Zur formalen Beschreibung von Aminosäureketten und Proteinen

Hermann von Issendorff
Institut für Netzwerkprogrammierung
D-21745 Hemmoor, Hauptstrasse 40
hviss@issendorff.de

Zusammenfassung. Proteine haben nicht nur eine räumliche Struktur, sondern sind auch in der Lage, bestimmte Funktionen auszuführen. Andererseits wird ein Proteine als Kette von Aminosäuren aufgebaut, die sich nach der Fertigstellung zu einem Protein zusammenzieht. In der Kette von Aminosäuren müssen daher bereits alle Eigenschaften des Proteins codiert sein. Die Aminosäurenkette kann somit auch als ein in einer bisher unbekannten Proteinsprache geschriebenes Programm gedeutet werden. In diesem Beitrag stellen wir eine formale Methode vor, Aktonalgebra genannt, mit der sich gleichermassen räumliche Strukturen und DV-Funktionen beschreiben lassen. Sie vermag vielleicht dazu beizutragen, die Proteinsprache schneller zu entschlüsseln.

1 Einleitung

Jede Zelle eines Organismus ist eine kleine Fabrik, in der u.a. fortwährend bestimmte Proteine erzeugt und verbraucht werden. Die bei der Proteinerzeugung ablaufenden Vorgänge sind recht genau bekannt. Ribosome suchen zu diesem Zweck bestimmte Abschnitte auf den DNA-Strängen, lesen den darin enthaltenen Gencode und erzeugen daraus eine Kette von Aminosäuren. Jede 3-er Folge von Nukleinsäuren beschreibt genau eine von 20 verschiedenen Aminosäuren.

Sobald die Aminosäurenkette freigegeben wird, zieht sie sich zusammen und bildet so ein Protein. Das Zusammenziehen geschieht mit grosser Geschwindigkeit und vor allem mit grosser Präzision und führt zu einer bestimmten räumlichen Struktur. Abhängig von der Reihenfolge der Aminosäuren und der Kettenlänge kann auf diese Weise eine enorme Zahl verschiedener Proteine gebildet werden.

Der eigentliche Zweck der verschiedenen Proteine besteht in den Funktionen, die sie innerhalb der Zelle ausüben. Die erwähnten Ribosome z.B. sind Proteine, die die Fähigkeit haben, auf einem DNA-Strang entlang zu wandern, dem Gencode entsprechend bestimmte Aminosäuren aus der Umgebung aufzunehmen und aneinander zu ketten. Andere Proteine, Proteasen genannt, betreiben z.B. so etwas wie Qualitätssicherung. Sie prüfen die Korrektheit der Proteine und zerschneiden sie in Einzelstücke, wenn sie fehlerhaft sind.

Die Struktur und die Funktionen eines Proteins hängen auf das Engste zusammen. Eine bestimmte Funktion kann nur ausgelöst werden, wenn ein anderes Molekül, etwa ein anderes Protein, mit seinen lokalen chemischen Bindungen genau zu dem Protein passt. Eine gängige Vorstellung ist die vom Schlüssel und dem Schloss.

Die 20 Aminosäuren, aus denen die Proteine gebildet werden, lassen sich als die Elemente einer formalen Sprache interpretieren, mit der sich nicht nur Funktionen beschreiben lassen, wie das von den konventionellen Programmiersprachen bekannt ist,

sondern auch räumliche Strukturen. Eine Kette von Aminosäuren ist unter dieser Interpretation ein Programm, das eine reale Maschine beschreibt, die in Wechselwirkung mit ihrer Umgebung bestimmte Funktionen ausführen kann. Eine Programmiersprache, die diese Eigenschaften hat, ist bisher nicht bekannt.

Wir wollen uns in dieser Arbeit mit genau dieser Frage befassen. Dabei wird zunächst die Frage nach der formalen Beschreibung räumlicher Strukturen im Vordergrund stehen. "Beschreibung" bedeutet dabei "Darstellung als (eindimensionaler) String". Eine derartige Sprache muss aber ausserdem die Eigenschaft haben, die auf oder in den Proteinen ablaufenden Vorgänge zu beschreiben.

Die zur Beschreibung der Strukturen und Vorgänge verwendete Sprache ist die Akton-Algebra, abgekürzt AA, über deren Vorstufen in diesem Workshop schon verschiedentlich berichtet wurde. Bei geeigneter semantischer Interpretation lassen sich mit der AA auch die räumlichen Strukturen von Biomolekülen beschreiben, u.a. auch Helices und Faltungen, die für Proteine typisch sind. Die AA benötigt dafür insgesamt 6 Grundsorten von strukturbildenden Aktonen und dazu mindestens eine Grundsorte funktioneller Aktonen. Die Natur verwendet für den gleichen Zweck 20 verschiedene Aminosäuren. Es besteht damit eine gewisse Aussicht, dass sich die strukturellen und funktionellen Eigenschaften der verschiedenen Aminosäuren als Akton-terme beschreiben lassen. Zumindest könnte die AA einen Hinweis zur Entschlüsselung des Aminosäurencodes geben.

