Computing in vivo
Vorlesung Bioinformatik und Informatik
Wintersemester 2009/2010

Thomas Hinze

LS Bioinformatik
Friedrich-Schiller-Universität Jena

thomas.hinze@uni-jena.de
http://users.minet.uni-jena.de/∼hinze

3. Membran- und zellbasierte Computingmodelle

Auswahl von Facetten „Computing in vivo“

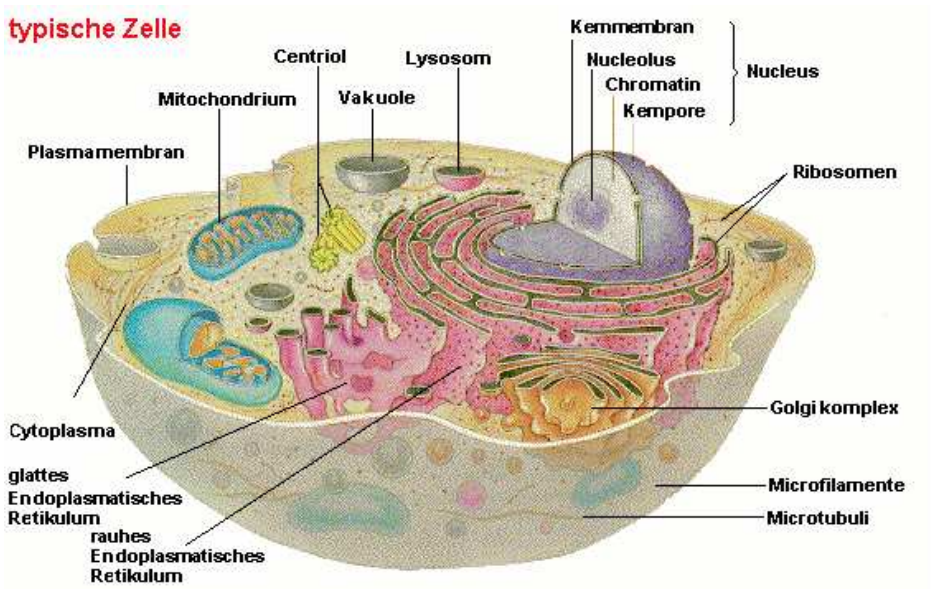
Mischung aus theorie- und praxisorientierten Inhalten

1. **Biologische Reaktionsnetzwerke als Recheneinheiten** (chemische, metabolische und Zellsignal-Netzwerke)
2. **Genetische Schaltkreise** (konstruiert aus Genregulationsnetzwerken)
3. **Membran- und zellbasierte Computingmodelle**
4. **Neuronale Netze** – Aufbau, Training, Anwendungen
5. **Gene Assembly** – Selbstorganisation nach Regeln
6. **Evolutionäres Computing** – eine universelle Heuristik
7. **Amorphes Computing** – komplexe Teamlösungen

„Rechnen im Raum“

Welche Vorteile bieten lebende Zellen in ihrer Gesamtheit für die Informationsverarbeitung?

- Kompartimentierung (Separation in gekoppelte Reaktionsräume)
- Strukturdynamik (gezielte Veränderung von Raumstrukturen)
- molekulare Adressierung (Messenger, „Befehlsmoleküle“)
- explizite Darstellung einzelner Moleküle und ihrer kombinatorischen Komplexbildung
- Einzelmolekülverfolgung (Tracing in Raum und Zeit)



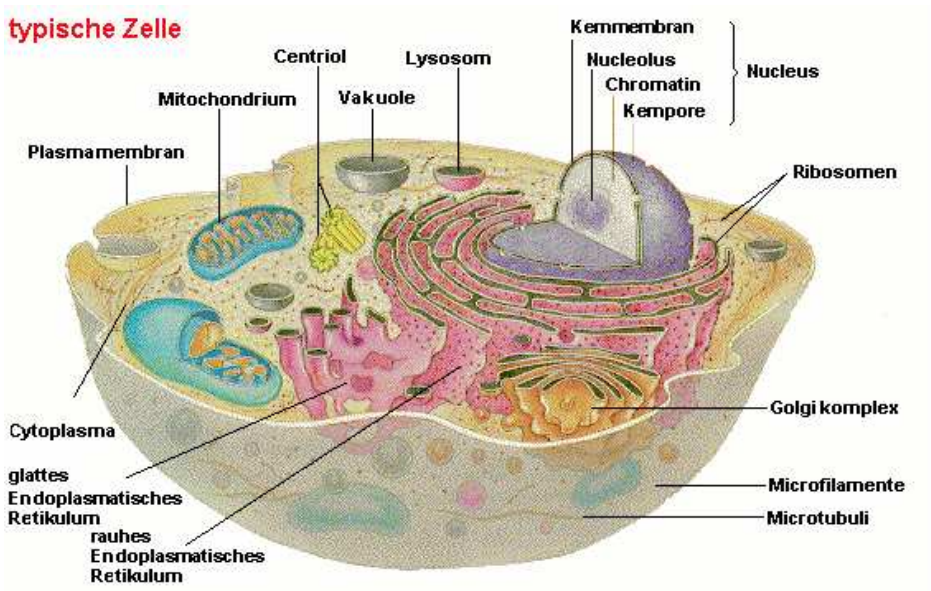
www.zum.de

- regelbasierte Modelle/Simulationen (Def. lokaler Interaktionen)
- direkter Zugang zu Berechnungsmodellen der Theor. Informatik

„Rechnen im Raum“

Welche Vorteile bieten lebende Zellen in ihrer Gesamtheit für die Informationsverarbeitung?

- Kompartimentierung (Separation in gekoppelte Reaktionsräume)
- Strukturdynamik (gezielte Veränderung von Raumstrukturen)
- molekulare Adressierung (Messenger, „Befehlsmoleküle“)
- explizite Darstellung einzelner Moleküle und ihrer kombinatorischen Komplexbildung
- Einzelmolekülverfolgung (Tracing in Raum und Zeit)
- regelbasierte Modelle/Simulationen (Def. lokaler Interaktionen)
- direkter Zugang zu Berechnungsmodellen der Theor. Informatik

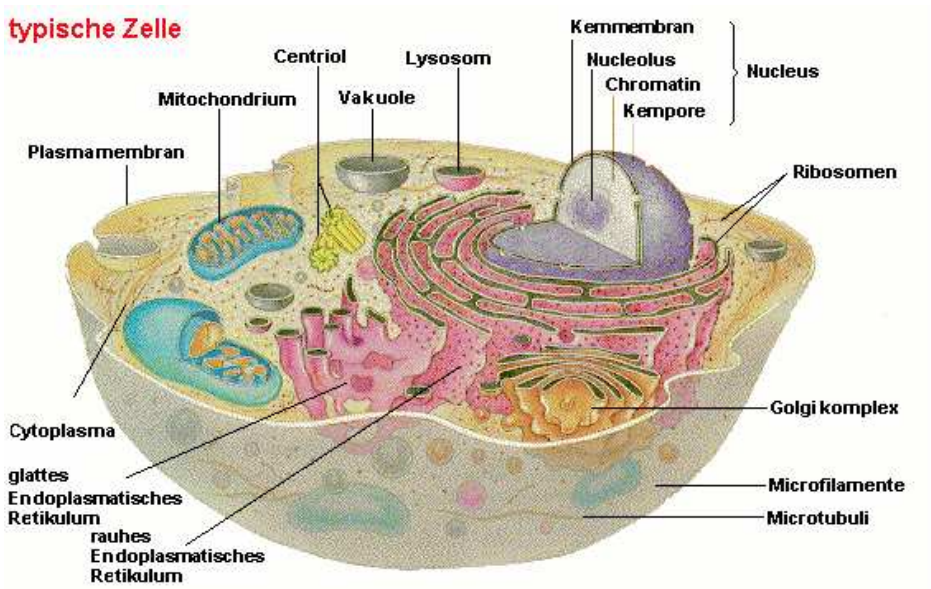


www.zum.de

„Rechnen im Raum“

Welche Vorteile bieten lebende Zellen in ihrer Gesamtheit für die Informationsverarbeitung?

- Kompartimentierung (Separation in gekoppelte Reaktionsräume)
- Strukturdynamik (gezielte Veränderung von Raumstrukturen)
- molekulare Adressierung (Messenger, „Befehlsmoleküle“)
- explizite Darstellung einzelner Moleküle und ihrer kombinatorischen Komplexbildung
- Einzelmolekülverfolgung (Tracing in Raum und Zeit)
- regelbasierte Modelle/Simulationen (Def. lokaler Interaktionen)
- direkter Zugang zu Berechnungsmodellen der Theor. Informatik

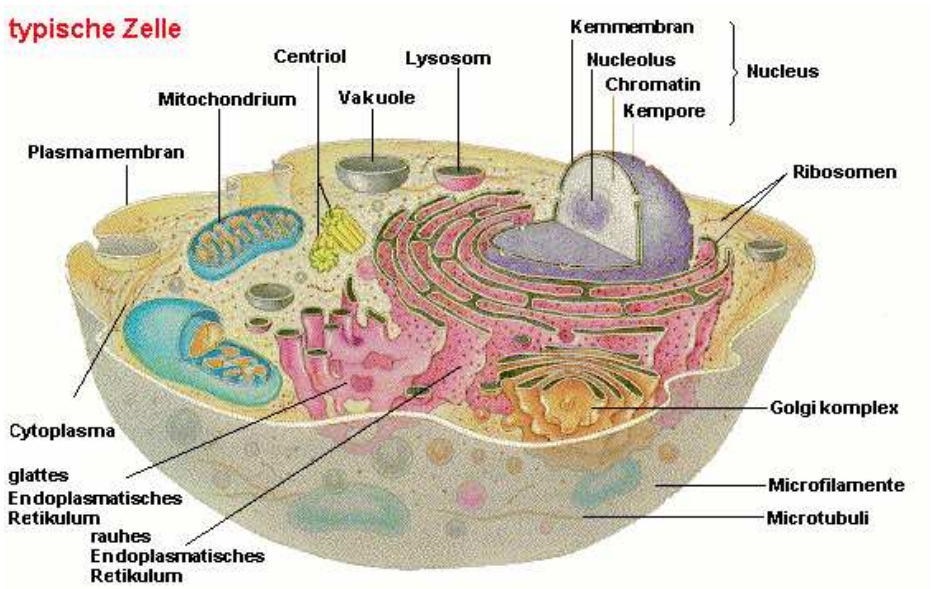


www.zum.de

„Rechnen im Raum“

Welche Vorteile bieten lebende Zellen in ihrer Gesamtheit für die Informationsverarbeitung?

- Kompartimentierung (Separation in gekoppelte Reaktionsräume)
- Strukturdynamik (gezielte Veränderung von Raumstrukturen)
- molekulare Adressierung (Messenger, „Befehlsmoleküle“)
- explizite Darstellung einzelner Moleküle und ihrer kombinatorischen Komplexbildung
- Einzelmolekülverfolgung (Tracing in Raum und Zeit)
- regelbasierte Modelle/Simulationen (Def. lokaler Interaktionen)
- direkter Zugang zu Berechnungsmodellen der Theor. Informatik

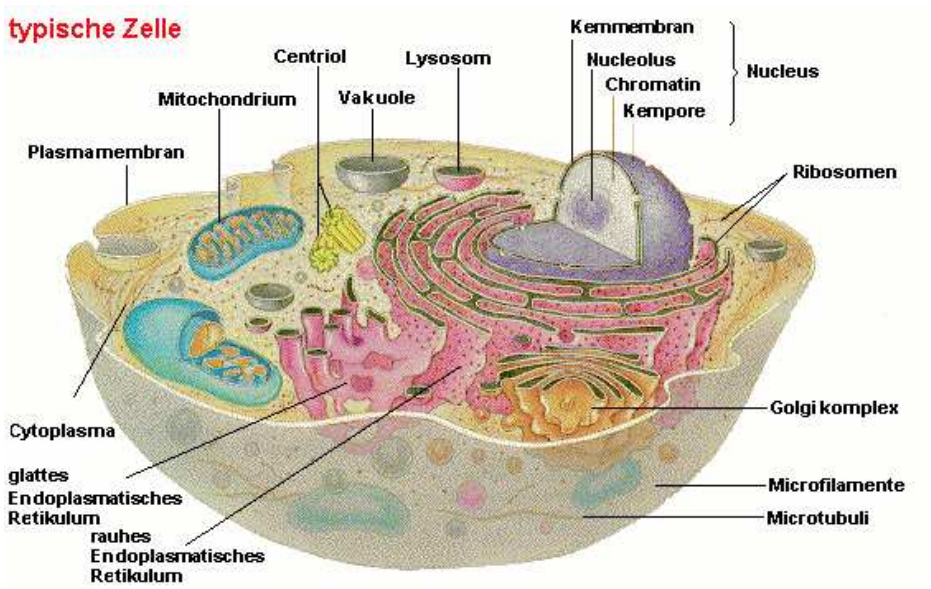


www.zum.de

„Rechnen im Raum“

Welche Vorteile bieten lebende Zellen in ihrer Gesamtheit für die Informationsverarbeitung?

- Kompartimentierung (Separation in gekoppelte Reaktionsräume)
- Strukturdynamik (gezielte Veränderung von Raumstrukturen)
- molekulare Adressierung (Messenger, „Befehlsmoleküle“)
- explizite Darstellung einzelner Moleküle und ihrer kombinatorischen Komplexbildung
- Einzelmolekülverfolgung (Tracing in Raum und Zeit)
- regelbasierte Modelle/Simulationen (Def. lokaler Interaktionen)
- direkter Zugang zu Berechnungsmodellen der Theor. Informatik

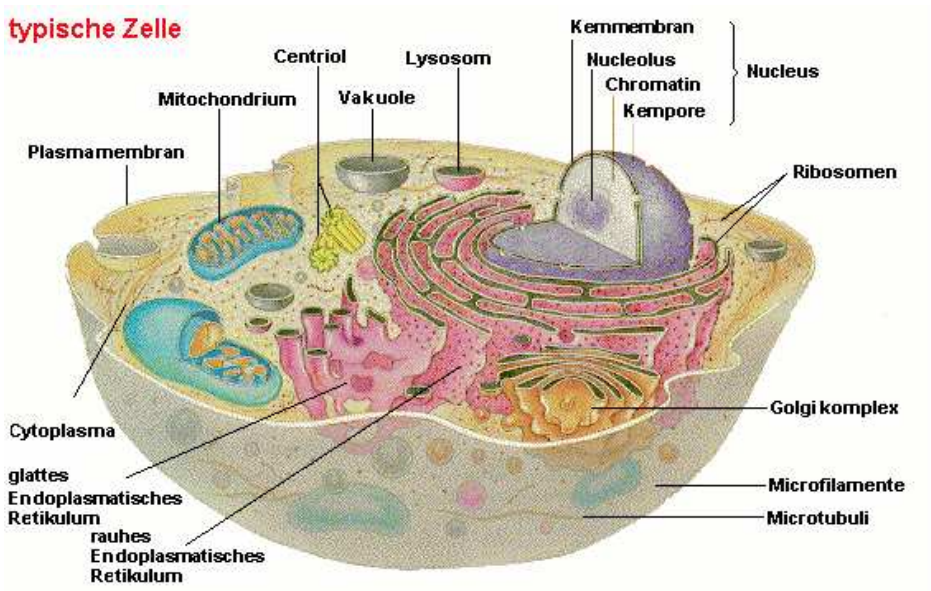


www.zum.de

„Rechnen im Raum“

Welche Vorteile bieten lebende Zellen in ihrer Gesamtheit für die Informationsverarbeitung?

- Kompartimentierung (Separation in gekoppelte Reaktionsräume)
- Strukturdynamik (gezielte Veränderung von Raumstrukturen)
- molekulare Adressierung (Messenger, „Befehlsmoleküle“)
- explizite Darstellung einzelner Moleküle und ihrer kombinatorischen Komplexbildung
- Einzelmolekülverfolgung (Tracing in Raum und Zeit)



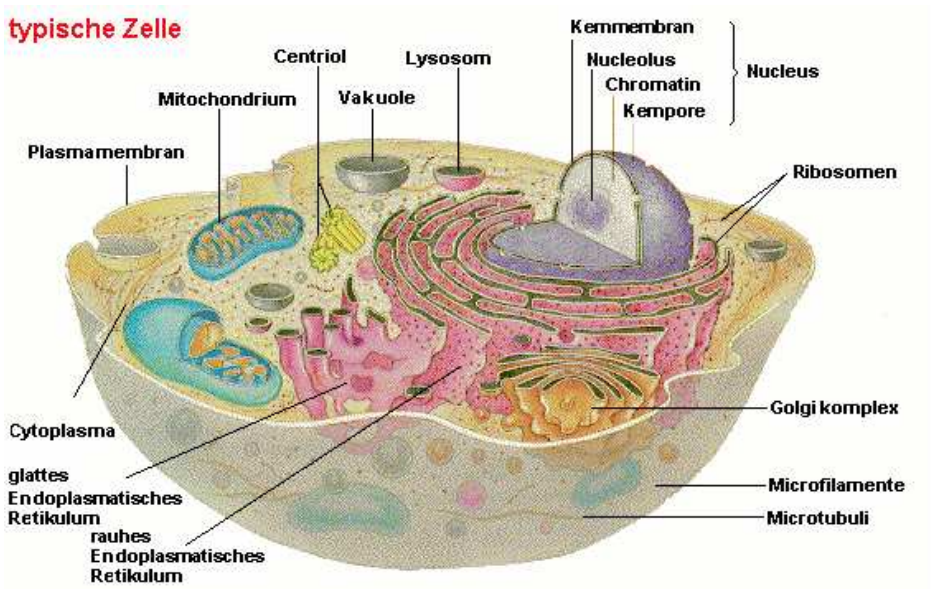
www.zum.de

- regelbasierte Modelle/Simulationen (Def. lokaler Interaktionen)
- direkter Zugang zu Berechnungsmodellen der Theor. Informatik

„Rechnen im Raum“

Welche Vorteile bieten lebende Zellen in ihrer Gesamtheit für die Informationsverarbeitung?

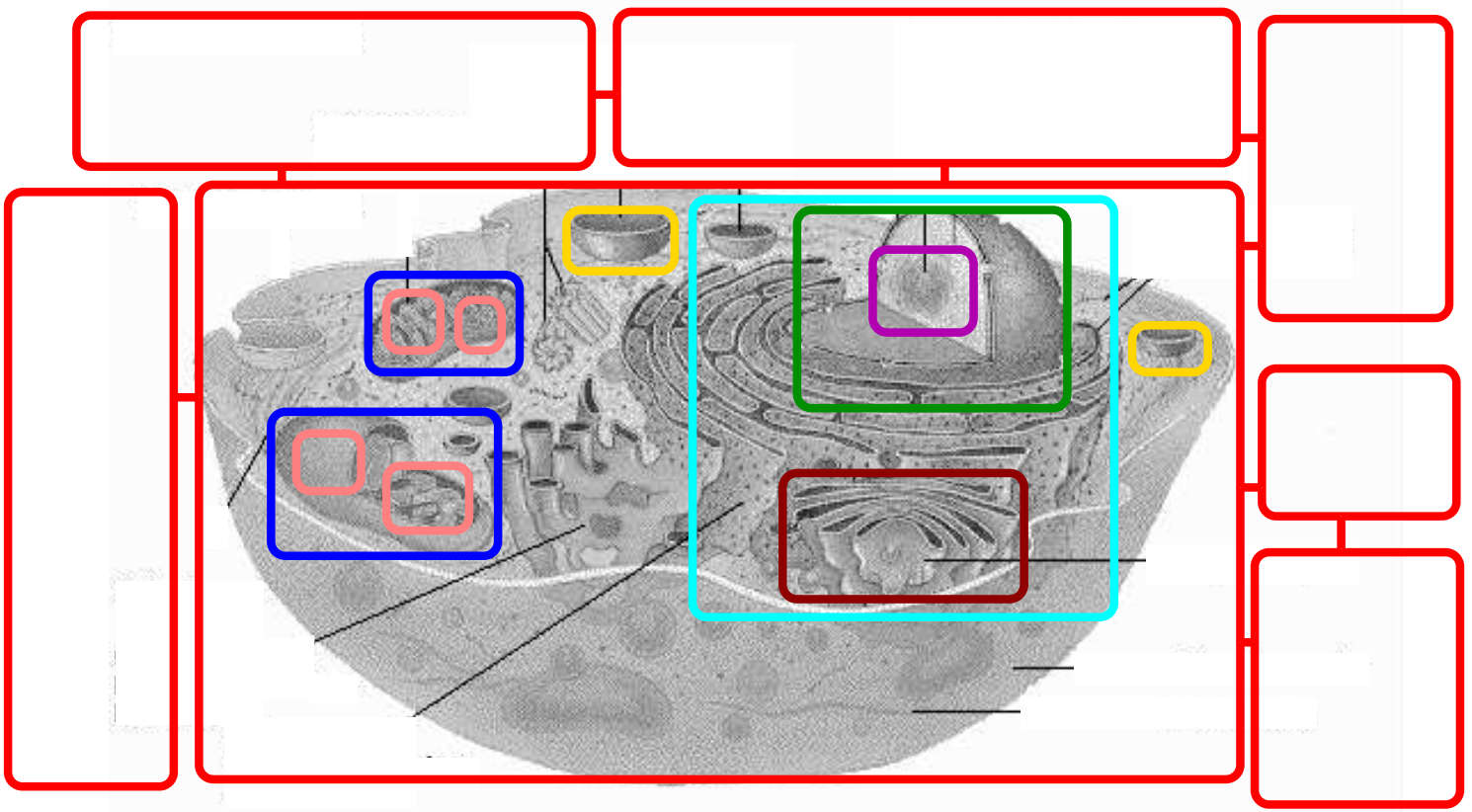
- Kompartimentierung (Separation in gekoppelte Reaktionsräume)
- Strukturdynamik (gezielte Veränderung von Raumstrukturen)
- molekulare Adressierung (Messenger, „Befehlsmoleküle“)
- explizite Darstellung einzelner Moleküle und ihrer kombinatorischen Komplexbildung
- Einzelmolekülverfolgung (Tracing in Raum und Zeit)



www.zum.de

- regelbasierte Modelle/Simulationen (Def. lokaler Interaktionen)
- direkter Zugang zu Berechnungsmodellen der Theor. Informatik

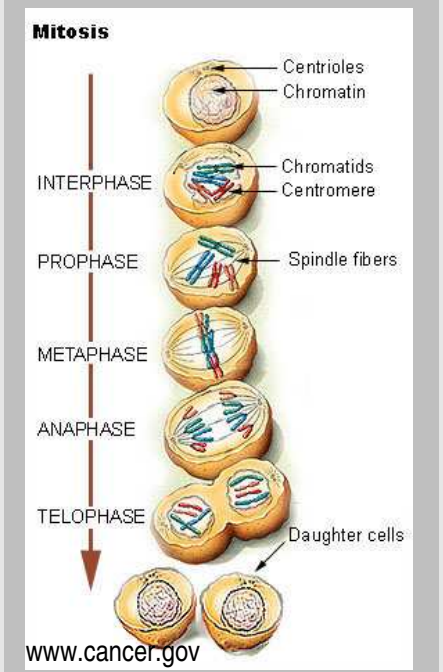
Kompartimentierung: eine abstrakte Sicht



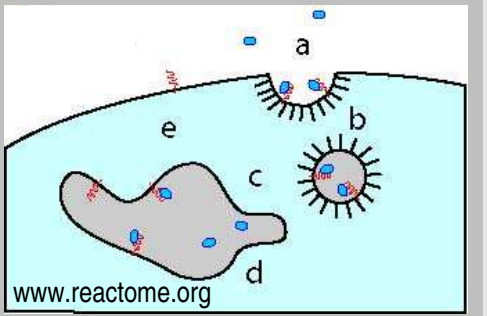
$\Longrightarrow$ verschachtelte und vernetzte Reaktionsräume

$\Longrightarrow$ umschlossen von rezeptorbesetzten Membranen

Dynamik in räumlichen Zellstrukturen



division

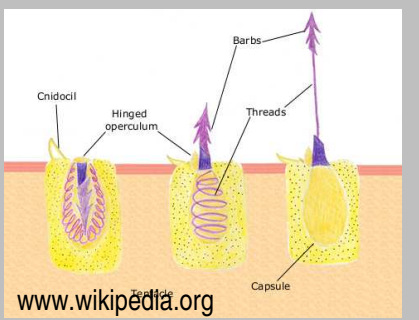


endocytosis

merge

exocytosis

separation



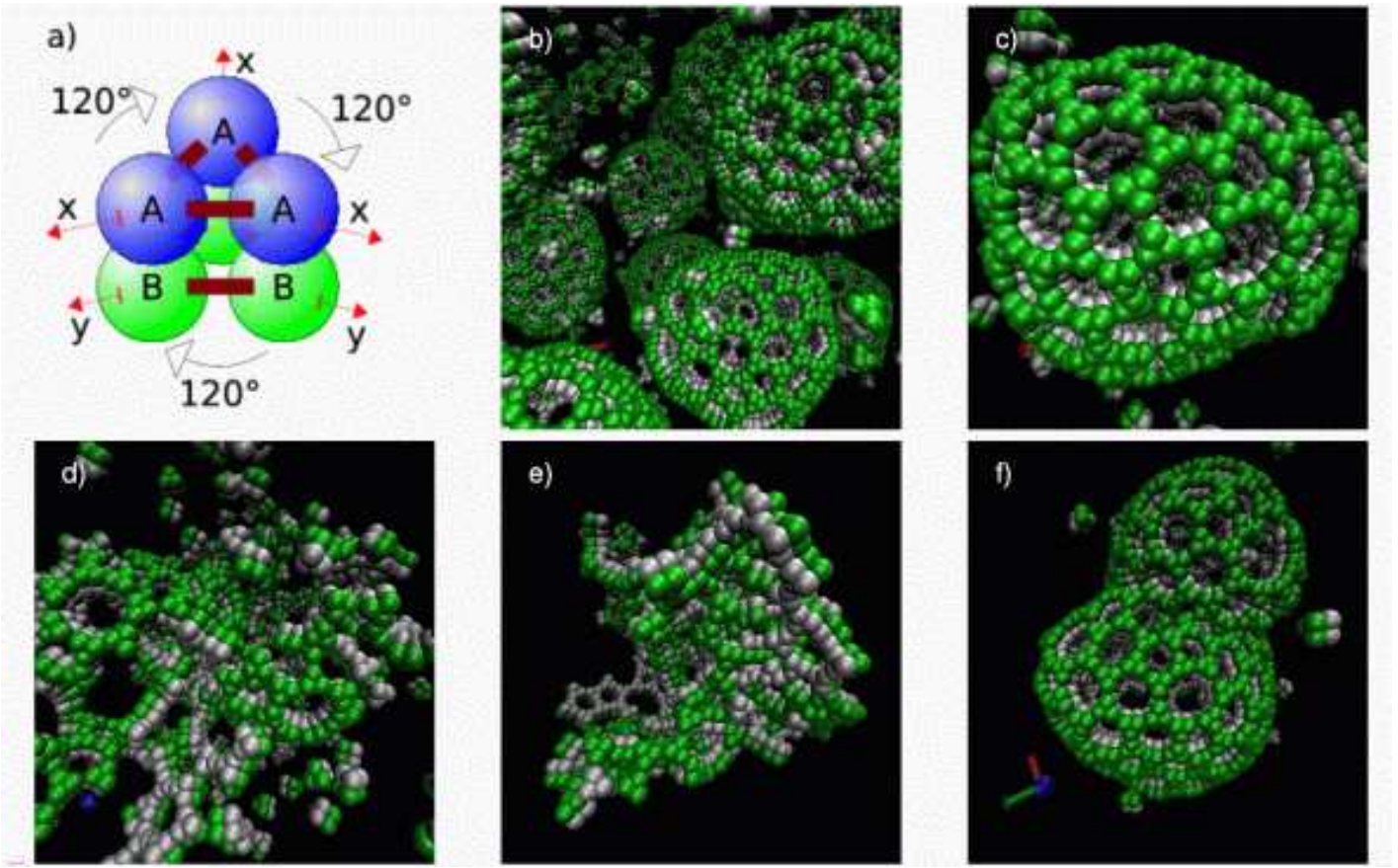
gemmation

creation

dissolution

$\Longrightarrow$ Plastizität durch triggerbare Reaktionsketten

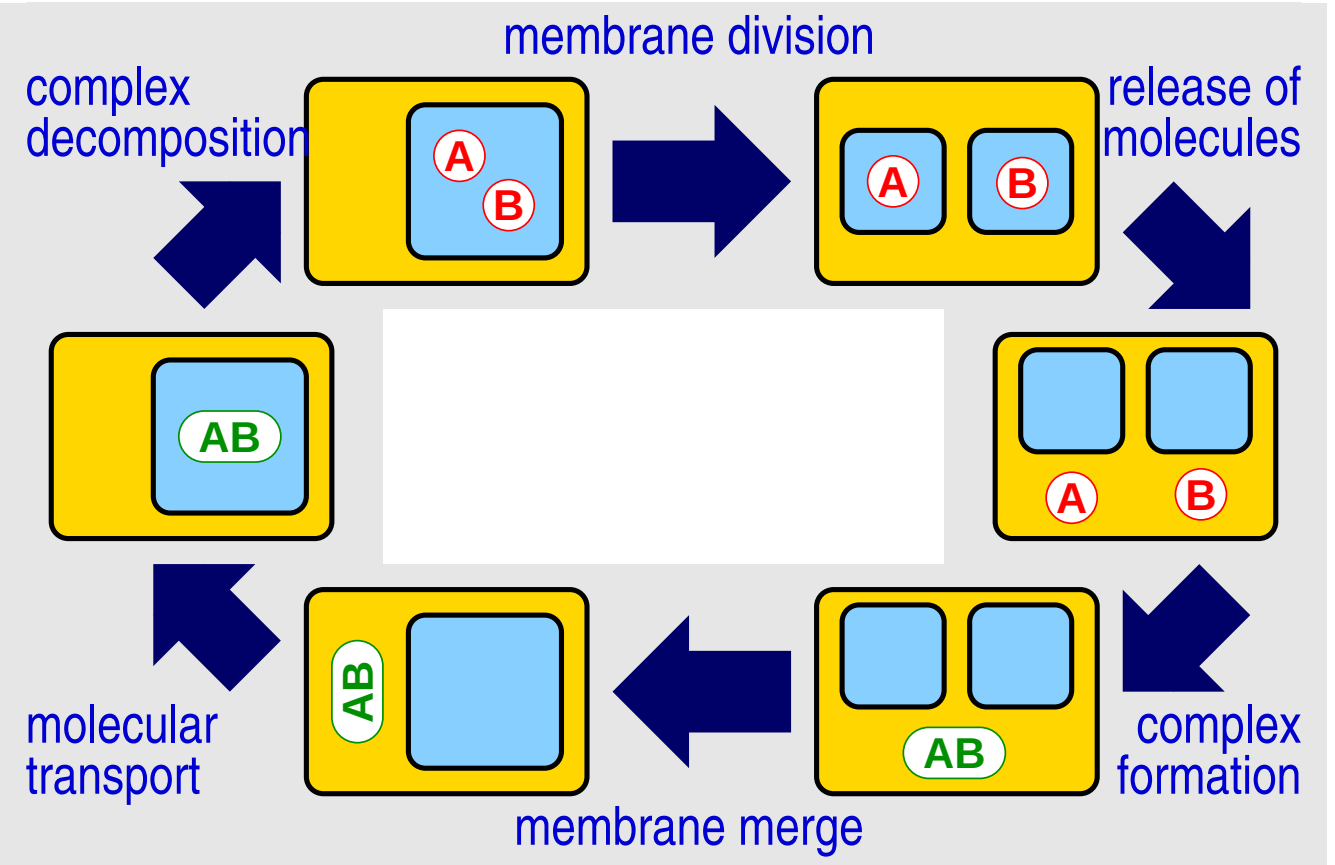
Dynamik in molekularen Raumstrukturen



G. Grünert, B. Ibrahim, T. Lenser, M. Lohel, T. Hinze, P. Dittrich: Rule-based spatial modeling with diffusing, geometrically constrained molecules, accepted for publication in BMC Bioinformatics, 2010

$\Longrightarrow$ Zusammensetzen komplexer Makromoleküle aus monomeren Einheiten und Liganden

Verzahnte Strukturdynamik im Gesamtsystemverhalten



$\Longrightarrow$ mikroskopisch-lokale Interaktionsregeln bewirken makroskopisch komplexes Verhalten

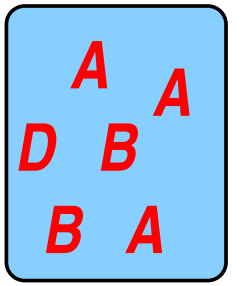
These

Universelles zellbasiertes Computing $\subseteq$ Strukturdynamik auf molekularer bis makroskopischer Ebene

Multimenge: Molekularer Inhalt einer Membran

Beispiel

$$\mathcal{F} = \{(A,3),(B,2),(C,0),(D,1)\}$$
$$\text{supp}(\mathcal{F}) = \{A,B,D\}$$
$$\text{card}(\mathcal{F}) = 6$$



Inhalt einer fiktiven Membran

Definitionen

Multimenge: Sei F eine Menge. Eine Multimenge über F ist eine Funktion $\mathcal{F} : F \to \mathbb{N}$, die jedem Element $a \in F$ seine Vielfachheit zuordnet.

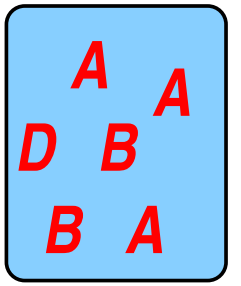
Support: Sei $\mathcal{F} : F \to \mathbb{N}$ eine Multimenge. Eine Menge $B \subseteq F$ heißt Support von $\mathcal{F}$ gdw. $B = \{b \in F \mid \mathcal{F}(b) > 0\}$.

Kardinalität: $\text{card}(\mathcal{F}) := \sum_{a \in F} \mathcal{F}(a)$

Multimenge: Molekularer Inhalt einer Membran

Beispiel

$$\begin{aligned} \mathcal{F} &= \{(A,3),(B,2),(C,0),(D,1)\} \\ \text{supp}(\mathcal{F}) &= \{A,B,D\} \\ \text{card}(\mathcal{F}) &= 6 \end{aligned}$$



Inhalt einer fiktiven Membran

Definitionen

Multimenge: Sei F eine Menge. Eine Multimenge über F ist eine Funktion $\mathcal{F} : F \to \mathbb{N}$, die jedem Element $a \in F$ seine Vielfachheit zuordnet.

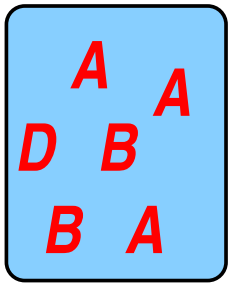
Support: Sei $\mathcal{F} : F \to \mathbb{N}$ eine Multimenge. Eine Menge $B \subseteq F$ heißt Support von $\mathcal{F}$ gdw. $B = \{b \in F \mid \mathcal{F}(b) > 0\}$.

Kardinalität: $\text{card}(\mathcal{F}) := \sum_{a \in F} \mathcal{F}(a)$

Multimenge: Molekularer Inhalt einer Membran

Beispiel

$$\begin{aligned} \mathcal{F} &= \{(A,3),(B,2),(C,0),(D,1)\} \\ \text{supp}(\mathcal{F}) &= \{A,B,D\} \\ \text{card}(\mathcal{F}) &= 6 \end{aligned}$$



Inhalt einer fiktiven Membran

Definitionen

Multimenge: Sei F eine Menge. Eine Multimenge über F ist eine Funktion $\mathcal{F} : F \to \mathbb{N}$, die jedem Element $a \in F$ seine Vielfachheit zuordnet.

Support: Sei $\mathcal{F} : F \to \mathbb{N}$ eine Multimenge. Eine Menge $B \subseteq F$ heißt Support von $\mathcal{F}$ gdw. $B = \{b \in F \mid \mathcal{F}(b) > 0\}$.

