Zellsignalnetzwerke – Biologische Computer mit universeller Berechnungsstärke

Thomas Hinze Thorsten Lenser Peter Dittrich

Friedrich-Schiller-Universität Jena
Arbeitsgruppe Biosystemanalyse
Ernst-Abbe-Platz 1–4, D-07743 Jena

{hinze,thlenser,dittrich}@minet.uni-jena.de

25. August 2006

Zusammenfassung

Leben ist durch die Verarbeitung von Signalen *in vivo* gekennzeichnet. In Form von Proteinen, Hilfsstoffen oder physikalischen Reizen an die äußere Membran einer Zelle herangeführt, durchlaufen diese Signale ein komplexes Netzwerk biochemischer Reaktionen, die zumeist in einer gezielten Ausschüttung spezifischer Proteine infolge einer Genexpression münden. Die so entstandene Zellantwort kann ein breites Wirkungsspektrum sowohl innerhalb als auch in der Umgebung der betroffenen Zelle entfalten. Das Aufdecken globaler Gesetzmäßigkeiten für die Vielzahl unterscheidbarer Zellsignalnetzwerke (CSNs) gehört zu den derzeitigen Aufgaben der Bioinformatik, die eine mathematische Modellierung und Analyse motivieren. P-Systeme des Membrane Computing haben sich als ein geeigneter Ansatz hierfür erwiesen. Nachfolgend wird die P-System-Klasse Π_{CSN} definiert, die Protein-Substrukturen sowie ein Matching zur Identifizierung reaktionsspezifischer Moleküle in eine deterministisch-dynamische Systembeschreibung einbettet. Es zeigt sich, dass derartige Systeme im berechenbarkeitstheoretischen Sinn universell sind. Der entsprechende Nachweis wird durch Simulation von Registermaschinen erbracht.

1 Einführung

Biochemische Signalnetzwerke lebender Zellen lassen sich als Berechnungsmodelle der Informatik auffassen. Ihre Informationsträger sind größtenteils Proteine, die zum Zweck einer intrazellularen Signalverarbeitung schrittweise modifiziert werden. Hierbei erfolgt eine gezielte Aktivierung ausgewähl-

ter Proteine, beispielsweise durch Phosphorylierung bestimmter Bindungsdomänen, durch Konformitätsänderungen oder durch Komplexbildungen [1]. Im Ergebnis von Aktivierungen kann eine Proteinsynthese im Zellkern angestoßen werden, wodurch eine Signalantwort mit Rückgriff auf genomisch gespeicherte Daten möglich ist [2]. Die bei der Zellsignalverarbeitung auftretenden biochemischen Reaktionen sind eng miteinander vernetzt und bilden ein feinabgestimmtes, rückkopplungsfähiges System. Störungen in diesem System gelten als Auslöser für eine Reihe von Erkrankungen. Dem Aufdecken von Gesetzmäßigkeiten der Zellsignalverarbeitung und dem Nachzeichnen ihrer evolutionären Entwicklung kommt deshalb eine besondere Bedeutung zu. Die Modellierung verfolgt verschiedene Ansätze, die sich in analytische, stochastische sowie algebraische Beschreibungen einteilen lassen. Letztere gestatten einen komponentenweisen Systemaufbau und widerspiegeln den diskreten Charakter der Zellsignalverarbeitung. Zu den algebraischen Beschreibungen zählen zustandsbasierte Modelle (wie *abstract machines* oder *X machines*), Prozesskalküle (wie *Petri nets*, π *calculus* oder *ambient calculus*) und Termersetzungssysteme [7], zu denen die *P systems* (P-Systeme) gehören [5]. Sie stellen einen flexiblen Modellierungsrahmen für Vorgänge des membranbasierten Computing dar, wobei eine Zelle in verschiedene Sektionen (Membranen) unterteilt wird, in denen chemische Reaktionsgleichungen auf einer Multimenge molekülkodierender Objekte zur Anwendung kommen. Entsprechend der Auswahl der jeweils in der Systembeschreibung berücksichtigten Modellierungsaspekte unterscheidet man eine Vielzahl von P-System-Ausprägungen [6]. Zur Erfassung und Verhaltensbeschreibung von Zellsignalnetzwerken dient u.a. die P-System-Klasse Π_{CSN}, die in [3] eingeführt wurde und aufgrund der verwendeten graphbasierten Systemtopologie als Subklasse der *tissue P systems* [4] angesehen werden kann. Die Besonderheit der Systeme Π_{CSN} liegt in der Kombination einer deterministischen Beschreibung des Zeitverhaltens, bei der die maximalparallele Anwendung von Reaktionsregeln durch eine Reaktionskinetik ersetzt wird, mit der Verarbeitung von Stringobjekten, die eine Erfassung sowie ein Matching von Protein-Substrukturen gestatten.