Auf dem Gebiet der klassischen Informatik sind bisher nur wenige Versuche bekannt, die räumliche Struktur von Systemen formal zu beschreiben, und diese sind sehr speziell. Beispiele dafür sind Ruby [1] und CADIC [2]. Beide Ansätze wurden für die Darstellung planarer Schaltungen entwickelt. Ruby ist auf die Beschreibung kombinatorischer Digitalschaltungen beschränkt, und damit auf gerichtete, azyklische, planare Strukturen. CADIC wurde für Layout-Zwecke entwickelt und beschreibt lediglich ungerichtete, planare Strukturen. Auf dem Gebiet der Bioinformatik gibt es Ansätze, planare Strukturen formal zu beschreiben, wie sie z.B. bei Faltungen auftreten [3, 4]. Diese Ansätze enthalten aber keinen Hinweis darauf, wie sie auf die Beschreibung räumlicher Strukturen oder gar Proteinfunktionen erweitert werden könnten.

2 Akton-Algebra

AA beschreibt ein physikalisches System, wie es von einem Beobachter gesehen wird, der zwischen rechts/links, oben/unten und vorne/hinten unterscheidet. Die Komponenten des Systems werden als gerichtet angenommen, d.h. Input und Output liegen auf entgegengesetzten Seiten. Dies ist keine Beschränkung, da Teilstrukturen mit mehreren Inputs und Outputs stets in Komponenten zerlegt werden können. Der Beobachter nimmt immer eine Position ein, sodass abhängige benachbarte Komponenten von links nach rechts angeordnet sind und nichtabhängige benachbarte Komponenten oben und unten. Bei Kreuzungen von Komponenten wird die hintere zerschnitten, und die Schnittenden werden markiert. Der Beobachter sieht damit ein physikalisches System ausgebreitet auf einer Ebene und von links nach rechts orientiert. Die originäre (dreidimensionale) Systemstruktur kann darin auf zwei verschiedene Weisen vollständig beschrieben werden, entweder durch Angabe der Raumwinkel zwischen verbundenen

Komponenten oder durch Annahme physikalischer oder chemischer Anziehungskräfte, wo immer die Systemstruktur in der Darstellung gedehnt oder zerschnitten ist.

AA ist eine mehrsortige Term-Algebra, deren Elemente Aktonen genannt werden. Ein Akton repräsentiert eine Systemkomponente als schwarzen Kasten, d.h. es zeigt nicht seine innere Struktur. Ein Akton hat einen geordneten Input und einen geordneten Output. Beide können leer sein oder eine beliebige (endliche) Menge von Elementen enthalten. Jedes Element hat eine eigene räumliche Position. Diese Position dient zur Identifikation des Elements und bildet das Kriterium für die Oben/Unten-Ordnung der Elemente im Input und Output. Wenn ein Akton Input- und Output-Elemente enthält, dann sind diese üblicherweise durch ein Verarbeitungsnetzwerk miteinander verbunden. Die Verarbeitung, d.h. die Erzeugung eines Output-Elementes aus einem oder mehreren Input-Elementen, dauert eine individuelle Zeit. Die Produktion verschiedener Output-Elemente eines Aktons ist üblicherweise nicht synchronisiert. Jedes Akton mit nichtleerem Output hat mindestens ein Akton als Nachfolger, und jedes Akton mit nichtleerem Eingang hat mindestens ein Akton als Vorgänger.

Ein Term repräsentiert ein Teilsystem als weissen Kasten, d.h. ein Term zeigt seine innere Struktur. Ansonsten gleicht ein Term einem Akton, d.h. es hat einen geordneten Input und einen geordneten Output. Ein Term besteht aus einem Akton oder jeder Aktonstruktur, die mit Hilfe von zwei binären Operatoren gebildet werden kann. Ein weisser Kasten kann immer zu einem schwarzen Kasten abstrahiert werden. Auf diese Weise können Aktonen beliebiger Grösse und Komplexität erzeugt werden.

Zwei benachbarte Terme *x* und *y* werden durch *x:y* beschrieben, wenn der Output von *x* die gleichen Elemente wie der Input von *y* enthält. Der Doppelpunkt ist ein *Next* genannter binärer Operator. *Next* definiert - der Schreibrichtung entsprechend - eine Richtung von links nach rechts.

Zwei benachbarte Terme *x* und *y*, die keine Input/Output-Beziehung haben, werden durch *x/y* beschrieben, wenn *x* oberhalb von *y* liegt. Der Schrägstrich ist ein *Juxta* genannter binärer Operator. Beide Terme *x* und *y* teilen sich in den Input des Term *x/y*. Term *x* bezieht seinen Anteil vom oberen Ende und Ter *y* vom unteren Ende. Überlappen sich beide Anteile, wird dieser Bereich gegabelt.

Die durch *Next* und *Juxta* definierten Links/Rechts-Richtungen und Oben/Unten-Richtungen spannen die Beobachtungsebene auf.

Die Infix-Notation der beiden binären Operatoren dient lediglich der besseren Lesbarkeit. Sie erfordert jedoch die Einführung von Klammern zur Abgrenzung der Terme. Wie üblich wird die Klammerzahl dadurch reduziert, dass für *Juxta* eine stärkere algebraische Bindung angenommen wird als für *Next*. Für eine maschinelle Bearbeitung ist natürlich die klammerfreie Polnische Notation vorzuziehen.

