Strukturbasierte Beschreibung und Suche von RNA-Familien in genomischen Sequenzdaten

I n a u g u r a l - D i s s e r t a t i o n
zur
Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf
vorgelegt von

Stefan Andreas Gräf
aus Essen

Düsseldorf
2005

Gedruckt mit der Genehmigung der Mathematisch-Naturwissenschaftlichen Fakultät der Heinrich-Heine-Universität Düsseldorf

Referent:	apl. Prof. Dr. G. Steger
Korreferent:	Univ.-Prof. Dr. R. Wagner

Tag der mündlichen Prüfung: 1. Februar 2005

“This string is the root data structure of an organism’s biology.“

Olsen (1995)

Meiner Familie

Danksagung

Herrn apl. Prof. Dr. Gerhard Steger gebührt mein besonderer Dank. Seine Unterstützung, die zahlreichen Anregungen und vielen fruchtbaren Diskussionen sowie die mir gewährte wissenschaftliche Freiheit haben maßgeblich zum Gelingen dieser Arbeit beigetragen.

Herrn Prof. Dr. Rolf Wagner möchte ich für seine spontane Bereitschaft danken, diese Arbeit als Korreferent zu beurteilen.

Herrn Prof. Dr. Detlev Riesner danke ich für die Gelegenheit, diese Arbeit im Institut für Physikalische Biologie anzufertigen. Seine stete Diskussionsbereitschaft gab wichtige Impulse.

Des weiteren bedanke ich mich bei Herrn Prof. Dr. Stefan Kurtz für die Zusammenarbeit und die vielen interessanten Exkursionen in die `(Bio)?informatik`.

Bei Christian Hammann bedanke ich mich ganz herzlich für die fantastische Kooperation, durch die diese Arbeit nicht unwesentlich an Gehalt gewonnen hat.

Ein großes Dankeschön geht an Michael Schmitz für die intensiven und unzähligen Diskussionen, besonders die zum Thema Statistik, die vielen Anregungen und seine stetige Hilfsbereitschaft *in vivo* und *in silico*.

Bedanken möchte ich mich auch ganz besonders bei Axel Schmitz, der mir immer als guter Freund mit Rat und Tat zur Seite stand und stets ein aufmunterndes Wort parat hatte.

Der Rechner-Gruppe, allen voran Andreas Wilm, danke ich für die ausgezeichnete Atmosphäre bei der alltäglichen Arbeit. Dem Korrektur-Trupp sei Dank.

Allen übrigen Mitgliedern und Ex-Mitgliedern des Instituts danke ich für das immer gute und freundschaftliche Arbeitsklima. Diese Zeit werde ich immer in bester Erinnerung behalten.

Danken möchte ich in besonderer Weise meiner Mutter, meiner Schwester Anja und meinen Schwiegereltern Maria und Klaus Stahnke für alles. Sie waren mir in vielerlei Hinsicht eine große Hilfe.

Meiner lieben Frau Katja danke ich von Herzen für ihre Liebe und die unendliche Unterstützung, die sie mir ganz besonders in der Endphase dieser Arbeit hat zuteil werden lassen. Sie und unser Sonnenschein Levin haben auf ganz besondere Weise zum Gelingen dieser Arbeit beigetragen.

Abkürzungsverzeichnis

A	Adenin
Abb.	Abbildung
Abschn.	Abschnitt
Abk.	Abkürzung
Bp	Basenpaar(e)
C	Cytosin
CDS	Kodierende DNA Sequenz
cDNA	komplementäre DNA
Chr.	Chromosom
DNA	Desoxyribonukleinsäure
dsRNA	doppelsträngige RNA
G	Guanin
GHz	Gigahertz
HHRz	Hammerhead-Ribozym
inkl.	inklusive
kB	Kilobasen
kBp	Kilobasenpaare
kcal	Kilokalorie
kDa	Kilo-Dalton
MB	Megabyte
miRNA	microRNA
Mögl.	Möglichkeit(en)
mRNA	*messenger RNA*
ncRNA	*non-coding RNA*
ORF	offener Leserahmen
o. a.	oben angegeben
RAM	Schreib-Lese-Speicher (*random access memory*)
rasiRNA	*repeat-associated siRNA*
RNA	Ribonukleinsäure
RNAi	RNA-Interferenz
RNAse	Ribonuklease
rRNA	ribosomale RNA
RNP	Ribonukleo-Protein-Partikel
siRNA	*short-interfering RNA*
snoRNA	*small nucleolar RNA*
snRNA	*small nuclear RNA*
sog.	sogenannt
ssRNA	einzelsträngige RNA
T	Thymidin
Tab.	Tabelle
Temp.	Temperatur
tRNA	transfer RNA
U	Uracil
UTR	Untranslatierte Region
v. a.	vor allem
vs.	versus
z. T.	zum Teil
z. Z.	zur Zeit