2 Formalsprachliche und multimengenarithmetische Grundlagen

Das Symbol ε bezeichnet das leere Wort. Die Verkettung der Sprachen L_1 und L_2 über dem gemeinsamen Alphabet Σ wird beschrieben durch $L_1 \otimes L_2 = \{uv \mid u \in L_1 \wedge v \in L_2\}$. $\mathcal{P}(L)$ verkörpert die Potenzmenge von L. Sei A eine beliebige Menge und $\mathbf{N}$ die Menge der natürlichen Zahlen einschl.

null. Eine Multimenge über A ist eine Abbildung $F : A \longrightarrow \mathbf{N} \cup \{\infty\}$. $F(a)$ gibt die Vielfachheit des Vorkommens von $a \in A$ in F an. Multimengen lassen sich als elementweise Aufzählung der Form $\{(a_1, F(a_1)), (a_2, F(a_2)), \ldots\}$ darstellen, wobei $\forall (a, b_1), (a, b_2) \in F : b_1 = b_2$. Der Support $\mathrm{supp}(F) \subseteq A$ von F ist definiert durch $\mathrm{supp}(F) = \{a \in A \mid F(a) > 0\}$. Eine Multimenge F über A ist leer gdw. $\forall a \in A : F(a) = 0$. Seien F_1 und F_2 Multimengen über A. F_1 ist eine Teilmenge von F_2, notiert $F_1 \subseteq F_2$, gdw. $\forall a \in A : (F_1(a) \leq F_2(a))$. Die Multimengen F_1 und F_2 sind gleich gdw. $F_1 \subseteq F_2 \wedge F_2 \subseteq F_1$. Die Schnittmenge $F_1 \cap F_2 = \{(a, F(a)) \mid a \in A \wedge F(a) = \min(F_1(a), F_2(a))\}$, die Multimengensumme $F_1 \uplus F_2 = \{(a, F(a)) \mid a \in A \wedge F(a) = F_1(a) + F_2(a)\}$ sowie die Multimengendifferenz $F_1 \ominus F_2 = \{(a, F(a)) \mid a \in A \wedge F(a) = \max(F_1(a) - F_2(a), 0)\}$ bilden Multimengenoperationen. Multiplikation einer Multimenge $F = \{(a, F(a)) \mid a \in A\}$ mit einem Skalar c, notiert $\mathrm{c} \cdot F$, ist definiert durch $\{(a, \mathrm{c} \cdot F(a)) \mid a \in A\}$. Der Term $\langle A \rangle = \{F : A \longrightarrow \mathbf{N} \cup \{\infty\}\}$ beschreibt die Menge aller Multimengen über A.

3 Systemdefinition

Ein P-System zur Beschreibung von Zellsignalnetzwerken ist ein Tupel

$$\Pi_{\mathrm{CSN}} = (V, V', E, M, n),$$

wobei V und V' zwei Alphabete bereitstellen, o.B.d.A. $\#, \neg, * \notin V \cup V'$. Die Komponenten E und M spezifizieren Kanäle (*edges*) und Reaktionsräume (*modules*). Deren Anzahl $n = |M| \in \mathbf{N} \setminus \{0\}$ gibt den Grad des Systems an. Die Syntax molekülkodierender Stringobjekte wird durch die reguläre Menge $S = V^+ \otimes \big(\{\#\} \otimes ((V')^+ \cup \{\neg\} \otimes (V')^+ \cup \{*\})\big)^*$ festgelegt.