Ausser den binären Operatoren gibt es zwei Typen von Aktonen, Strukuraktonen und Funktionsaktonen genannt.

Insgesamt gibt es 6 Strukturaktonen, die verschiedene Grundstrukturen darstellen. Als Erstes werden die komplemtentären Strukturaktonen *Entry* (∗) und *Exit* (∗') eingeführt. Sie dienen als Systembegrenzer. *Exit* exportiert Elemente in die sichtbare Umgebung und *Entry* importiert Elemente von dort. Demgemäss hat *Exit* einen leeren Output und einen nichtleeren Input. Bei *Entry* ist es umgekehrt. Wenn der Output von *Entry* dem Input von *Exit* gleicht und beide planar benachbart sind, wird das Paar pas-

send (matching) genannt. Nichtpassende, planar benachbarte *Entries* und *Exits* werden durch unterschiedliche Indices unterschieden.

Ein nächstes Paar von Strukturaktonen wird *Up(*o*)* und *Down(*o*')* genannt. *Up* hat die gleichen Eigenschaften wie *Entry* und *Down* wie *Exit*, ausser dass sie eine Verbindung zu einer Umgebung darstellen, die unterhalb der Beobachtungsebene liegt und damit nicht sichtbar ist. Ein passendes (matching) *Down/Up*-Paar ist daher räumlich benachbart. Wie später noch behandelt wird, lassen sich mit *Down/Up*-Paaren gerichtete Kreuzungen eindeutig beschreiben, einschliesslich des Drehsinns.

Ein weiteres Strukturakton wird *Link ($)* genannt. Sein Input und sein Output haben die gleiche Ordnung und die gleiche Kardinalität. Ein *Link* dient dazu, räumlich getrennte abhängige Aktonen miteinander zu verbinden und hat damit die Eigenschaft eines Verbindungskabels.

Das letzte Strukturakton wird *Gap* (#) genannt. Es hat einen leeren Input und einen leeren Output und importiert oder exportiert nichts. Es kann als ein Stück Materie betrachtet werden, das als Abstandshalter dient.

Die Strukturaktonen und die für ihre algebraische Darstellung verwendeten Symbole sind in der nachstehenden Tabelle zusammengefasst:

Entry	*Exit*	*Up*	*Down*	*Link*	*Gap*
*	*'	o	o'	*$*	*#*

Funktionsaktonen haben geordnete Inputs und Outputs. Jedes Output-Element ist über ein Verarbeitungsnetzwerk mit Input-Elementen und/oder aktoninternen Konstanten verbunden. Im Unterschied zum *Link* produziert ein Funktionsakton im Allgemeinen einen Output, der sich vom Input unterscheidet. Im einfachsten Fall besteht ein Funktionsakton aus einem digitalen Gatter oder einem Inverter. Der Output enthält in diesem Fall ein einziges Element. Funktionsaktonen sind aber nicht auf diese Schaltelemente beschränkt, sondern können Verarbeitungsnetzwerke jeder Grösse und Komplexität enthalten.

Eine Besonderheit der Funktionsaktonen ist, dass sie über eine explizit angegebene Bedingung kontrolliert werden können. Diese Bedingung ist eine dreiwertige Boolesche Variable, die konjunktiv auf alle Output-Elemente wirkt. Die drei Werte sind *1, 0* und *u (undefiniert)*. Die Bedingung kann in eckigen Klammern an ein Funktionsakton angehängt werden. Das heisst, wenn *a* der Name eines Funktionsaktons ist und α die Bedingung, dann kann das bedingte Akton durch $a[\alpha]$ bezeichnet werden. Da üblicherweise Boolesche Variable verwendet werden, die nur die Werte *1* und *0* annehmen, führen wir eine spezielle Funktion *s* ein, die den Wert *0* in *u* überführt. Das heisst, wenn $p \in \{1, 0\}$ *ist*, dann ist $s(p) \in \{1, u\}$.

Wir verwenden in dieser Arbeit die Buchstaben *u,v,w,x,y,z* für Terme. Andere Buchstaben dienen als Bezeichner für Aktonen und andere Objekte.

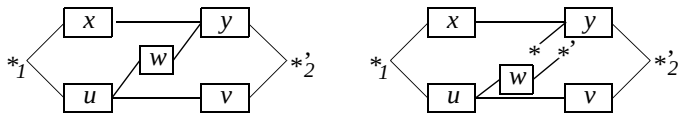
3 Vom Raum zur AA

Die Beschreibung von physikalischen Systemen durch die AA beruht auf zwei wohldefinierten Abbildungen. Die erste Abbildung reduziert die drei Dimensionen des phy-

sikalischen Systems auf die zwei Dimensionen einer planaren Darstellung, und die zweite die planare Darstellung auf eine lineare. Die lineare Darstellung wird durch die AA wiedergegeben. Beide Abbildungen können durch ein Gummibandmodell visualisiert werden, in dem die Gummibänder die Input/Output-Beziehungen zwischen den Komponenten veranschaulichen. Dabei ist jedoch zu beachten, dass die Gummibänder nur virtuelle Beziehungen darstellen und sonst keine physikalische Bedeutung haben.

