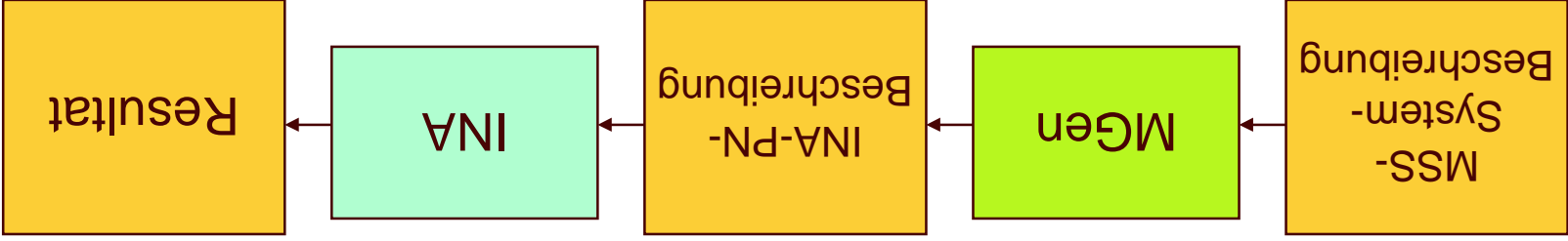
# MSS — Spezifikation mit timed petrinets



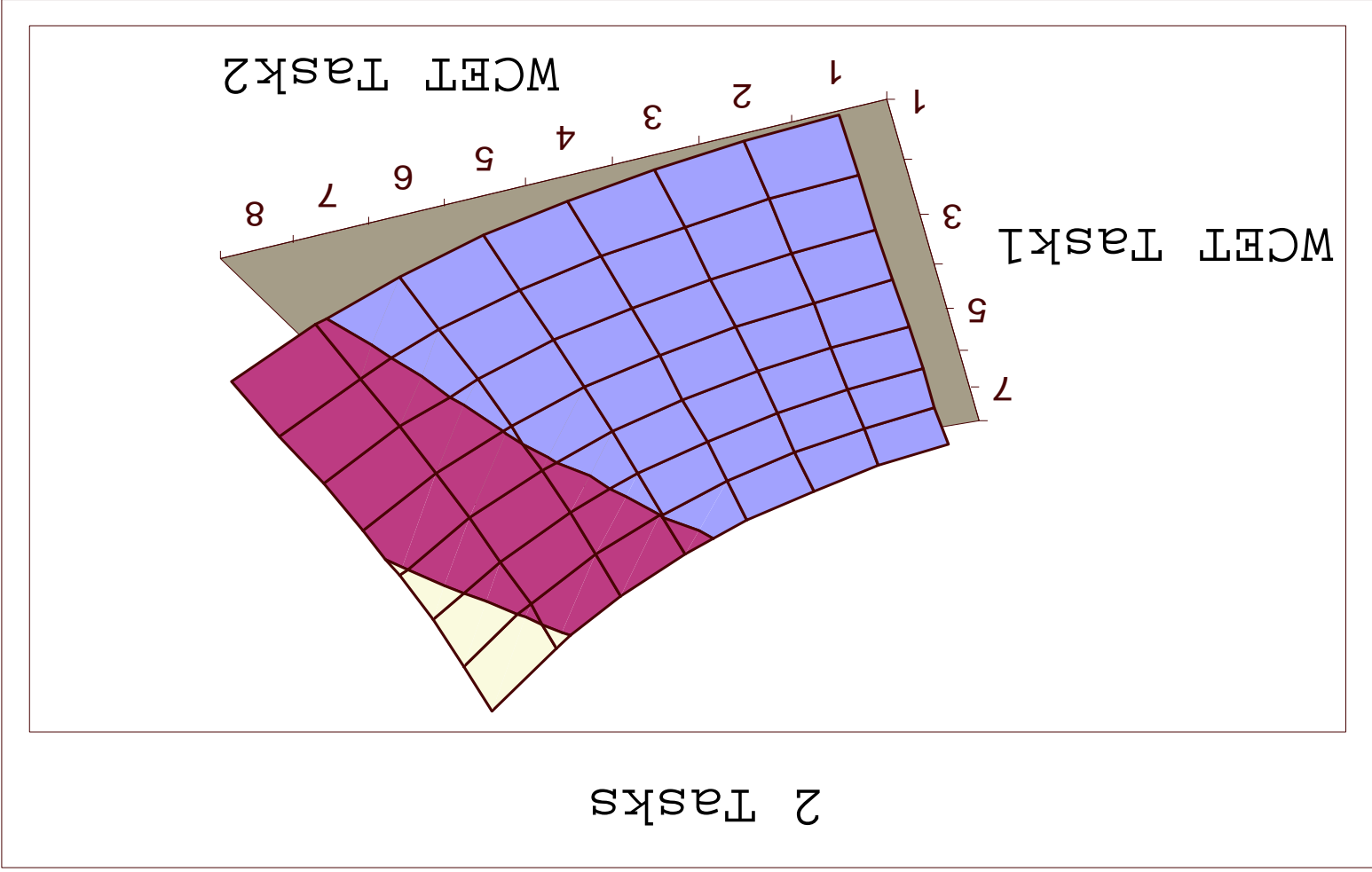
# MSS — Spezifikation mit timed petri nets