Kardinalität: $\text{card}(\mathcal{F}) := \sum_{a \in F} \mathcal{F}(a)$

Operationen auf Multimengen

Seien $\mathcal{F} : F \to \mathbb{N}$ und $\mathcal{G} : G \to \mathbb{N}$ Multimengen.

Echte Teilmultimenge: $\mathcal{F} \subset \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subset \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \,.\, \mathcal{F}(a) < \mathcal{G}(a)$

Teilmultimenge: $\mathcal{F} \subseteq \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subseteq \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \,.\, \mathcal{F}(a) \leq \mathcal{G}(a)$

Gleichheit: $\mathcal{F} = \mathcal{G}$ gdw. $\mathcal{F} \subseteq \mathcal{G}$ und $\mathcal{G} \subseteq \mathcal{F}$

Vereinigung: $\mathcal{F} \cup \mathcal{G} = \{(a, \max(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cup G\}$

Durchschnitt: $\mathcal{F} \cap \mathcal{G} = \{(a, \min(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cap G\}$

Multimengensumme: $\mathcal{F} \uplus \mathcal{G} = \{(a, \mathcal{F}(a) + \mathcal{G}(a)) \mid a \in F \cup G\}$

Multimengendifferenz:
$\mathcal{F} \ominus \mathcal{G} = \{(a, \max(\mathcal{F}(a) - \mathcal{G}(a), 0)) \mid a \in F \setminus G\}$

Operationen auf Multimengen

Seien $\mathcal{F} : F \to \mathbb{N}$ und $\mathcal{G} : G \to \mathbb{N}$ Multimengen.

Echte Teilmultimenge: $\mathcal{F} \subset \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subset \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \,.\, \mathcal{F}(a) < \mathcal{G}(a)$

Teilmultimenge: $\mathcal{F} \subseteq \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subseteq \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \,.\, \mathcal{F}(a) \leq \mathcal{G}(a)$

Gleichheit: $\mathcal{F} = \mathcal{G}$ gdw. $\mathcal{F} \subseteq \mathcal{G}$ und $\mathcal{G} \subseteq \mathcal{F}$

Vereinigung: $\mathcal{F} \cup \mathcal{G} = \{(a, \max(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cup G\}$

Durchschnitt: $\mathcal{F} \cap \mathcal{G} = \{(a, \min(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cap G\}$

Multimengensumme: $\mathcal{F} \uplus \mathcal{G} = \{(a, \mathcal{F}(a) + \mathcal{G}(a)) \mid a \in F \cup G\}$

Multimengendifferenz:
$$\mathcal{F} \ominus \mathcal{G} = \{(a, \max(\mathcal{F}(a) - \mathcal{G}(a), 0)) \mid a \in F \setminus G\}$$

Operationen auf Multimengen

Seien $\mathcal{F} : F \to \mathbb{N}$ und $\mathcal{G} : G \to \mathbb{N}$ Multimengen.

Echte Teilmultimenge: $\mathcal{F} \subset \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subset \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \,.\, \mathcal{F}(a) < \mathcal{G}(a)$

Teilmultimenge: $\mathcal{F} \subseteq \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subseteq \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \,.\, \mathcal{F}(a) \leq \mathcal{G}(a)$

Gleichheit: $\mathcal{F} = \mathcal{G}$ gdw. $\mathcal{F} \subseteq \mathcal{G}$ und $\mathcal{G} \subseteq \mathcal{F}$

Vereinigung: $\mathcal{F} \cup \mathcal{G} = \{(a, \max(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cup G\}$

Durchschnitt: $\mathcal{F} \cap \mathcal{G} = \{(a, \min(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cap G\}$

Multimengensumme: $\mathcal{F} \uplus \mathcal{G} = \{(a, \mathcal{F}(a) + \mathcal{G}(a)) \mid a \in F \cup G\}$

Multimengendifferenz:
$$\mathcal{F} \ominus \mathcal{G} = \{(a, \max(\mathcal{F}(a) - \mathcal{G}(a), 0)) \mid a \in F \setminus G\}$$

Operationen auf Multimengen

Seien $\mathcal{F} : F \to \mathbb{N}$ und $\mathcal{G} : G \to \mathbb{N}$ Multimengen.

Echte Teilmultimenge: $\mathcal{F} \subset \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subset \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \;.\; \mathcal{F}(a) < \mathcal{G}(a)$

Teilmultimenge: $\mathcal{F} \subseteq \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subseteq \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \;.\; \mathcal{F}(a) \leq \mathcal{G}(a)$

Gleichheit: $\mathcal{F} = \mathcal{G}$ gdw. $\mathcal{F} \subseteq \mathcal{G}$ und $\mathcal{G} \subseteq \mathcal{F}$

Vereinigung: $\mathcal{F} \cup \mathcal{G} = \{(a, \max(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cup G\}$

Durchschnitt: $\mathcal{F} \cap \mathcal{G} = \{(a, \min(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cap G\}$

Multimengensumme: $\mathcal{F} \uplus \mathcal{G} = \{(a, \mathcal{F}(a) + \mathcal{G}(a)) \mid a \in F \cup G\}$

Multimengendifferenz:
$$\mathcal{F} \ominus \mathcal{G} = \{(a, \max(\mathcal{F}(a) - \mathcal{G}(a), 0)) \mid a \in F \setminus G\}$$

Operationen auf Multimengen

Seien $\mathcal{F} : F \to \mathbb{N}$ und $\mathcal{G} : G \to \mathbb{N}$ Multimengen.

Echte Teilmultimenge: $\mathcal{F} \subset \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subset \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \;.\; \mathcal{F}(a) < \mathcal{G}(a)$

Teilmultimenge: $\mathcal{F} \subseteq \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subseteq \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \;.\; \mathcal{F}(a) \leq \mathcal{G}(a)$

Gleichheit: $\mathcal{F} = \mathcal{G}$ gdw. $\mathcal{F} \subseteq \mathcal{G}$ und $\mathcal{G} \subseteq \mathcal{F}$

Vereinigung: $\mathcal{F} \cup \mathcal{G} = \{(a, \max(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cup G\}$

Durchschnitt: $\mathcal{F} \cap \mathcal{G} = \{(a, \min(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cap G\}$

Multimengensumme: $\mathcal{F} \uplus \mathcal{G} = \{(a, \mathcal{F}(a) + \mathcal{G}(a)) \mid a \in F \cup G\}$

Multimengendifferenz:
$\mathcal{F} \ominus \mathcal{G} = \{(a, \max(\mathcal{F}(a) - \mathcal{G}(a), 0)) \mid a \in F \setminus G\}$

Operationen auf Multimengen

Seien $\mathcal{F} : F \to \mathbb{N}$ und $\mathcal{G} : G \to \mathbb{N}$ Multimengen.

Echte Teilmultimenge: $\mathcal{F} \subset \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subset \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \,.\, \mathcal{F}(a) < \mathcal{G}(a)$

Teilmultimenge: $\mathcal{F} \subseteq \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subseteq \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \,.\, \mathcal{F}(a) \leq \mathcal{G}(a)$

Gleichheit: $\mathcal{F} = \mathcal{G}$ gdw. $\mathcal{F} \subseteq \mathcal{G}$ und $\mathcal{G} \subseteq \mathcal{F}$

Vereinigung: $\mathcal{F} \cup \mathcal{G} = \{(a, \max(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cup G\}$

Durchschnitt: $\mathcal{F} \cap \mathcal{G} = \{(a, \min(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cap G\}$

Multimengensumme: $\mathcal{F} \uplus \mathcal{G} = \{(a, \mathcal{F}(a) + \mathcal{G}(a)) \mid a \in F \cup G\}$

Multimengendifferenz:
$\mathcal{F} \ominus \mathcal{G} = \{(a, \max(\mathcal{F}(a) - \mathcal{G}(a), 0)) \mid a \in F \setminus G\}$

Operationen auf Multimengen

Seien $\mathcal{F} : F \to \mathbb{N}$ und $\mathcal{G} : G \to \mathbb{N}$ Multimengen.

Echte Teilmultimenge: $\mathcal{F} \subset \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subset \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \; . \; \mathcal{F}(a) < \mathcal{G}(a)$

Teilmultimenge: $\mathcal{F} \subseteq \mathcal{G}$ gdw. $\text{supp}(\mathcal{F}) \subseteq \text{supp}(\mathcal{G})$ und
$\forall a \in \text{supp}(\mathcal{F}) \; . \; \mathcal{F}(a) \leq \mathcal{G}(a)$

Gleichheit: $\mathcal{F} = \mathcal{G}$ gdw. $\mathcal{F} \subseteq \mathcal{G}$ und $\mathcal{G} \subseteq \mathcal{F}$

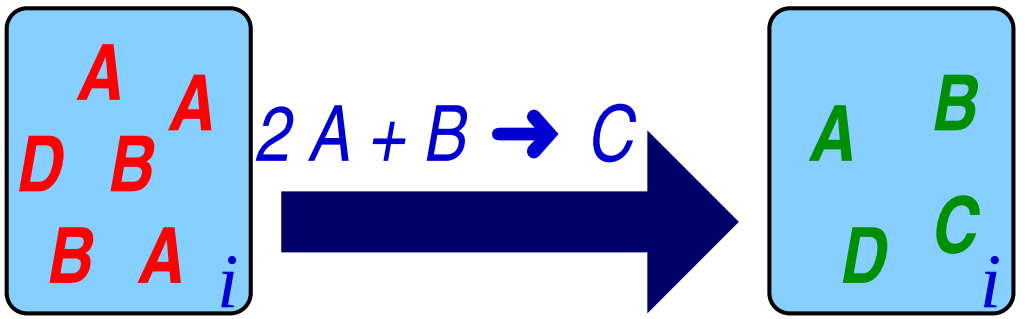
Vereinigung: $\mathcal{F} \cup \mathcal{G} = \{(a, \max(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cup G\}$

Durchschnitt: $\mathcal{F} \cap \mathcal{G} = \{(a, \min(\mathcal{F}(a), \mathcal{G}(a))) \mid a \in F \cap G\}$

Multimengensumme: $\mathcal{F} \uplus \mathcal{G} = \{(a, \mathcal{F}(a) + \mathcal{G}(a)) \mid a \in F \cup G\}$

Multimengendifferenz:
$$\mathcal{F} \ominus \mathcal{G} = \{(a, \max(\mathcal{F}(a) - \mathcal{G}(a), 0)) \mid a \in F \setminus G\}$$

Chemische Reaktionen als Operationen auf Multimengen innerhalb einer Membran *i*



Chemische Reaktion: Darstellung durch zwei Multimengen

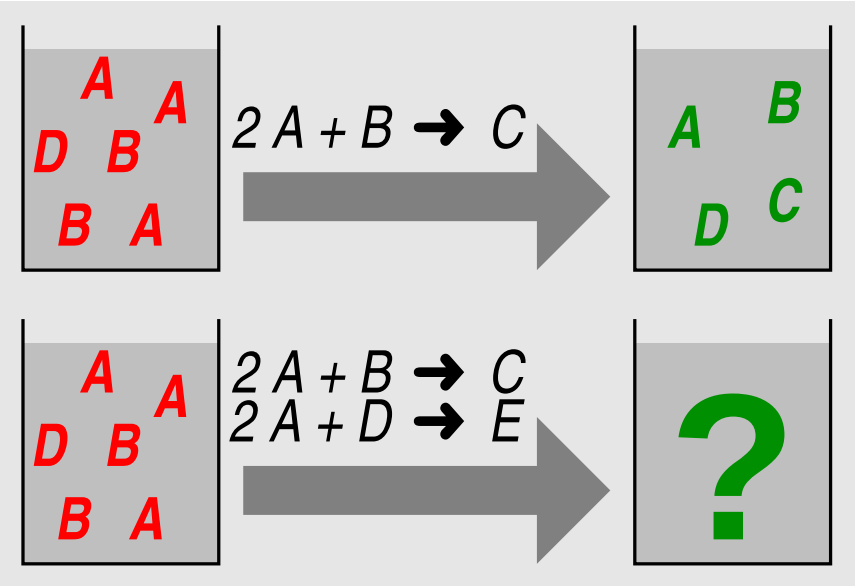
$$2A + B \longrightarrow C$$

$$\big(\{(A,2),(B,1)\} \quad , \quad \{(C,1)\}\big)$$

Reaktionsausführung:
Abfolge von Multimengenoperationen (vereinfacht)

$$\{(A,3),(B,2),(C,0),(D,1)\} \ominus \{(A,2),(B,1)\} \uplus \{(C,1)\} = \{(A,1),(B,1),(C,1),(D,1)\}$$

Konflikthandling: Strategien zur Prozesssteuerung



Auswahl der Reaktionen

Nichtdeterministisch: maximal parallele Betrachtung aller möglichen Szenarien

Priorisierung der Reaktionsregeln: deterministische Arbeitsweise durch vordefinierte Abfolge, in der Reaktionen ausgeführt werden

Stochastisch: zufällige Auswahl einer abgesättigten Reaktion

Reaktionssystem (Membran) in der Zeit

Zeitdiskretes Iterationsschema

- chemische Reaktionen in einer Membran gegeben
- initialer Membraninhalt als *Anfangskonfiguration* $\mathcal{L}_0$ gegeben
- aufeinanderfolgende Iterationsschritte $\mathcal{L}_t \rightarrow \mathcal{L}_{t+1}$. Jeder Iterationsschritt zwischen den diskreten Zeitpunkten t und $t+1$ entspricht einer (realen) Zeitspanne $\Delta\tau$
- Innerhalb jedes Iterationsschrittes werden ausführbare chemische Reaktionen identifiziert und jeweils einmal oder mehrfach angewendet (Mehrfachanwendung z.B. bei Einbeziehung kinetischer Funktionen $\mathrm{f} : \mathcal{L} \rightarrow \mathbb{N}$)
- Im Ergebnis entsteht eine Ableitungsfolge bzw. ein Ableitungsbaum von Membraninhalten (*Konfigurationen*) als Knoten.

Reaktionssystem (Membran) in der Zeit

Zeitdiskretes Iterationsschema

- chemische Reaktionen in einer Membran gegeben
- initialer Membraninhalt als *Anfangskonfiguration* $\mathcal{L}_0$ gegeben
- aufeinanderfolgende Iterationsschritte $\mathcal{L}_t \rightarrow \mathcal{L}_{t+1}$. Jeder Iterationsschritt zwischen den diskreten Zeitpunkten t und $t+1$ entspricht einer (realen) Zeitspanne $\Delta\tau$
- Innerhalb jedes Iterationsschrittes werden ausführbare chemische Reaktionen identifiziert und jeweils einmal oder mehrfach angewendet (Mehrfachanwendung z.B. bei Einbeziehung kinetischer Funktionen $\mathrm{f} : \mathcal{L} \rightarrow \mathbb{N}$)
- Im Ergebnis entsteht eine Ableitungsfolge bzw. ein Ableitungsbaum von Membraninhalten (*Konfigurationen*) als Knoten.

Reaktionssystem (Membran) in der Zeit

Zeitdiskretes Iterationsschema

- chemische Reaktionen in einer Membran gegeben
- initialer Membraninhalt als *Anfangskonfiguration* $\mathcal{L}_0$ gegeben
- aufeinanderfolgende Iterationsschritte $\mathcal{L}_t \rightarrow \mathcal{L}_{t+1}$. Jeder Iterationsschritt zwischen den diskreten Zeitpunkten t und $t+1$ entspricht einer (realen) Zeitspanne $\Delta\tau$
- Innerhalb jedes Iterationsschrittes werden ausführbare chemische Reaktionen identifiziert und jeweils einmal oder mehrfach angewendet (Mehrfachanwendung z.B. bei Einbeziehung kinetischer Funktionen $\mathrm{f} : \mathcal{L} \rightarrow \mathbb{N}$)
- Im Ergebnis entsteht eine Ableitungsfolge bzw. ein Ableitungsbaum von Membraninhalten (*Konfigurationen*) als Knoten.

Reaktionssystem (Membran) in der Zeit

Zeitdiskretes Iterationsschema

- chemische Reaktionen in einer Membran gegeben
- initialer Membraninhalt als *Anfangskonfiguration* $\mathcal{L}_0$ gegeben
- aufeinanderfolgende Iterationsschritte $\mathcal{L}_t \rightarrow \mathcal{L}_{t+1}$. Jeder Iterationsschritt zwischen den diskreten Zeitpunkten t und $t+1$ entspricht einer (realen) Zeitspanne $\Delta\tau$
- Innerhalb jedes Iterationsschrittes werden ausführbare chemische Reaktionen identifiziert und jeweils einmal oder mehrfach angewendet (Mehrfachanwendung z.B. bei Einbeziehung kinetischer Funktionen $\mathrm{f} : \mathcal{L} \rightarrow \mathbb{N}$)
- Im Ergebnis entsteht eine Ableitungsfolge bzw. ein Ableitungsbaum von Membraninhalten (*Konfigurationen*) als Knoten.

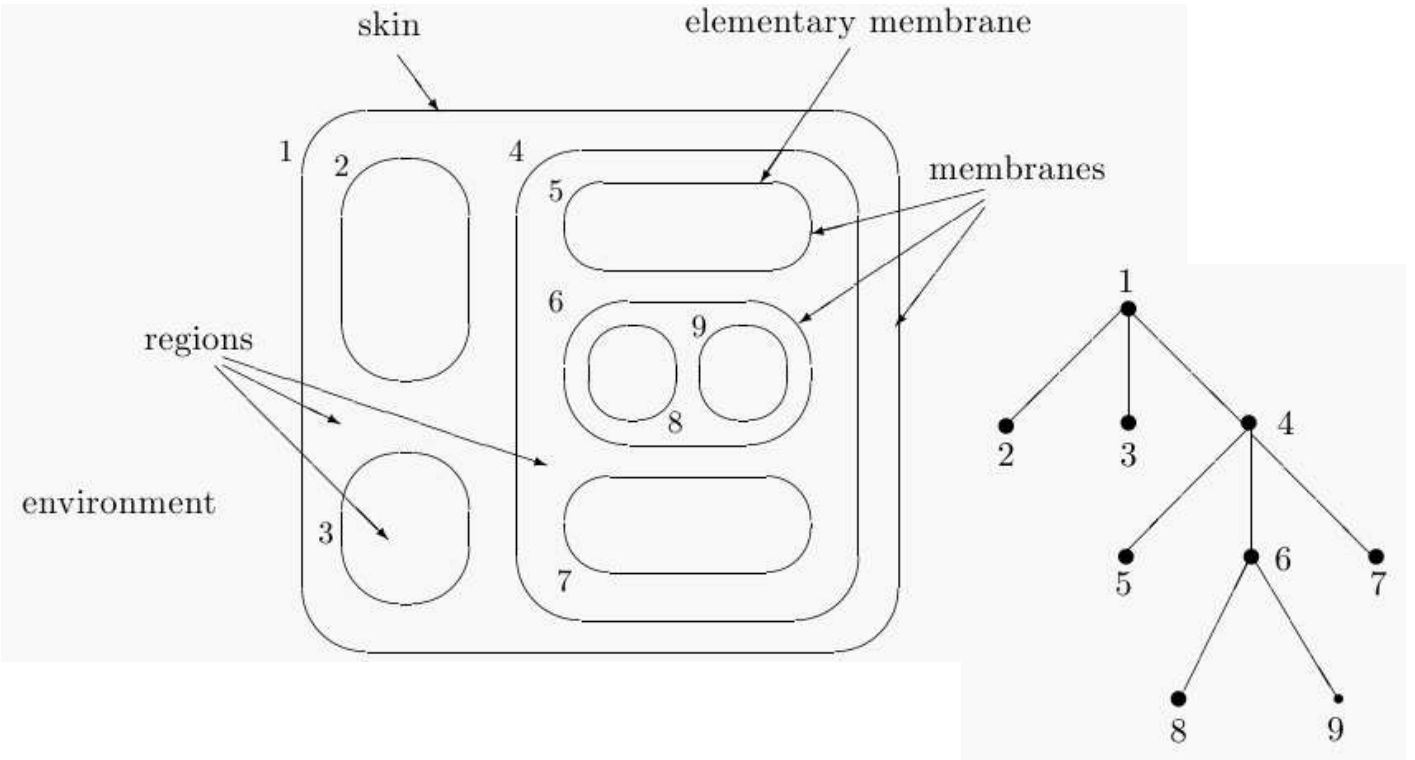
Reaktionssystem (Membran) in der Zeit

Zeitdiskretes Iterationsschema

- chemische Reaktionen in einer Membran gegeben
- initialer Membraninhalt als *Anfangskonfiguration* $\mathcal{L}_0$ gegeben
- aufeinanderfolgende Iterationsschritte $\mathcal{L}_t \rightarrow \mathcal{L}_{t+1}$. Jeder Iterationsschritt zwischen den diskreten Zeitpunkten t und $t+1$ entspricht einer (realen) Zeitspanne $\Delta\tau$
- Innerhalb jedes Iterationsschrittes werden ausführbare chemische Reaktionen identifiziert und jeweils einmal oder mehrfach angewendet (Mehrfachanwendung z.B. bei Einbeziehung kinetischer Funktionen $\mathrm{f} : \mathcal{L} \rightarrow \mathbb{N}$)
- Im Ergebnis entsteht eine Ableitungsfolge bzw. ein Ableitungsbaum von Membraninhalten (*Konfigurationen*) als Knoten.

Von der Einzelmembran zur Membranstruktur

Hierarchisch verschachtelte Membrane notierbar als Baum



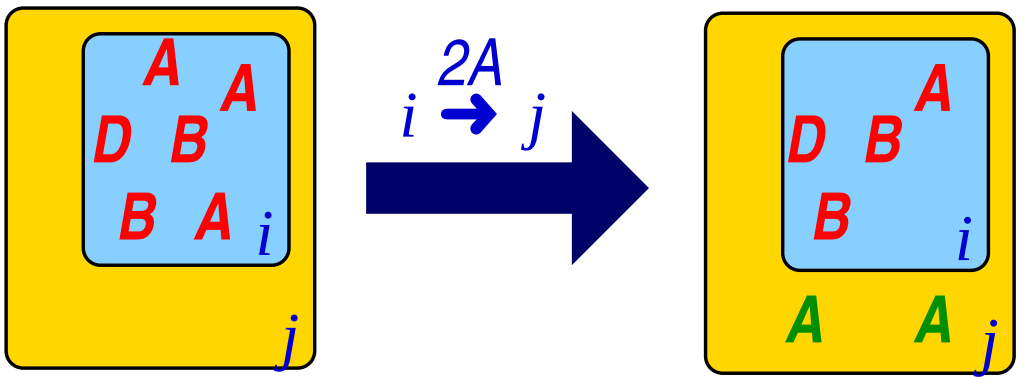
Darstellung als Zeichenkette oder als Graph

$$\mu = [_1[_2]_2[_3]_3[_4[_5]_5[_6[_8]_8[_9]_9]_6[_7]_7]_4]_1$$

G. Păun. Membrane Computing: An Introduction. Springer Verlag, 2002

Molekülaustausch zwischen Membranen

Transportregeln / Diffusionsregeln



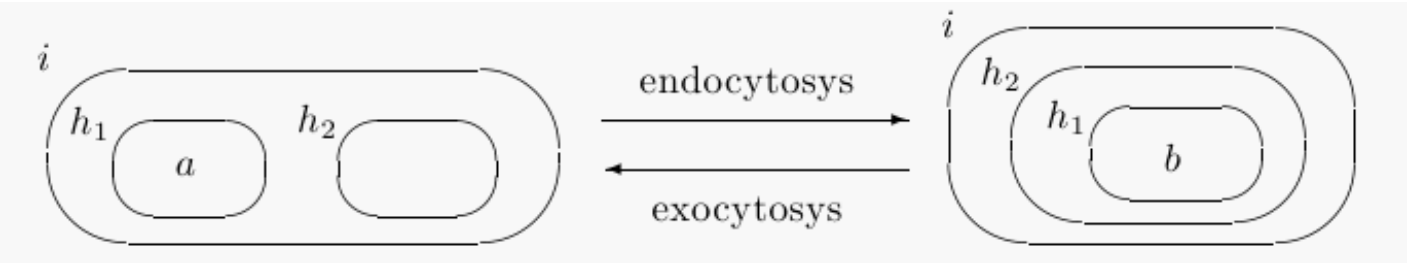
$$[_4]_4 : \{(A, 2)\} \longrightarrow [_5]_5$$

speziell zugeschnitte Regeldefinitionen für

- Diffusion (unspezifischer Transport zwischen benachbarten Membranen)
- Rezeptoren (molekulare Filter)
- Symport/Antiport (interaktiver Molekülaustausch)

Veränderbare (aktive) Membrane

Regeln für Veränderungen der Membranstruktur



G. Păun. Introduction to Membrane Computing. Romanian Academy, 2001

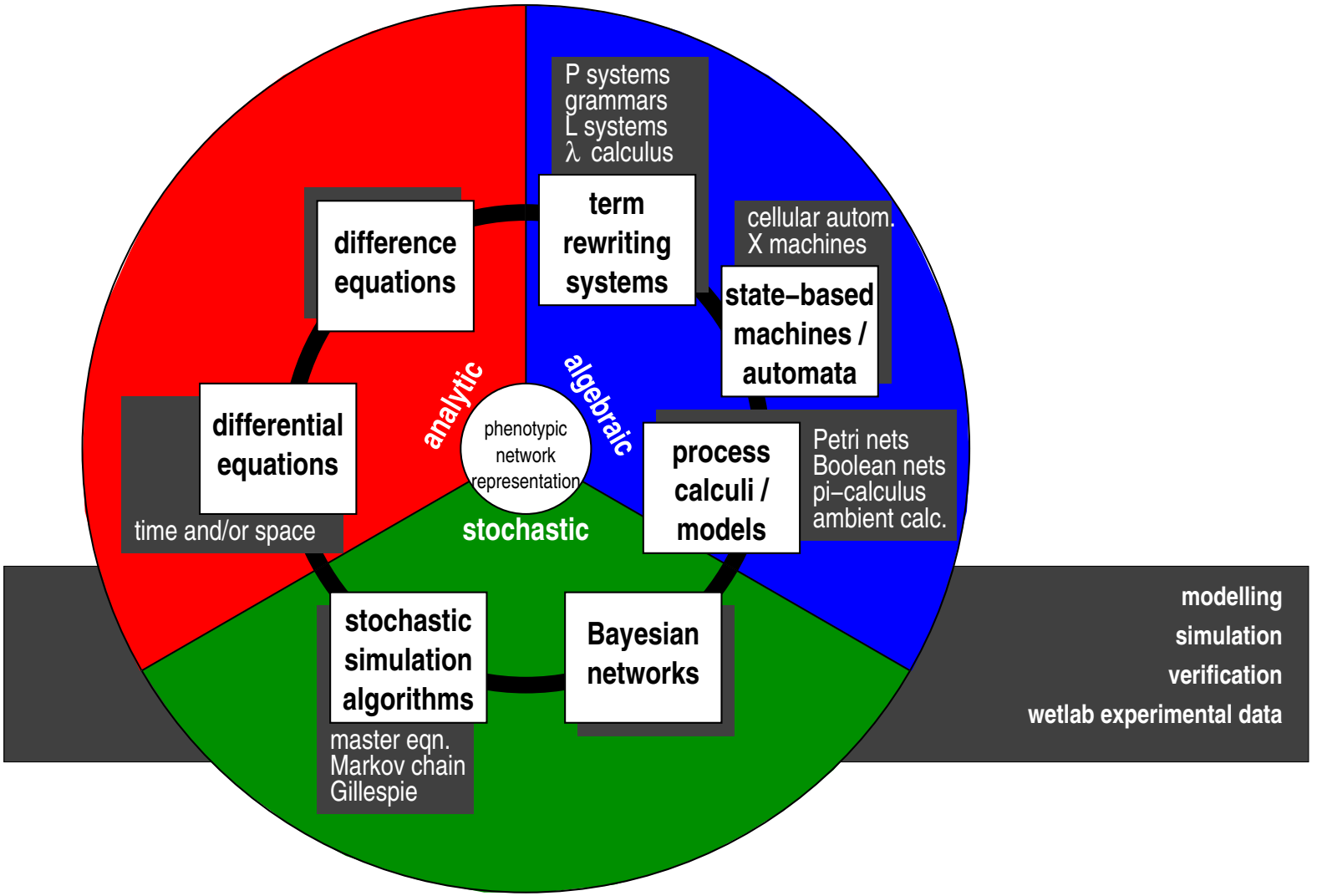
$$[_{h_1} a]_{h_1} [_{h_2}]_{h_2} \longleftrightarrow [_{h_2} [_{h_1} b]_{h_1}]_{h_2}$$

Bei Spezifikation beachten:

- Änderungen der Membranstruktur μ beeinflussen weitere Systemkomponenten
- Anpassung der Konfigurationen $\mathcal{L}$, Reaktionsregeln $\mathcal{R}$ und Transportregeln in den betroffenen Membranen notwendig
- Membraneigenschaften (wie z.B. elektrische Ladung $[]^+, []^-$) darstellbar

Modellierung biologischer Reaktionsnetzwerke

Phänotypische Beschreibung – dynamische Simulation – Analyse



Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submolekulare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

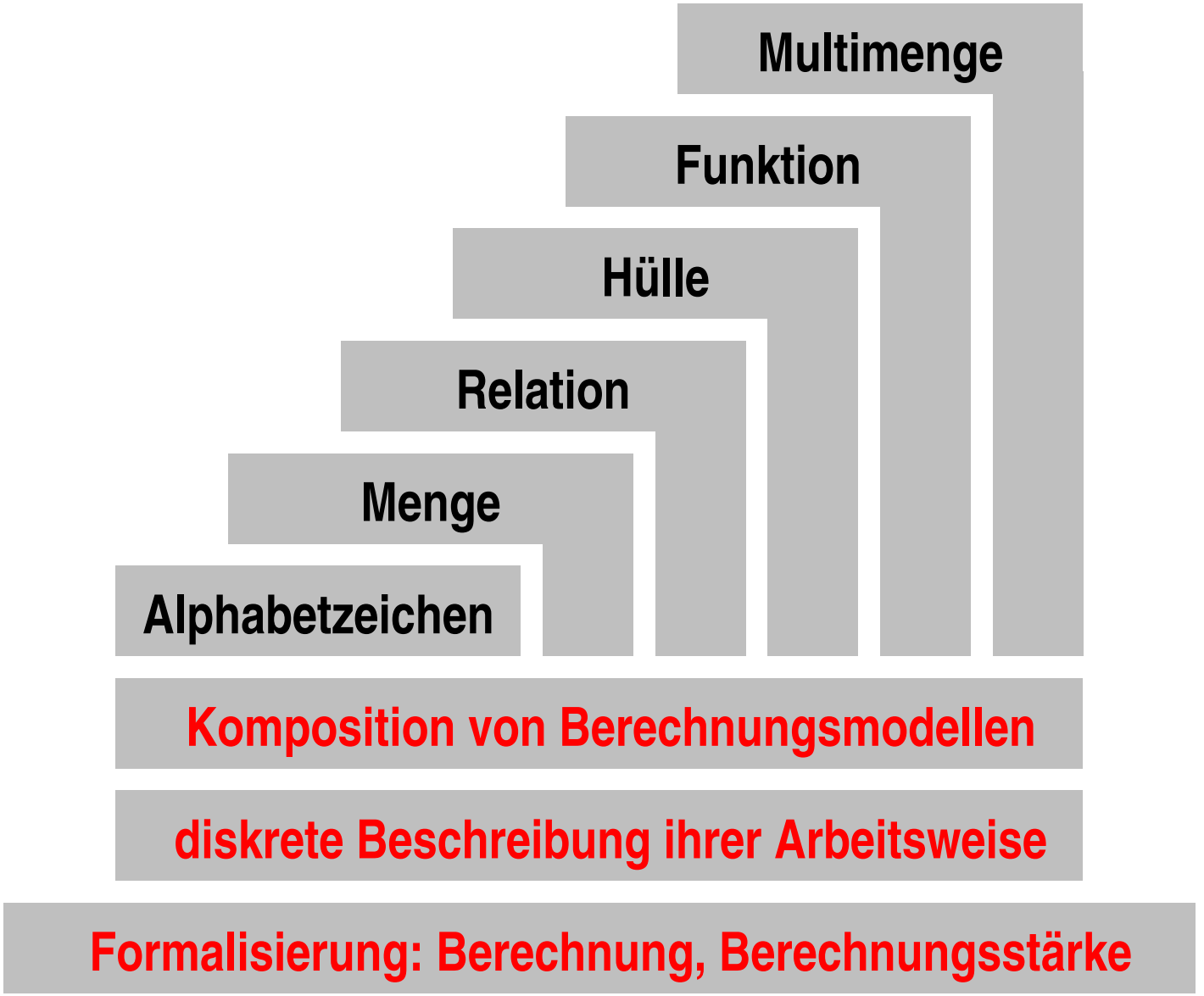
Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Analytische vs. algebraische Beschreibungsansätze

analytische Modelle		algebraische Modelle
gewöhnl. DGL-Systeme (ODE) partielle DGL-Systeme (PDE)	**Notation**	Termersetzungssysteme zustandsbasierte Modelle (Automaten, Maschinen), Kalküle
wertkontinuierlich zeitkontinuierlich	**Granularität**	wertdiskret zeitdiskret
gemittelte homogene Stoffkonzentrationen, keine innere Stoffstruktur	**elementare Objekte**	Einzelmolekül bzw. submoleku- lare Komponenten, ggf. mit zusätzlichem Parametersatz
nicht erfasst	**modulare Systemtopologie**	erfassbar
überabzählbar unendliche Mengen $Z \subseteq \mathbb{R}^n$	**Zustandsraum** Z (Konfigurationen, Phasen)	abzählbar unendliche Mengen $\hat{=}$ rekursiv aufzählbare formale Sprachen ($Z \subseteq \mathbb{N}^n, Z \subseteq (\Sigma^*)^n$)
unendlich viele Moleküle, explizit definiert	**Annahme über Grundgesamtheit**	Konstruktionsprinzip induktiv definierbar
analytische/numerische Lsg. des DGL-Systems	**Fortschreibung der Simulation**	Ausführung regelbasierter Transitionen (Arbeitsschritt, Hülle)
implizit enthalten	**Reaktionskinetik**	zusätzliche Constraints
effizient simulierbar (#-t-x-Diagramm)	**dynamisches Verhalten**	hoher Rechenaufwand, da Einzelmolekülbetrachtung
Mittelwertverfolgung	**Monitoring**	Einzelmolekülverfolgung

Algebraische Grundbegriffe für Berechnungsmodelle



Membransysteme als Termersetzungssysteme

„Die Berechnungsmodelle, mit denen wir uns bisher in der Theoretischen Informatik beschäftigt haben, verwenden keine Multimengen. Kann man Membransysteme auch ohne Multimengen beschreiben?“

Ja, wenn man jedes Molekül nur durch ein *Symbol* (ohne innere Struktur, sog. Symbolobjekt) darstellt und die Ablaufsteuerung vereinfacht (z.B. maximal parallele Regelanwendung ohne Kinetik). Dann entsteht unmittelbar ein *Termersetzungssystem* auf *Zeichenketten (Wörtern)*:

$AAABBD$

und gleichwertig alle Permutationen

$$AAB \longrightarrow C$$

$ABCD$

Membransysteme als Termersetzungssysteme

„Die Berechnungsmodelle, mit denen wir uns bisher in der Theoretischen Informatik beschäftigt haben, verwenden keine Multimengen. Kann man Membransysteme auch ohne Multimengen beschreiben?“

Ja, wenn man jedes Molekül nur durch ein *Symbol* (ohne innere Struktur, sog. Symbolobjekt) darstellt und die Ablaufsteuerung vereinfacht (z.B. maximal parallele Regelanwendung ohne Kinetik). Dann entsteht unmittelbar ein *Termersetzungssystem* auf *Zeichenketten (Wörtern)*:

AAABBD

und gleichwertig alle Permutationen

$$AAB \longrightarrow C$$

ABCD

Membransysteme als Termersetzungssysteme

„Die Berechnungsmodelle, mit denen wir uns bisher in der Theoretischen Informatik beschäftigt haben, verwenden keine Multimengen. Kann man Membransysteme auch ohne Multimengen beschreiben?“

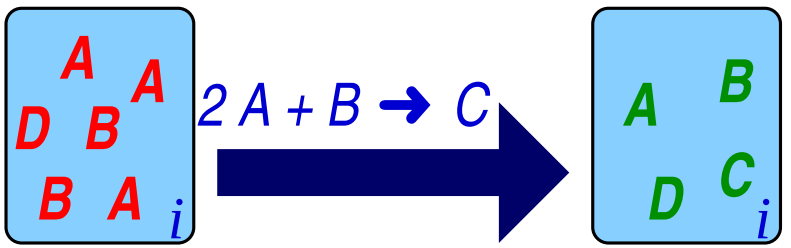
Ja, wenn man jedes Molekül nur durch ein *Symbol* (ohne innere Struktur, sog. Symbolobjekt) darstellt und die Ablaufsteuerung vereinfacht (z.B. maximal parallele Regelanwendung ohne Kinetik). Dann entsteht unmittelbar ein *Termersetzungssystem* auf *Zeichenketten (Wörtern)*:

AAABBD

und gleichwertig alle Permutationen

$$AAB \longrightarrow C$$

ABCD



Inhalt einer fiktiven Membran

Formale Sprachen

Alphabet: nichtleere, endliche Menge Σ von Zeichen $a \in \Sigma$

Wort: endliche Folge (Tupel) von Symbolen aus einem Alphabet Σ, Notation: $(a_1, a_2, \ldots, a_n) =: a_1 a_2 \ldots a_n$

leeres Wort: Wort der Länge 0. Notation: ε

Menge aller Wörter über einem Alphabet: Σ^*. Vollständige Enumeration (induktiv). Beachte: $\varepsilon \in \Sigma^*$

Formale Sprache: Sei Σ ein Alphabet. Eine beliebige Menge $L \subseteq \Sigma^*$ heißt formale Sprache. Jede Zeichenkette $x \in L$ wird Wort der formalen Sprache L genannt.