Das Gummibandmodell nimmt an, dass alle Verbindungen zwischen Aktonen beliebig gestreckt oder gebogen werden können. Die erste Abbildung beinhaltet die Ausrichtung aller Teilstrukturen von links nach rechts. Dies gilt insbesondere für topologische Schleifen und Rückkopplungszyklen. Das System wird so weit in Links/Rechts- und Oben/Unten-Richtung gedehnt, dass alle Aktonen für den Beobachter vollständig sichtbar werden. Dies ergibt ein System, das bezüglich der Aktonen planar ist, das aber noch lokale dreidimensionale Strukturen enthält, d.h. Kreuzungen, die von sich kreuzenden unabhängigen Verbindungen, topologischen Schleifen oder Rückkopplungszyklen stammen. Diese lokalen dreidimensionalen Strukturen können nun dadurch beseitigt werden, dass jede unterführende Verbindung zerschnitten wird und durch ein passendes *Down/Up*-Paar ersetzt wird. Das Endresultat dieser Abbildung ist eine gerichtete, partiell geordnete, planare Struktur.

Die zweite Abbildung dehnt die planare Struktur weiter in Links/Rechts-Richtung, bis alle Aktonen aufgereiht sind. Das geschieht dadurch, dass das oberste *Entry* oder *Up* an der linken Seite und das unterste *Exit* oder *Down* an der rechten Seite auseinander gezogen werden. Auf diese Weise lassen sich benachbarte unabhängige Teilstrukturen ineinander verschachteln (interleaving), und zwar so, dass obere Teilstrukturen vor den unteren liegen. Es gibt jedoch auch benachbarte Teilstrukturen, die untereinander verbunden und damit nicht unabhängig sind. Solche Teilstrukturen können nicht in eindeutiger Weise verschachtelt werden. Das Problem ist in der Abbildung 1 dargestellt.



Planare Aktonstruktur Modifizierte planare Aktonstruktur

$*_1:(x/*:y)/(u:(w:*')/v):*'_2$

Lineare Aktonstruktur

Abbildung 1. Nichtverschachtelbare Aktonstruktur, verschachtelbare Aktonstruktur und AA-Ausdruck

Die Darstellung der Aktonenstrukturen in Abbildung 1 und allen folgenden Abbildungen muss erläutert werden. Funktionsaktonen und Funktionsaktonen enthaltende Teilstrukturen werden durch benannte Kästchen dargestellt und Strukturaktonen durch ihre Symbole. Die Linien sind die Gummibänder. Durchgehende Linien kennzeichnen explizite Abhängigkeiten zwischen Teilstrukturen. Implizite Beziehungen passender *Exit/Entry*-Paare können durch gebrochene Linien angedeutet werden und die passen-

der *Down/Up*-Paare durch gepunktete Linien.

Die planare Struktur links in Abbildung 1 besteht aus einer oberen Teilstruktur *x,y*, einer unteren *u,v* und einer sie verbindenden Teilstruktur *w*. Teilstruktur *w* hat die Eigenschaft, sowohl unter *x* zu liegen als auch über *v* und kann deshalb nicht eindeutig in eine Kette eingeordnet werden, dass sie vor *v* und hinter *x* liegt. In der planaren Struktur rechts in Abbildung 1 ist die Verbindung zwischen w und y aufgeschnitten und ein *Exit/Entry*-Paar eingesetzt. Diese Struktur lässt sich linearisieren und damit umkehrbar eindeutig durch den AA-Ausdruck unten in der Abbildung beschreiben.

4 Physikalische Strukturen

Die Strukturelemente der AA haben bisher nur eine bescheidene Semantik, die sich auf einige allgemeine funktionale und räumliche Eigenschaften bezieht. Um physikalische Systeme vollständig beschreiben zu können, muss man aber noch weitere Eigenschaften definieren. Ein wesentlicher Unterschied besteht z.B. zwischen multiplanaren und echten räumlichen Systemen. Obgleich in dieser Arbeit die Beschreibung echter räumlicher Systeme im Vordergrund steht, erscheint es sinnvoll, multiplanare Systeme nicht ganz unbeachtet zu lassen. Dies vor allem angesichts der enormen Bedeutung, die sie in der Schaltkreistechnik haben.

In der multiplanaren Schaltkreistechnik wird *Next* als eine flexible Kopplung zwischen den verknüpften Teilstrukturen interpretiert, die nur Platz beansprucht, wenn beide Teilstrukturen nicht die gleiche Richtung haben. Für jede andere Richtung gilt der minimale Platz. *Juxta* wird als vertikale Ausrichtung sich berührender Teilstrukturen interpretiert. Ein passendes *Down/Up*-Paar wird als zwei Mengen von Durchkontaktierungen angenommen, die auf einer zweiten Ebene verbunden sind. Ein passendes *Exit/Entry*-Paar wird als Verbindung ohne Platzbedarf interpretiert, ein nichtpassendes *Exit/Entry*-Paar dagegen als die beiden Seiten eines Steckers. *Links* sind echte Leitungen, die in geeigneter Länge überall dort eingesetzt werden, wo unmittelbar abhängige Aktonen nicht aneinander grenzen.