Jeder Reaktionsraum aus der endlichen Menge $M = \{M_1, \ldots, M_n\}$ definiert ein Tupel $M_i = (R_{i1}, \ldots, R_{ir_i}, \mathrm{f}_{i1}, \ldots, \mathrm{f}_{ir_i}, A_i)$, wobei jedes $R_{ij} \in \langle S \rangle \times \langle S \rangle$ eine Reaktionsgleichung, bestehend aus den Multimengen der Edukte und Produkte, angibt, die Funktion $\mathrm{f}_{ij} : \langle S \rangle \longrightarrow \mathbf{N}$ die Kinetik der Reaktion R_{ij} bestimmt sowie die Multimenge $A_i \in \langle S \rangle$ den Reaktionsraum M_i mit Stringobjekten initialisiert. Die Menge der Kanäle ist gegeben durch $E \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\} \times \mathcal{P}(S \times \{\mathrm{g} : \langle S \rangle^2 \longrightarrow \mathbf{N}\}) \times \mathbf{N}$. Jeder Kanal $e_{ij} = (i, j, I_{ij}, \mathrm{d}_{ij}) \in E$ repräsentiert eine gerichtete Verbindung von M_i nach M_j, die von Stringobjekten passiert werden kann, die den Filter (*interface*) $I_{ij} \subseteq \{(w, \mathrm{g}_{w,ij}) \mid w \in S \wedge \mathrm{g}_{w,ij} : \langle S \rangle^2 \longrightarrow \mathbf{N}\}$ matchen. Der Rezeptor w gibt die Struktur dieser Stringobjekte vor, während die Funktion g die maximale Kanalkapazität liefert. Die Zahl d_{ij} notiert die für die Passage von e_{ij} benötigte Zeit (Systemschrittanzahl, *delay*). Da die Syntax S der Stringobjekte bei der Erfassung von Moleküleigenschaften Platzhaltersymbole $*$ und Ausschlussterme, eingeleitet mit $\neg$, in den durch

getrennten Substrings zulässt [3], wird ein Matching (Ähnlichkeitstest) von Stringobjekten notwendig, um die Filterung und die Objektauswahl bei Anwendung der Reaktionsgleichungen zu modellieren. Dies geschieht durch eine geeignete Matching-Relation $Match_x \subseteq S \times S$ und ihre Erweiterung $Match(w) = \{s \in S \mid (w,s) \in Match_x\}$. O.B.d.A. gelte für alle Filter: $\bigcap_{j\in\{1,\ldots,n\}} Match(\mathrm{supp}(I_{ij})) = \emptyset \; \forall i \in \{1,\ldots,n\}$. Das dynamische Systemverhalten entsteht unter Nutzung eines globalen Taktes $t \in \mathbf{N}$. Die Multimenge $L_i(t)$ bezeichnet den Inhalt von Reaktionsraum M_i zum Zeitpunkt t:

$$\begin{aligned} L_i(0) &= A_i, \quad L_i(\tau) = \emptyset \text{ für } \tau < 0 \\ L_i'(t) &= L_i(t) \ominus Educts_i(t) \uplus Prod_i(t) \\ L_i(t+1) &= L_i'(t) \ominus Outgoing_i(t) \uplus Incoming_i(t) \end{aligned}$$