$\Longrightarrow$ Formale Sprachen dürfen leer sein. Oft enthalten sie jedoch abzählbar unendlich viele Wörter.

$\Longrightarrow$ Alle Datenstrukturen der Informatik lassen sich durch formale Sprachen darstellen.

$\Longrightarrow$ Eingaben bzw. Ausgaben beliebiger Berechnungen

Formale Sprachen

Alphabet: nichtleere, endliche Menge Σ von Zeichen $a \in \Sigma$

Wort: endliche Folge (Tupel) von Symbolen aus einem Alphabet Σ, Notation: $(a_1, a_2, \ldots, a_n) =: a_1 a_2 \ldots a_n$

leeres Wort: Wort der Länge 0. Notation: ε

Menge aller Wörter über einem Alphabet: Σ^*. Vollständige Enumeration (induktiv). Beachte: $\varepsilon \in \Sigma^*$

Formale Sprache: Sei Σ ein Alphabet. Eine beliebige Menge $L \subseteq \Sigma^*$ heißt formale Sprache. Jede Zeichenkette $x \in L$ wird Wort der formalen Sprache L genannt.

$\Longrightarrow$ Formale Sprachen dürfen leer sein. Oft enthalten sie jedoch abzählbar unendlich viele Wörter.

$\Longrightarrow$ Alle Datenstrukturen der Informatik lassen sich durch formale Sprachen darstellen.

$\Longrightarrow$ Eingaben bzw. Ausgaben beliebiger Berechnungen

Formale Sprachen

Alphabet: nichtleere, endliche Menge Σ von Zeichen $a \in \Sigma$

Wort: endliche Folge (Tupel) von Symbolen aus einem Alphabet Σ, Notation: $(a_1, a_2, \ldots, a_n) =: a_1 a_2 \ldots a_n$

leeres Wort: Wort der Länge 0. Notation: ε

Menge aller Wörter über einem Alphabet: Σ^*. Vollständige Enumeration (induktiv). Beachte: $\varepsilon \in \Sigma^*$

Formale Sprache: Sei Σ ein Alphabet. Eine beliebige Menge $L \subseteq \Sigma^*$ heißt formale Sprache. Jede Zeichenkette $x \in L$ wird Wort der formalen Sprache L genannt.

$\Longrightarrow$ Formale Sprachen dürfen leer sein. Oft enthalten sie jedoch abzählbar unendlich viele Wörter.

$\Longrightarrow$ Alle Datenstrukturen der Informatik lassen sich durch formale Sprachen darstellen.

$\Longrightarrow$ Eingaben bzw. Ausgaben beliebiger Berechnungen

Formale Sprachen

Alphabet: nichtleere, endliche Menge Σ von Zeichen $a \in \Sigma$

Wort: endliche Folge (Tupel) von Symbolen aus einem Alphabet Σ, Notation: $(a_1, a_2, \ldots, a_n) =: a_1 a_2 \ldots a_n$

leeres Wort: Wort der Länge 0. Notation: ε

Menge aller Wörter über einem Alphabet: Σ^*. Vollständige Enumeration (induktiv). Beachte: $\varepsilon \in \Sigma^*$

Formale Sprache: Sei Σ ein Alphabet. Eine beliebige Menge $L \subseteq \Sigma^*$ heißt formale Sprache. Jede Zeichenkette $x \in L$ wird Wort der formalen Sprache L genannt.

$\Longrightarrow$ Formale Sprachen dürfen leer sein. Oft enthalten sie jedoch abzählbar unendlich viele Wörter.

$\Longrightarrow$ Alle Datenstrukturen der Informatik lassen sich durch formale Sprachen darstellen.

$\Longrightarrow$ Eingaben bzw. Ausgaben beliebiger Berechnungen

Formale Sprachen

Alphabet: nichtleere, endliche Menge Σ von Zeichen $a \in \Sigma$

Wort: endliche Folge (Tupel) von Symbolen aus einem Alphabet Σ, Notation: $(a_1, a_2, \ldots, a_n) =: a_1 a_2 \ldots a_n$

leeres Wort: Wort der Länge 0. Notation: ε

Menge aller Wörter über einem Alphabet: Σ^*. Vollständige Enumeration (induktiv). Beachte: $\varepsilon \in \Sigma^*$

Formale Sprache: Sei Σ ein Alphabet. Eine beliebige Menge $L \subseteq \Sigma^*$ heißt formale Sprache. Jede Zeichenkette $x \in L$ wird Wort der formalen Sprache L genannt.

$\Longrightarrow$ Formale Sprachen dürfen leer sein. Oft enthalten sie jedoch abzählbar unendlich viele Wörter.

$\Longrightarrow$ Alle Datenstrukturen der Informatik lassen sich durch formale Sprachen darstellen.

$\Longrightarrow$ Eingaben bzw. Ausgaben beliebiger Berechnungen

Formale Sprachen

Alphabet: nichtleere, endliche Menge Σ von Zeichen $a \in \Sigma$

Wort: endliche Folge (Tupel) von Symbolen aus einem Alphabet Σ, Notation: $(a_1, a_2, \ldots, a_n) =: a_1 a_2 \ldots a_n$

leeres Wort: Wort der Länge 0. Notation: ε

Menge aller Wörter über einem Alphabet: Σ^*. Vollständige Enumeration (induktiv). Beachte: $\varepsilon \in \Sigma^*$

Formale Sprache: Sei Σ ein Alphabet. Eine beliebige Menge $L \subseteq \Sigma^*$ heißt formale Sprache. Jede Zeichenkette $x \in L$ wird Wort der formalen Sprache L genannt.

$\Longrightarrow$ Formale Sprachen dürfen leer sein. Oft enthalten sie jedoch abzählbar unendlich viele Wörter.

$\Longrightarrow$ Alle Datenstrukturen der Informatik lassen sich durch formale Sprachen darstellen.

$\Longrightarrow$ Eingaben bzw. Ausgaben beliebiger Berechnungen

Formale Sprachen

Alphabet: nichtleere, endliche Menge Σ von Zeichen $a \in \Sigma$

Wort: endliche Folge (Tupel) von Symbolen aus einem Alphabet Σ, Notation: $(a_1, a_2, \ldots, a_n) =: a_1 a_2 \ldots a_n$

leeres Wort: Wort der Länge 0. Notation: ε

Menge aller Wörter über einem Alphabet: Σ^*. Vollständige Enumeration (induktiv). Beachte: $\varepsilon \in \Sigma^*$

Formale Sprache: Sei Σ ein Alphabet. Eine beliebige Menge $L \subseteq \Sigma^*$ heißt formale Sprache. Jede Zeichenkette $x \in L$ wird Wort der formalen Sprache L genannt.

$\Longrightarrow$ Formale Sprachen dürfen leer sein. Oft enthalten sie jedoch abzählbar unendlich viele Wörter.

$\Longrightarrow$ Alle Datenstrukturen der Informatik lassen sich durch formale Sprachen darstellen.

$\Longrightarrow$ Eingaben bzw. Ausgaben beliebiger Berechnungen

Reguläre Sprachen

Seien L, L_1, L_2 formale Sprachen über gemeinsamem Alphabet Σ.

Vereinigung: $L_1 \cup L_2 = \{u \mid u \in L_1 \vee u \in L_2\}$

Verkettung: $L_1 \otimes L_2 = \{vw \mid v \in L_1 \wedge w \in L_2\}$

reflexive transitive Kleenesche Hülle $L^\star$:

Sei $L^0 = \{\varepsilon\}$, $L^{n+1} = L \otimes L^n$. Dann $L^\star = \bigcup_{n \in \mathbb{N}} L^n$

$\Longrightarrow$ *Reguläre Sprachen* sind spezielle formale Sprachen mit besonders einfachem Aufbauprinzip:

- Die Sprachen $\emptyset$, $\{\varepsilon\}$ und $\{a\}$ mit $a \in \Sigma$ sind regulär.
- Wenn L_1 und L_2 regulär sind, dann auch $(L_1 \cup L_2)$, $(L_1 \otimes L_2)$, $(L_1^\star)$.

$\Longrightarrow$ Jede endliche formale Sprache ist regulär.

$\Longrightarrow$ Reguläre Sprachen können abzählbar unendlich viele Wörter enthalten und durch endliche Automaten erkannt werden.

$\Longrightarrow$ Viele formale Sprachen aus Σ^* sind komplizierter aufgebaut, und ihre Erzeugung erfordert mächtigere Berechnungsmodelle.

Reguläre Sprachen

Seien L, L_1, L_2 formale Sprachen über gemeinsamem Alphabet Σ.

Vereinigung: $L_1 \cup L_2 = \{u \mid u \in L_1 \vee u \in L_2\}$

Verkettung: $L_1 \otimes L_2 = \{vw \mid v \in L_1 \wedge w \in L_2\}$

reflexive transitive Kleenesche Hülle $L^\star$:

Sei $L^0 = \{\varepsilon\}$, $L^{n+1} = L \otimes L^n$. Dann $L^\star = \bigcup_{n \in \mathbb{N}} L^n$

$\Longrightarrow$ *Reguläre Sprachen* sind spezielle formale Sprachen mit besonders einfachem Aufbauprinzip:

- Die Sprachen $\emptyset$, $\{\varepsilon\}$ und $\{a\}$ mit $a \in \Sigma$ sind regulär.
- Wenn L_1 und L_2 regulär sind, dann auch $(L_1 \cup L_2)$, $(L_1 \otimes L_2)$, $(L_1^\star)$.

$\Longrightarrow$ Jede endliche formale Sprache ist regulär.

$\Longrightarrow$ Reguläre Sprachen können abzählbar unendlich viele Wörter enthalten und durch endliche Automaten erkannt werden.

$\Longrightarrow$ Viele formale Sprachen aus Σ^* sind komplizierter aufgebaut, und ihre Erzeugung erfordert mächtigere Berechnungsmodelle.

Reguläre Sprachen

Seien L, L_1, L_2 formale Sprachen über gemeinsamem Alphabet Σ.

Vereinigung: $L_1 \cup L_2 = \{u \mid u \in L_1 \vee u \in L_2\}$

Verkettung: $L_1 \otimes L_2 = \{vw \mid v \in L_1 \wedge w \in L_2\}$

reflexive transitive Kleenesche Hülle $L^\star$:

Sei $L^0 = \{\varepsilon\}$, $L^{n+1} = L \otimes L^n$. Dann $L^\star = \bigcup_{n \in \mathbb{N}} L^n$

$\Longrightarrow$ *Reguläre Sprachen* sind spezielle formale Sprachen mit besonders einfachem Aufbauprinzip:

- Die Sprachen $\emptyset$, $\{\varepsilon\}$ und $\{a\}$ mit $a \in \Sigma$ sind regulär.
- Wenn L_1 und L_2 regulär sind, dann auch $(L_1 \cup L_2)$, $(L_1 \otimes L_2)$, $(L_1^\star)$.

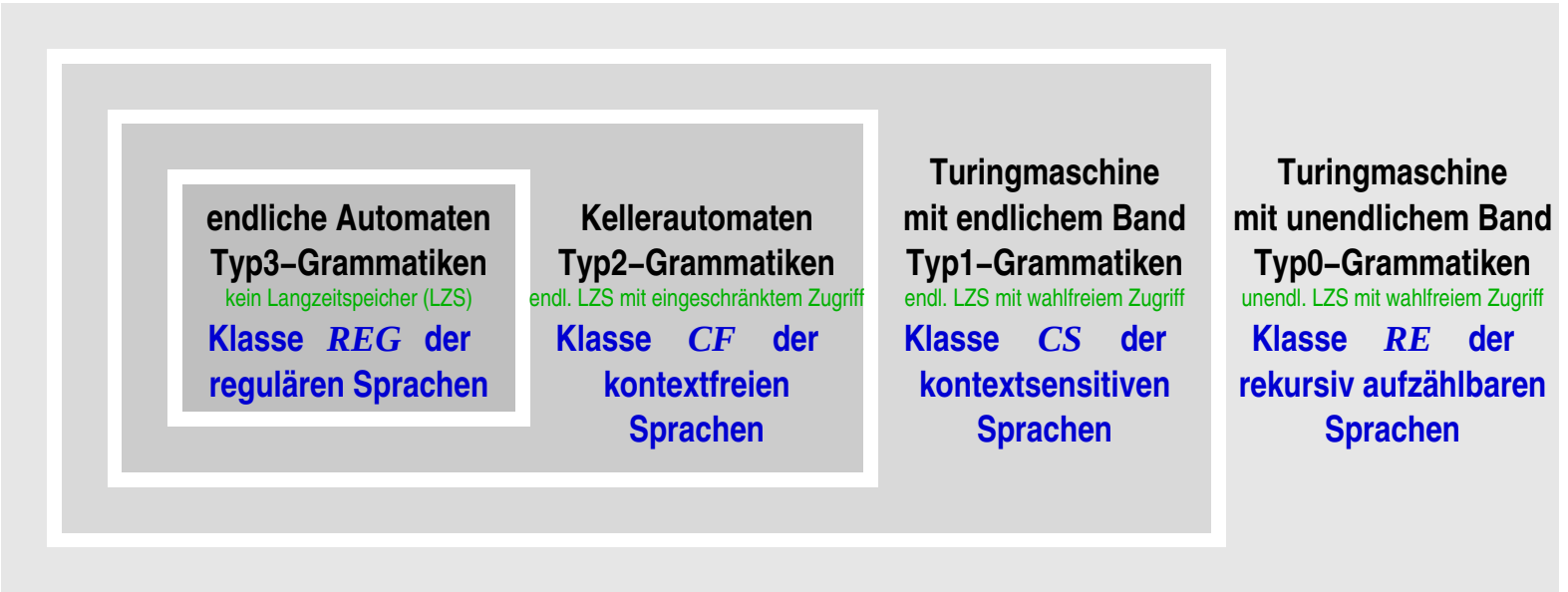
$\Longrightarrow$ Jede endliche formale Sprache ist regulär.

$\Longrightarrow$ Reguläre Sprachen können abzählbar unendlich viele Wörter enthalten und durch endliche Automaten erkannt werden.

$\Longrightarrow$ Viele formale Sprachen aus Σ^* sind komplizierter aufgebaut, und ihre Erzeugung erfordert mächtigere Berechnungsmodelle.

Berechnungsstärken gemäß Sprachklassen

Berechnung: Generieren oder Akzeptieren formaler Sprachen



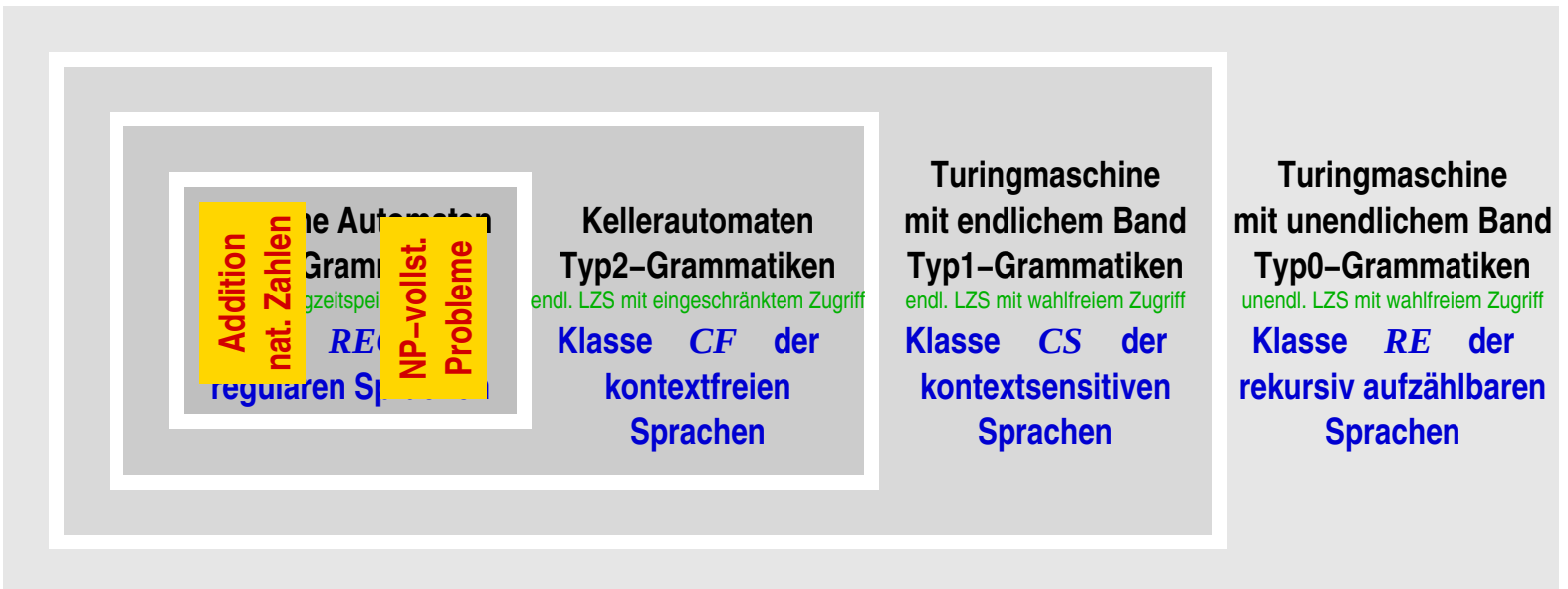
Es gilt:

$FIN \subset REG \subset CF \subset CS \subset RE$

$\text{card}(RE) = \text{card}(\mathbb{N}) < \text{card}(\mathbb{R})$

Berechnungsstärken gemäß Sprachklassen

Berechnung: Generieren oder Akzeptieren formaler Sprachen



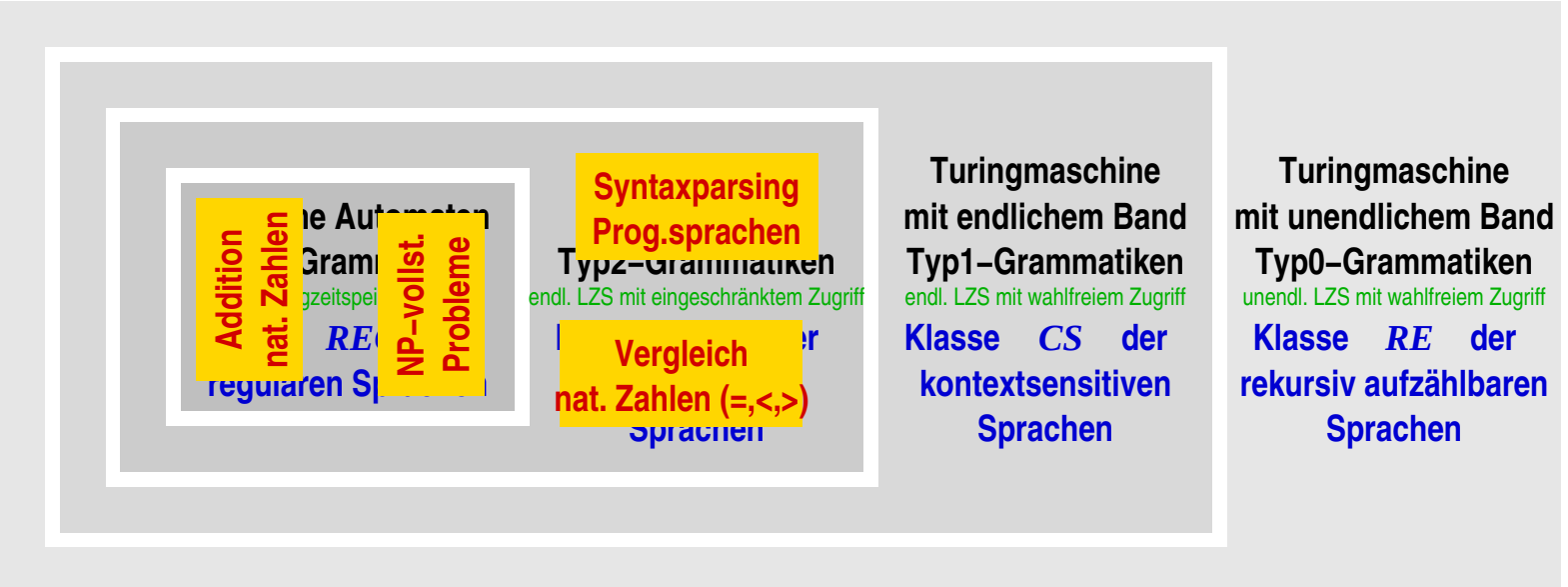
Es gilt:

$FIN \subset REG \subset CF \subset CS \subset RE$

$\text{card}(RE) = \text{card}(\mathbb{N}) < \text{card}(\mathbb{R})$

Berechnungsstärken gemäß Sprachklassen

Berechnung: Generieren oder Akzeptieren formaler Sprachen



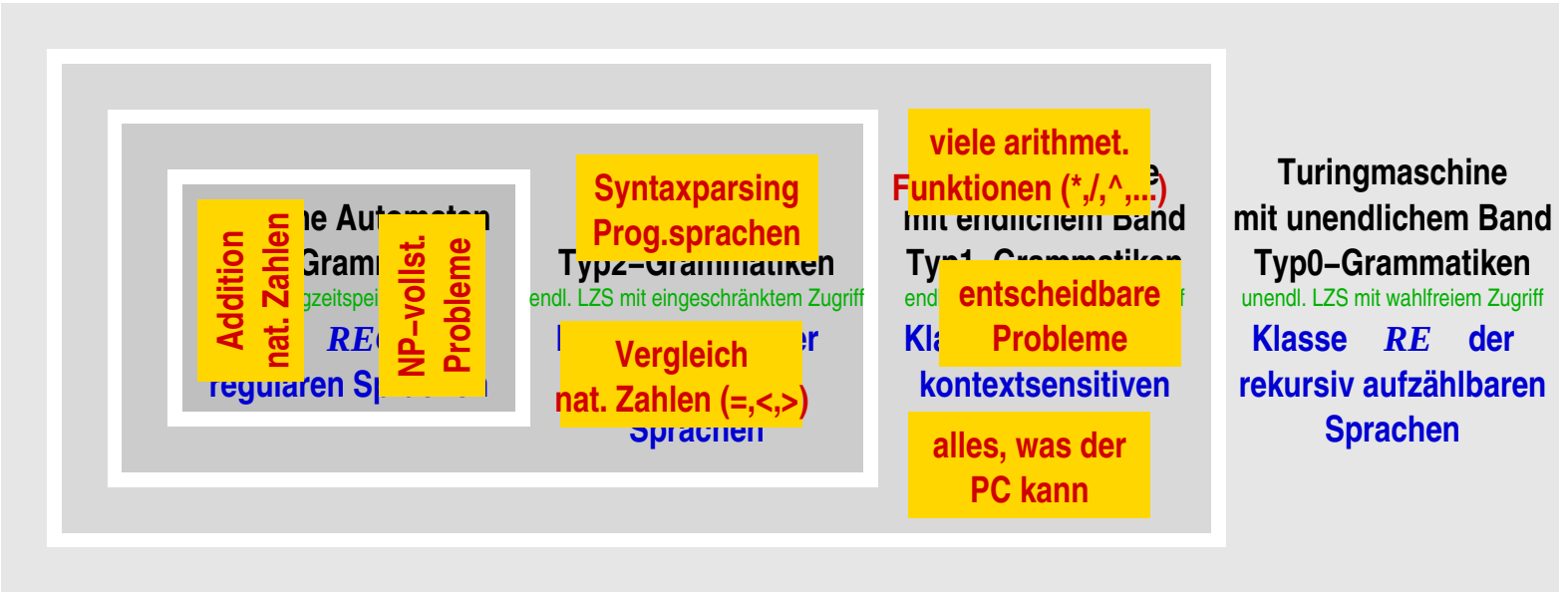
Es gilt:

$FIN \subset REG \subset CF \subset CS \subset RE$

$\text{card}(RE) = \text{card}(\mathbb{N}) < \text{card}(\mathbb{R})$

Berechnungsstärken gemäß Sprachklassen

Berechnung: Generieren oder Akzeptieren formaler Sprachen



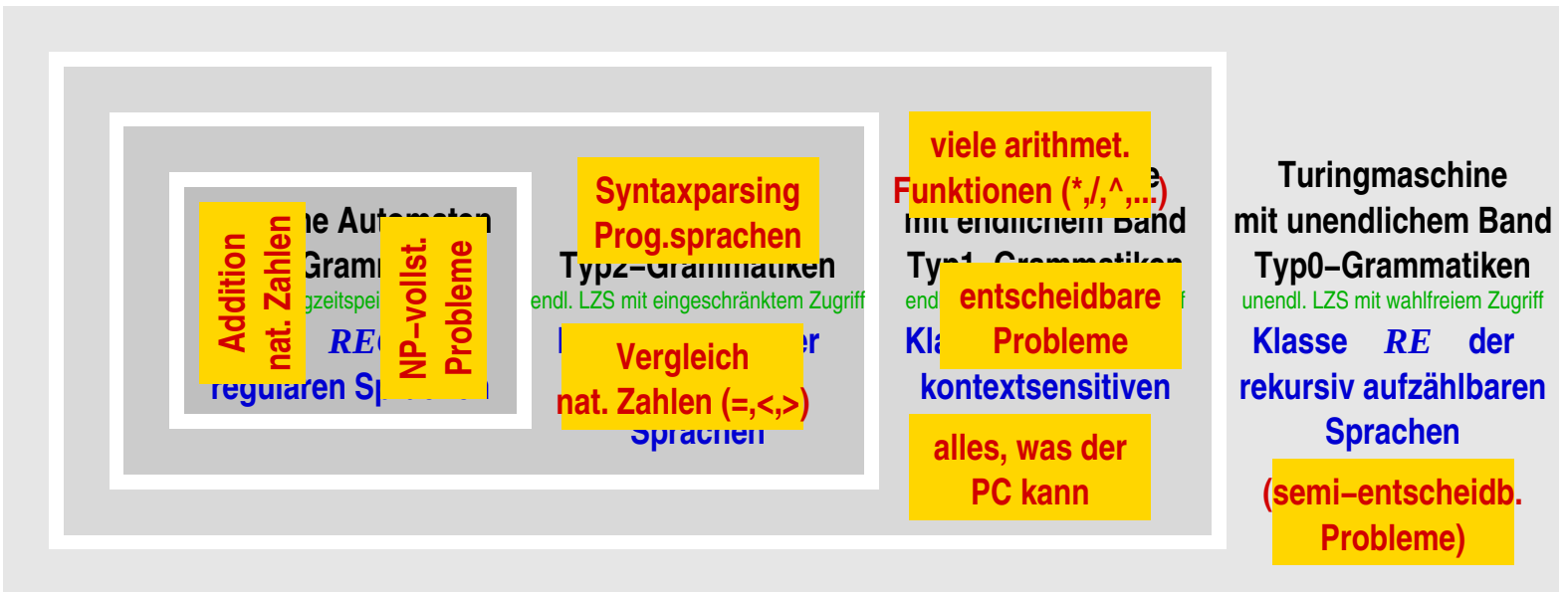
Es gilt:

$FIN \subset REG \subset CF \subset CS \subset RE$

$\text{card}(RE) = \text{card}(\mathbb{N}) < \text{card}(\mathbb{R})$

Berechnungsstärken gemäß Sprachklassen

Berechnung: Generieren oder Akzeptieren formaler Sprachen



Es gilt:

$FIN \subset REG \subset CF \subset CS \subset RE$

$\text{card}(RE) = \text{card}(\mathbb{N}) < \text{card}(\mathbb{R})$

Berechnungsmodell Chomsky-Grammatik

$\Longrightarrow$ endliches Erzeugungssystem für beliebige formale Sprachen

$\Longrightarrow$ Berechnung durch Ersetzungsregeln gesteuert, mithin nichtdeterministisch und massiv datenparallel realisierbar

Definition

Eine Chomsky-Grammatik G ist ein Quadrupel $G = (V, \Sigma, P, S)$ mit folgender Bedeutung der Komponenten:

- V: Alphabet der Nichtterminalsymbole (Variablen)
- Σ: Alphabet der Terminalsymbole (Zeichenvorrat, aus dem sich die generierte Sprache zusammensetzt)
- $P \subset ((V \cup \Sigma)^*) \otimes V \otimes ((V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$: nichtleere endliche Regelmenge (Produktionen)
- S: Startsymbol, $S \in V$. Zusätzlich gelte: $V \cap \Sigma = \emptyset$

Berechnungsmodell Chomsky-Grammatik

$\Longrightarrow$ endliches Erzeugungssystem für beliebige formale Sprachen

$\Longrightarrow$ Berechnung durch Ersetzungsregeln gesteuert, mithin nichtdeterministisch und massiv datenparallel realisierbar

Definition

Eine Chomsky-Grammatik G ist ein Quadrupel $G = (V, \Sigma, P, S)$ mit folgender Bedeutung der Komponenten:

- V: Alphabet der Nichtterminalsymbole (Variablen)
- Σ: Alphabet der Terminalsymbole (Zeichenvorrat, aus dem sich die generierte Sprache zusammensetzt)
- $P \subset ((V \cup \Sigma)^*) \otimes V \otimes ((V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$: nichtleere endliche Regelmenge (Produktionen)
- S: Startsymbol, $S \in V$. Zusätzlich gelte: $V \cap \Sigma = \emptyset$

Arbeitsweise einer Chomsky-Grammatik

Sei $G = (V, \Sigma, P, S)$ eine (gegebene) Chomsky-Grammatik.

- *Ableitungsschritt* (Regelanwendung, Produktion) vom Wort x nach Wort y ist eine Relation $\vdash_G \subset ((V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$, definiert durch: $x \vdash_G y = \{(x, y) \mid x = x_1 u x_2 \wedge y = x_1 v x_2 \wedge \exists x_1, x_2 \in (V \cup \Sigma)^*.((u, v) \in P)\}$.
  D.h., das Teilwort u in x wird durch v ersetzt, so dass das Wort y entsteht, wenn es eine Regel $(u, v) \in P$ gibt.
- *Ableitung* eines Wortes w_n ist eine endliche Folge von Ableitungsschritten, ausgehend vom Startsymbol:
  $S \vdash_G w_1 \vdash_G w_2 \vdash_G \ldots \vdash_G w_n$
- Die durch G beschriebene (generierte) *Sprache* $L(G)$ ist definiert durch: $L(G) = \{x \in \Sigma^* \mid S \vdash_G^* x\}$.
  D.h., die Bildung der reflexiven transitiven Hülle $\vdash_G^*$ beschreibt die Hintereinanderausführung beliebig vieler Ableitungsschritte.

$\Longrightarrow$ Die erzeugte Sprache $L(G)$ ist das Berechnungsergebnis der Chomsky-Grammatik G.

Arbeitsweise einer Chomsky-Grammatik

Sei $G = (V, \Sigma, P, S)$ eine (gegebene) Chomsky-Grammatik.

- *Ableitungsschritt* (Regelanwendung, Produktion) vom Wort x nach Wort y ist eine Relation $\vdash_G \subset ((V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$, definiert durch: $x \vdash_G y = \{(x, y) \mid x = x_1 u x_2 \wedge y = x_1 v x_2 \wedge \exists x_1, x_2 \in (V \cup \Sigma)^*.((u, v) \in P)\}$.
  D.h., das Teilwort u in x wird durch v ersetzt, so dass das Wort y entsteht, wenn es eine Regel $(u, v) \in P$ gibt.
- *Ableitung* eines Wortes w_n ist eine endliche Folge von Ableitungsschritten, ausgehend vom Startsymbol:
  $S \vdash_G w_1 \vdash_G w_2 \vdash_G \ldots \vdash_G w_n$
- Die durch G beschriebene (generierte) *Sprache* $L(G)$ ist definiert durch: $L(G) = \{x \in \Sigma^* \mid S \vdash_G^* x\}$.
  D.h., die Bildung der reflexiven transitiven Hülle $\vdash_G^*$ beschreibt die Hintereinanderausführung beliebig vieler Ableitungsschritte.

$\Longrightarrow$ Die erzeugte Sprache $L(G)$ ist das Berechnungsergebnis der Chomsky-Grammatik G.

Arbeitsweise einer Chomsky-Grammatik

Sei $G = (V, \Sigma, P, S)$ eine (gegebene) Chomsky-Grammatik.

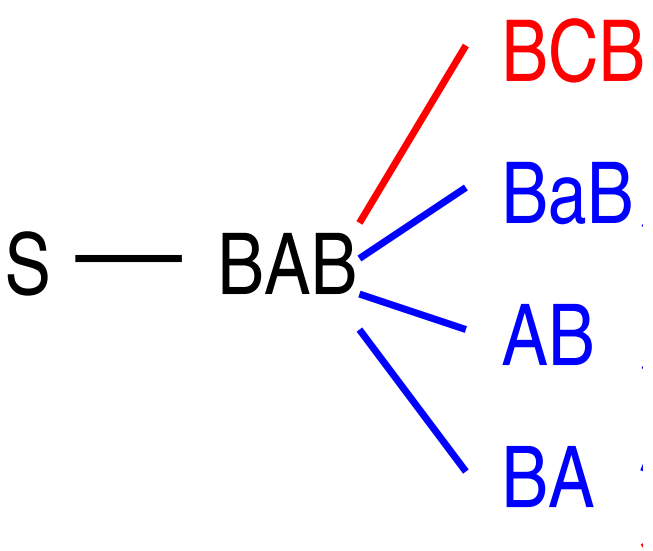
- *Ableitungsschritt* (Regelanwendung, Produktion) vom Wort x nach Wort y ist eine Relation $\vdash_G \subset ((V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$, definiert durch: $x \vdash_G y = \{(x, y) \mid x = x_1 u x_2 \wedge y = x_1 v x_2 \wedge \exists x_1, x_2 \in (V \cup \Sigma)^*.((u, v) \in P)\}$.
  D.h., das Teilwort u in x wird durch v ersetzt, so dass das Wort y entsteht, wenn es eine Regel $(u, v) \in P$ gibt.
- *Ableitung* eines Wortes w_n ist eine endliche Folge von Ableitungsschritten, ausgehend vom Startsymbol:
  $S \vdash_G w_1 \vdash_G w_2 \vdash_G \ldots \vdash_G w_n$
- Die durch G beschriebene (generierte) *Sprache* $L(G)$ ist definiert durch: $L(G) = \{x \in \Sigma^* \mid S \vdash_G^* x\}$.
  D.h., die Bildung der reflexiven transitiven Hülle $\vdash_G^*$ beschreibt die Hintereinanderausführung beliebig vieler Ableitungsschritte.