Im Fall echter räumlicher Strukturen werden *Next* und *Juxta* als physikalisch/chemische Bindungen interpretiert. *Down/Up*-Paare und *Exit/Entry*-Paare werden jeweils als Objekte angesehen, die sich auf Grund physikalisch/chemischer Kräfte gegenseitig anziehen. Passende Paare bilden auf diese Weise Verbindungen innerhalb des betrachteten Systems, während einzelne *Down*-, *Up*-, *Exit*- oder *Entry*-Aktonen als Andockstellen zu anderen Systemen dienen. Die Kräfte mögen sich in der Reihenfolge *Next*, *Juxta*, *Exit/Entr* , *Down/Up* abschwächen.

Mit diesen Interpretationen beschreibt AA ein lineares physikalisches System, das die Fähigkeit hat, sich selbst in wohldefinierter Weise in ein räumliches physikalisches System zu verwandeln, genauso, wie eine Kette von Aminosäuren sich in ein Protein verwandelt.

Es gibt zwei Basisstrukturen, die aus der Sicht eines Beobachters in nahezu jedem physikalischen System auftreten, die Kreuzung und die topologische Schleife. Eine Kreuzung besteht aus zwei unabhängigen Teilstrukturen, die sich überlagern. Eine Kreuzung ist damit eine dreidimensionale Struktur, in der eine Teilstruktur von der anderen überdeckt wird. Abhängig davon, welche der Teilstrukturen die andere über-

deckt, gibt es zwei komplementäre Kreuzungen. In AA lässt sich mittels des *Down/ Up*-Paars eindeutig beschreiben, welche der beiden Teilstrukturen unterhalb der anderen liegt. AA hat damit die Fähigkeit, die Chiralität, d.h. den Drehsinn, von Strukturen zu beschreiben.

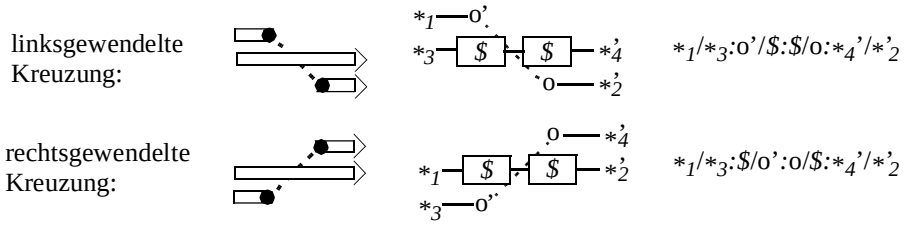


Abbildung 2. Physikalische Struktur, Aktonstruktur und AA-Ausdruck von komplementären Kreuzungen

Abbildung 2 zeigt oben eine linksgewendelte und darunter eine rechtsgewendelte Kreuzung. Dargestellt sind jeweils die eigentliche Struktur, das Gummibandmodell und der AA-Ausdruck. Die schwarzen Punkte in der eigentlichen Struktur markieren *Down* und *Up* und die gepunktete Linie dazwischen ihre unterführende Verbindung. In biplanarer Umgebung stellen die schwarzen Punkte Durchkontaktierungen dar, in allgemeiner dreidimensionalen Umgebung physikalische Objekte, die sich gegenseitig anziehen.

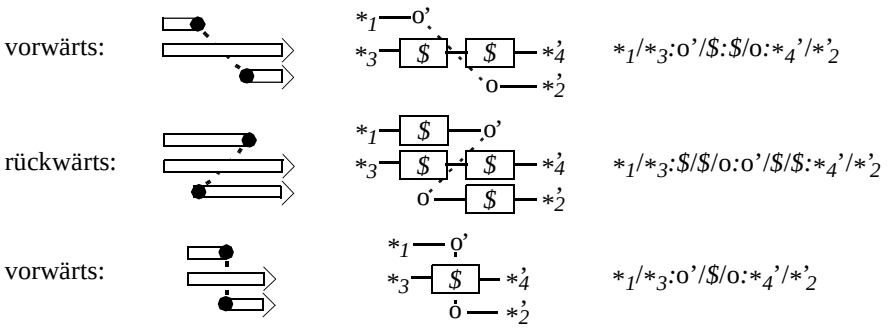
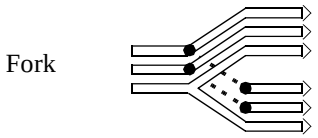


Abbildung 3. Vorwärts- und Rückwärtstypen einer linksgewendelten Kreuzung

Bei genauerer Betrachtung stellen die in Abbildung 2 gezeigten Kreuzungen nur einen Typ dar, der dadurch gekennzeichnet ist, dass das *Down* vor dem *Up* liegt. Entsprechend soll er Vorwärtstyp genannt werden. Dazu gibt es aber auch einen Rückwärtstyp, bei dem *Up* vor *Down* liegt. Der dritte Kreuzungstyp, bei de *Down* und *Up* gleichauf liegen, wird dem Vorwärtstyp zugeschlagen. Abbildung 3 zeigt die drei Typen einer linksgewendelten Kreuzung, wieder dargestellt als eigentliche Struktur, Gummibandmodell und AA-Ausdruck.