Die einzelnen Phasen jedes Systemschrittes werden synchron in den Reaktionsräumen ausgeführt. In $R_{ij} = (F_A, F_B)$ bildet $\mathrm{supp}(F_A) = \{a_1,\ldots,a_p\}$ die Menge der Edukte und $\mathrm{supp}(F_B) = \{b_1,\ldots,b_q\}$ die Menge der Produkte. Es gilt: $Educts_{ij}(t) = \biguplus_{e_1\in Match(a_1)} \cdots \biguplus_{e_p\in Match(a_p)} \mathrm{f}_{ij}\big(\{(e_1,\infty),\ldots,(e_p,\infty)\} \cap L_i(t)\big) \cdot \big\{(e_1, F_{A_{ij}}(a_1)),\ldots,(e_p,F_{A_{ij}}(a_p))\big\}$ und $Educts_i(t) = \biguplus_{j\in\{1,\ldots,r_i\}} Educts_{ij}(t)$ sowie $Prod_{ij}(t) = \biguplus_{e_1\in Match(a_1)} \cdots \biguplus_{e_p\in Match(a_p)} \mathrm{f}_{ij}\big(\{(e_1,\infty),\ldots,(e_p,\infty)\} \cap L_i(t)\big) \cdot \big\{(b_1, F_{B_{ij}}(b_1)),\ldots,(b_q,F_{B_{ij}}(b_q))\big\}$ und $Prod_i(t) = \biguplus_{j\in\{1,\ldots,r_i\}} Prod_{ij}(t)$. Ferner gilt: $Outgoing_{ij}(t) = L_i'(t) \cap \big\{(v, \mathrm{g}_{w,ij}(L_i'(t), L_j'(t))) \big| v \in S \wedge w \in \mathrm{supp}(I_{ij}) \wedge v \in Match(w)\big\}$ und $Outgoing_i(t) = \biguplus_{j\in\{1,\ldots,n\}} Outgoing_{ij}(t)$ sowie $Incoming_i(t) = \biguplus_{k\in\{1,\ldots,n\}} Outgoing_{ki}(t - \mathrm{d}_{ki})$, so dass $L(\Pi_{\mathrm{CSN}}) = \mathrm{supp}\left(\biguplus_{t=0}^{\infty}\left(\biguplus_{i=1}^{n} L_i(t)\right)\right)$.

4 Nachweis der Universalität

Die Universalität von Systemen Π_{CSN} lässt sich durch Simulation von Registermaschinen (RAMs) nachweisen, die ihrerseits als universelles Berechnungsmodell bekannt sind [8]. Gegeben sei eine RAM $= (R, L, P, l_1)$ mit $m \in \mathbf{N}$ Registern $R = \{r_1,\ldots,r_m\}$, ihren Anfangsinhalten $r_k \in \mathbf{N}$, der Menge der Sprungmarken $L = \{l_1,\ldots,l_c\}$, o.B.d.A. fortlaufend nummeriert, dem Programm P sowie der Sprungmarke l_1 des ersten abzuarbeitenden Befehls. Es stehen die Befehle $l_i : \texttt{INCR}(r_k), l_j$ (Inhalt des Registers r_k inkrementieren und zu l_j springen), $l_i : \texttt{DECR}(r_k), l_j$, der Test $l_i : r_k \neq 0, l_j, l_p$ (Falls $r_k \neq 0$, springe zu l_j, sonst zu l_p) sowie der Haltbefehl $l_i : \texttt{HALT}$ zur Verfügung. Für zwei Befehle $l_a : x$ und $l_b : y$ einer RAM gelte: $l_a \neq l_b$ (Determinismus). Die Menge P enthält alle Befehle einer RAM. Bei Erreichen des Haltbefehls stoppt die Abarbeitung mit Ausgabe der Registerinhalte r_1 bis r_m.

Für jeden der c Befehle aus P und für jedes der m Register aus R werden in $\Pi_{\mathrm{CSN}} = (V, V', E, M, c+m)$ jeweils ein separater Reaktionsraum und Kanäle angelegt:

RAM	Π_{CSN}
Befehl $l_i : \texttt{INCR}(r_k), l_j$	$M_i = \left(PC\#i \to PC\#j + A, 1, A_i = \begin{cases} \{(PC\#1, 1)\} \text{ falls } i = 1 \\ \emptyset \text{ sonst} \end{cases}\right)$ $(i, j, \{(PC\#j, 1)\}, 0),\ (i, c+k, \{(A, 1)\}, 0) \in E$
Befehl $l_i : \texttt{DECR}(r_k), l_j$	$M_i = \left(PC\#i \to PC\#j + B, 1, A_i = \begin{cases} \{(PC\#1, 1)\} \text{ falls } i = 1 \\ \emptyset \text{ sonst} \end{cases}\right)$ $(i, j, \{(PC\#j, 1)\}, 0),\ (i, c+k, \{(B, 1)\}, 0) \in E$
Befehl $l_i : r_k \neq 0, l_j, l_p$	$M_i = \left(R_{i1}, R_{i2}, R_{i3}, R_{i4}, 1, 1, 1, 1, A_i = \begin{cases} \{(PC\#1, 1)\} \text{ falls } i=1 \\ \emptyset \text{ sonst} \end{cases}\right)$ $R_{i1} : PC\#i \to PC\#X + D_i, \quad R_{i2} : PC\#X \to PC\#W + D_i,$ $R_{i3} : PC\#W + 2E_i \to PC\#j, \quad R_{i4} : PC\#W + H_i \to PC\#p$ $(i, j, \{(PC\#j, 1)\}, 0), (i, p, \{(PC\#p, 1)\}, 0), (i, c+k, \{(D_i, 1)\}, 0) \in E$
Befehl $l_i : \texttt{HALT}$	$M_i = \left(\emptyset, \emptyset, A_i = \begin{cases} \{(PC\#1, 1)\} \text{ falls } i = 1 \\ \emptyset \text{ sonst} \end{cases}\right)$
Register r_k initialisiert mit α	$M_{c+k} = (R_{c+k,1}, R_{c+k,2}, ..., R_{c+k,2c+1}, 1, ..., 1, A_{c+k} = \{(A, \alpha)\})$ $R_{c+k,1}$: $B+A \to \emptyset$, $R_{c+k,2\mu}$: $D_\mu + A \to A + E_\mu$, $R_{c+k,2\mu+1}$: $2D_\mu \to H_\mu$ $(c+k, \mu, \{(E_\mu, 1), (H_\mu, 1)\}, 0) \in E \qquad \forall \mu = 1, \ldots, c$
	$V = \{A, B, PC\} \cup \{D_\mu, E_\mu, H_\mu \mid \mu = 1, \ldots, c\}$ $V' = \{W, X\} \cup \{\mu \mid \mu = 1, \ldots, c\}$

Das umlaufende Programm-Counter-Molekül $PC\#i$ verweist auf den aktuell ausgeführten Befehl l_i, während jedes Molekülexemplar A eine Zähleinheit der Registerinhalte verkörpert. Als abzählbar unendliche Ressource dient die Molekülanzahl A in den registerkodierenden Reaktionsräumen.

Danksagung. Die vorliegende Arbeit ist Teil des Forschungsprojektes ESIGNET (Evolving Cell Signalling Networks in Silico), welches aus Mitteln des 6. Rahmenprogrammes der EU gefördert wird (Projektnr. 12789).

Literatur

[1] B.L. Cooper, N. Schonbrunner, G. Krauss. *Biochemistry of signal transduction and regulation.* Wiley-VCH, 2001

[2] T. Hinze, M. Sturm. *Rechnen mit DNA – Eine Einführung in Theorie und Praxis.* Oldenbourg-Wissenschaftsverlag München, Wien, 2004

[3] T. Hinze, T. Lenser, P. Dittrich. *A Protein Substructure Based P System for Description and Analysis of Cell Signalling Networks.* Prel. Proceedings Seventh Workshop on Membrane Computing, Leiden, 2006, accepted for LNCS

[4] C. Martin-Vide, G. Păun, J. Pazos, A. Rodriguez-Paton. *Tissue P Systems.* Theoretical Computer Science **296(2)**:295-326, 2003

[5] G. Păun. *Computing with Membranes.* Journal of Computer and System Sciences **61(1)**:108-143, 2000

[6] G. Păun. *Membrane Computing: An Introduction.* Springer-Verlag Berlin 2002

[7] G. Rozenberg, A. Salomaa (Eds.). *Handbook of formal languages 1-3.* Springer-Verlag Berlin, 1999

[8] J.C. Shepherdson, H.E. Sturgis. *Computability of recursive functions.* Journal of the Association for Computing Machinery **10(2)**:217–255, 1963