$\Longrightarrow$ Die erzeugte Sprache $L(G)$ ist das Berechnungsergebnis der Chomsky-Grammatik G.

Chomsky-Grammatik: Beispiel

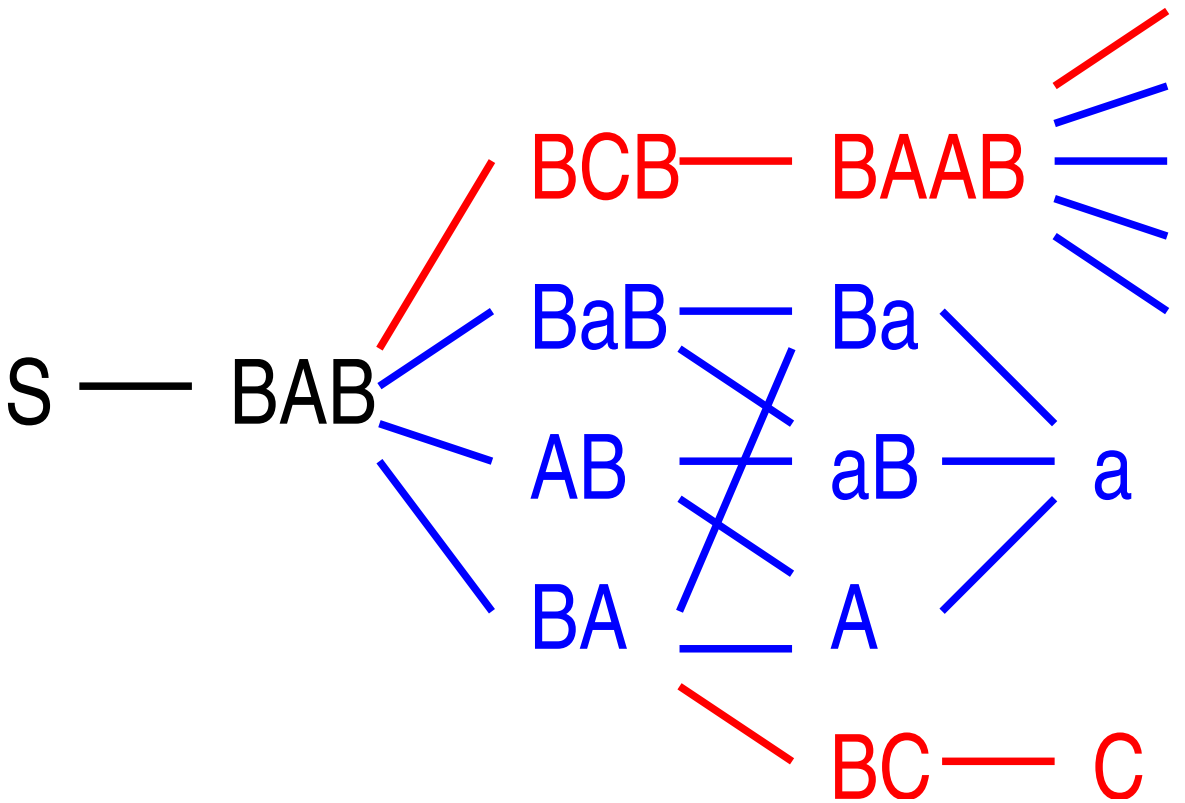
$\Longrightarrow$ Für Produktionen $(u, v) \in P$ schreibt man häufig $u \to v$.

Grammatik $G = (\{\mathbf{S}, \mathbf{A}, \mathbf{B}, \mathbf{C}\}, \{\mathbf{a}\}, P, \mathbf{S})$ mit den Regeln $P = \{\mathbf{S} \to \mathbf{BAB}, \mathbf{BA} \to \mathbf{BC}, \mathbf{CA} \to \mathbf{AAC}, \mathbf{CB} \to \mathbf{AAB}, \mathbf{B} \to \varepsilon, \mathbf{A} \to \mathbf{a}\}$.
Welche Sprache $L(G)$ berechnet G?



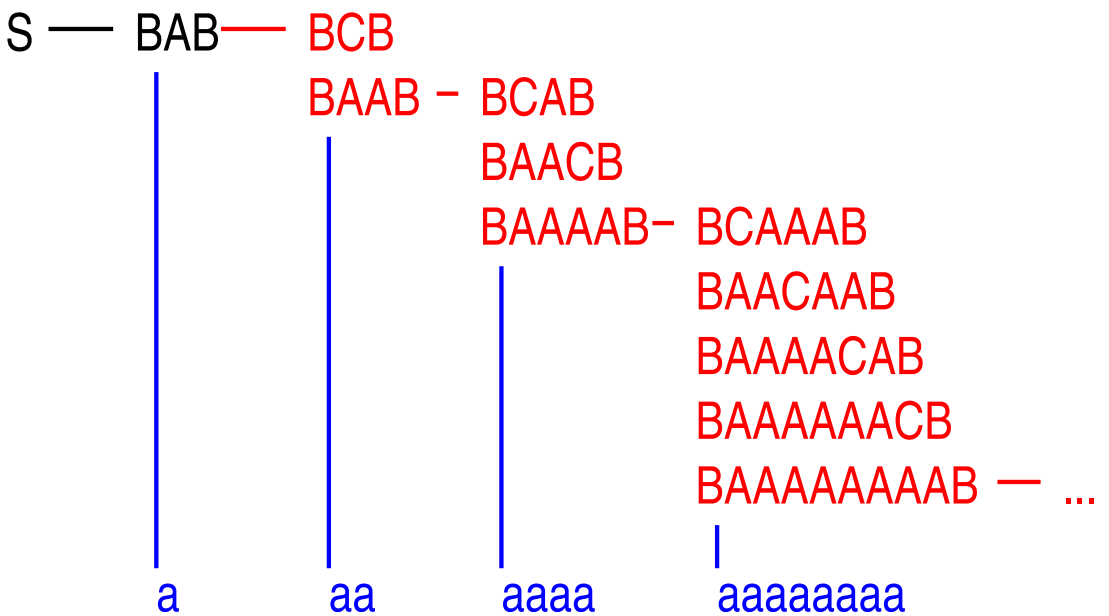
Chomsky-Grammatik: Beispiel

Grammatik $G = (\{\mathbf{S}, \mathbf{A}, \mathbf{B}, \mathbf{C}\}, \{\mathbf{a}\}, P, \mathbf{S})$ mit den Regeln $P = \{\mathbf{S} \to \mathbf{BAB}, \mathbf{BA} \to \mathbf{BC}, \mathbf{CA} \to \mathbf{AAC}, \mathbf{CB} \to \mathbf{AAB}, \mathbf{B} \to \varepsilon, \mathbf{A} \to \mathbf{a}\}$.
Welche Sprache $L(G)$ berechnet G?



Chomsky-Grammatik: Beispiel

Grammatik $G = (\{\mathbf{S}, \mathbf{A}, \mathbf{B}, \mathbf{C}\}, \{\mathbf{a}\}, P, \mathbf{S})$ mit den Regeln $P = \{\mathbf{S} \to \mathbf{BAB}, \mathbf{BA} \to \mathbf{BC}, \mathbf{CA} \to \mathbf{AAC}, \mathbf{CB} \to \mathbf{AAB}, \mathbf{B} \to \varepsilon, \mathbf{A} \to \mathbf{a}\}$.
Welche Sprache $L(G)$ berechnet G?

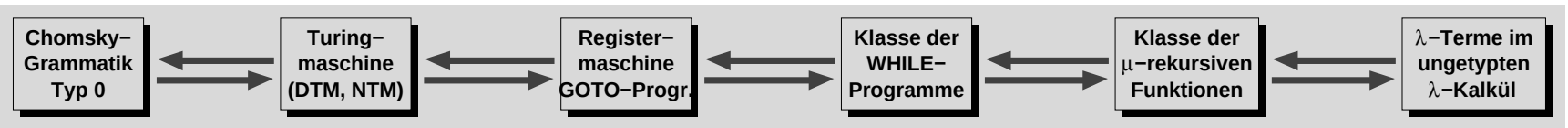


$L(G) = \{\underbrace{\mathbf{a}\ldots\mathbf{a}}_{2^n} \mid n \in \mathbb{N}\}$ Berechnung aller Zweierpotenzen

Grenze der Berechenbarkeit

- Registermaschinen können jede abzählbar unendliche Teilmenge natürlicher Zahlen erzeugen $L(RAM) \subseteq \mathbb{N}$.
- Chomsky-Grammatiken können alle rekursiv aufzählbaren formalen Sprachen generieren $L(G) \subseteq \Sigma^*$, oBdA $\Sigma = \{a\}$,
- wobei die Kodierung „Wort $\underbrace{a \ldots a}_{n-\text{mal}} \hat{=}$ Zahl n“ zugrunde liegt.

$\Longrightarrow$ Chomsky-Grammatiken und Registermaschinen können sich gegenseitig simulieren, ebenso weitere Berechnungsmodelle:



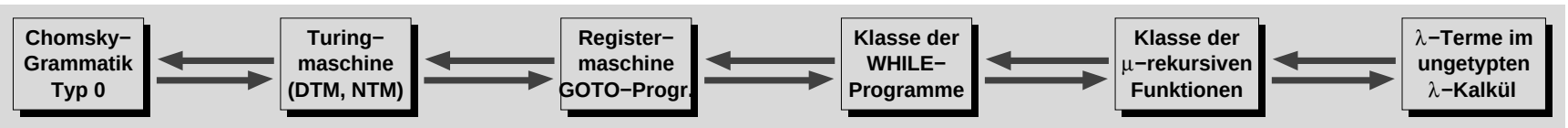
$\Longrightarrow$ Gemeinsamkeit all dieser Modelle: beliebig großer Langzeitspeicher mit wahlfreiem Zugriff

$\Longrightarrow$ Grenze der Berechenbarkeit: Suchraum für ein Problem abzählbar unendlich, dann kann er mittels Berechnung (algorithmisch) durchforstet werden (Turing).

Grenze der Berechenbarkeit

- Registermaschinen können jede abzählbar unendliche Teilmenge natürlicher Zahlen erzeugen $L(RAM) \subseteq \mathbb{N}$.
- Chomsky-Grammatiken können alle rekursiv aufzählbaren formalen Sprachen generieren $L(G) \subseteq \Sigma^*$, oBdA $\Sigma = \{a\}$,
- wobei die Kodierung „Wort $\underbrace{a \ldots a}_{n-\text{mal}} \hat{=}$ Zahl n“ zugrunde liegt.

$\Longrightarrow$ Chomsky-Grammatiken und Registermaschinen können sich gegenseitig simulieren, ebenso weitere Berechnungsmodelle:



$\Longrightarrow$ Gemeinsamkeit all dieser Modelle: beliebig großer Langzeitspeicher mit wahlfreiem Zugriff

$\Longrightarrow$ Grenze der Berechenbarkeit: Suchraum für ein Problem abzählbar unendlich, dann kann er mittels Berechnung (algorithmisch) durchforstet werden (Turing).

Membran- und zellbasiertes Computing

vs. Cluster-Computing

Mensch

200 Billionen Zellen

200 Zellarten

Signalnetzwerke

Rezeptoren

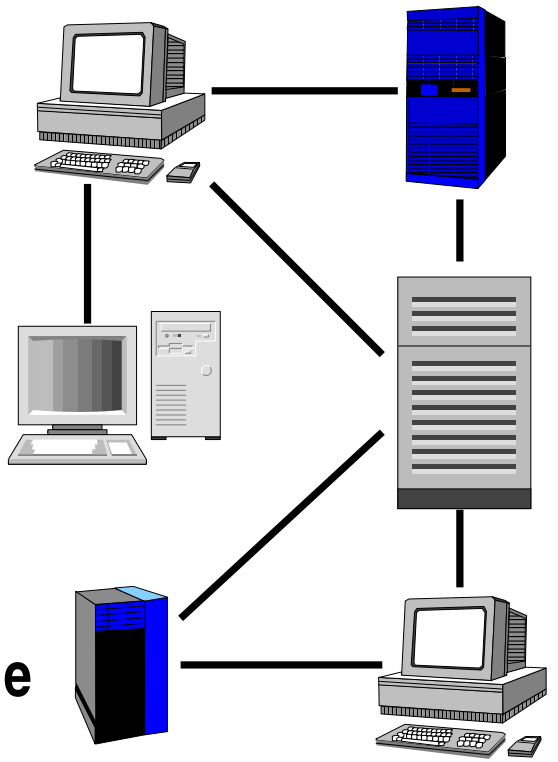
Computer–Cluster

10.000 Einzelrechner

1.000 Normteilarten

Kommunikationskanäle

Adressierung

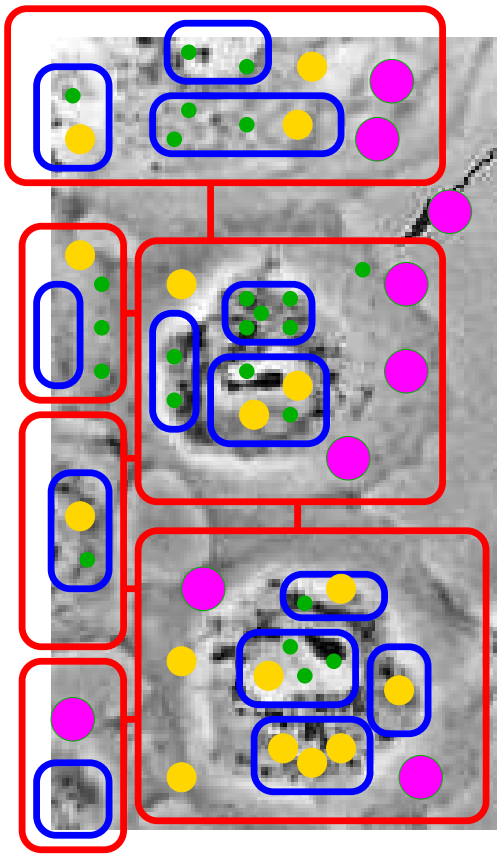


Organismus (Zellverband, Gewebe)

Computer–Cluster

Von Komponenten zu Kompartimenten

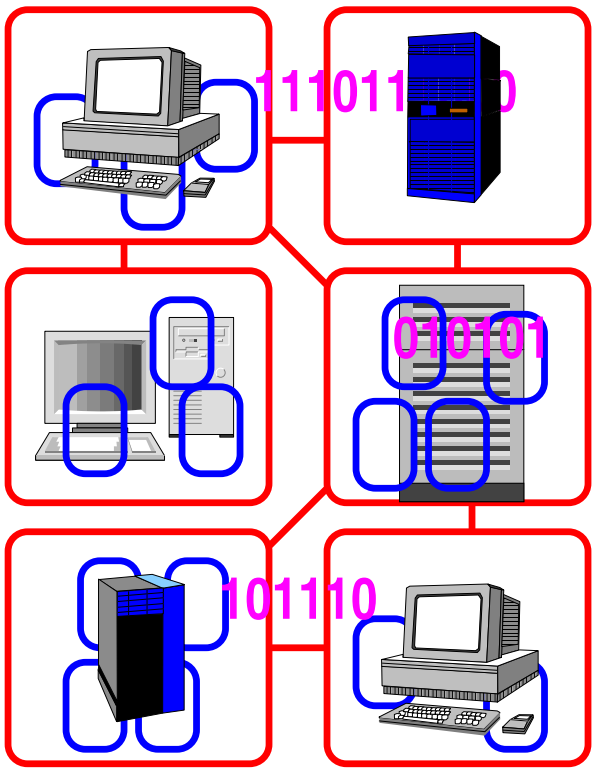
Hierarchische Komposition funktioneller Einheiten



diskrete Signale

Kommunikations–struktur: hierarchisch oder netzbasiert

Berechnung: Signalumwandlung und/oder Kommunikation



Organismus (Zellverband, Gewebe)

Computer–Cluster

Ein Membransystem mit fester Membranstruktur

Betrachten wir das Membransystem
$\Pi = (O, \mu, w_1, w_2, R_1, R_2, i_O)$ mit den Komponenten

$O = \{a, b, c\}$ Alphabet der **Symbolobjekte**, also drei Sorten Moleküle

$\mu = [_1[_2\,]_2]_1$ Struktur der Reaktionsräume (Membranen), fortlaufend nummeriert, bei zellbasierten Membransystemen immer als **Baum** darstel

$w_1 = aa$ **Anfangsbelegung** Membran 1: zwei Moleküle a

$w_2 = \varepsilon$ Anfangsbelegung Membran 2: leer

$R_1 = \{a \to a(b, in_2)(c, in_2)(c, in_2), aa \to (a, out)(a, out)\}$

Reaktionsregeln in Membran 1. *in, out:* Zielmembranen

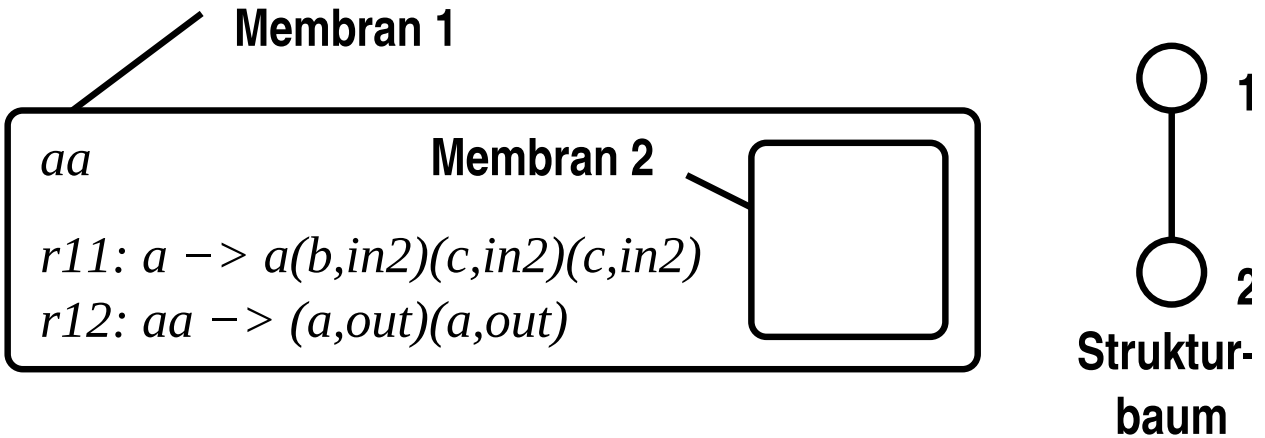
$R_2 = \emptyset$ Reaktionsregeln in Membran 2: keine

$i_O = 2$ Nummer der **Ausgabemembran**

$\Longrightarrow$ Wie arbeitet dieses System und was berechnet es?

Beispielsystem – Arbeitsweise

Initialkonfiguration

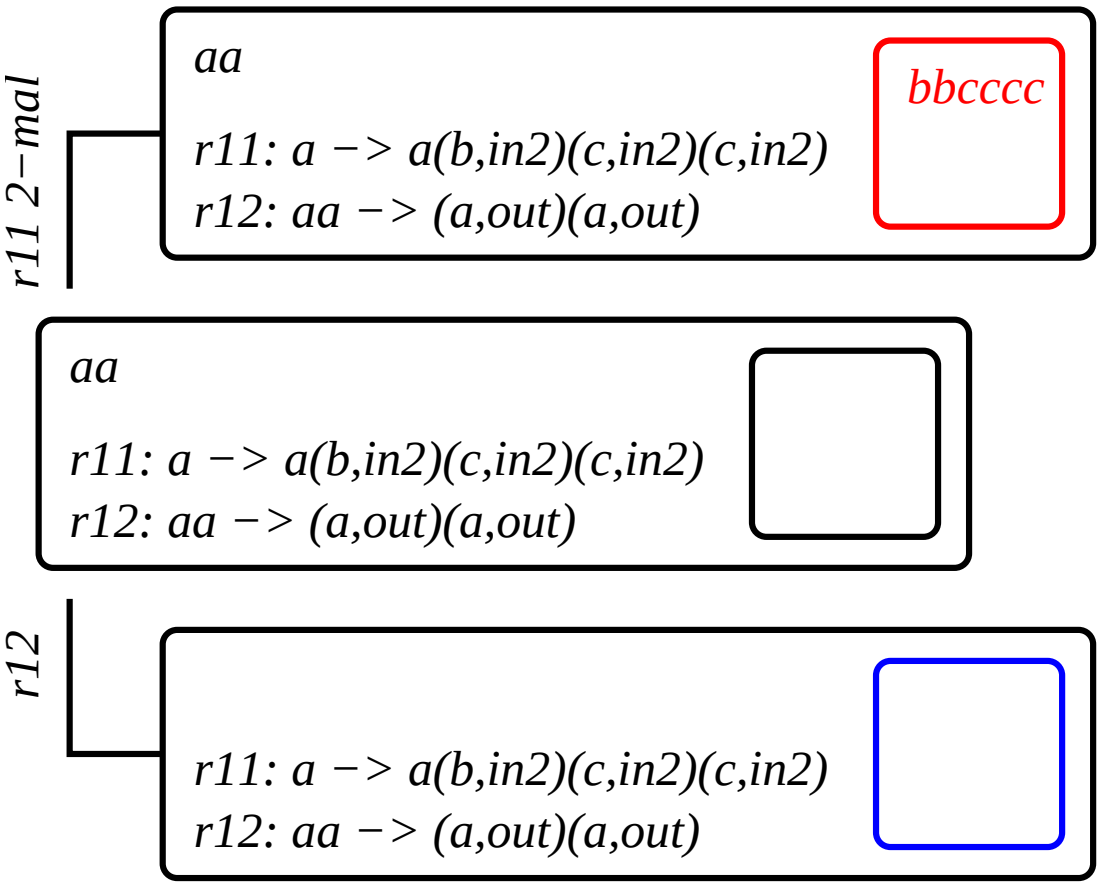


r11: a –> a + b + 2c, wobei die b und 2c in Membran 2 wandern
r12: 2a –> 2a, beide a nach außen abgegeben

$\Longrightarrow$ Maximal parallele Ausführung der Reaktionsregeln in allen Membranen

Beispielsystem – Arbeitsweise

Ableitungsbaum nach erstem Reaktionsschritt

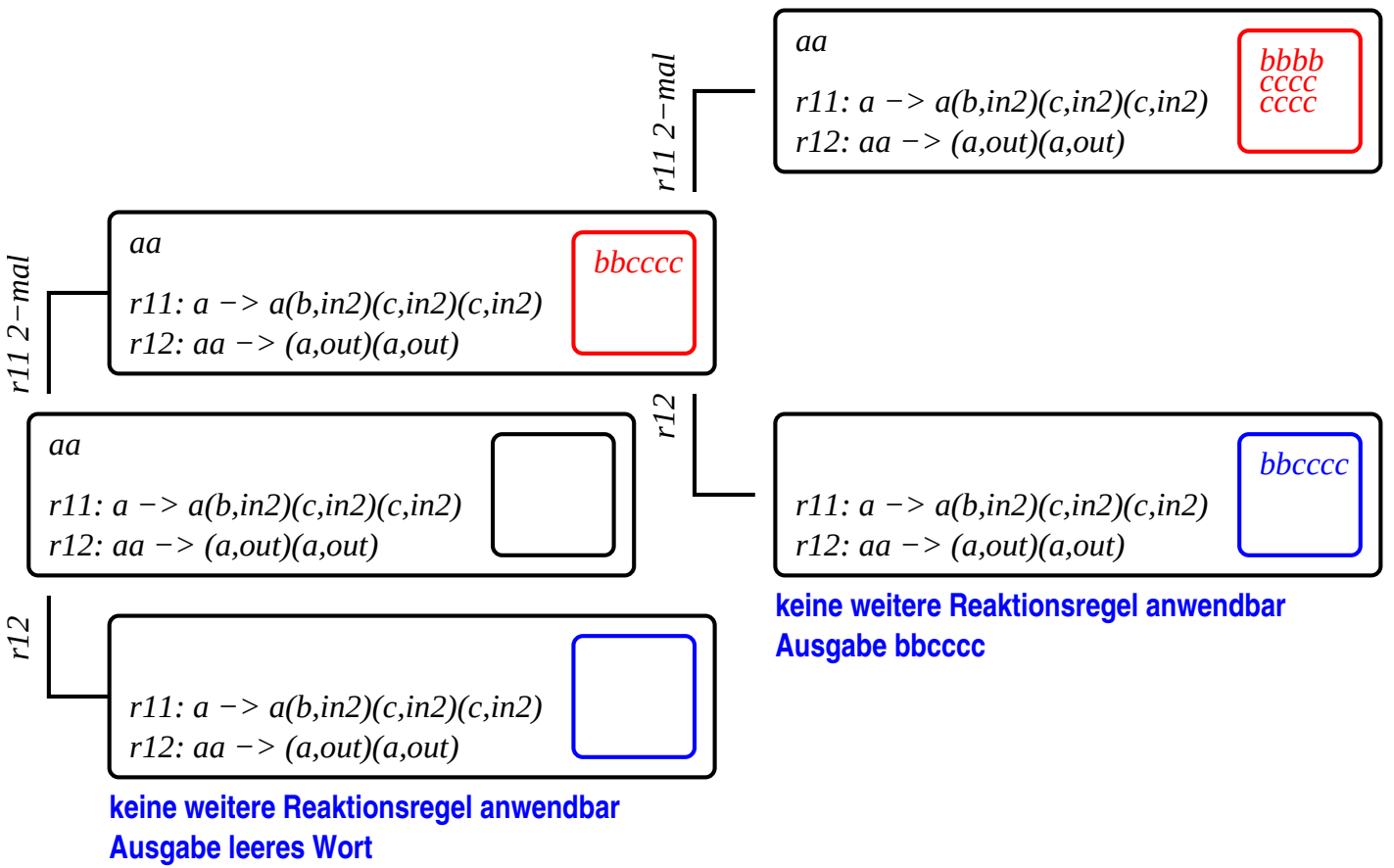


keine weitere Reaktionsregel anwendbar

Ausgabe leeres Wort

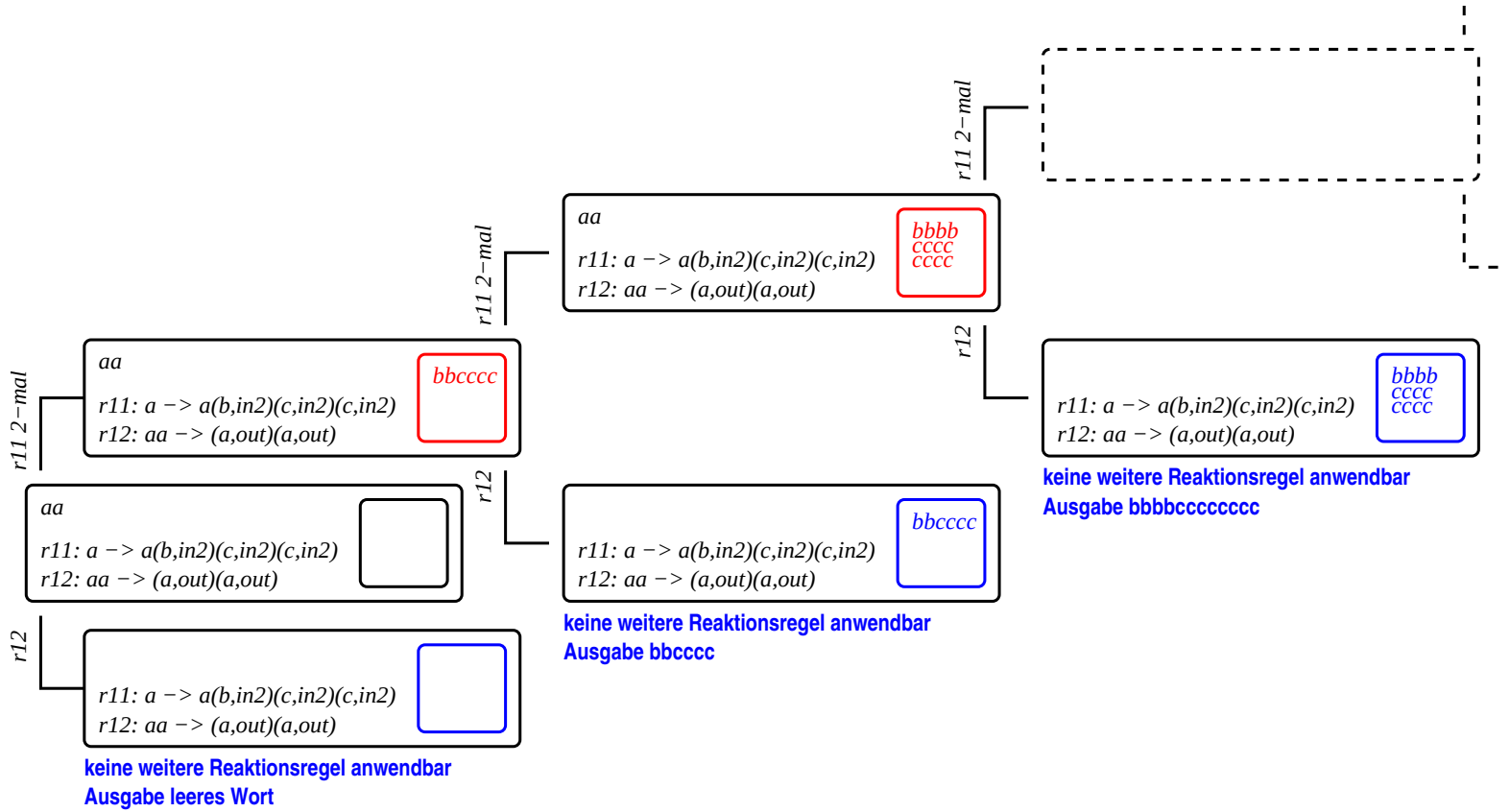
Beispielsystem – Arbeitsweise

Ableitungsbaum nach zweitem Reaktionsschritt



Beispielsystem – Arbeitsweise

Ableitungsbaum nach drittem Reaktionsschritt



$$L(\Pi) = \{\varepsilon, bbcccc, bbbbcccccccc, \ldots\} = \{b^{2n}c^{4n} \mid n \in \mathbb{N}\}$$
$$N(\Pi) = \{\mathrm{lgth}(w) \mid w \in L(\Pi)\} = \{6n \mid n \in \mathbb{N}\}$$

Membransystem mit variabler Membranstruktur

- Beispiel: Lösung von SAT (Andrei Păun, 2000)
- SAT (satisfiability): *Erfüllbarkeitstest der Aussagenlogik*; NP-vollständiges Problem
- *gegeben:* boolesche Formel in konjunktiver Normalform

$$\begin{aligned} \Gamma &= C_1 \wedge \ldots \wedge C_m \quad (m \text{ Klauseln}), m \geq 1 \\ C_i &= (y_{i,1} \vee \ldots \vee y_{i,p_i}) \quad \text{für } p_i \geq 1, i = 1, \ldots, m, \text{ wobei} \\ & \quad \text{Literale } y_{i,j} \in \{x_k, \overline{x_k}\}, k = 1, \ldots, n \ (n \text{ Variablen}) \end{aligned}$$

  Jede Variable ist mit true (t) oder false (f) belegbar.
- *gefragt:* Ist unter den 2^n Variablenbelegungen mindestens eine, so dass der gesamte boolesche Ausdruck Γ den Wert true annimmt?
- Beispiel: $\Gamma = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$ ist erfüllbar.

$\Longrightarrow$ Jede boolesche Formel lässt sich effizient algorithmisch in konjunktive Normalform bringen.

Membransystem mit variabler Membranstruktur

- Beispiel: Lösung von SAT (Andrei Păun, 2000)
- SAT (satisfiability): *Erfüllbarkeitstest der Aussagenlogik*; NP-vollständiges Problem
- *gegeben:* boolesche Formel in konjunktiver Normalform

$$\begin{aligned} \Gamma &= C_1 \wedge \ldots \wedge C_m \quad (m \text{ Klauseln}), m \geq 1 \\ C_i &= (y_{i,1} \vee \ldots \vee y_{i,p_i}) \quad \text{für } p_i \geq 1, i = 1, \ldots, m, \text{ wobei} \\ & \quad \text{Literale } y_{i,j} \in \{x_k, \overline{x_k}\}, k = 1, \ldots, n \ (n \text{ Variablen}) \end{aligned}$$

  Jede Variable ist mit true (t) oder false (f) belegbar.
- *gefragt:* Ist unter den 2^n Variablenbelegungen mindestens eine, so dass der gesamte boolesche Ausdruck Γ den Wert true annimmt?
- Beispiel: $\Gamma = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$ ist erfüllbar.

$\Longrightarrow$ Jede boolesche Formel lässt sich effizient algorithmisch in konjunktive Normalform bringen.

Membransystem mit variabler Membranstruktur

- Beispiel: Lösung von SAT (Andrei Păun, 2000)
- SAT (satisfiability): *Erfüllbarkeitstest der Aussagenlogik*; NP-vollständiges Problem
- *gegeben:* boolesche Formel in konjunktiver Normalform

$$\begin{aligned} \Gamma &= C_1 \wedge \ldots \wedge C_m \quad (m \text{ Klauseln}), m \geq 1 \\ C_i &= (y_{i,1} \vee \ldots \vee y_{i,p_i}) \quad \text{für } p_i \geq 1, i = 1, \ldots, m, \text{ wobei} \\ & \quad \text{Literale } y_{i,j} \in \{x_k, \overline{x_k}\}, k = 1, \ldots, n \ (n \text{ Variablen}) \end{aligned}$$

  Jede Variable ist mit true (t) oder false (f) belegbar.
- *gefragt:* Ist unter den 2^n Variablenbelegungen mindestens eine, so dass der gesamte boolesche Ausdruck Γ den Wert true annimmt?
- Beispiel: $\Gamma = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$ ist erfüllbar.

$\Longrightarrow$ Jede boolesche Formel lässt sich effizient algorithmisch in konjunktive Normalform bringen.

Membransystem mit variabler Membranstruktur

- Beispiel: Lösung von SAT (Andrei Păun, 2000)
- SAT (satisfiability): *Erfüllbarkeitstest der Aussagenlogik*; NP-vollständiges Problem
- *gegeben:* boolesche Formel in konjunktiver Normalform

$$\begin{aligned} \Gamma &= C_1 \wedge \ldots \wedge C_m \quad (m \text{ Klauseln}), m \geq 1 \\ C_i &= (y_{i,1} \vee \ldots \vee y_{i,p_i}) \quad \text{für } p_i \geq 1, i = 1, \ldots, m, \text{ wobei} \\ & \quad \text{Literale } y_{i,j} \in \{x_k, \overline{x_k}\}, k = 1, \ldots, n \ (n \text{ Variablen}) \end{aligned}$$

  Jede Variable ist mit true (t) oder false (f) belegbar.
- *gefragt:* Ist unter den 2^n Variablenbelegungen mindestens eine, so dass der gesamte boolesche Ausdruck Γ den Wert true annimmt?
- Beispiel: $\Gamma = (x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$ ist erfüllbar.

$\Longrightarrow$ Jede boolesche Formel lässt sich effizient algorithmisch in konjunktive Normalform bringen.