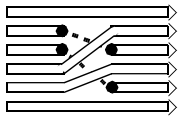
Die Vorwärtskreuzung ist ein essenzieller Bestandteil von zwei Grundstrukturen, die sowohl in biplanaren als auch in allgemeinen räumlichen Systemen auftreten. Sie sollen Fork und Shuffle benannt werden. Fork gabelt ein geordnetes Bündel von Ver-

bindungen in zwei Bündel, die die gleiche Ordnung haben. Shuffle vertauscht die Verbindungen von zwei geordneten Bündeln gleicher Kardinalität so, dass die Ordnung auf Verbindungspaare übertragen wird. Abbildung 4 zeigt beide (linksgewendelten) Grundstrukturen in physikalischer Darstellung und als AA-Ausdruck. Ebenso lassen sich natürlich auch rechtsgewendelte Grundstrukturen beschreiben.



Fork

*1/*2/*3:
($/o')/($/o')/($/$):($/$/$)/(o/o/$):
(*'4/*'5/*'6)/(*'7/*'8/*'9)



Shuffle (S^+):

*1/*2/*3/*4/*5/*6:
($/o'/o')/($/$/$):($/$)/(o/$)/(o/$):
'7/'8/*'9/*'10/*'11/*'12

Abbildung 4. Fork und Shuffle, zwei essenzielle Grundstrukturen

Fork tritt im Allgemeinen nicht explizit in Erscheinung, da es integraler Bestandteil von *Juxta* ist. Fork kann jedoch explizit gemacht werden, wenn das erforderlich ist. Shuffle dagegen ist ein eigenständiges Akton, das dementsprechend für die AA-Beschreibung eine eindeutige Bezeichnung braucht. Es soll mit dem Symbol S^+ bezeichnet werden, wobei das '+' bedeutet, dass es so viele Linienpaare wie notwendig erzeugt. In der Datenverarbeitung tritt ein Shuffle gewöhnlich mit einer nachfolgenden Menge von *And*- oder *Or*-Gattern auf, die jeweils ein Verbindungspaar zu einer Verbindung reduzieren. Diese multiplen Kombinationen können entsprechend zu SA^+ und SO^+ zusammengefasst werden.

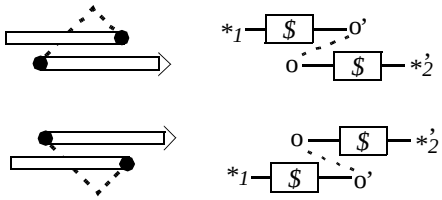


Abbildung 5. Physikalische Struktur, AA-Struktur und -Ausdruck komplementärer offener Schleifen

Eine topologische Schleife ist nur strukturell. Eine Rückkopplung dagegen ist strukturell und funktionell. Rückkopplungen werden im nächsten Abschnitt behandelt. Eine topologische Schleife kann offen oder geschlossen sein. Eine offene Schleife hat eine enge Beziehung zu Kreuzungsstrukturen. Während jedoch zwei sich überlagernde unabhängige Teilstrukturen sich i.Allg. vorwärts überkreuzen, überkreuzt eine offene Schleife sich selbst immer rückwärts. Eine offene Schleife hat wie eine Kreuzung eine Chiralität. In Abbildung 5 sind eine links- und eine rechtsgewendelte offene Schleife dargestellt, zusammen mit ihrer AA-Struktur und ihrem AA-Ausdruck. Das unterfüh-

rende Verbindungsband wird wie bisher schon durch eine gepunktete Linie dargestellt. Seine geknickte Form dient lediglich dem Zweck, die Unterführung deutlicher zu zeigen.

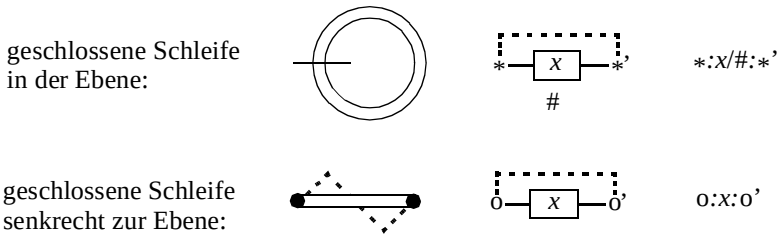


Abbildung 6. Zwei Sichten einer geschlossenen Schleife

Geschlossene Schleifen können sowohl mittels *Exit/Entry*-Paaren als auch *Down/Up*-Paaren beschrieben werden. Im ersten Fall liegt die geschlossene Schleife in der Beobachtungsebene, im zweiten Fall senkrecht dazu. Beide Fälle sind in Abbildung 6 dargestellt. Bei der AA-Beschreibung im ersten Fall ist allerdings eine Besonderheit zu beachten. Der einfache AA-Ausdruck *∗:x:∗'* zeigt nicht, ob der Ring in der Beobachtungsebene links herum oder rechts herum zu schliessen ist, d.h. keinen Drehsinn. Dies Manko lässt sich aber durch Einführung eines *Gap* beheben, da dadurch der planare Abstand zwischen dem *Entry* und dem *Exit* unterschiedlich lang wird. Der zweite Fall hat diese Besonderheit nicht.