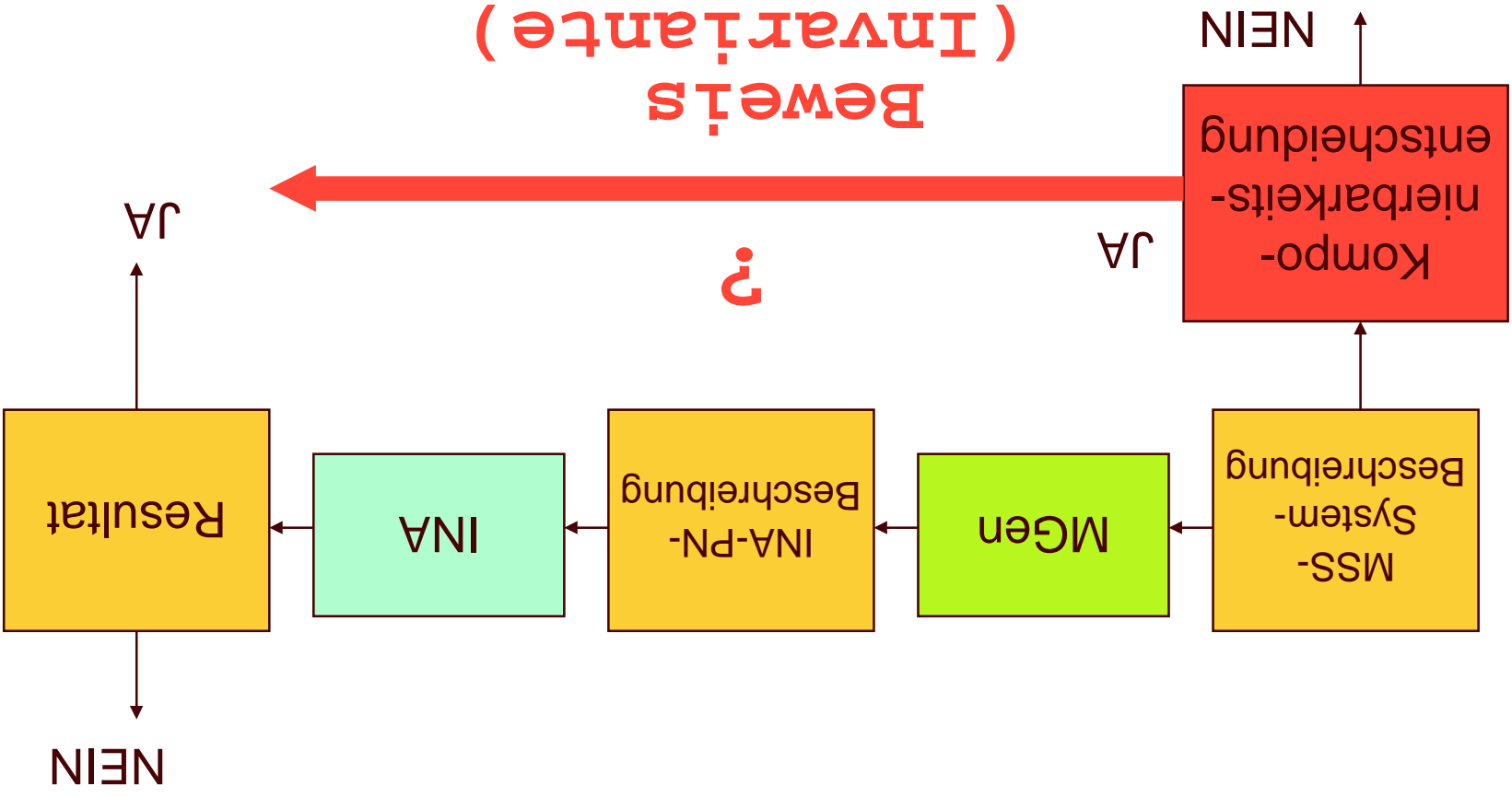
# MSS — Automatische Analyse



# MSS — Komplexität der Analyse



# MSS — Ziel der Analyse



## MSS — Weitere Schritte

- Weiterentwicklung der Beweistechniken
- Verifikation der Behauptungen
- Verallgemeinerung der Komponierbarkeitstechniken
  - Andere Protokolle oder Architekturen
  - Andere Eigenschaften

# Vergleich MSS vs. TTA

## MSS - Vorteile

- einfache Erweiterbarkeit
- unanfällig gegen Störungen auf dem Medium
- geringes globales Wissen

## MSS - Nachteile

- Jitterfreiheit kaum erreichbar
- erreichbare Auslastung niedriger

## TTA - Vorteile

- jitterfreie Ausgaben möglich
- gut geeignet für statische Einsatzgebiete
- hohe Auslastung erzielbar

## TTA - Nachteile

- Erweiterbarkeit aufwendig
- anfällig gegen Störungen auf dem Medium
- viel globales Wissen