Algorithmische Idee

- massiv-datenparalleles Brute-Force über Membranstrukturen
- Verwendung aktiver Membrane
  (Teilen und Auflösen von Reaktionsräumen)
- Membrane ladungsbehaftet machen $(+, -, 0)$,
  Ladung in Reaktionsregeln einbeziehen
- Aufbauphase (alle Belegungen erzeugen),
  dann Selektionsphase (erfüllende auswählen)

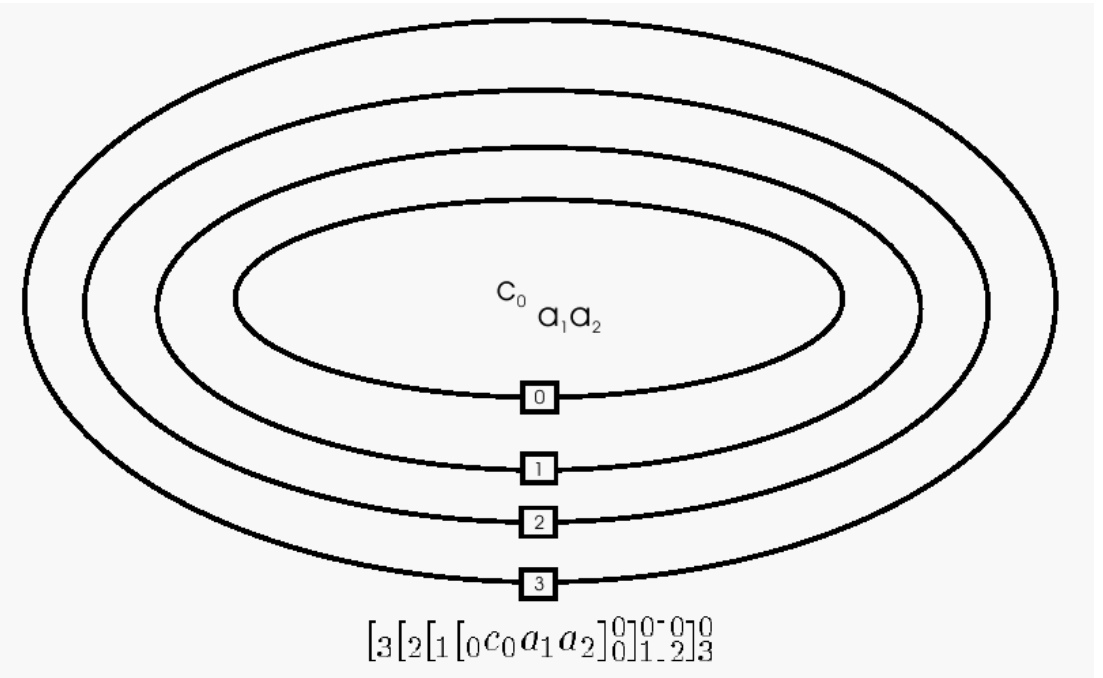
Aufbauphase

- Beginne mit $m + 2$ ineinander geschachtelten Reaktionsräumen, Initialmolekül zum Zählen der Aufbauschritte im innersten
- Fortgesetztes Teilen der Reaktionsräume erzeugt mindestens 2^n Membranstrukturen, wobei die jeweils enthaltenen Moleküle eine Variablenbelegung kodieren. Dabei Membranstrukturen klauselweise ineinander verschachtelt.
- Steuere diesen Vorgang durch ein mit jeder Teilung schrittweise aufzubauendes (Zähler)Molekül, das nach Abschluss der Aufbauphase die Selektionsphase einleitet.

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

Startkonfiguration

- Initialmoleküle $c_0 a_1 a_2$, wobei c_0 Zähler für Aufbauphase, und die a Platzhalter für Belegungen jeder booleschen Variable
- Konfiguration: $[_3[_2[_1[_0 c_0 a_1 a_2]^0_0]^0_1]^0_2]^0_3$

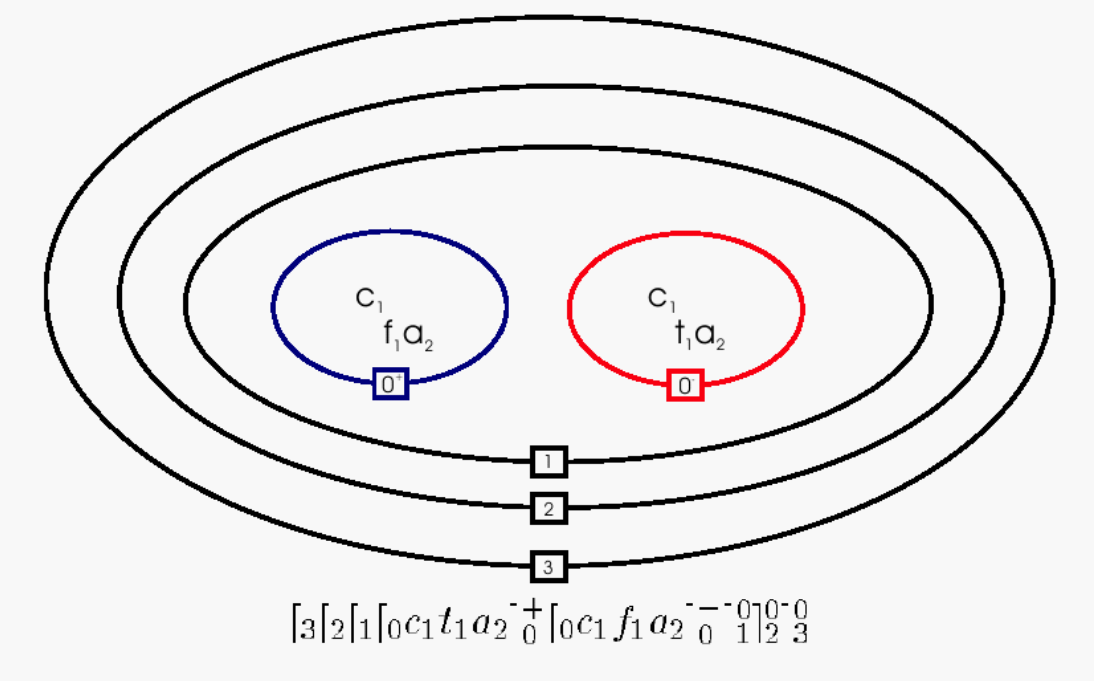


anwendbare Regeln:

- Aufbauzähler inkrementieren, Regel $[_0 c_0 \rightarrow c_1]^0_0$
- Teilung: $[_0 a_1]^0_0 \rightarrow [_0 t_1]^+_0 [_0 f_1]^-_0$, d.h. Belegungen für x_1 fixieren

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

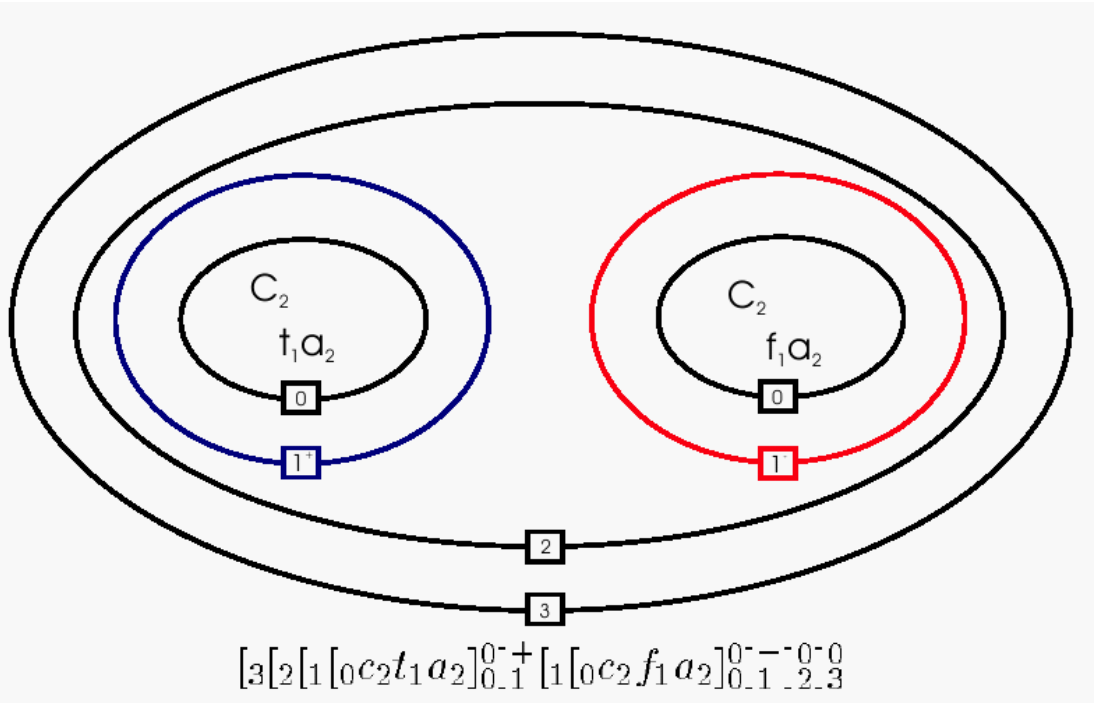
Konfiguration nach 1. Schritt



Aufbau der Membransubstrukturen: 1-Membran teilt sich in zwei gegensätzlich geladene Membranen, die die 0-Membranen jeweils in sich aufnehmen (einschachteln). Ladung der eingeschachtelten 0-Membranen wird neutral. Regeln: $[_1[_0]^+_0[_0]^-_0]^0_1 \rightarrow [_1[_0]^0_0]^+_1[_1[_0]^0_0]^-_1$ und $[_0c_1 \rightarrow c_2]^0_0$ (Aufbauzähler inkr.)

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

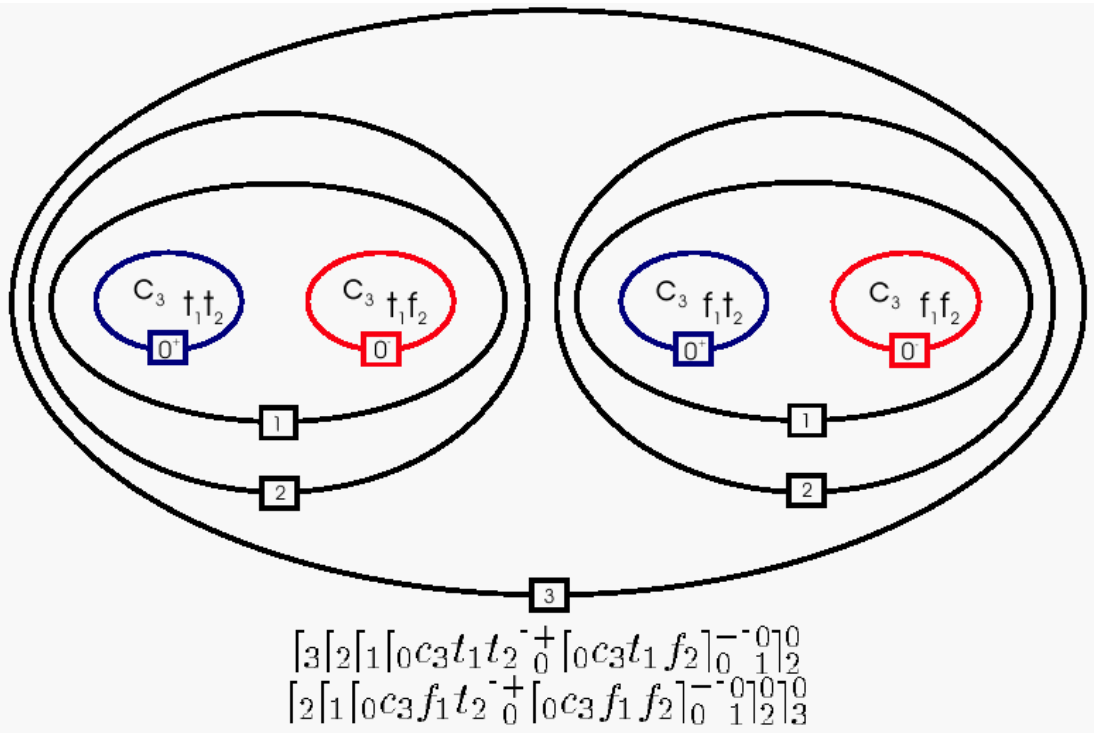
Konfiguration nach 2. Schritt



Erneute Teilung der 0-Membranen zum Fixieren der Belegungen für x_2, Regel: $[_0 a_2]_0^0 \rightarrow [_0 t_2]_0^+ [_0 f_2]_0^-$, simultan Einschachtelung der nächsten Membransubstruktur (2-Membran), Regel: $[_2 [_1]_1^+ [_1]_1^-]_2^0 \rightarrow [_2 [_1]_1^0]_2^0 [_2 [_1]_1^0]_2^0$ sowie Aufbauzähler inkrementieren, Regel: $[_0 c_2 \rightarrow c_3]_0^0$

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

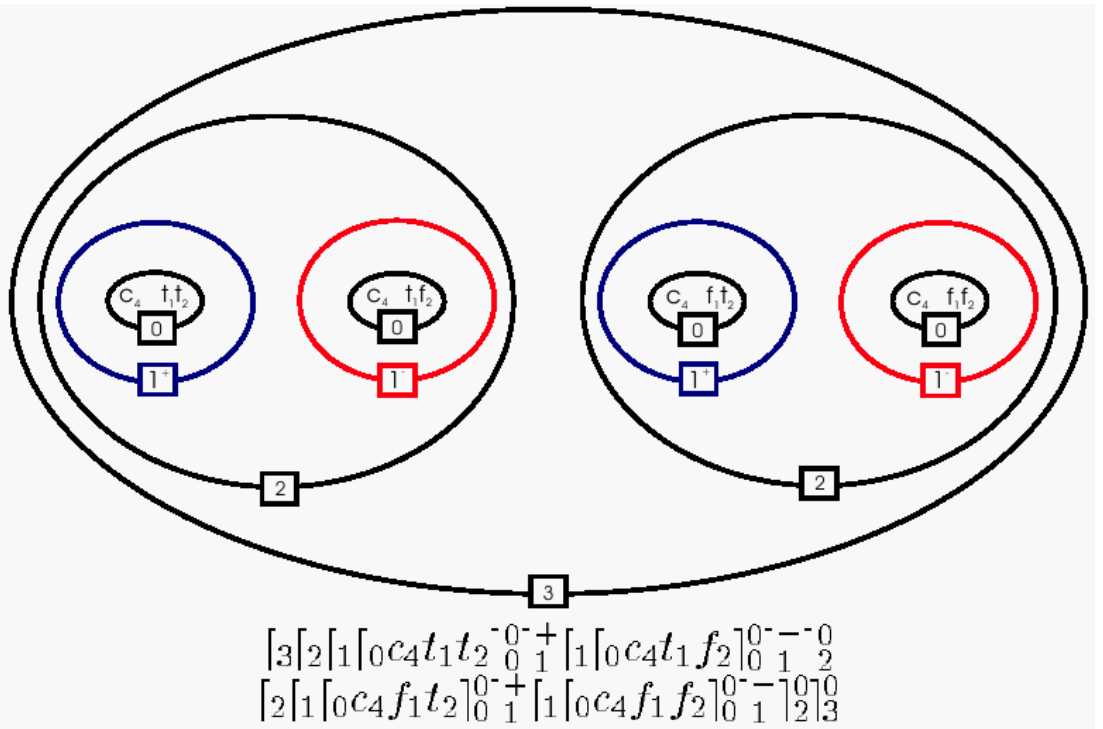
Konfiguration nach 3. Schritt



1-Membranen teilen sich in jeweils zwei gegensätzlich geladene Membranen, die die 0-Membranen umschließen. Ladung der eingeschachtelten 0-Membranen wird neutral, Regel:
$[_1[_0]_0^+[_0]_0^-]_1^0 \rightarrow [_1[_0]_0^0]_1^+[_1[_0]_0^0]_1^-$ sowie Aufbauzähler inkr.: $[_0 c_3 \rightarrow c_4]_0^0$

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

Konfiguration nach 4. Schritt

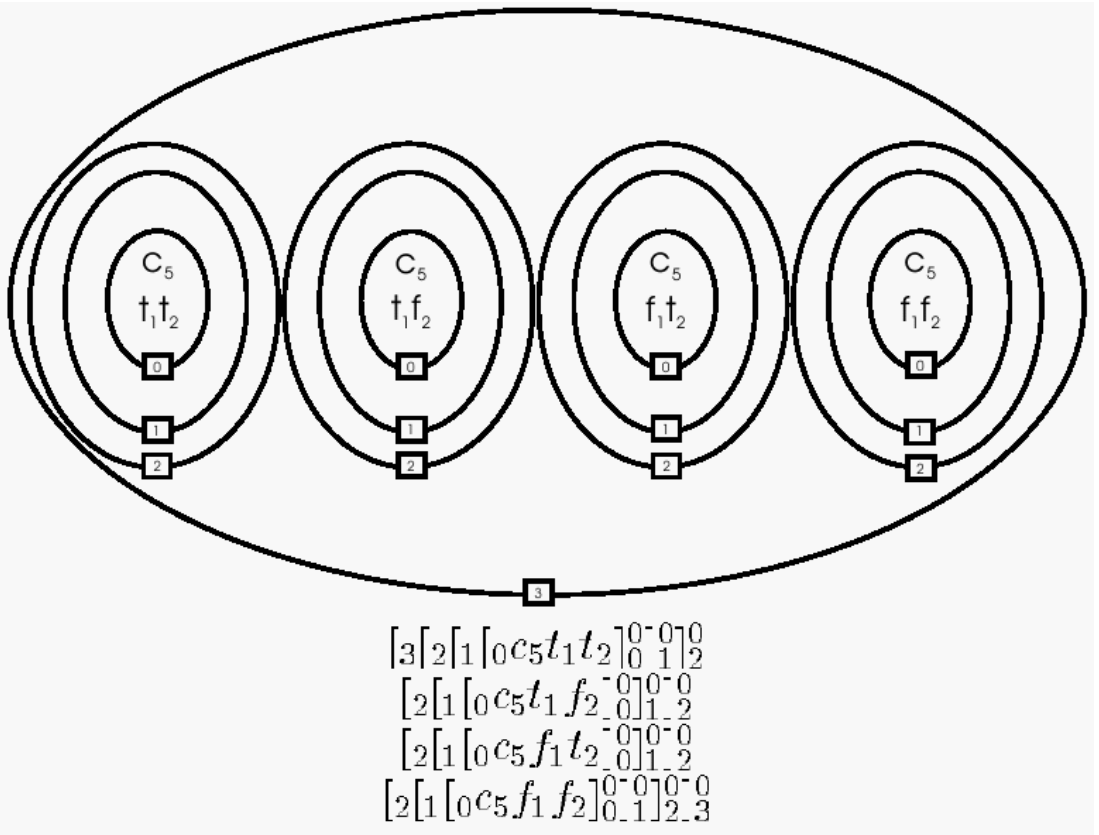


$$[_3[_2[_1[_0 c_4 t_1 t_2\ ]^{0-}_{0}\ ]^{+}_{1}\ [_1[_0 c_4 t_1 f_2\ ]^{0-}_{0}\ ]^{-}_{1}\ ]^{0}_{2}$$
$$[_2[_1[_0 c_4 f_1 t_2\ ]^{0-}_{0}\ ]^{+}_{1}\ [_1[_0 c_4 f_1 f_2\ ]^{0-}_{0}\ ]^{-}_{1}\ ]^{0}_{2}\ ]^{0}_{3}$$

2-Membranen teilen sich in jeweils zwei Membranen, die die 1-Membranen umschließen. Ladung der nunmehr eingeschachtelten 1-Membranen wird neutral, Regel: $[_2[_1]^{+}_{1}\,[_1]^{-}_{1}\,]^{0}_{2} \rightarrow [_2[_1]^{0}_{1}]^{0}_{2}[_2[_1]^{0}_{1}]^{0}_{2}$ sowie Aufbauzähler inkr.: $[_0 c_4 \rightarrow c_5]^{0}_{0}$

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

Konfiguration nach 5. Schritt



$$[_3[_2[_1[_0 c_5 t_1 t_2]_{0}^{0\text{-}0}]_{0\_1}^{0}]_{2}^{0}$$
$$[_2[_1[_0 c_5 t_1 f_{2\_0}]_{0}^{0\text{-}0}]_{1\_2}^{0\text{-}0}$$
$$[_2[_1[_0 c_5 f_1 t_{2\_0}]_{0}^{0\text{-}0}]_{1\_2}^{0\text{-}0}$$
$$[_2[_1[_0 c_5 f_1 f_2]_{0\_1}^{0\text{-}0}]_{2\_3}^{0\text{-}0}$$

Aufbauphase jetzt abgeschlossen, Selektionsphase beginnt. Zähler c_5 wandelt sich zum Klauselerfüllbarkeitsmarker t, innerste Membranen (0-Membranen) werden aufgelöst, Regel: $[_0 c_5]_0^0 \rightarrow t$.

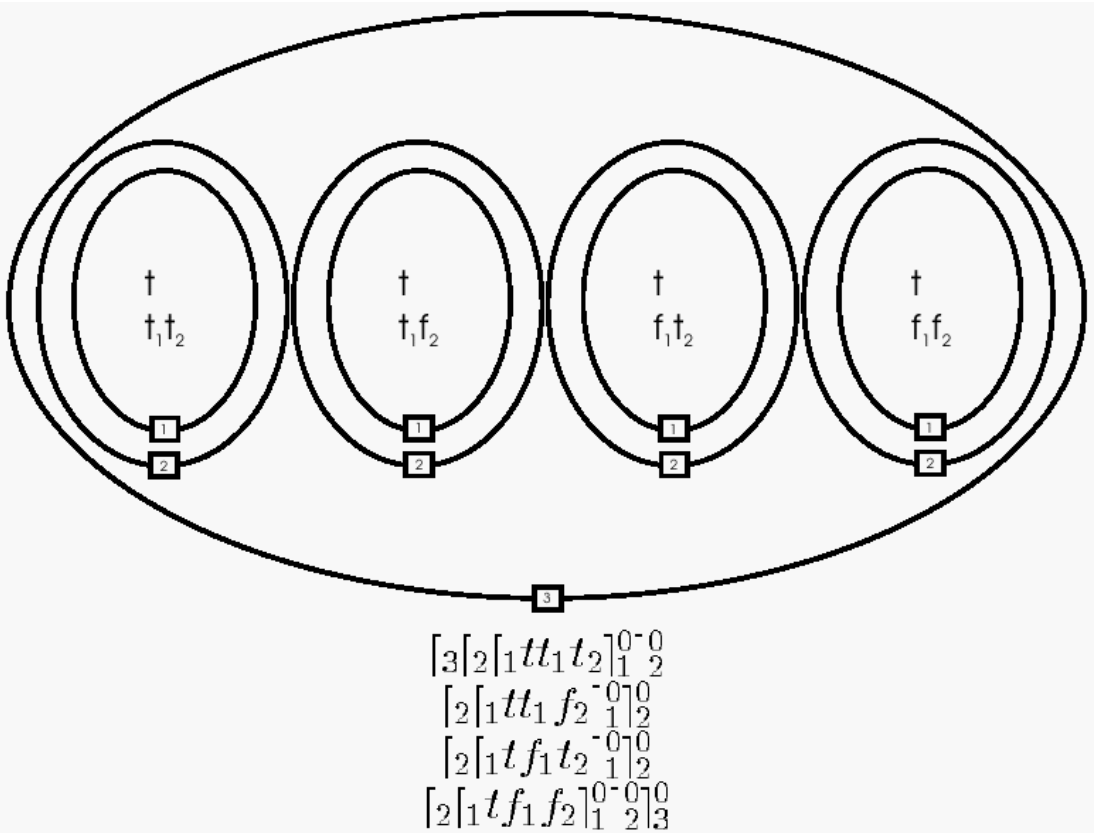
Selektionsphase

Selektionsphase

- Membranstrukturen klauselweise von innen nach außen daraufhin überprüfen, ob jeweilige Klausel erfüllbar. Falls ja: Membran aufgelöst und erfüllende Belegungen (Moleküle) in nächstäußere Membran abgeben. Falls nein: keine Membranauflösung, Belegung bleibt verkapselt.
- Ladungen der Membranen benutzt, um Überprüfungsende der Klauseln anzuzeigen (dann geladen)
- Kann ein true-kodierendes Molekül innerhalb von $2n + 2m + 1$ Reaktionsschritten die Skin-Membran verlassen: Lösung „ja“, sonst Lösung „nein“.

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

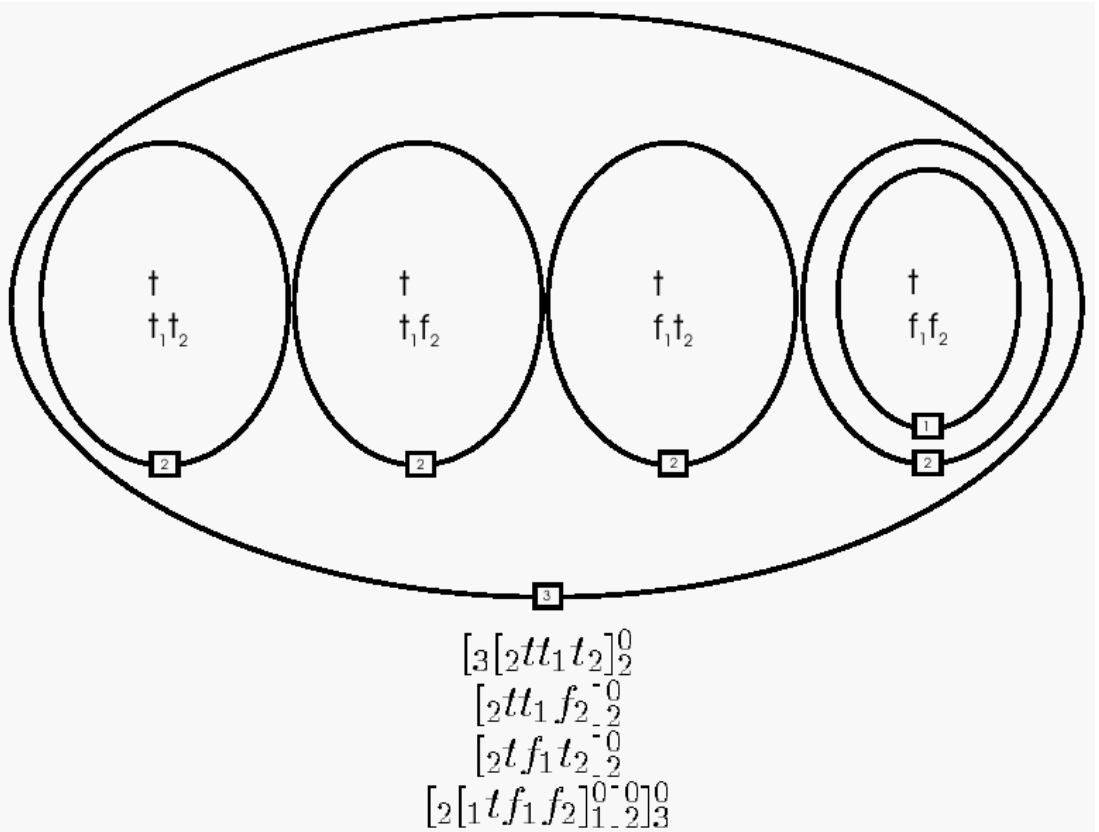
Konfiguration nach 6. Schritt



Prüfung der ersten Klausel $(x_1 \vee x_2)$ auf Erfüllbarkeit in den 1-Membranen. Bei Erfüllbarkeit umgebende 1-Membranen aufgelöst, Regeln: $[_1 t_1]^0_1 \rightarrow t_1$ (für $x_1 = t$) und $[_1 t_2]^0_1 \rightarrow t_2$ (für $x_2 = t$)

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

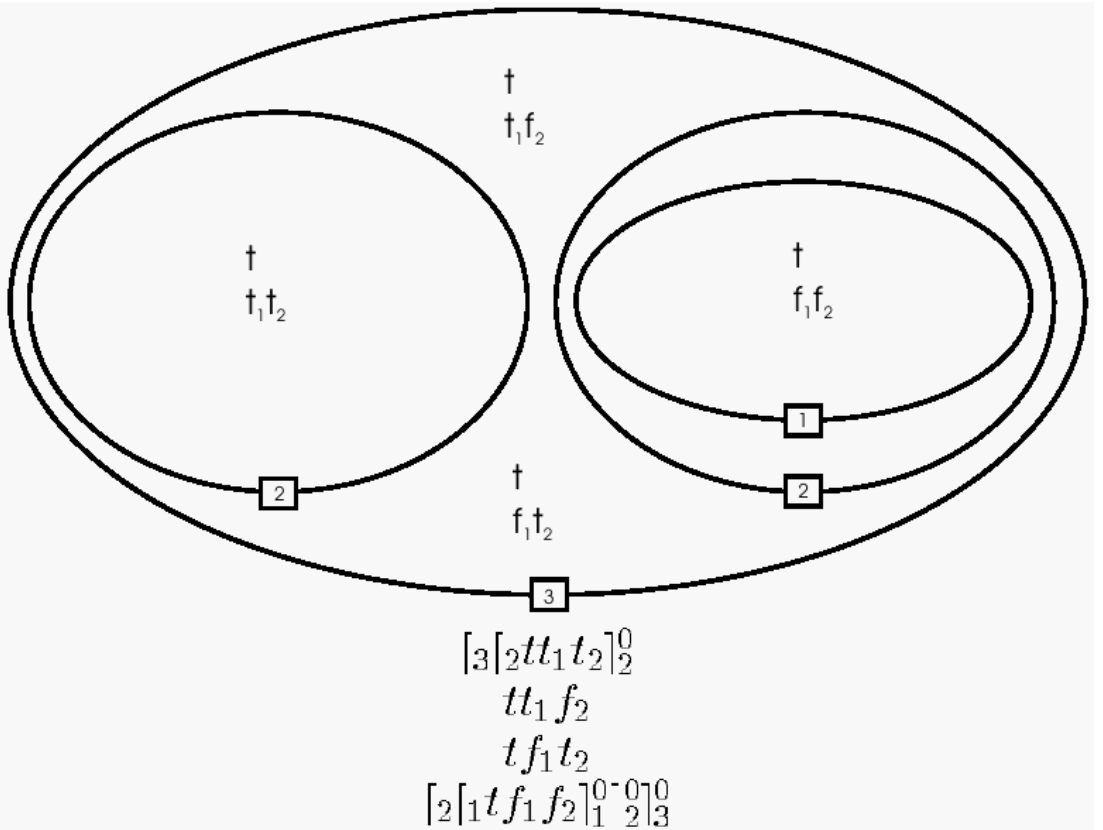
Konfiguration nach 7. Schritt



Prüfung der zweiten Klausel $(\overline{x_1} \vee \overline{x_2})$ auf Erfüllbarkeit in den 2-Membranen. Bei Erfüllbarkeit umgebende 2-Membranen aufgelöst, Regeln: $[_1 f_1]^0_1 \rightarrow f_1$ (für $x_1 = f$) und $[_1 f_2]^0_1 \rightarrow f_2$ (für $x_2 = f$)

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

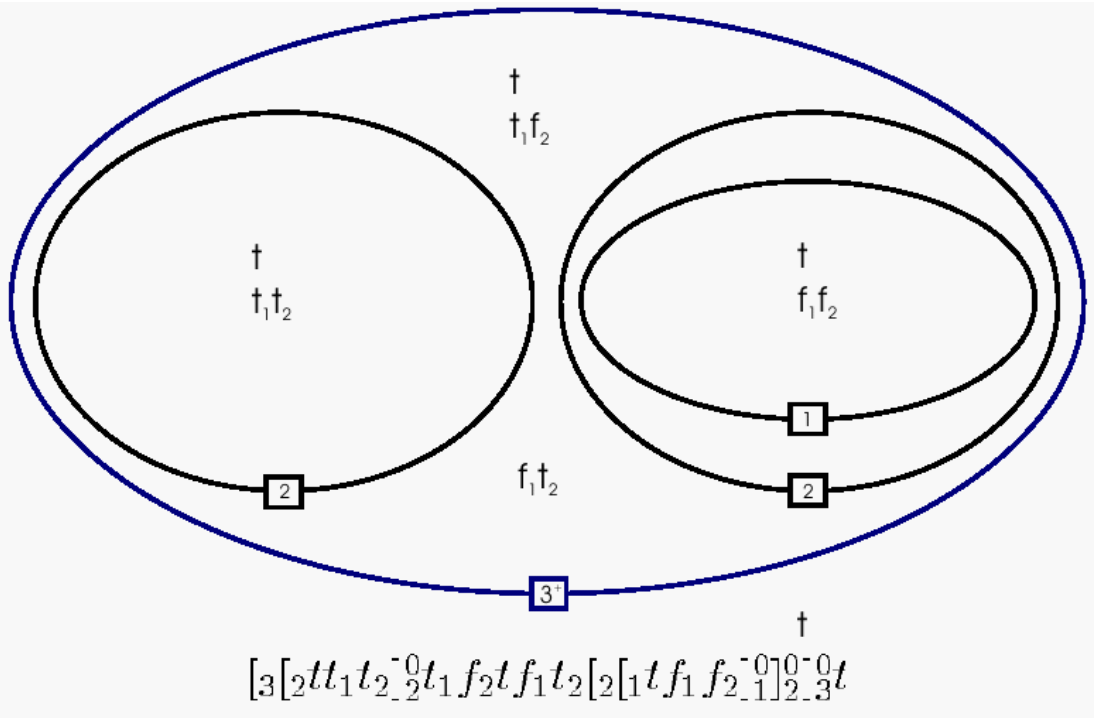
Konfiguration nach 8. Schritt



Eines der beiden Moleküle, die für das Erfüllen aller Klauseln stehen und sich jetzt in der Skin-Membran (3-Membran) befinden, zur Anzeige der Lösung „ja“ nach außen abgeben: $[_3 t]_3^0 \rightarrow [_3]_3^+ t$.

Beispiel $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2})$

Endkonfiguration nach 9. Schritt



Lösung „ja“ liegt vor. Anzahl Regeln linear zu m und n.
Anzahl Zeitschritte ebenfalls linear zu m und n.

Exponentieller Ressourcenverbrauch auf Anzahl Membranen und Moleküle (Platz) verschoben.