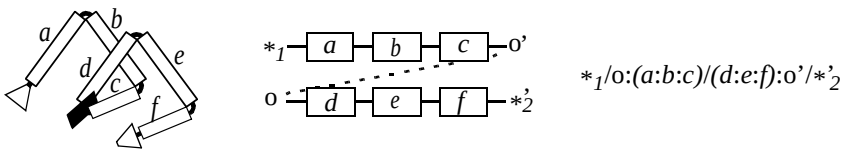
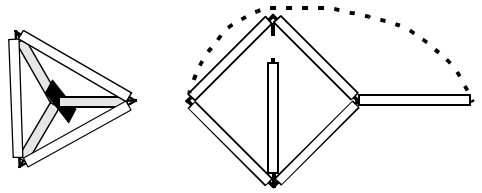


Abbildung 7. Räumliche Struktur, AA-Darstellung und -Beschreibung einer Helix

Wir demonstrieren im Folgenden die Beschreibung offener und geschlossener dreidimensionaler Strukturen durch je ein Beispiel. Das erste Beispiel ist eine Helix, d.h. eine Struktur, die von der Natur sowohl in DNA als auch in Proteinen verwendet wird. Die in Abbildung 7 gezeigte Helix ist recht primitiv, enthält aber sonst keine Beschränkungen. Sie besteht aus zwei Schleifen mit je drei Aktonen, *a, b, c* in der ersten Schleife und *d, e, f* in der zweiten. Für die Beschreibung werden die beiden Schleifen durch Einführung eines *Down/Up*-Paares formal voneinander getrennt. In Abbildung 7 links ist die Trennstelle durch eine kleine schwarze Fläche angedeutet. Diese Trennung erlaubt die Ausbreitung der Helix in der Ebene, wie in der AA-Darstellung in der Mitte gezeigt. Diese wiederum wird umkehrbar eindeutig durch den AA-Ausdruck rechts beschrieben. Hervorzuheben ist, dass die Aktonenbezeichnungen lediglich zur Typunterscheidung dienen. Grundsätzlich ist jedes Akton bereits eindeutig durch seine relative räumliche Position spezifiziert. Im Beispiel wurden die Aktonbezeichnungen lediglich eingeführt, um die Beziehungen zwischen der physikalischen Struktur, der

AA-Struktur und dem AA-Ausdruck zu verdeutlichen.



o:*(e:e/∗)/(e:(e:∗')/e):e:*o'

Abbildung 8. Räumliche Struktur, planare Darstellung und AA-Ausdruck eines Tetraeders

Als Beispiel für eine dreidimensionale geschlossene Struktur soll ein Tetraeder behandelt werden, wie in Abbildung 8 dargestellt. Dazu sei angenommen, dass er gleichlange Kanten hat, die deshalb einheitlich die Typbezeichnug *e* erhalten sollen. Der erste Schritt zur AA-Beschreibung ist die Ausbreitung des Tetraeders auf einer Ebene. Dies lässt sich durch Auftrennung des Tetraeders und Einsetzung eine *Down/Up*-Paars auf seiner Rückseite erreichen, z.B. an der Stelle, die in Abbildung 8 links durch eine kleine schwarze Fläche gekennzeichnet ist. Die sich ergebende planare Struktur enthält noch nicht verschachtelbar. Dies wird erst durch einen weiteren Schnitt und Einsetzung eine *Exit/Entry*-Paars erreicht. Das Ergebnis der beiden Schnitte ist die in der Mitte von Abbildung 8 gezeigte planare Darstellung, die umkehrbar eindeutig durch den AA-Ausdruck links beschrieben wird. Hervorzuheben ist, dass der AA-Ausdruck sogar festlegt, dass bei einer Rekonstruktion des Tetraeders *Down* und *Up* nach hinten zusammengeführt werden. Dies beinhaltet, dass AA auch beschreibt, was an einer Struktur innen und aussen ist.

5 Datenverarbeitung in Proteinen

Im vorigen Abschnitt wurde gezeigt, wie sich unter Verwendung von *Down/Up*- und *Exit/Entry*-Paaren verschiedene räumliche Strukturen beschreiben lassen, darunter die Helix, die einerseits die Basisstruktur von DNA und RNA ist, aber auch in Proteinen eine besondere Rolle spielt. DNA und RNA bestehen aus Doppelhelices, die zwischen sich die Erbinformation tragen. Diese Doppelhelices können ohne Schwierigkeiten mit der AA beschrieben werden. Während aber DNA und RNA nach heutigem Kenntnisstand nur zur Speicherung von Erbinformation dienen und damit passiv sind, führen Proteine Funktionen aus, die eine Datenverarbeitung beinhalten. Wie diese Datenverarbeitung geschieht, ist heute noch weitgehend unklar. Sicher ist nur, dass sie in Form chemisch/physikalischer Effekte in und zwischen Molekülen erfolgt und damit weitaus komplexer ist als die auf elektrischen Effekten beruhende in Computern. Nichtsdestoweniger gibt es keinen Grund anzunehmen, dass die biologische Datenverarbeitung in anderer Weise geschieht als die klassische.

Eine vollständige Beschreibung von Proteinen muss nicht nur die Strukturen, sondern auch die Datenverarbeitung und -speicherung abdecken. AA bietet auch diese Eigenschaften, wie in diesem Abschnitt gezeigt werden soll.