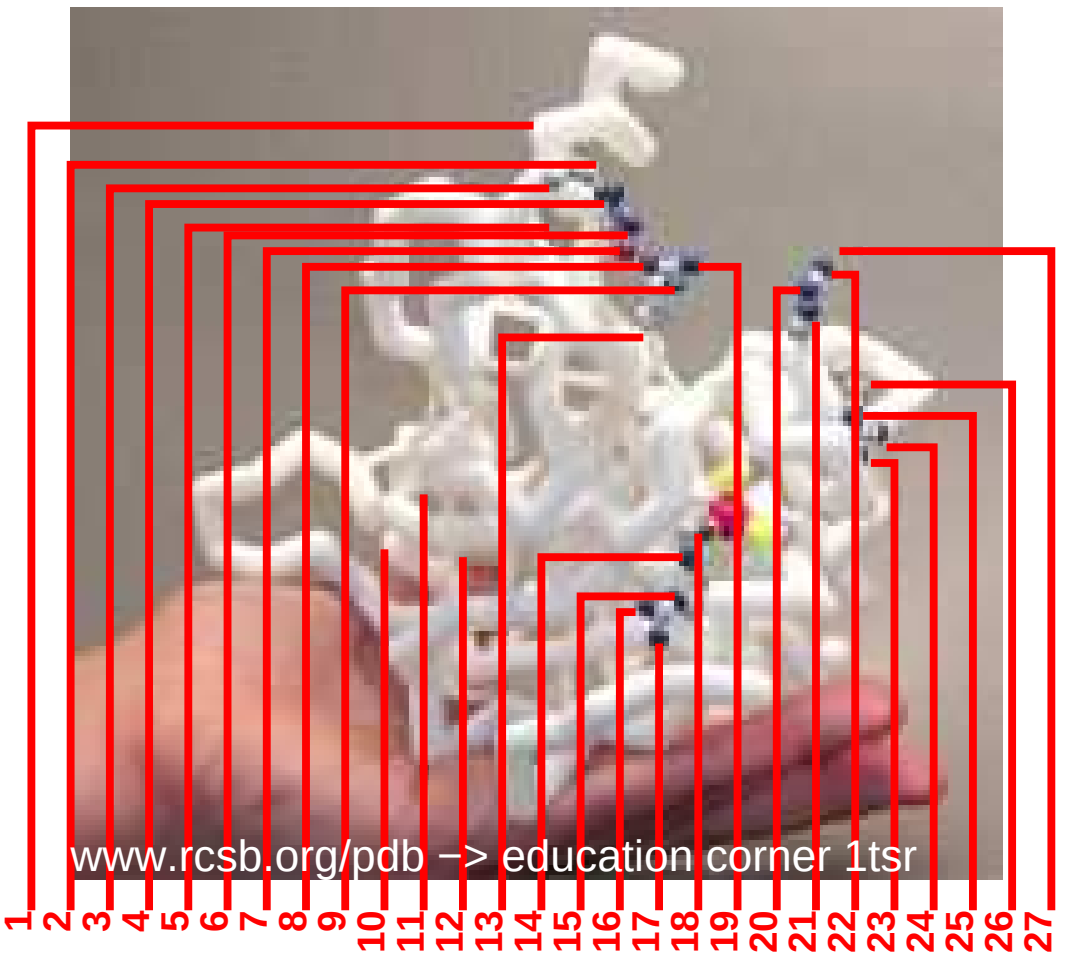
Entwurfsziele für Zellsignalsysteme

- innere Struktur von Molekülen und ihre chemischen Veränderungen erfassen
- Ablaufsteuerung entsprechend (realistischer) Reaktionskinetiken

- Einzelmolekülbetrachtung
- Modellierung durch Multimengen
- deterministische Arbeitsweise (Priorisierung der Reaktionsregeln)
- Simulation des Systemverhaltens
- Anwendung in der Systembiologie
- spätere Kopplung von Reaktionsräumen (Zellen, Membrane) durch Transduktionskanäle

Combinatorial Explosion of Protein Activation States

- Tumor suppressor protein p53: 27 phosphorylation sites
- Up to $2^{27} = 134,217,728$ distinguishable activation states
- Each state: individual constituent of reaction network



Cell Signalling Module

Characteristics

- Intracellular reaction network acting as functional unit
- Composed of proteins carrying phosphorylation sites
- Interactions between individual activation states

Facts

- Dynamical behaviour essential to understand function
- Often partially unknown
- Reconstruction as challenging task in systems biology
- Reverse engineering by integrative approach

Idea to manage complexity:
Capturing each protein by a specific string-object
instead of separate species per activation state

Specification of String-Objects

Assuming two alphabets: V (for protein names), V' (for protein properties); w.l.o.g $\#, \neg, * \notin V \cup V'$

Syntax for string-objects by regular set

$$S = V^+ \cdot \left(\{\#\} \cdot ((V')^+ \cup \{\neg\} \cdot (V')^+ \cup \{*\})\right)^*$$

Protein properties

- x: property x present (e.g. specific phosphate attached)
- $\neg x$: property x absent (e.g. specific phosphate removed)
- $*$: placeholder for arbitrary property setting

Examples

- $\mathrm{prot1\#p\#} * \#\neg\mathrm{p}$ (subsumes activation states of prot1)
- $\mathrm{KaiC\#\neg KaiA\#KaiB\#4}$ (prot. complex, 4 ligands attached)

$\Longrightarrow$ **Application of reaction rules requires string matching**

Π_{CSM}: System Components

Let $\langle S \rangle$ be the set of all multisets over S.

$$\Pi_{\mathrm{CSM}} = (V, V', R_1, \ldots, R_r, \mathrm{f}_1, \ldots, \mathrm{f}_r, A, \mathrm{C}, \Delta\tau)$$

with

$R_i \in \langle S \rangle \times \langle S \rangle$ ..... is a reaction rule composed of two finite multisets

$\mathrm{f}_i : \langle S \rangle \longrightarrow \mathbb{N}$ ..... is a function corresponding to discrete kinetics of reaction R_i

$A \in \langle S \rangle$ ..... is a multiset of axioms representing the initial molecular configuration

$\mathrm{C} \in \mathbb{R}_+$ spatial capacity of the module (vessel or compartment)

$\Delta\tau \in \mathbb{R}_+$ ..... time discretisation interval

Π_{CSM}: Matching

Let S be a string-object syntax. Two string-objects match iff there is at least one common wild card-free representation:

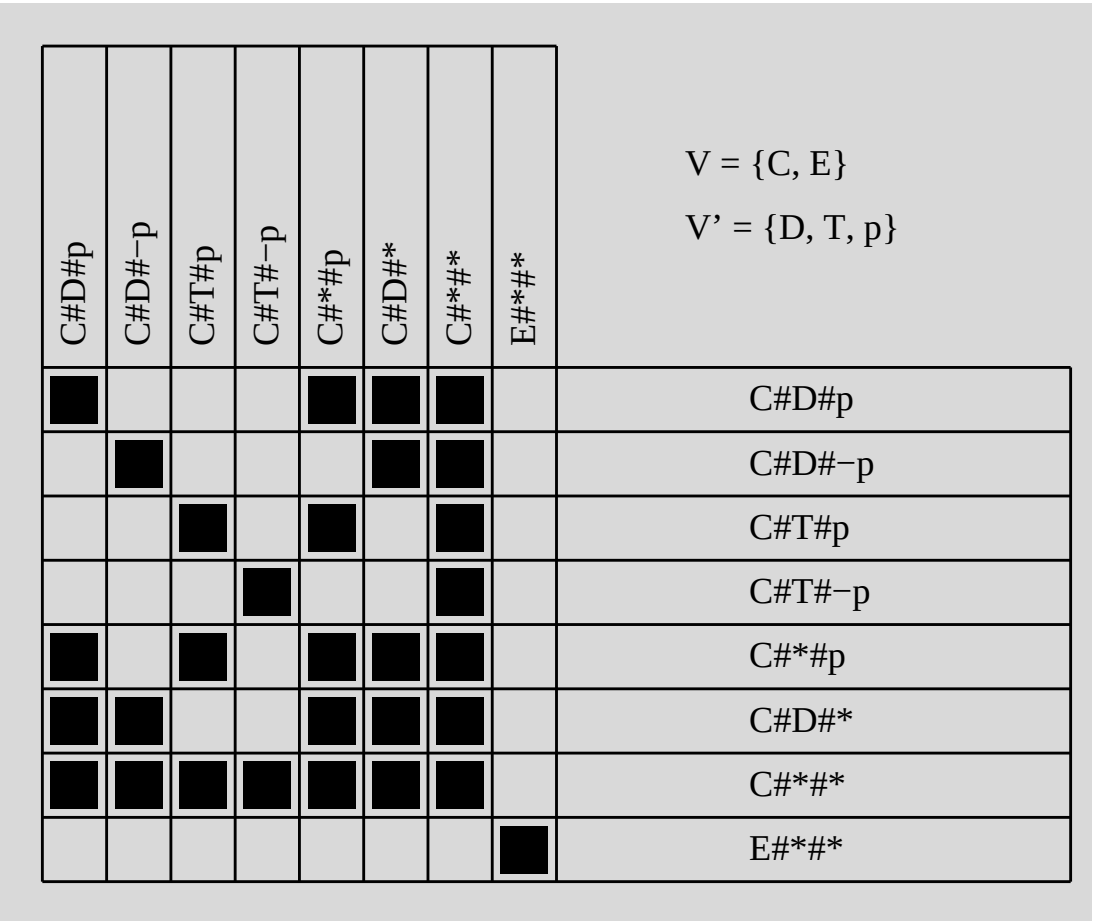
$$
\begin{aligned}
\mathit{Match} &\subseteq S \times S \\
\mathit{Match} &= \bigcup_{m \in \mathbb{N}} \{(p\#p_1\#p_2 \ldots \#p_m,\ s\#s_1\#s_2 \ldots \#s_m) \mid (p = s) \wedge \\
&\qquad \forall j \in \{1, \ldots, m\} : [(p_j = s_j) \vee (p_j = *) \vee (s_j = *) \vee \\
&\qquad\qquad ((p_j = \neg q) \wedge (s_j \neq q)) \vee ((s_j = \neg q) \wedge (p_j \neq q))]\}
\end{aligned}
$$

- *Match* is a symmetric relation
- Requires minimal similarity between string-objects with incomplete information
- Uncertainty interpreted as arbitrary replacement by available properties

Π_{CSM}: Matching

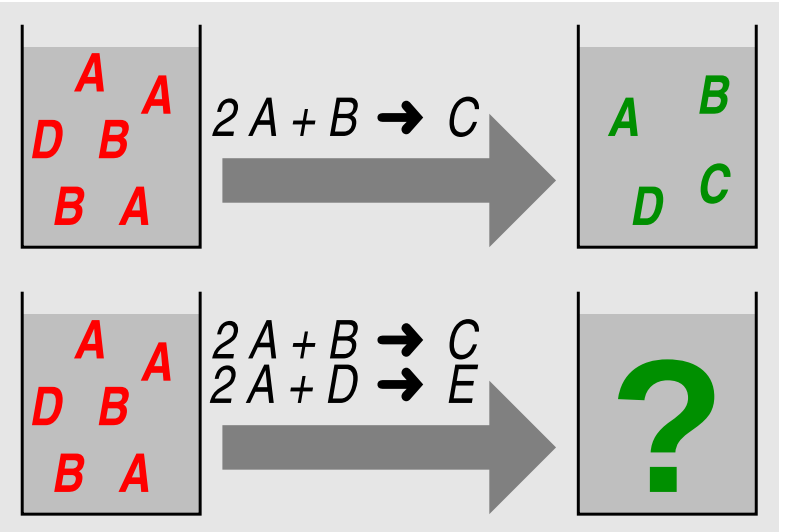
Let S be a string-object syntax. Two string-objects match iff there is at least one common wild card-free representation:

Example



Π_{CSM}: Dynamical System Behaviour (I)

- Successive progression of configuration $L_t \in \langle S \rangle$ over time $t \in \mathbb{N}$ starting from axioms A
- $\Delta\tau$: span between t and $t+1$
- Conflict handling by prioritisation of reaction rules



$$L_0 = L_{0,0} = A$$

$$L_{t,1} = \begin{cases} L_{t,0} \ominus Reactants_{t,1} \uplus Products_{t,1} & \text{if } Reactants_{t,1} \subseteq L_{t,0} \\ L_{t,0} & \text{otherwise} \end{cases}$$

$$\vdots$$

$$L_{t+1} = L_{t,r} = \begin{cases} L_{t,r-1} \ominus Reactants_{t,r} \uplus Products_{t,r} & \text{if } Reactants_{t,r} \subseteq L_{t,r-1} \\ L_{t,r-1} & \text{otherwise} \end{cases}$$

Π_{CSM}: Dynamical System Behaviour (II)

Estimation of multisets $Reactants_{t,j}$ and $Products_{t,j}$ at time t concerning reaction $R_j = (A_j, B_j) \in \langle S \rangle \times \langle S \rangle$ denoted

$$A_j(a_1)\ a_1 + \ldots + A_j(a_p)\ a_p \longrightarrow B_j(b_1)\ b_1 + \ldots + B_j(b_q)\ b_q$$

includes

- Matching between string-objects in L_t and those in A_j
- Consideration of stoichiometry captured by multisets A_j, B_j
- Evaluation of kinetic law expressed by scalar function f_j

$$Reactants_{t,j} = \biguplus_{e_1 \in Match(a_1)} \ldots \biguplus_{e_p \in Match(a_p)} \mathrm{f}_j\left(\{(e_1,\infty),\ldots,(e_p,\infty)\} \cap L_{t,j-1}\right) \cdot \{(e_1, A_j(a_1)),\ldots,(e_p, A_j(a_p))\}$$

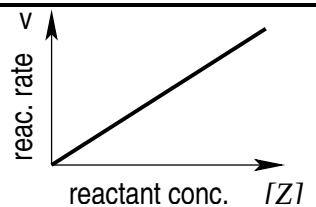
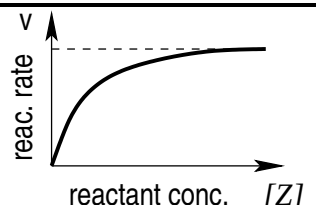
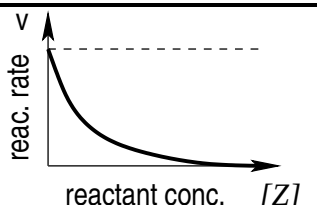
$$Products_{t,j} = \biguplus_{e_1 \in Match(a_1)} \ldots \biguplus_{e_p \in Match(a_p)} \mathrm{f}_j\left(\{(e_1,\infty),\ldots,(e_p,\infty)\} \cap L_{t,j-1}\right) \cdot \{(b_1, B_j(b_1)),\ldots,(b_q, B_j(b_q))\}$$

Π_{CSM}: Discrete Reaction Kinetics

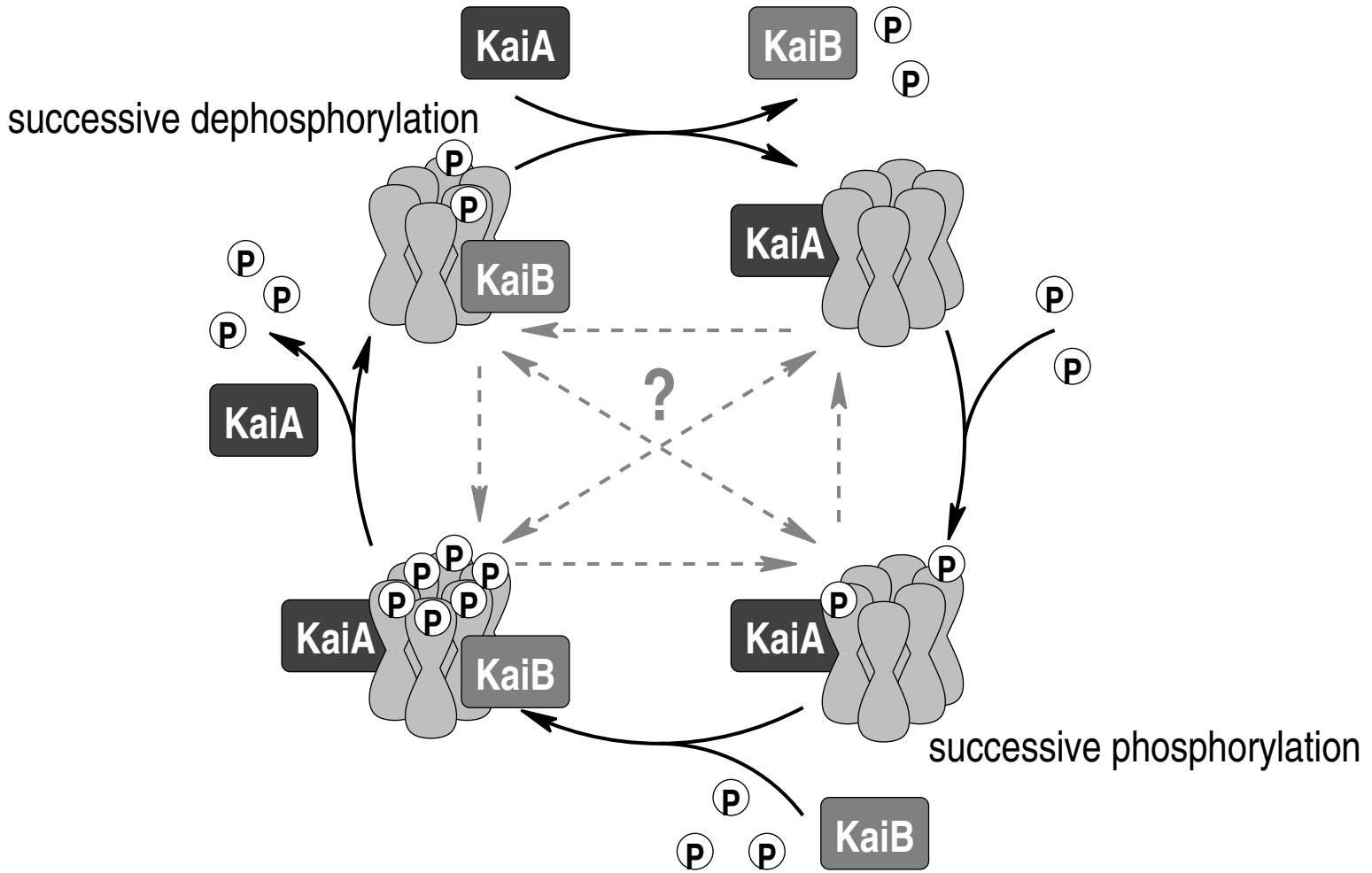
Scalar function f_j provides number of turns for application of reaction rule R_j. Rate constant: $k_j = \hat{k}_j \cdot C \cdot \Delta\tau$ (Euler).

$$f_j(L_t) = \left\lfloor k_j \prod_{\forall \alpha \in Match(A_j) \cap Match(L_t) : (R_j = (A_j, B_j))} \hat{f}(L_t(\alpha))^{|Match(A_j) \cap \{(\alpha, \infty)\}|} \right\rfloor$$

Selected kinetic laws $\hat{f}([Z])$

Kinetics	Activation	Repression
Mass-Action (no saturation)	 $\hat{f}([Z]) = [Z]$	–
Michaelis-Menten (saturation)	 $\hat{f}([Z]) = \frac{[Z]}{\Theta + [Z]}$	 $\hat{f}([Z]) = \left(1 - \frac{[Z]}{\Theta + [Z]}\right)$

KaiABC Circadian Oscillator: Reaction Cycle



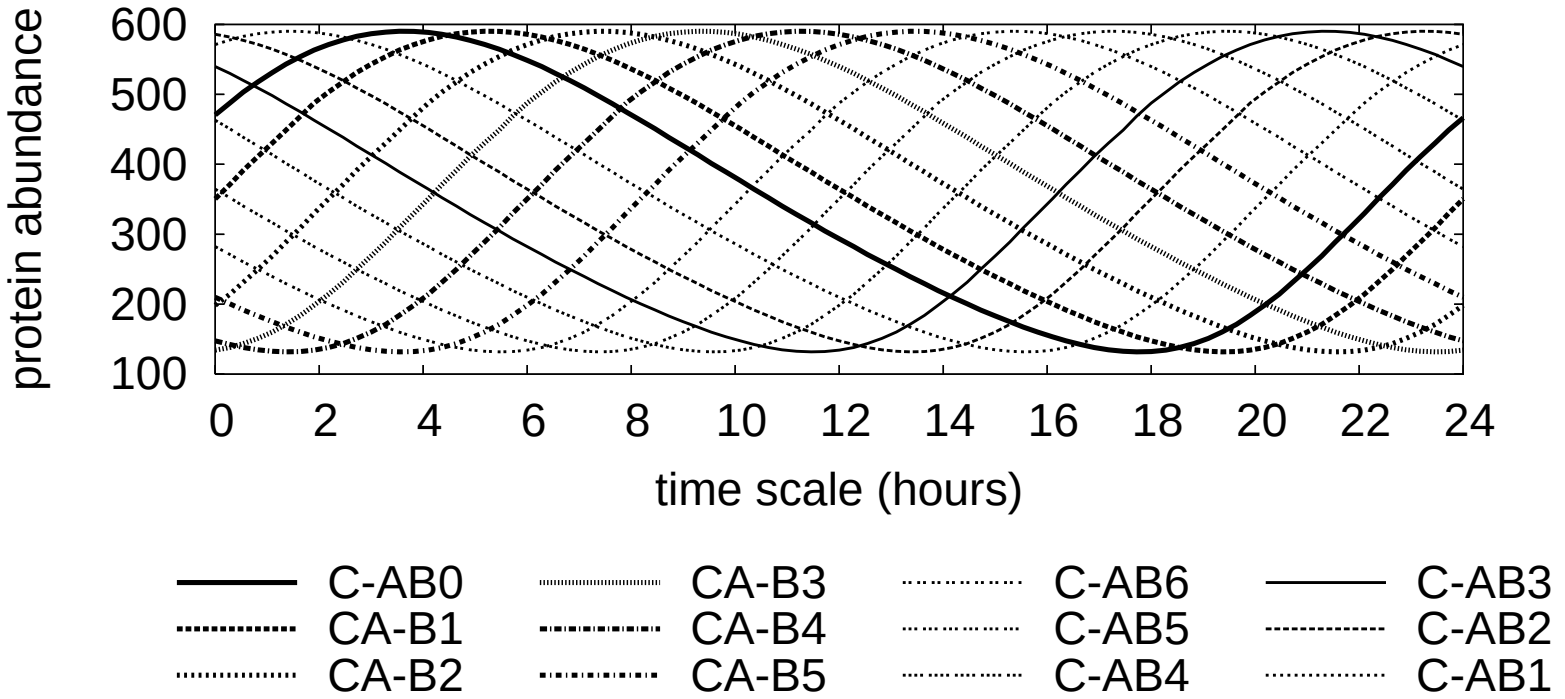
Incomplete information about interphase feedback loops

The Model Π_{KaiABC} at a Glance

$$
\begin{aligned}
\Pi_{KaiABC} &= (V, V', R_1, \ldots, R_{17}, \mathrm{f}_1, \ldots, \mathrm{f}_{17}, A, \mathrm{C}, \Delta\tau) \\
V &= \{\mathrm{A}, \mathrm{B}, \mathrm{C}\} \ldots\ldots\ldots\ldots\ldots\ldots \text{identifiers of proteins KaiA, KaiB, KaiC} \\
V' &= \{\mathrm{A}, \mathrm{B}\} \cup \ldots\ldots\ldots\ldots\ldots \text{KaiA, KaiB within a complex associated to KaiC} \\
&\quad \{0, 1, 2, 3, 4, 5, 6\} \ldots\ldots \text{number of attached phosphates} \\
R_1 &= \mathrm{C}\#\neg\mathrm{A}\#\mathrm{B}\#0 + \mathrm{A} \longrightarrow \mathrm{C}\#\mathrm{A}\#\neg\mathrm{B}\#1 + \mathrm{B} \\
R_2 &= \mathrm{C}\#\mathrm{A}\#*\#1 + \mathrm{A} \longrightarrow \mathrm{C}\#\mathrm{A}\#*\#2 + \mathrm{A} \\
R_3 &= \mathrm{C}\#\mathrm{A}\#*\#2 + \mathrm{A} \longrightarrow \mathrm{C}\#\mathrm{A}\#*\#3 + \mathrm{A} \\
R_4 &= \mathrm{C}\#\mathrm{A}\#*\#3 + \mathrm{A} \longrightarrow \mathrm{C}\#\mathrm{A}\#*\#4 + \mathrm{A} \\
R_5 &= \mathrm{C}\#\mathrm{A}\#*\#4 + \mathrm{A} \longrightarrow \mathrm{C}\#\mathrm{A}\#*\#5 + \mathrm{A} \\
R_6 &= \mathrm{C}\#\mathrm{A}\#\neg\mathrm{B}\#5 + \mathrm{B} \longrightarrow \mathrm{C}\#\neg\mathrm{A}\#\mathrm{B}\#6 + \mathrm{A} \\
R_7 &= \mathrm{C}\#*\#\mathrm{B}\#6 + \mathrm{B} \longrightarrow \mathrm{C}\#*\#\mathrm{B}\#5 + \mathrm{B} \\
R_8 &= \mathrm{C}\#*\#\mathrm{B}\#5 + \mathrm{B} \longrightarrow \mathrm{C}\#*\#\mathrm{B}\#4 + \mathrm{B} \\
R_9 &= \mathrm{C}\#*\#\mathrm{B}\#4 + \mathrm{B} \longrightarrow \mathrm{C}\#*\#\mathrm{B}\#3 + \mathrm{B} \\
R_{10} &= \mathrm{C}\#*\#\mathrm{B}\#3 + \mathrm{B} \longrightarrow \mathrm{C}\#*\#\mathrm{B}\#2 + \mathrm{B} \\
R_{11} &= \mathrm{C}\#*\#\mathrm{B}\#2 + \mathrm{B} \longrightarrow \mathrm{C}\#*\#\mathrm{B}\#1 + \mathrm{B} \\
R_{12} &= \mathrm{C}\#*\#\mathrm{B}\#1 + \mathrm{B} \longrightarrow \mathrm{C}\#*\#\mathrm{B}\#0 + \mathrm{B} \\
R_{13} &= \mathrm{C}\#\neg\mathrm{A}\#\mathrm{B}\#* + \mathrm{A} \longrightarrow \mathrm{C}\#\mathrm{A}\#\neg\mathrm{B}\#* + \mathrm{B} \\
R_{14} &= \mathrm{C}\#\mathrm{A}\#\neg\mathrm{B}\#* + \mathrm{B} \longrightarrow \mathrm{C}\#\neg\mathrm{A}\#\mathrm{B}\#* + \mathrm{A} \\
R_{15} &= \mathrm{A} \longrightarrow \emptyset \\
R_{16} &= \mathrm{B} \longrightarrow \emptyset \\
R_{17} &= \mathrm{C}\#*\#*\#* \longrightarrow \emptyset \\
&\ldots
\end{aligned}
$$

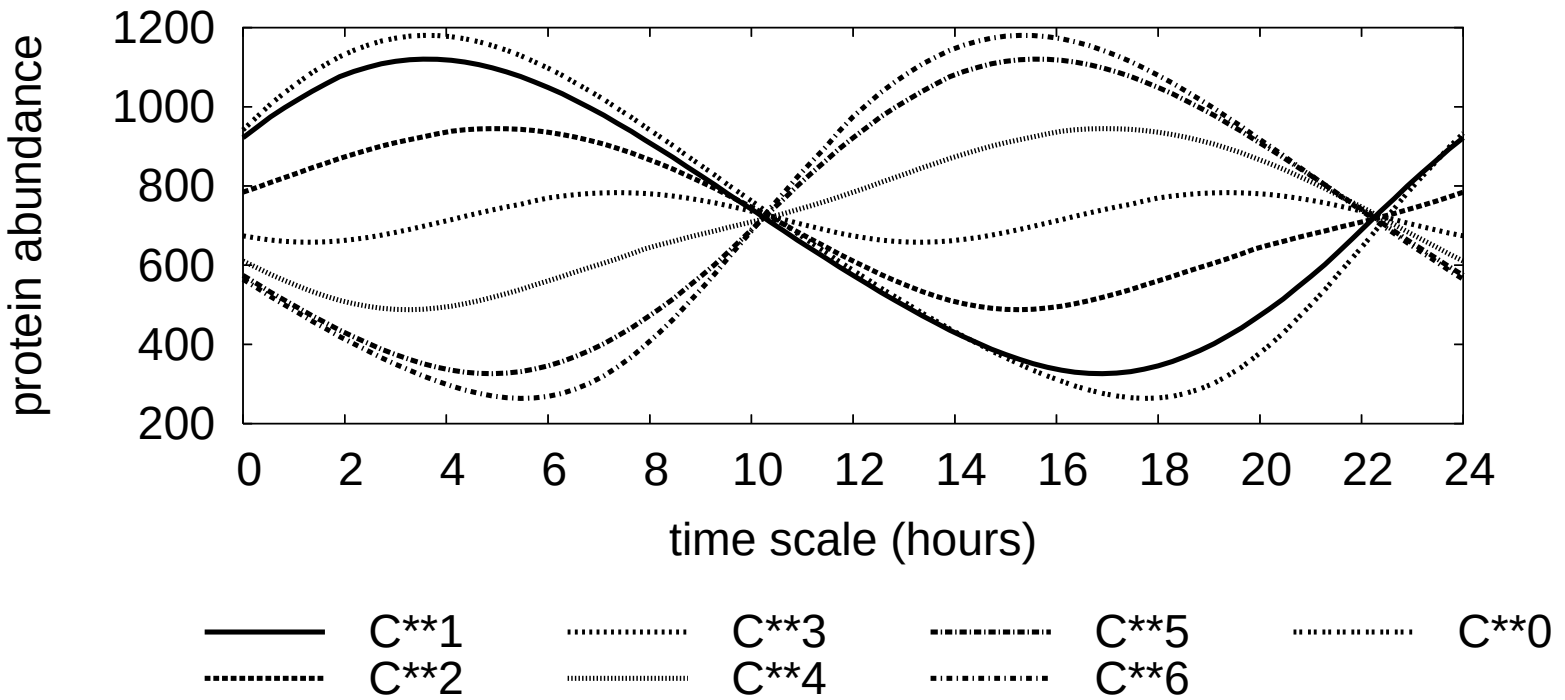
Discrete Michaelis-Menten kinetics

Simulation Results: Individual KaiABC Subproducts



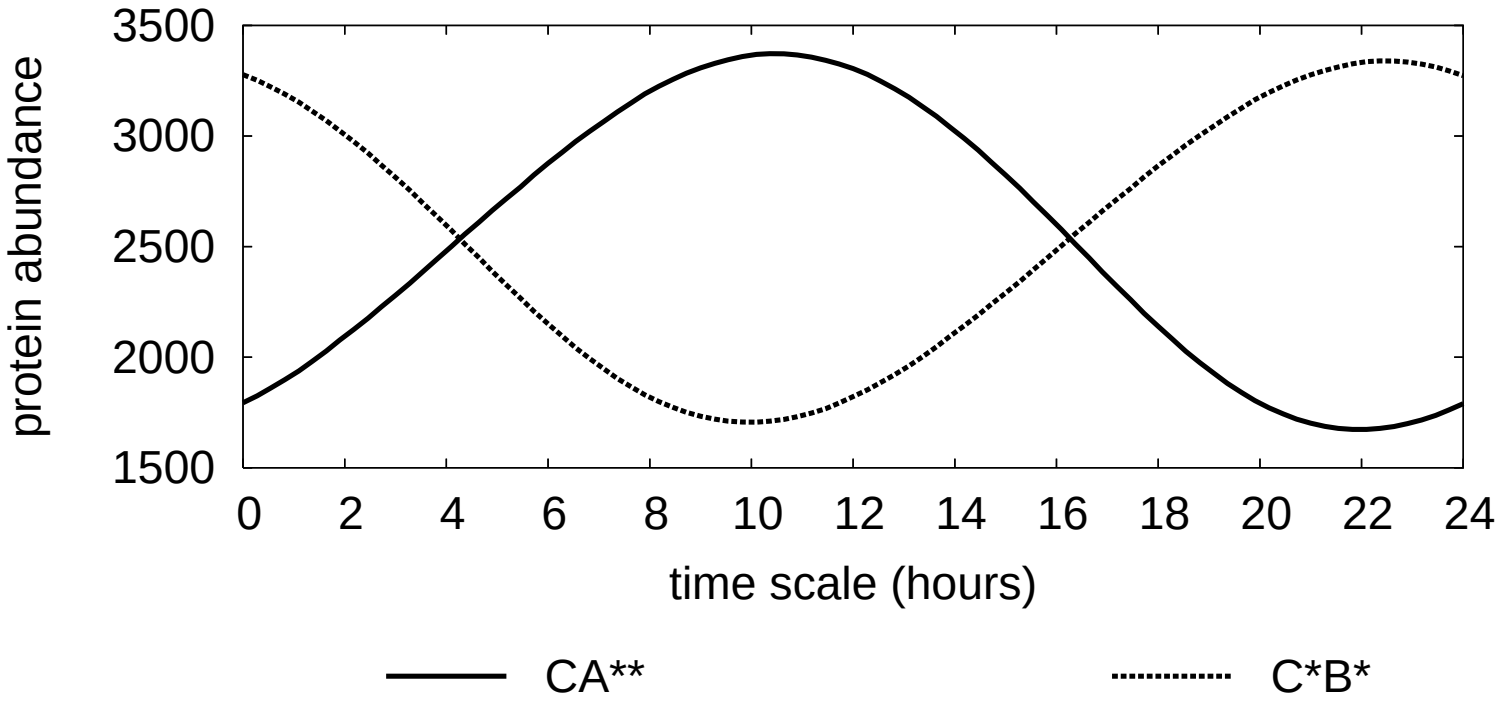
Temporal courses of 12 specific KaiABC subproducts representing the process status of the reaction cycle. Kinetic parameters and initial amounts adjusted in a way to obtain a period of $\approx$ 24 hours and symmetry among individual oscillations.

Focussing on the Level of Phosphorylation



Temporal courses of KaiABC subproducts subsumed by their level of phosphorylation ranging from 0 to 6. Kinetic parameters and initial amounts adjusted in a way to obtain a period of $\approx$ 24 hours and symmetry among individual oscillations.

Focussing on KaiABC Complex Formation



Temporal courses of KaiABC subproducts separated into two groups by association of KaiA resp. KaiB to KaiC. Kinetic parameters and initial amounts adjusted in a way to obtain a period of $\approx$ 24 hours and symmetry among individual oscillations.

Systemerweiterung auf mehrere vernetze Module

- System $\Pi_{\mathrm{CSN}} = (V, V', E, M, n)$
  - V: Alphabet der Proteinnamen
  - V': Alphabet der Proteinuntereinheiten bzw. -eigenschaften
  - M: **Module** $\longrightarrow$ funktionelle Reaktionsräume
  - E: Graph $\longrightarrow$ (Transport-) **Kanäle** zwischen Modulen
  - n: Modulanzahl (Grad des P-Systems)
- Module $M_i = (R_{i1}, \ldots, R_{ir_i}, \mathrm{f}_{i1}, \ldots \mathrm{f}_{ir_i}, A_i) \in M$
  - R_{ij}: Reaktionsregel $\longrightarrow$ Multimengen der Edukte, Produkte Metasymbole erlaubt $\longrightarrow$ **Matching** notwendig
  - f_{ij}: **kinetische** Funktion zur Regel R_{ij}, Zahl der umzusetzenden Eduktobjekte pro Reaktionsschritt
  - A_i: Multimenge der Axiome $\longrightarrow$ Initialinhalt M_i
- Kanäle $e_{ij} = (i, j, I_{ij}, \mathrm{d}_{ij}) \in E$
  - von Modul i zum Modul j gerichtet, bewertet
  - I_{ij}: Filter-Interface (**Rezeptor** und **Konzentrationsgradient**)
  - d_{ij}: **Zeitbedarf** (Anzahl **Schritte**) für Passage

Systemerweiterung auf mehrere vernetze Module

- System $\Pi_{\mathrm{CSN}} = (V, V', E, M, n)$
  - V: Alphabet der Proteinnamen
  - V': Alphabet der Proteinuntereinheiten bzw. -eigenschaften
  - M: **Module** $\longrightarrow$ funktionelle Reaktionsräume
  - E: Graph $\longrightarrow$ (Transport-) **Kanäle** zwischen Modulen
  - n: Modulanzahl (Grad des P-Systems)
- Module $M_i = (R_{i1}, \ldots, R_{ir_i}, \mathrm{f}_{i1}, \ldots \mathrm{f}_{ir_i}, A_i) \in M$
  - R_{ij}: Reaktionsregel $\longrightarrow$ Multimengen der Edukte, Produkte Metasymbole erlaubt $\longrightarrow$ **Matching** notwendig
  - f_{ij}: **kinetische** Funktion zur Regel R_{ij}, Zahl der umzusetzenden Eduktobjekte pro Reaktionsschritt
  - A_i: Multimenge der Axiome $\longrightarrow$ Initialinhalt M_i
- Kanäle $e_{ij} = (i, j, I_{ij}, \mathrm{d}_{ij}) \in E$
  - von Modul i zum Modul j gerichtet, bewertet
  - I_{ij}: Filter-Interface (**Rezeptor** und **Konzentrationsgradient**)
  - d_{ij}: **Zeitbedarf** (Anzahl **Schritte**) für Passage

Systemerweiterung auf mehrere vernetze Module

- System $\Pi_{\mathrm{CSN}} = (V, V', E, M, n)$
  - V: Alphabet der Proteinnamen
  - V': Alphabet der Proteinuntereinheiten bzw. -eigenschaften
  - M: **Module** $\longrightarrow$ funktionelle Reaktionsräume
  - E: Graph $\longrightarrow$ (Transport-) **Kanäle** zwischen Modulen
  - n: Modulanzahl (Grad des P-Systems)
- Module $M_i = (R_{i1}, \ldots, R_{ir_i}, \mathrm{f}_{i1}, \ldots \mathrm{f}_{ir_i}, A_i) \in M$
  - R_{ij}: Reaktionsregel $\longrightarrow$ Multimengen der Edukte, Produkte Metasymbole erlaubt $\longrightarrow$ **Matching** notwendig
  - f_{ij}: **kinetische** Funktion zur Regel R_{ij}, Zahl der umzusetzenden Eduktobjekte pro Reaktionsschritt
  - A_i: Multimenge der Axiome $\longrightarrow$ Initialinhalt M_i
- Kanäle $e_{ij} = (i, j, I_{ij}, \mathrm{d}_{ij}) \in E$
  - von Modul i zum Modul j gerichtet, bewertet
  - I_{ij}: Filter-Interface (**Rezeptor** und **Konzentrationsgradient**)
  - d_{ij}: **Zeitbedarf** (Anzahl **Schritte**) für Passage

Dynamisches Systemverhalten

- Inhalt Modul M_i zur globalen Zeit $t \in \mathbb{N}$: Multimenge $\mathcal{L}_i(t)$
- Arbeitsschritt in jedem Modul M_i:

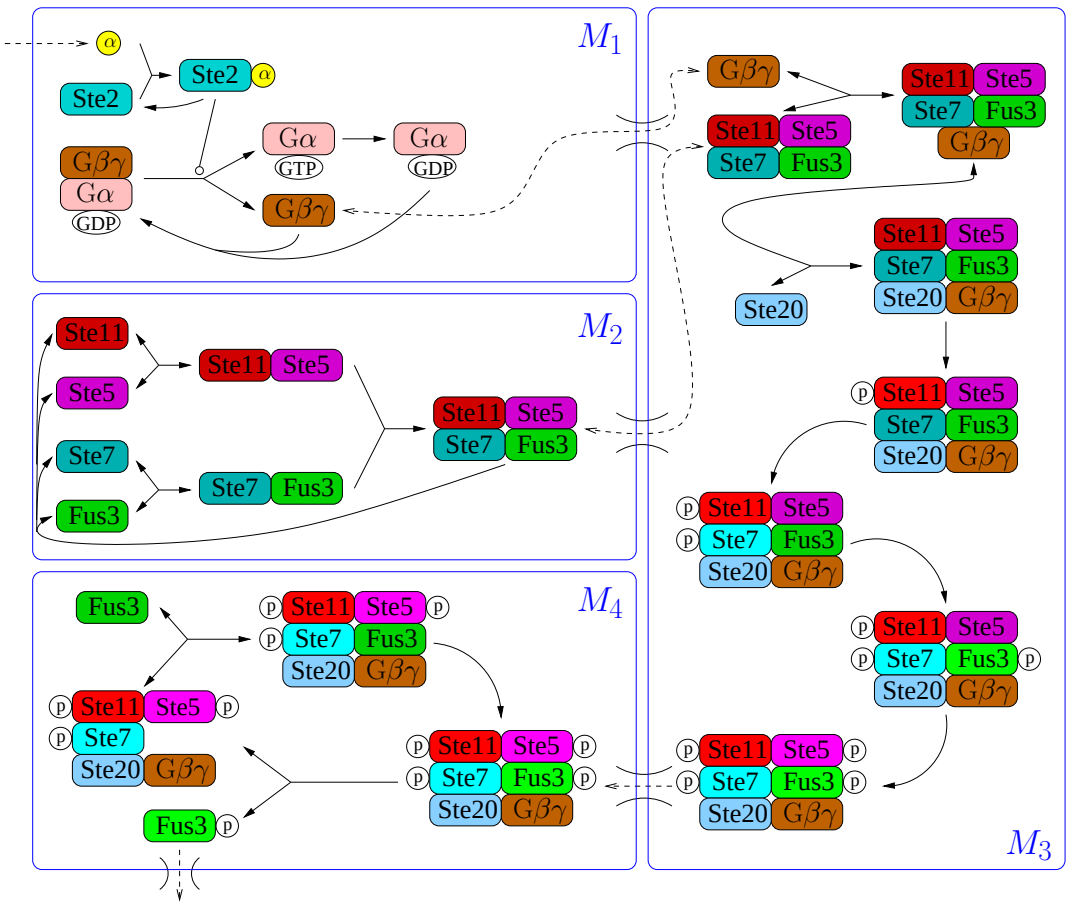
$$\begin{aligned} \mathcal{L}_i(0) &= \mathcal{A}_i \\ \mathcal{L}'_i(t) &= \mathcal{L}_i(t) \ominus \mathit{Educts}_i(t) \uplus \mathit{Products}_i(t) \\ \mathcal{L}_i(t+1) &= \mathcal{L}'_i(t) \ominus \mathit{Outgoing}_i(t) \uplus \mathit{Incoming}_i(t) \end{aligned}$$

1. Bestimme Multimenge der Edukte mit Hilfe von $\mathcal{L}_i(t)$, $R_{i1}, \ldots, R_{ir_i}, \mathrm{f}_{i1}, \ldots \mathrm{f}_{ir_i}$; matche dabei
2. Entferne Eduktobjekte aus Modulinhalt
3. Bestimme Multimenge der Reaktionsprodukte, erhalte $\mathcal{L}'_i(t)$
4. Bestimme in andere Module zu sendende Objekte, nutze dabei $\mathcal{L}'_i(t)$ und I jedes wegführenden Kanals, matche
5. Füge über Kanäle eintreffende Objekte hinzu, berücks. d

Fallbeispiel: Yeast Pheromone Pathway

Signaltransduktion in *Saccharomyces cerevisiae*

biologisch gut untersuchtes System, Fortpflanzung Hefezellen



$$\Pi = (V, V', E, M, 4)$$

$$V = \{\mathrm{Ste2}, \alpha, \mathrm{G}\beta\gamma, \mathrm{G}\alpha, \ldots\}$$

$$V' = \{\mathrm{a}, \mathrm{GDP}, \mathrm{GTP}, \mathrm{p}\}$$

$$M = \{M_1, M_2, M_3, M_4\}$$

$$M_1 = (R_{11}, R_{12}, R_{13}, R_{14}, R_{15}, \mathrm{f}_{11}, \mathrm{f}_{12}, \mathrm{f}_{13}, \mathrm{f}_{14}, \mathrm{f}_{15}, A_1)$$

$$R_{11} = \mathrm{Ste2}\#\neg\mathrm{a} + \alpha \rightarrow \mathrm{Ste2}\#\mathrm{a}$$

$$R_{12} = \mathrm{Ste2}\#\mathrm{a} \rightarrow \mathrm{Ste2}\#\neg\mathrm{a}$$

$$\mathrm{f}_{11} = \lfloor \mathrm{k}_{11}[\mathrm{Ste2}\#\neg\mathrm{a}][\alpha]/\mathrm{V}_1^2 \rfloor$$

$$\vdots$$

Simulation von Registermaschinen (RAM)

- Definition der Komponenten
  $RAM = (R, L, P, I_1)$
  - I_1: Sprungmarke Startbefehl
  - P: Programm (endlicher Befehlssatz)
  - L: endl. Menge von Sprungmarken $L = \{I_1, \ldots, I_c\}$
  - R: endl. Menge von Registern $R = \{r_1, \ldots, r_m\}, \quad r_k \in \mathbb{N}$
- verfügbare Befehle
  - I_i : INCR$(r_k), I_j$ Register r_k inkrementieren, gehe zu I_j
  - I_i : DECR$(r_k), I_j$ Register r_k dekrementieren, gehe zu I_j
  - $I_i : r_k \neq 0, I_j, I_p$ Falls $r_k \neq 0$ gehe zu I_j sonst zu I_p
  - I_i : HALT Programmende
- Festlegungen
  - jedes Register speichert eine natürliche Zahl
  - bei Start alle Register mit Eingabewerten initialisiert
  - Ausgaberegister festgelegt, seine Werte: erzeugte Sprache
  - deterministische oder nichtdeterministische Arbeitsweise

Simulation von Registermaschinen (RAM)

- Definition der Komponenten
  $RAM = (R, L, P, l_1)$
  - l_1: Sprungmarke Startbefehl
  - P: Programm (endlicher Befehlssatz)
  - L: endl. Menge von Sprungmarken $L = \{l_1, \ldots, l_c\}$
  - R: endl. Menge von Registern $R = \{r_1, \ldots, r_m\}, \quad r_k \in \mathbb{N}$
- verfügbare Befehle
  - l_i : INCR(r_k), l_j — Register r_k inkrementieren, gehe zu l_j
  - l_i : DECR(r_k), l_j — Register r_k dekrementieren, gehe zu l_j
  - l_i : $r_k \neq 0, l_j, l_p$ — Falls $r_k \neq 0$ gehe zu l_j sonst zu l_p
  - l_i : HALT — Programmende
- Festlegungen
  - jedes Register speichert eine natürliche Zahl
  - bei Start alle Register mit Eingabewerten initialisiert
  - Ausgaberegister festgelegt, seine Werte: erzeugte Sprache
  - deterministische oder nichtdeterministische Arbeitsweise

Simulation von Registermaschinen (RAM)

- Definition der Komponenten
  $RAM = (R, L, P, l_1)$

- verfügbare Befehle
  - l_i : INCR(r_k), l_j Register r_k inkrementieren, gehe zu l_j
  - l_i : DECR(r_k), l_j Register r_k dekrementieren, gehe zu l_j
  - l_i : $r_k \neq 0, l_j, l_p$ Falls $r_k \neq 0$ gehe zu l_j sonst zu l_p
  - l_i : HALT Programmende
- Festlegungen
  - jedes Register speichert eine natürliche Zahl
  - bei Start alle Register mit Eingabewerten initialisiert
  - Ausgaberegister festgelegt, seine Werte: erzeugte Sprache
  - deterministische oder nichtdeterministische Arbeitsweise

Simulation von Registermaschinen durch Π_{CSN}

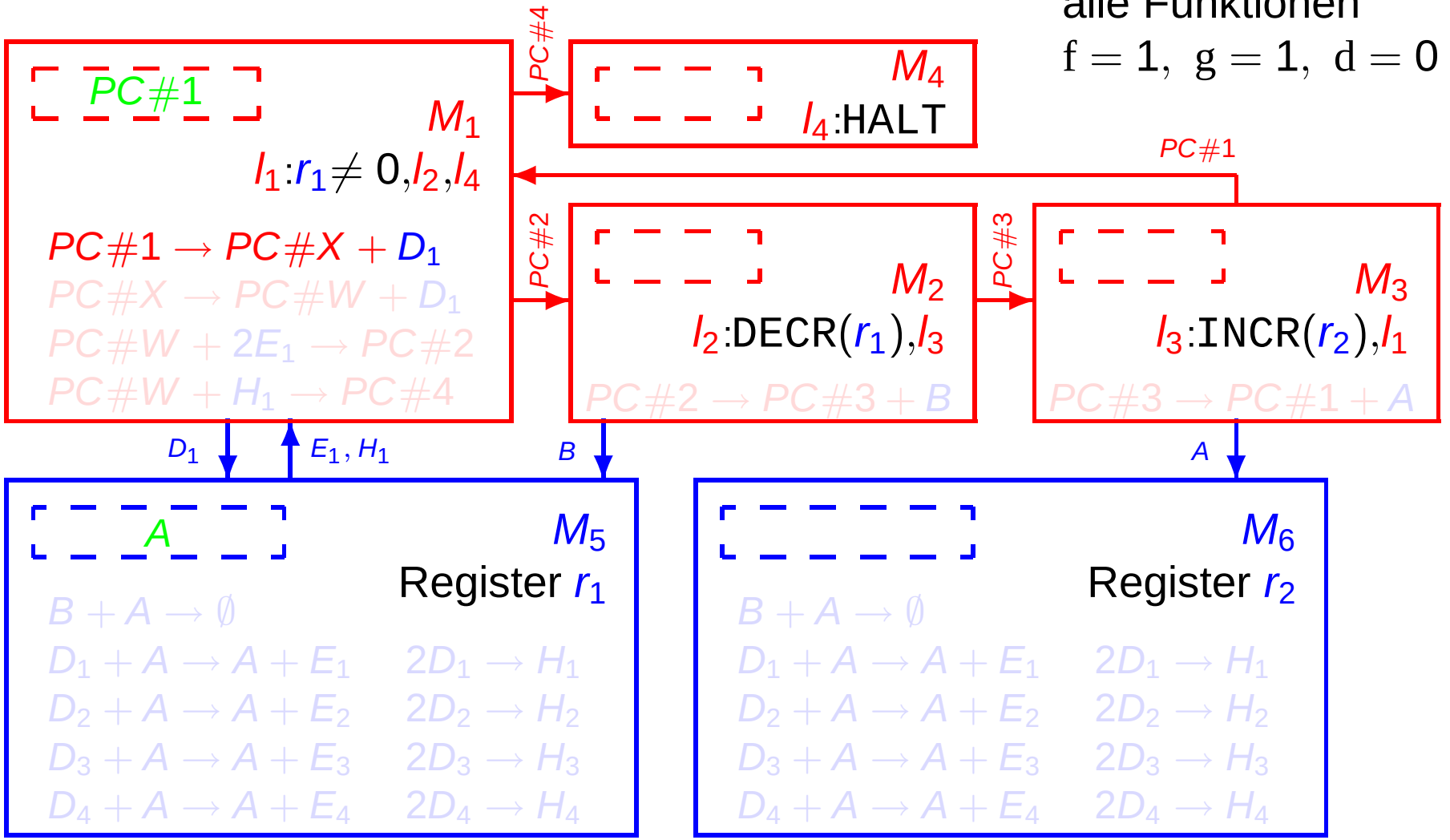
Universalitätsnachweis und Beispiel einer Berechnung

- Betrachte für Simulation und Systemtransformation

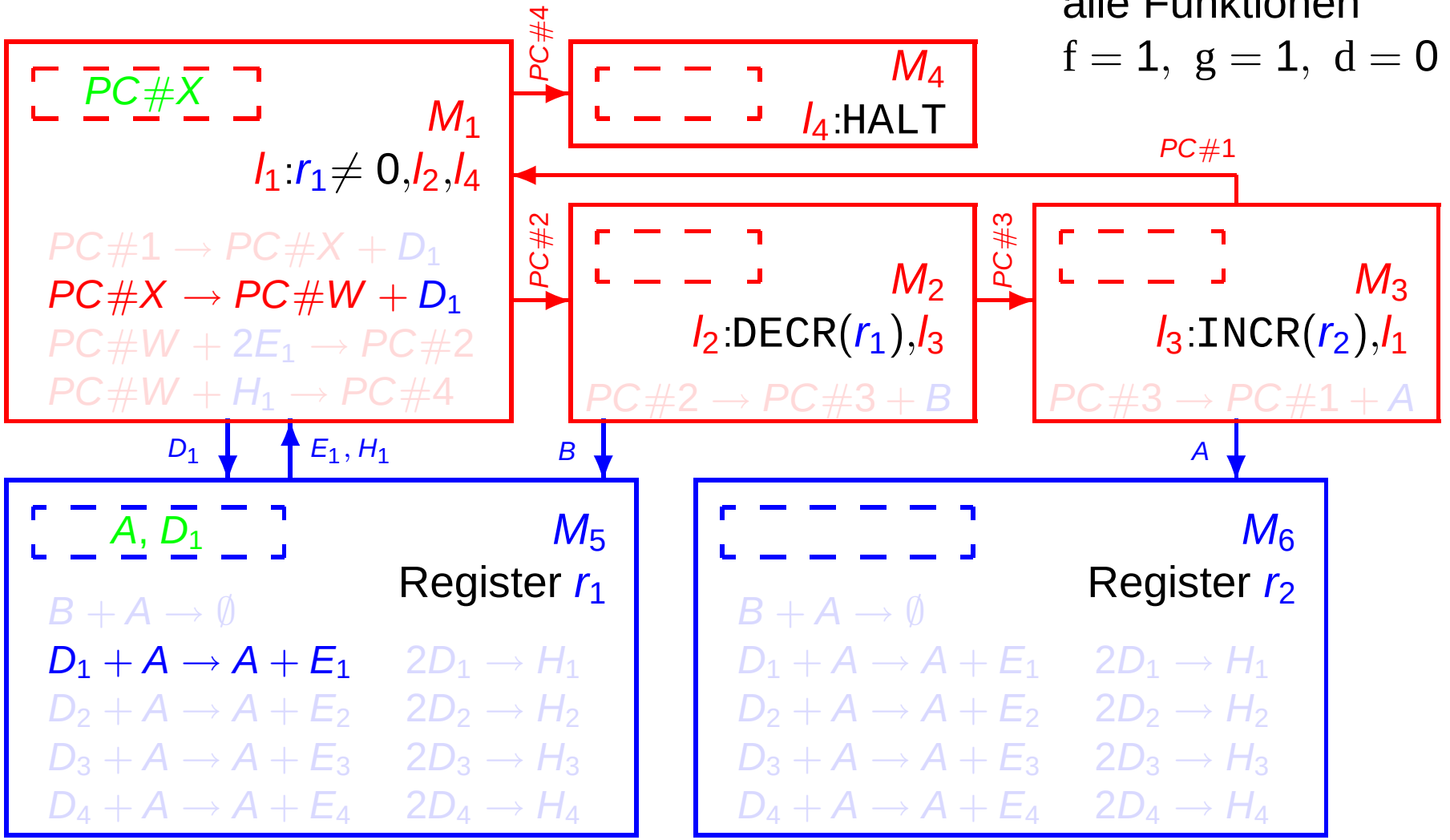
$$
\begin{aligned}
RAM &= (R, L, P, l_1) \\
R &= \{r_1, r_2\} \\
L &= \{l_1, l_2, l_3, l_4\} \\
P &= \{l_1 : r_1 \neq 0, l_2, l_4, \\
&\quad\ l_2 : \texttt{DECR}(r_1), l_3, \\
&\quad\ l_3 : \texttt{INCR}(r_2), l_1, \\
&\quad\ l_4 : \texttt{HALT}\}
\end{aligned}
$$

- Initialisierung: $r_1 = 1$ und $r_2 = 0$ (beliebig gewählt)
- Programm addiert Inhalt von r_1 schrittweise auf r_2 (Ausgabe)
- RAM besteht aus $m = 2$ Registern und $c = 4$ Befehlen
- Jedes Register und jeder Befehl wird separates Modul des P-Systems $\Pi_{\text{CSN}} = (V, V', E, M, c + m)$

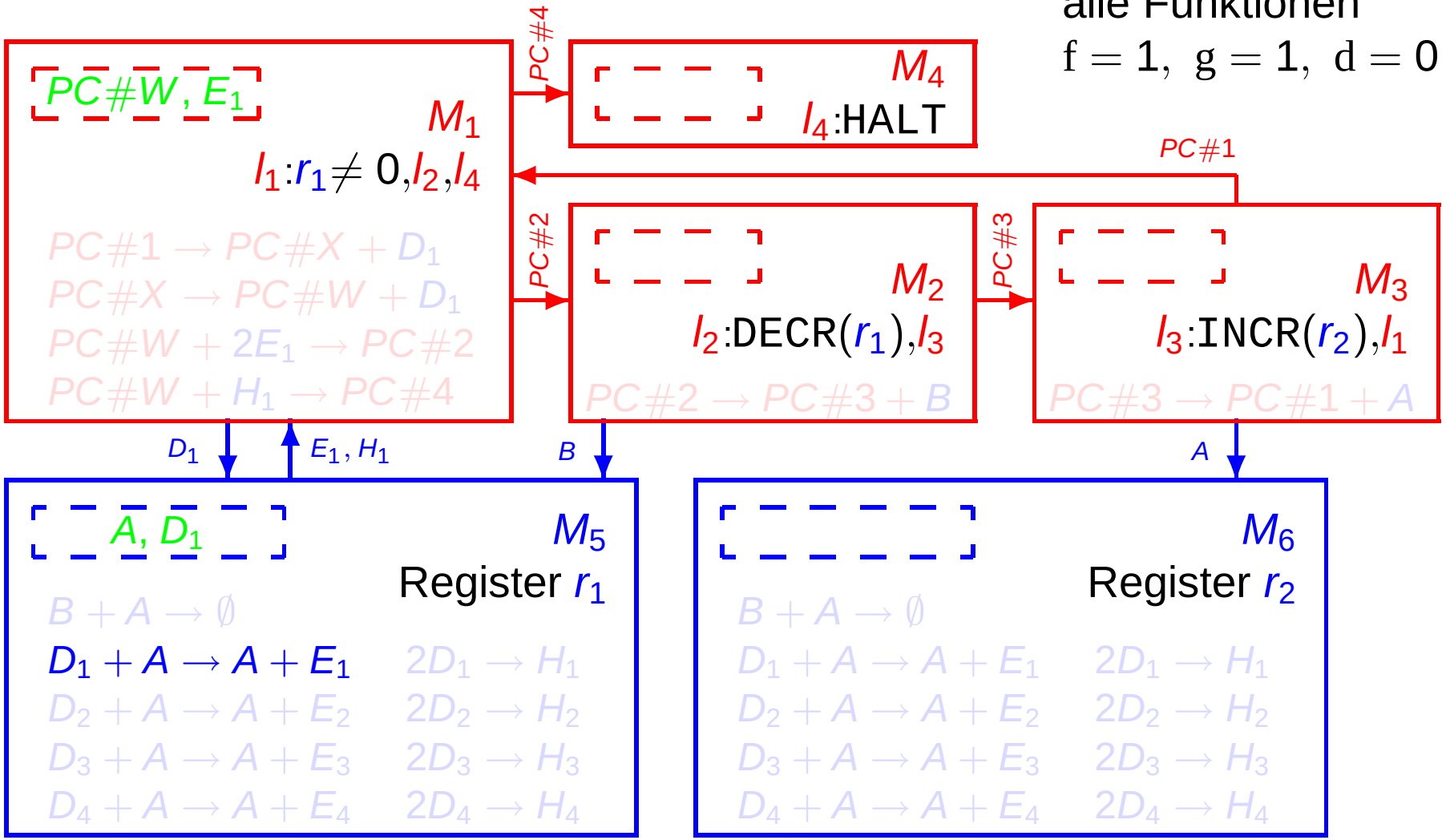
Simulation von Registermaschinen durch Π_{CSN}



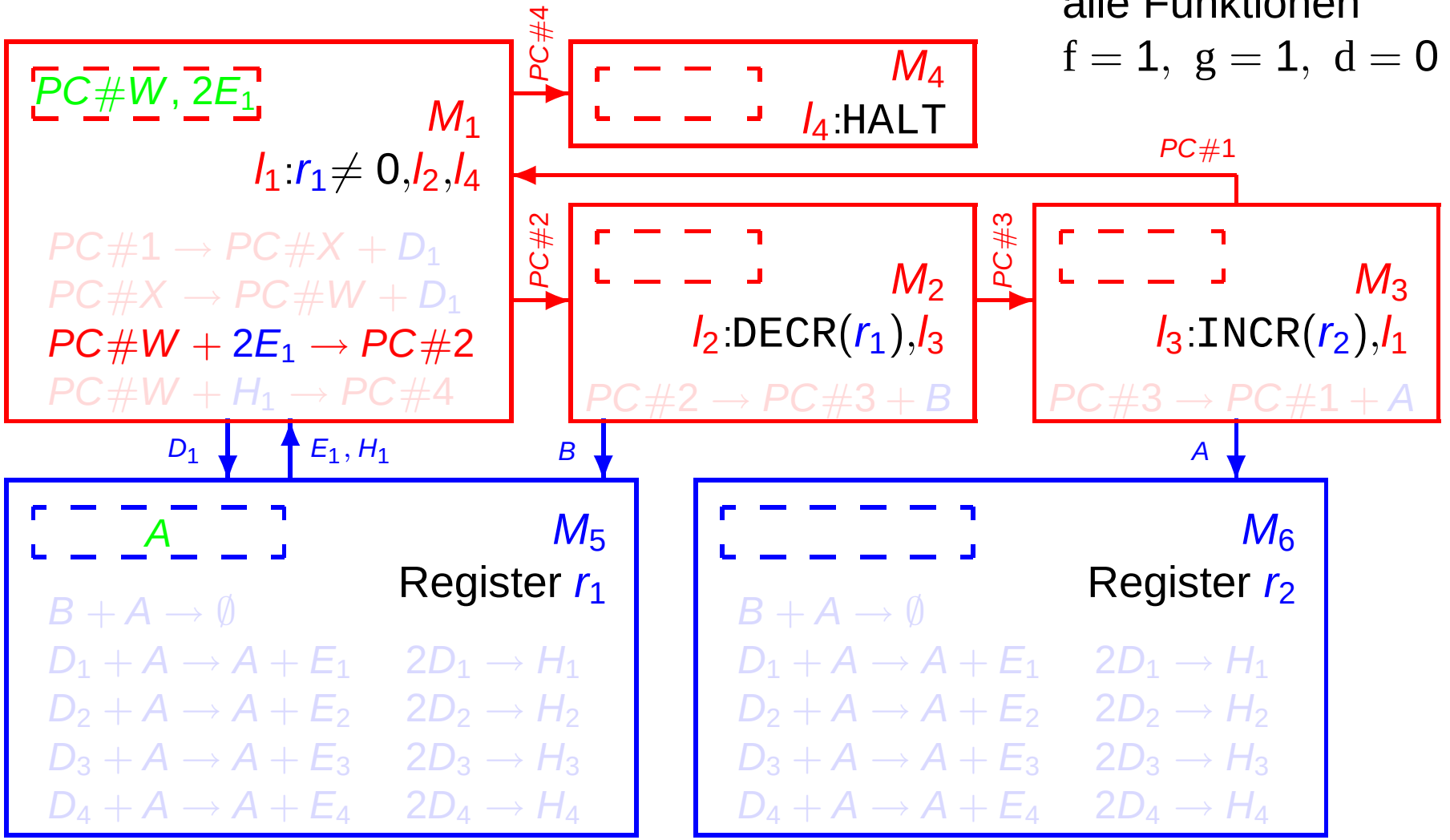
Simulation von Registermaschinen durch Π_{CSN}



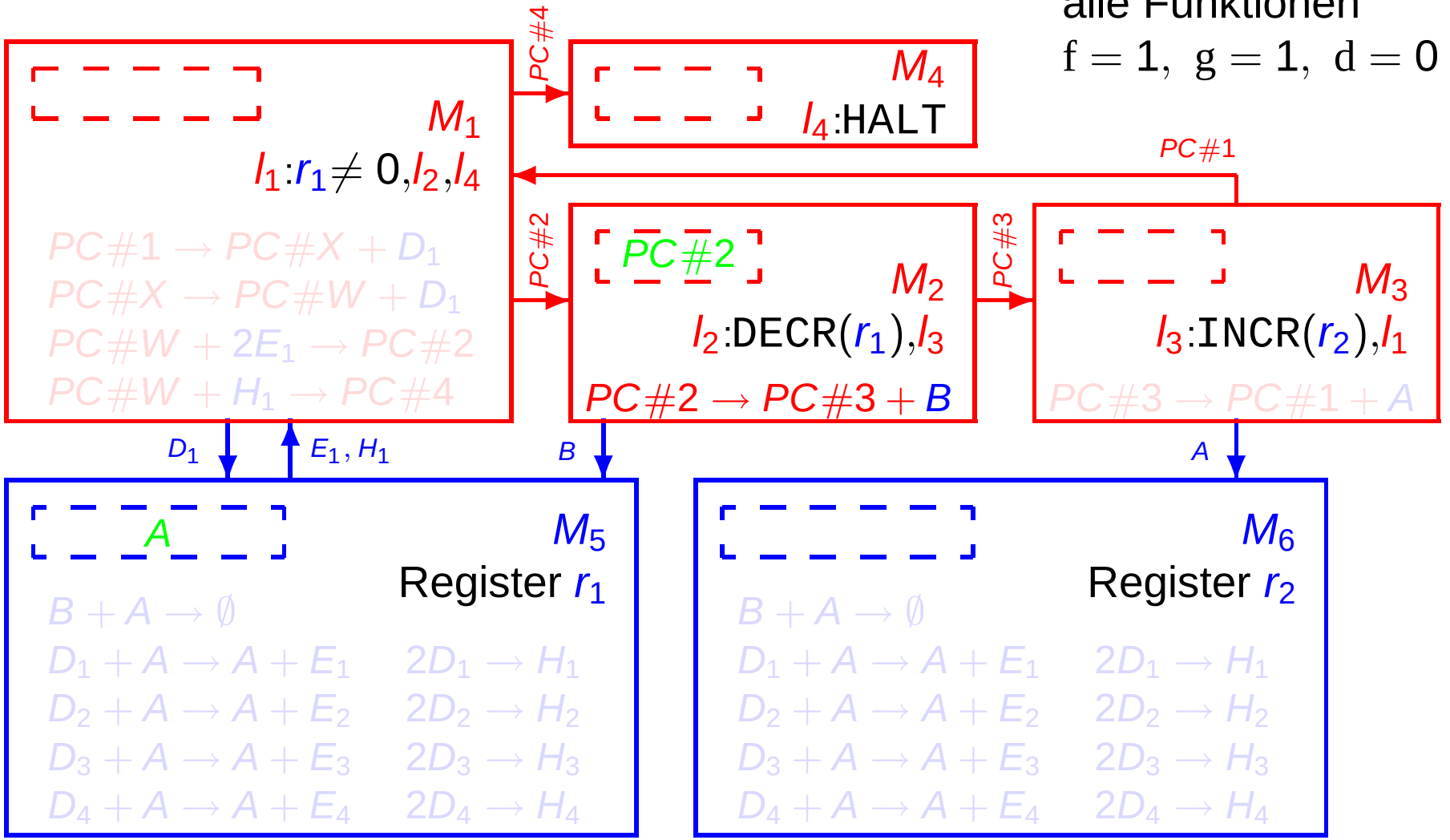
Simulation von Registermaschinen durch Π_{CSN}



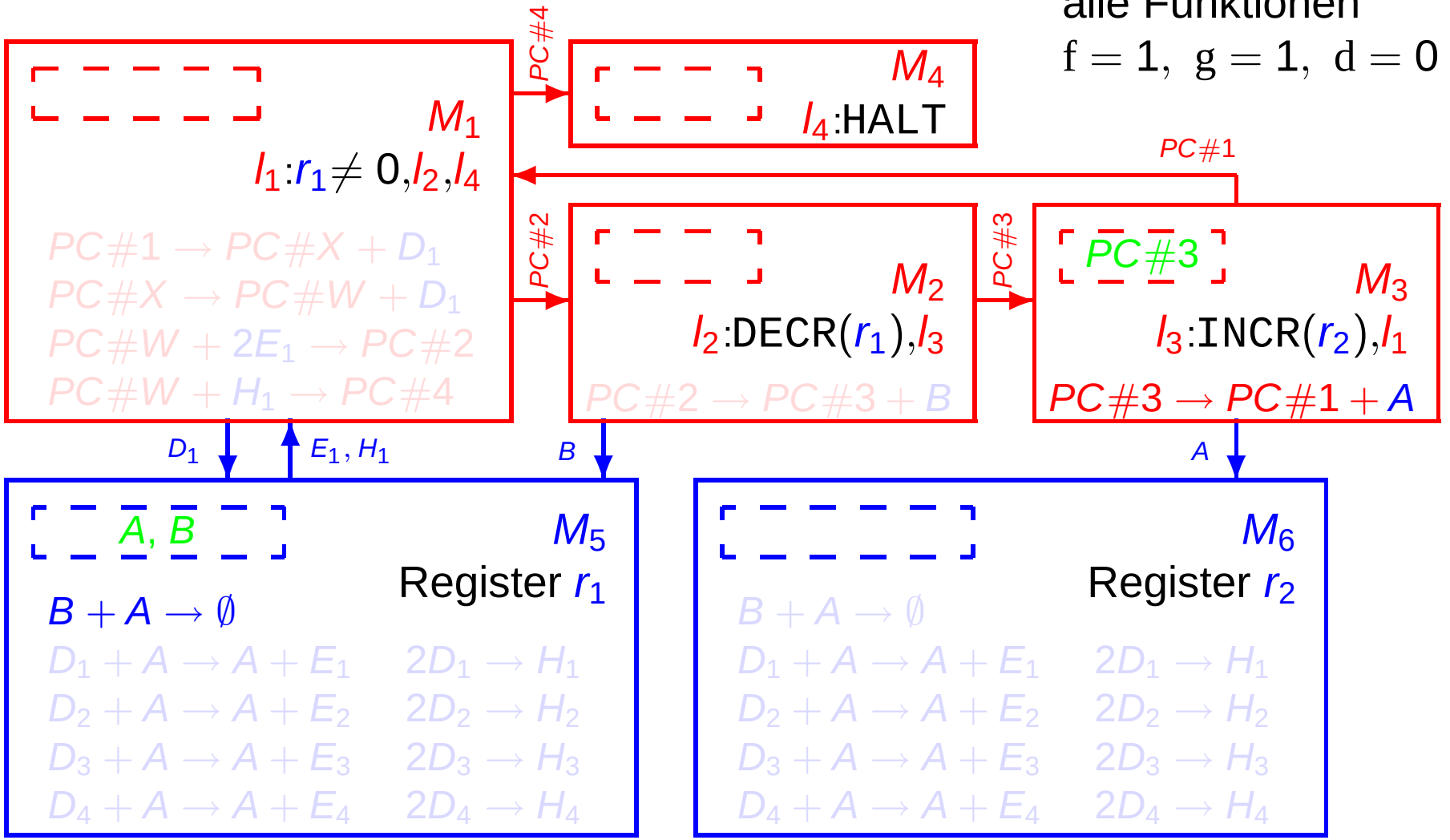
Simulation von Registermaschinen durch Π_{CSN}



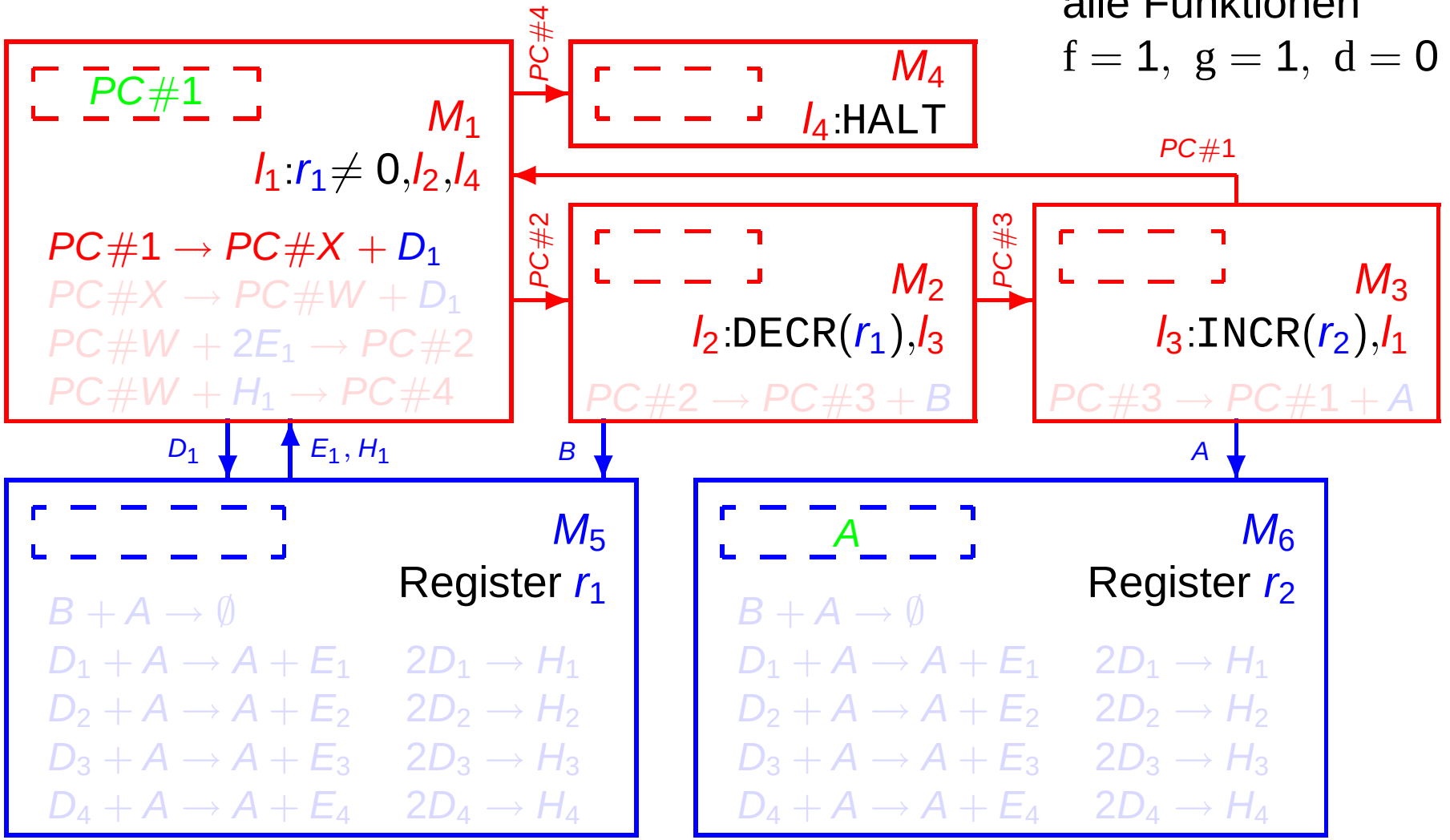
Simulation von Registermaschinen durch Π_{CSN}