Generell lassen sich in der Datenverarbeitung zwei gerichtete Strukturen unterscheiden, lineare (kombinatorische) Netze und Rückkopplungsnetze, die in der Schal-

tungstechnik als Schaltnetze und Schaltwerke bezeichnet werden. Schaltnetze können mit den Mitteln der Schaltalgebra stückweise durch Auflistung Boolescher Gleichungen beschrieben werden, für Schaltwerke gibt es bisher keine formale Sprache. Mit der AA dagegen können sowohl lineare als auch Rückkopplungsnetze vollständig beschrieben werden. Dies soll an zwei Beispielen demonstriert werden.

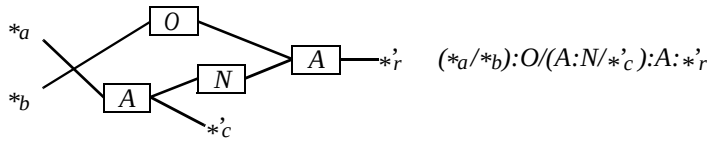


Abbildung 9. AA-Struktur und -Ausdruck eines Halbaddierers

Als Beispiel für ein lineares Netz wählen wir einen Halbaddierer. Seine AA-Struktur und sein AA-Ausdruck sind in Abbildung 9 gezeigt. Die Entries $*_a/*_b$ liefern den Input, der *Exit* $*'_c$ den Übertrag und *Exit* $*'_r$ das Ergebnis. Die Buchstaben *A*, *O*, *N* bezeichnen ein *And*-Gatter, ein *Or*-Gatter und einen Inverter. Die gezeigte AA-Struktur ist topologisch identisch mit der physikalischen Struktur. Die Funktionen und die Struktur des Halbaddierers werden durch den AA-Ausdruck rechts in der Abbildung präzise beschrieben.

Zum Vergleich sei die schaltalgebraische Beschreibung erwähnt. Sie erfordert zwei Boolesche Gleichungen $a \wedge b = c$ und $(a \vee b) \wedge \neg(a \wedge b) = r$ und sagt zudem nichts über die Schaltungsstruktur aus.

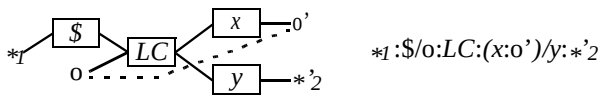
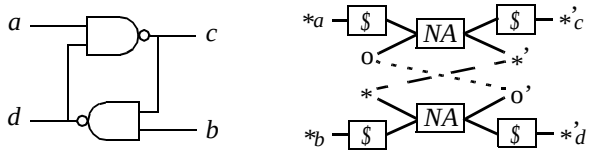


Abbildung 10. Eine (linksgewendelte) Rückkopplungsschleife, kontrolliert durch das Akton *LC*

Als zweites Beispiel betrachten wir eine allgemeine (linksgewendelte) Rückkopplungsschleife, wie in Abbildung 10 gezeigt. Eine Rückkopplungsschleife ist eine aktive offene Schleife, deren Input und Output gemeinsam kontrolliert bzw. verarbeitet wird. In der Abbildung ist das Kontrollakton mit *LC* bezeichnet. Ein primitives *LC* besteht nur aus eine *And*- oder *Nand*-Gatter und erzeugt damit entweder eine positive oder negative Rückkopplung. Erstere führt zu einem einzigen stabilen Zustand, letztere zur Oszillation. Eine positive Rückkopplung allein hat keine sinnvolle Anwendung, die Kombination von zwei verschachtelten positiven Rückkopplungen in Form von zwei *Nand*-Gattern dagegen hat zwei stabile Zustände und ist damit als Speicher geeignet, der als RS-Flipflop bezeichnet wird .

Abbildung 11 zeigt ein solches RS-Flipflop. Die AA-Struktur rechts gleicht in ihrer planaren Topologie der üblichen Schaltbilddarstellung, in der die beiden Eingänge links und die beiden Ausgänge rechts angeordnet sind. Bei räumlicher Interpretation erzwingt jedoch die in Abschnitt 4 eingeführte Attraktionssemantik der *Up/Down*- und *Entry/Exit*-Paare ein Umklappen der unteren Teilstruktur gegenüber der oberen, wie links in der Abbildung gezeigt. Dies ist aber nur eine Struktur unter vielen anderen, die

ebenfalls präzise beschrieben werden können.



$$(*_a{:}\$/\mathrm{o}{:}NA{:}\$/*'{:}*'_c)/(*_b{:}*/\${:}NA{:}\mathrm{o}'/\${:}*'_d)$$

Abbildung 11. Räumliche Struktur, AA-Struktur und -Ausdruck eines RS-Flipflops

Literatur

1. Jones, G., Sheeran, M.: Circuit Design in Ruby. In: Staunstrup, J. (ed.): Formal Methods of VLSI Design. North Holland (1990) 13-70
2. Kolla, R., Molitor, P., Osthof, H.G.: Einführung in den VLSI-Entwurf. Teubner-Verlag, Stuttgart (1989)
3. Rivas, E., Addy, S.R.: The language of RNA: a formal grammar that includes pseudoknots. Bioinformatics, Vol. 16, No. 4, , (2000) 334-340
4. Searls, D.B.: Formal Language Theory and Bioloical Macromolecules. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Vol. 47 (1999)