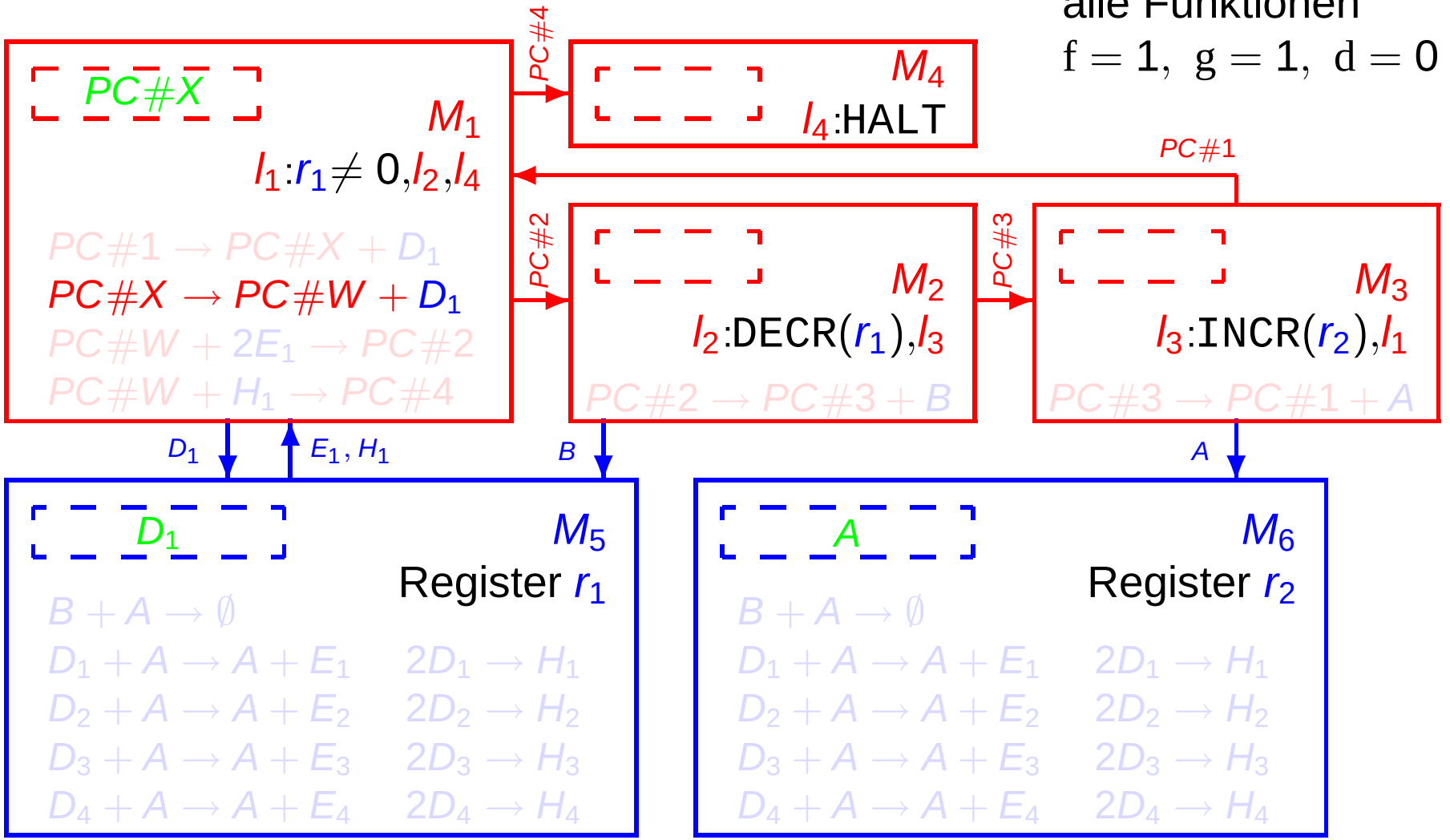
Simulation von Registermaschinen durch Π_{CSN}



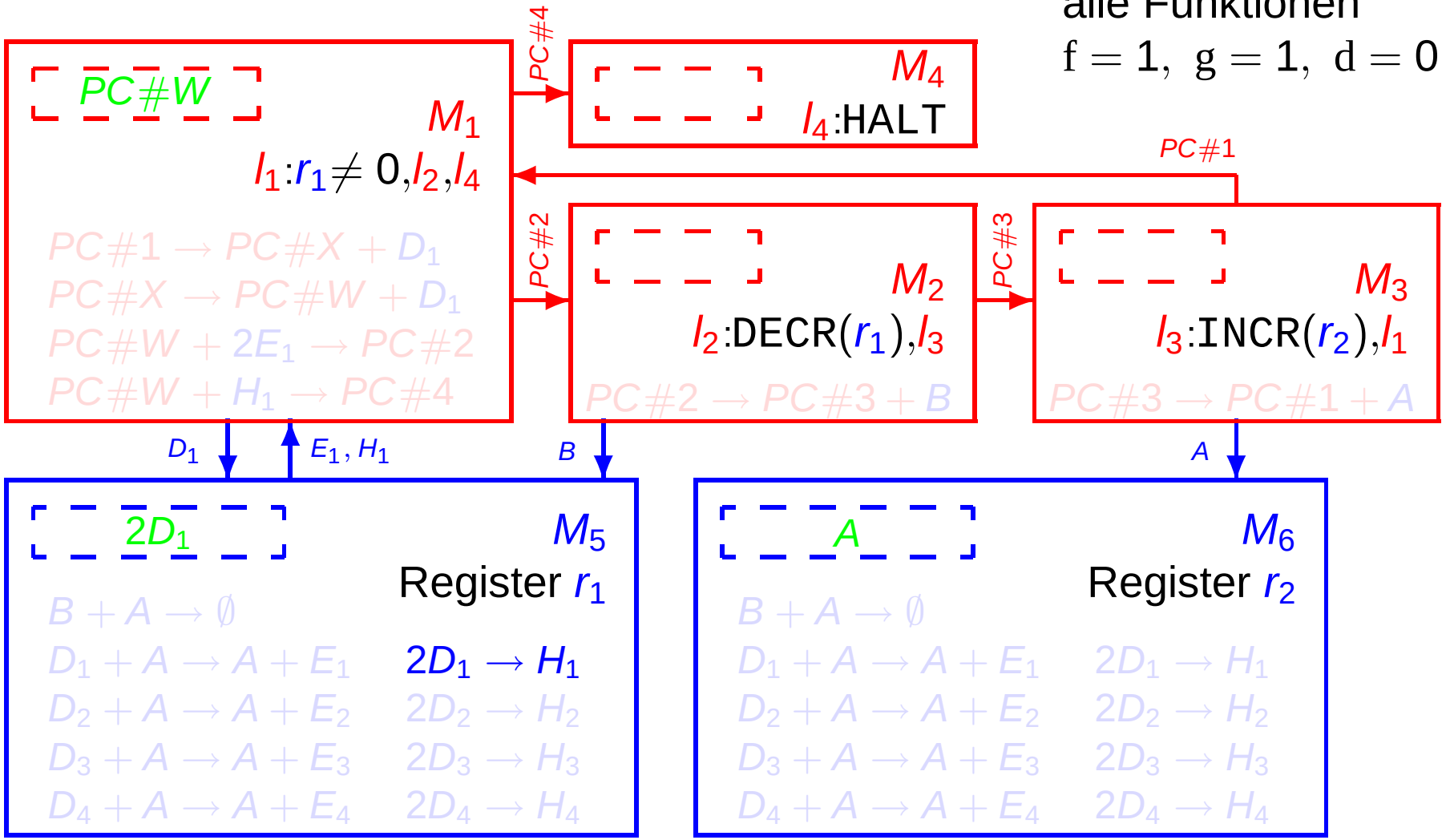
Simulation von Registermaschinen durch Π_{CSN}



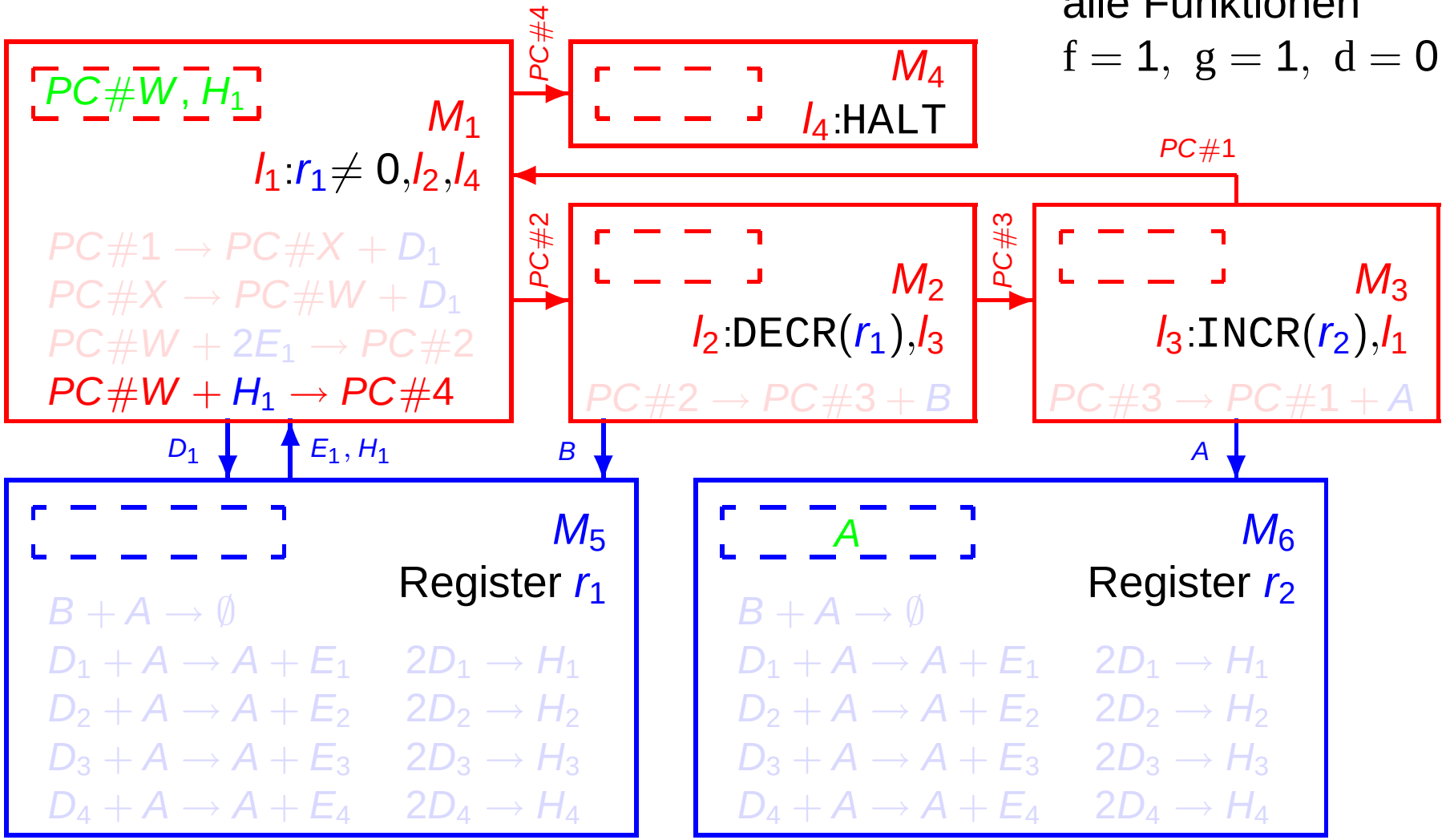
Simulation von Registermaschinen durch Π_{CSN}



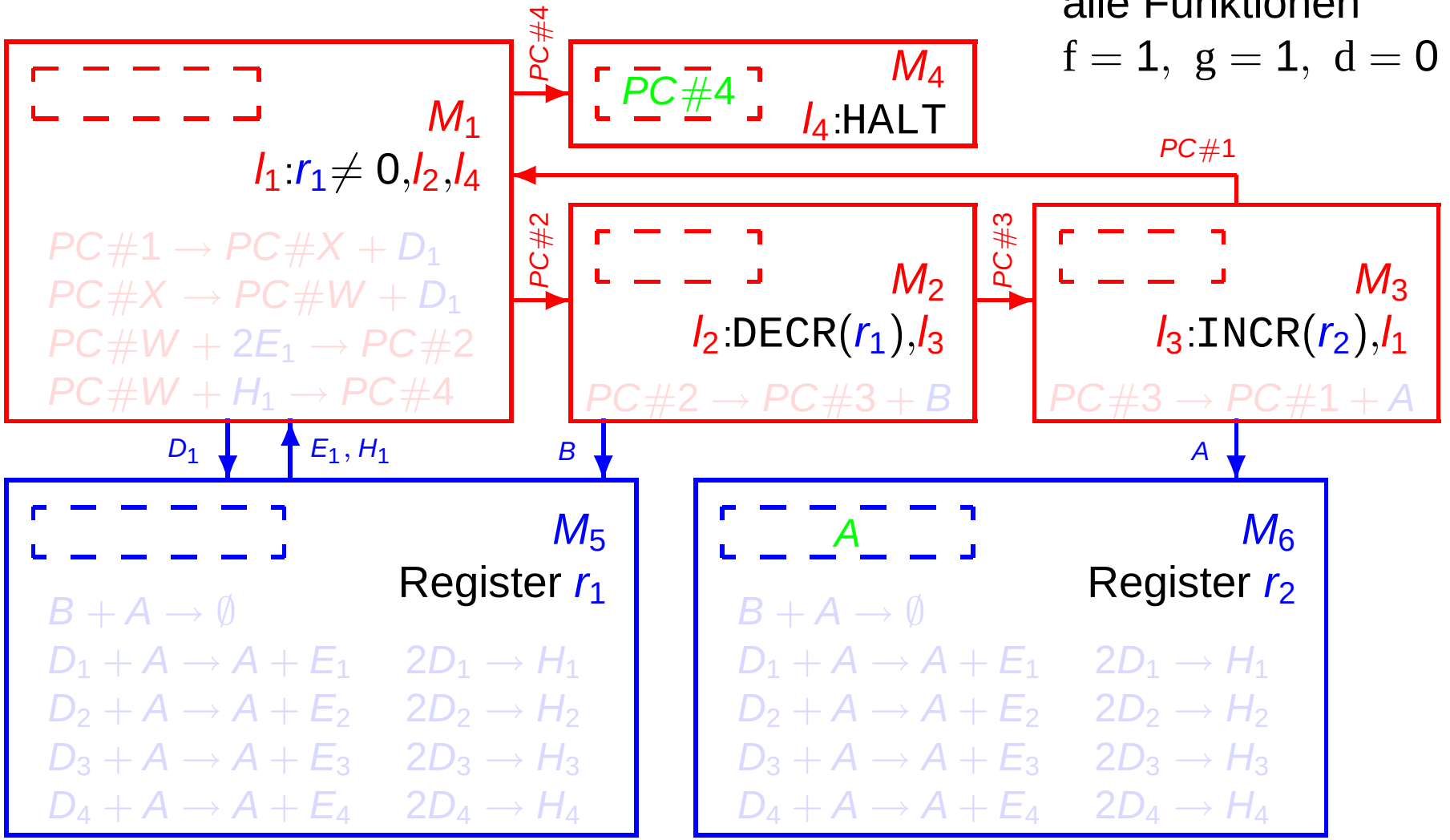
Simulation von Registermaschinen durch Π_{CSN}



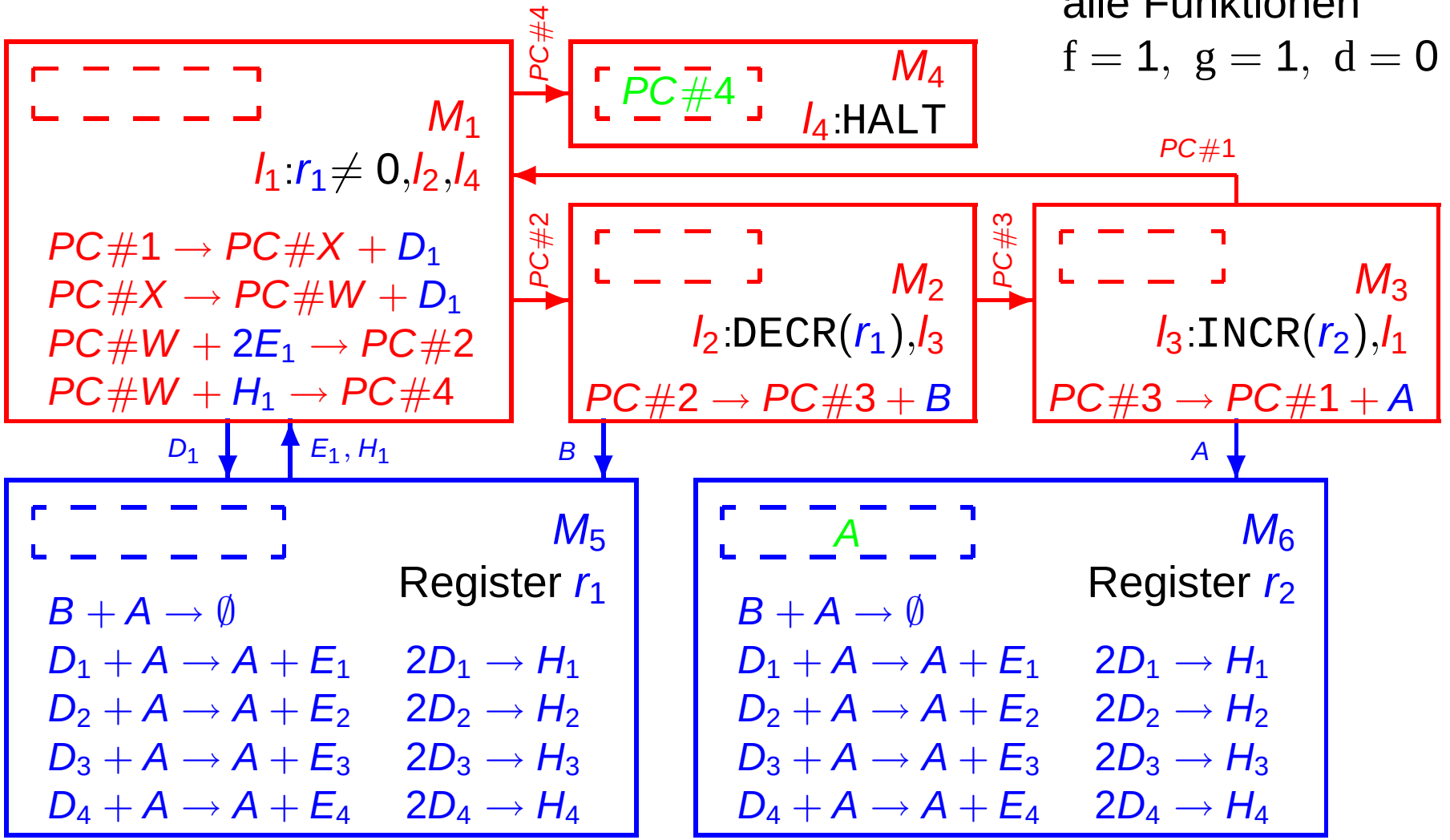
Simulation von Registermaschinen durch Π_{CSN}



Simulation von Registermaschinen durch Π_{CSN}



Simulation von Registermaschinen durch Π_{CSN}



Klassifikation von Membransystemen

- **Anordnung der Reaktionsräume**
  - homogen (ein Reaktionsraum)
  - zellbasiert (hierarchisch)
  - gewebebasiert (als Graph)
  - populationsbasiert (objektspezifisch)

- **Strukturierung der Objekte**
  - Symbolobjekte
  - Stringobjekte
  - Graphobjekte
  - parameterbehaftete Objekte

- **Strukturierung der Reaktions- und Transportregeln**
  - einzelne chemische Reaktion
  - einzelner Stofftransport (z.B. Diffusion)
  - Symport/Antiport, Promoters/Inhibitors
  - Auflösen, Teilen, Erzeugen von Reaktionsräumen
  - Filtern/Matchen von Objekten
  - Generieren neuer Objekte

www.amazon.de

Klassifikation von Membransystemen

- **Anordnung der Reaktionsräume**
  - homogen (ein Reaktionsraum)
  - zellbasiert (hierarchisch)
  - gewebebasiert (als Graph)
  - populationsbasiert (objektspezifisch)

- **Strukturierung der Objekte**
  - Symbolobjekte
  - Stringobjekte
  - Graphobjekte
  - parameterbehaftete Objekte

- **Strukturierung der Reaktions- und Transportregeln**
  - einzelne chemische Reaktion
  - einzelner Stofftransport (z.B. Diffusion)
  - Symport/Antiport, Promoters/Inhibitors
  - Auflösen, Teilen, Erzeugen von Reaktionsräumen
  - Filtern/Matchen von Objekten
  - Generieren neuer Objekte

www.amazon.de

Klassifikation von Membransystemen

- **Anordnung der Reaktionsräume**
  - homogen (ein Reaktionsraum)
  - zellbasiert (hierarchisch)
  - gewebebasiert (als Graph)
  - populationsbasiert (objektspezifisch)

- **Strukturierung der Objekte**
  - Symbolobjekte
  - Stringobjekte
  - Graphobjekte
  - parameterbehaftete Objekte

- **Strukturierung der Reaktions- und Transportregeln**
  - einzelne chemische Reaktion
  - einzelner Stofftransport (z.B. Diffusion)
  - Symport/Antiport, Promoters/Inhibitors
  - Auflösen, Teilen, Erzeugen von Reaktionsräumen
  - Filtern/Matchen von Objekten
  - Generieren neuer Objekte

www.amazon.de

Klassifikation von Membransystemen

- **Strukturierung der Ablaufsteuerung**
  - maximal parallel (nichtdeterministisch)
  - Prioritäten zwischen Transitionsregeln (determ.)
  - konditionale Ersetzung (Kinetiken, determ.)
  - Kommunikation (Ereignisse)
  - probabilistisch (Wahrscheinlichkeitsverteilung)
  - stochastisch
  - Splicing, Insertion/Deletion
- **Zeitfortschreibung**
  - globale Uhr, äquidistante vs. variable Schrittlänge
  - Warteschlange lokaler Ereignisse

www.amazon.de

Klassifikation von Membransystemen

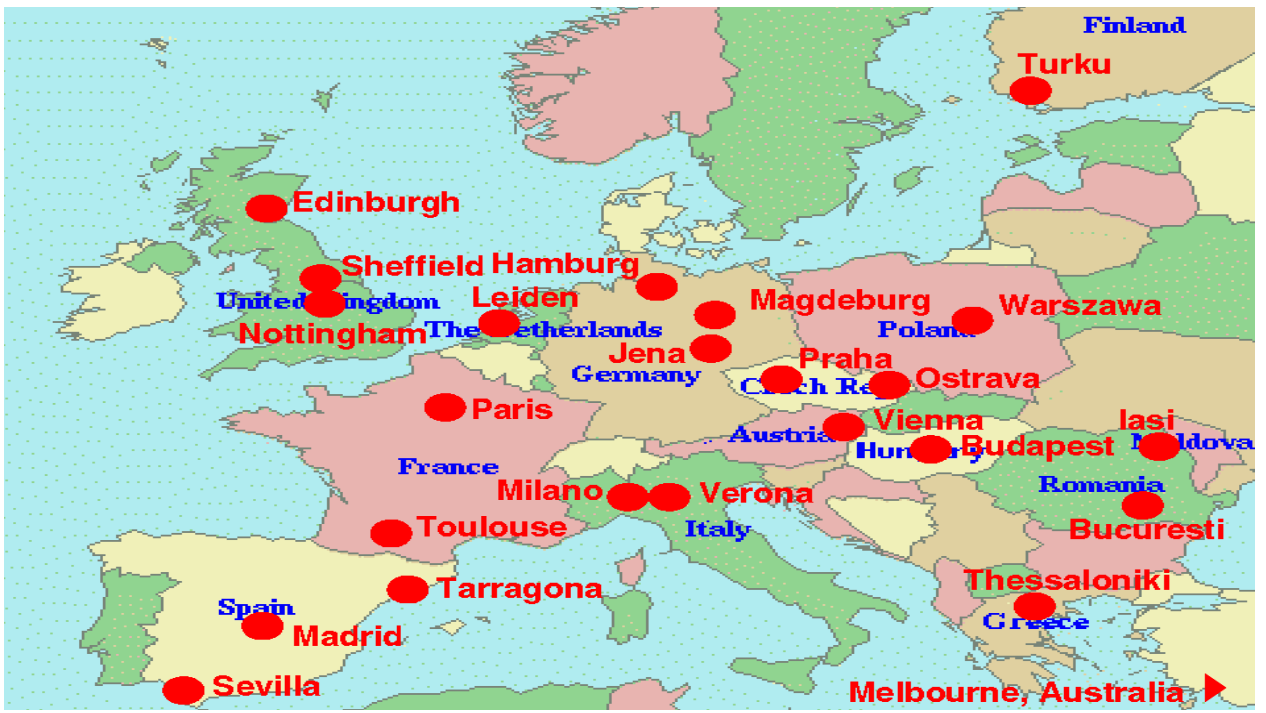
- **Strukturierung der Ablaufsteuerung**
  - maximal parallel (nichtdeterministisch)
  - Prioritäten zwischen Transitionsregeln (determ.)
  - konditionale Ersetzung (Kinetiken, determ.)
  - Kommunikation (Ereignisse)
  - probabilistisch (Wahrscheinlichkeitsverteilung)
  - stochastisch
  - Splicing, Insertion/Deletion
- **Zeitfortschreibung**
  - globale Uhr, äquidistante vs. variable Schrittlänge
  - Warteschlange lokaler Ereignisse

www.amazon.de

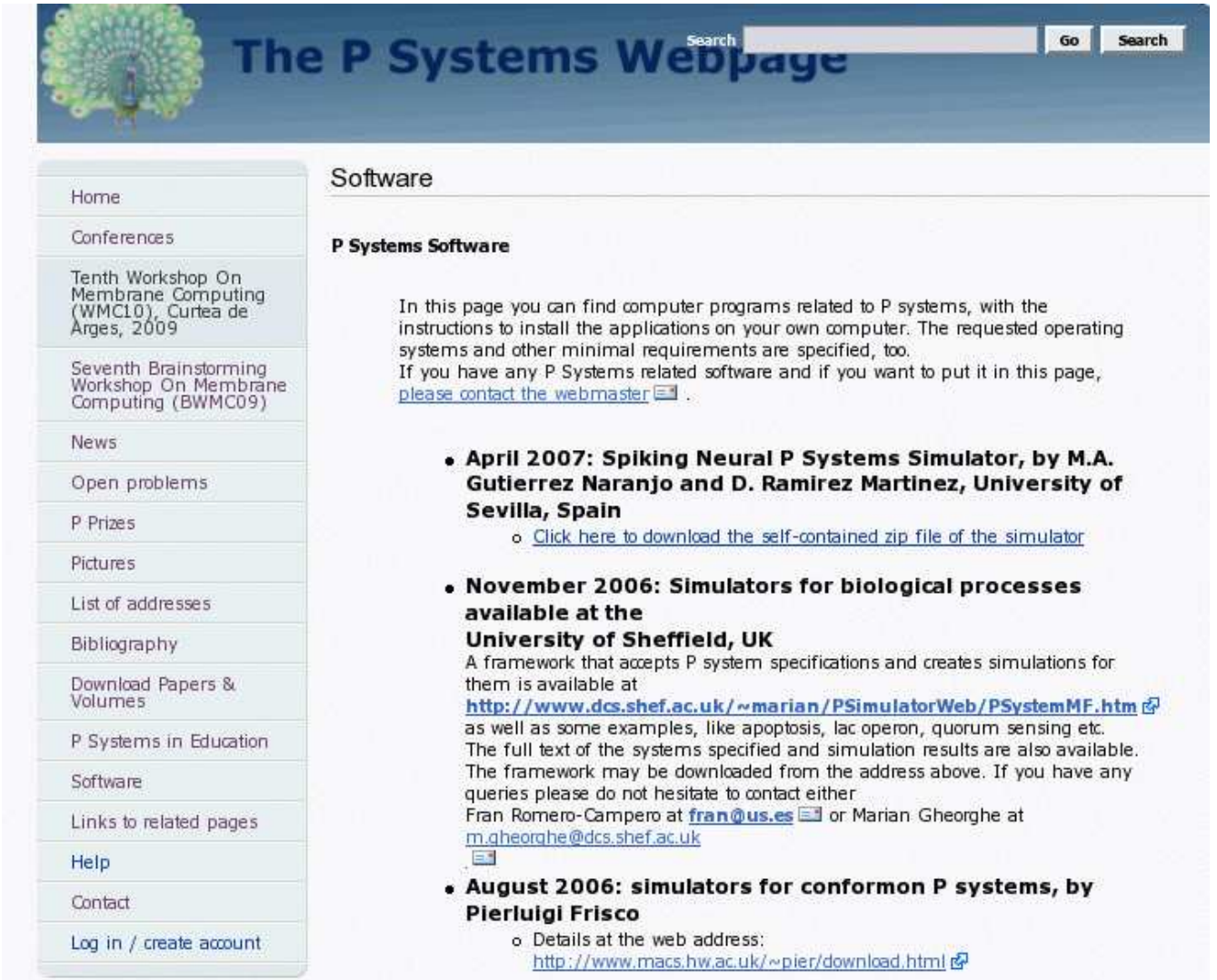
Membrane Systems: International Dimension

- Pioneered in 1998 by Gheorghe Paun
- $\approx$ 10 years scientific evolution
- $\approx$ 100 active researchers in community
- $\approx 1,500$ publications up to now

Gheorghe Păun
www.imar.ro/∼paun



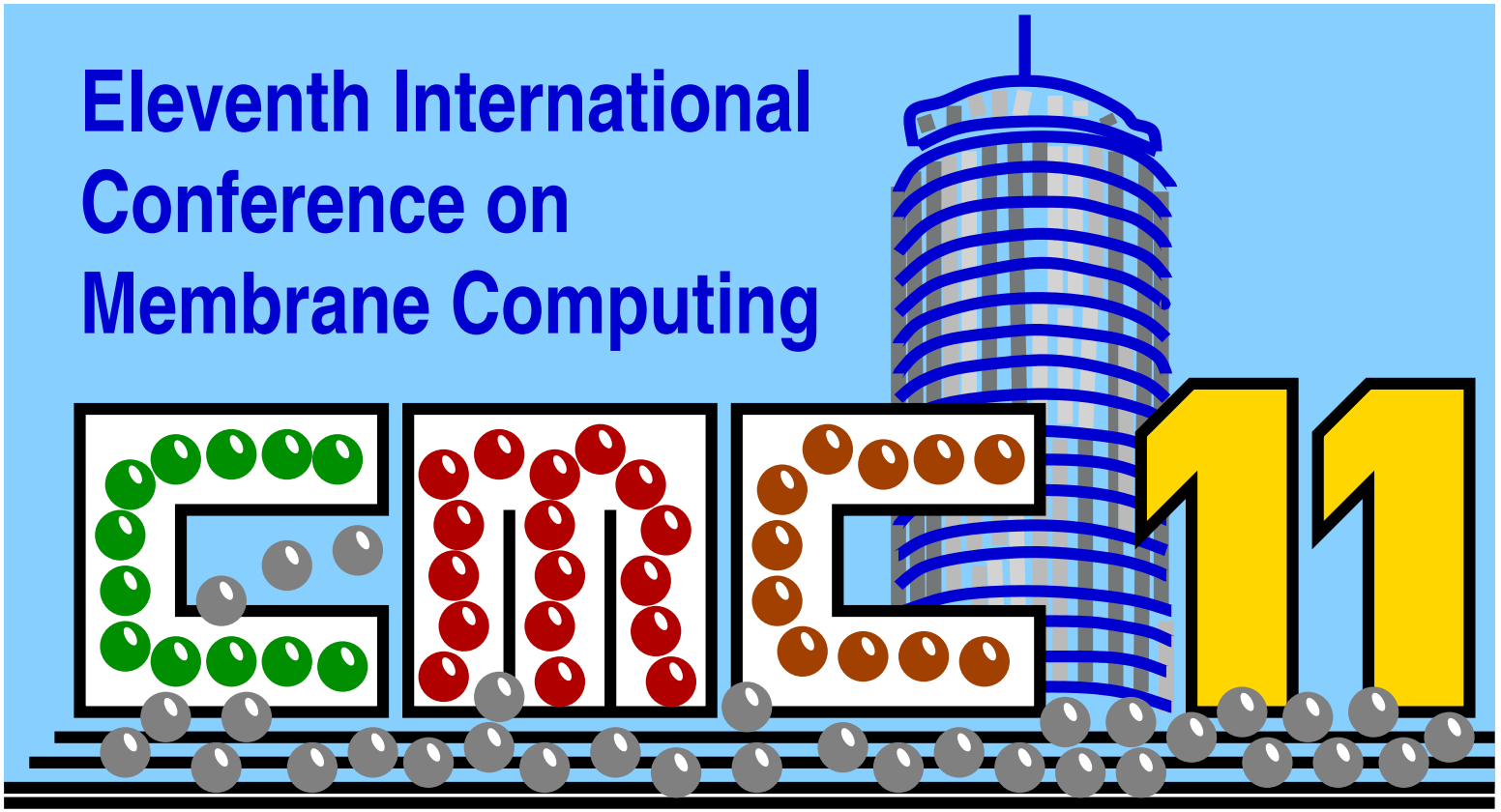
The P Page: A Repository and More . . .



http://ppage.psystems.eu

Eleventh International Conference on Membrane Computing (CMC11)

24-27 August 2010, Jena, Germany



http://cmc11.uni-jena.de

Zusammenfassung

- Zur Beschreibung und Untersuchung biologischer Reaktionsnetzwerke eignen sich neben analytischen Ansätzen auch algebraische Modelle wie Membransysteme.
- Vorteile dieser Modellierungstechnik liegen in der Erfassung von Strukturdynamik, der Einbeziehung räumlicher Gegebenheiten und der expliziten Darstellung einzelner Moleküle.
- Algebraische Modelle ordnen sich unmittelbar in Berechnungsmodelle der theoretischen Informatik ein.
- Nichtdeterministische Modelle gestatten eine massiv-datenparallele Arbeitsweise, mit deren Hilfe z.B. NP-vollständige Probleme zeiteffizient gelöst werden können (jedoch exponentiell viel Speicherplatz nötig ist).
- Beschreibung biologischer Systeme offenbart zugrunde liegende Programme der Informationsverarbeitung.