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1

Einleitung

Das von Francis Crick 1958 aufgestellte zentrale Dogma der Molekularbiologie besagt, dass die genetische Information, die in der Desoxyribonukleinsäure (DNA) gespeichert ist, mit Hilfe von Ribonukleinsäure (RNA) als Informationsüberträger in Proteine übersetzt wird (Crick, 1958, 1970). Die Funktion von DNA und RNA sei also einzig und alleine auf die Speicherung und Übertragung der genetischen Information beschränkt, während die Proteine die chemischen Prozesse des Lebens in der Zelle katalysieren. Damals waren drei Klassen von RNAs (messenger-RNA (mRNA), transfer-RNA (tRNA) und ribosomale RNA (rRNA)) bekannt, die alle an der Protein-Biosynthese beteiligt sind. Zu Beginn der 80er Jahre wurden dann die ersten RNAs entdeckt, die katalytische Eigenschaften besitzen (Guerrier-Takada *et al.*, 1983; Kruger *et al.*, 1982), wofür Sidney Altman und Thomas R. Cech 1989 den Nobelpreis in Chemie erhielten. Seither wurden verschiedene weitere Klassen von funktionellen, nicht-Protein-kodierenden RNAs (ncRNAs) gefunden. Diese können strukturelle oder katalytische Funktion besitzen. Andererseits werden RNAs seit einigen Jahren auch regulatorische Aufgaben zugeschrieben. Nach dem heutigen Kenntnisstand wird der Fluss der genetischen Information in der Zelle mit dem zentralen Dogma der Molekularbiologie nur unzureichend beschrieben. Ein Beispiel dafür sind die genomisch kodierten microRNAs (miRNAs), die entwicklungsabhängig die Expression bestimmter Gene auf post-transkriptioneller Ebene regulieren (Übersichtsartikel: He & Hannon, 2004). Mattick & Gagen (2001) postulieren deshalb ein Netzwerk zur Regulation der Genaktivität, in dem RNAs mit DNA, mRNA und Proteinen interagieren. Das daraus resultierende Potential an Komplexität könnte die Unterschiede zwischen einfachen und komplexeren Organismen erklären, die sich in der Anzahl an Protein-kodierenden Genen z. T. nur wenig unterscheiden. Beispielweise besitzt der Mensch mit wahrscheinlich 20000 bis 25000 Genen (Human Genome Sequencing Consortium, 2004) nur etwa 5000 bis 10000 mehr als Nematoden der Spezies *Caenorhabditis elegans* (*Caenorhabditis elegans* Sequencing Consortium, 1998) oder die Fruchtfliege *Drosophila melanogaster* (Adams *et al.*, 2000), welche selbst nur etwa zweimal soviele Protein-kodierende Gene besitzen wie die Hefe *Saccharomyces cerevisiae*. Diese Zahlen konnten durch die Analyse dieser sequenzierten Genome

mit Programmen, wie GENSCAN (Burge & Karlin, 1997), Genie (Reese *et al.*, 2000) oder GeneWise (Birney *et al.*, 2004c; Birney & Durbin, 2000), zur Vorhersage von Protein-kodierenden Genen abgeschätzt werden. In diesem Zusammenhang wird es in Zukunft von Interesse sein, beliebige RNA-Familien in genomischen Sequenzdaten zuverlässig und effizient zu finden und zu annotieren. Die o. a. Werkzeuge können allerdings nicht für die Suche nach ncRNA-Genen eingesetzt werden. Ein möglicher Ansatz zur Vorhersage von ncRNA-Genen sollte in der hier vorliegenden Arbeit entwickelt werden. Das zu implementierende Bioinformatik-Werkzeug HyPa soll die deklarative Beschreibung, effiziente Suche und benutzerfreundliche Annotation v. a. von RNA, aber auch von DNA und Proteinen erlauben.

1.1 Die große Welt der RNA

RNA Moleküle erfüllen in der Zelle verschiedene Aufgaben von zentraler Bedeutung: Als Primer sind sie in die Replikation von DNA involviert. Sie transportieren die genetische Information in Form von mRNAs zu den Ribosomen, die aus rRNAs und Proteinen aufgebaut sind. An den Ribosomen erfolgt die Translation der mRNAs in Proteine mit Hilfe von tRNAs als Adapter-Molekülen. Im Ribosom selbst wird die Knüpfung der Peptidbindung durch die ribosomale 23 S RNA katalysiert und auch die Wechselwirkung des Ribosoms mit der mRNA und den tRNAs wird durch rRNA vermittelt (Moore & Steitz, 2002; Nissen *et al.*, 2000; Ogle *et al.*, 2001; Yusupov *et al.*, 2001; Yusupova *et al.*, 2001).

Katalytisch aktive RNAs werden im Allgemeinen als Ribozyme bezeichnet, die durch ihre dreidimensionale Struktur positionsspezifisch chemische Reaktionen in *cis* oder *trans* durchführen können. Das chemische Repertoire von Ribozymen reicht von der Umesterung über die Hydrolyse bis hin zur Knüpfung von Peptidbindungen. Das erste Ribozym wurde von Kruger *et al.* (1982) in einer sich selbst spaltenden prä-rRNA von *Tetrahymena thermophila* entdeckt (Gruppe I Intron). In den darauffolgenden Jahren konnte gezeigt werden, dass auch die RNA-Komponente der Ribonuklease P Primär-Transkripte von tRNAs, 5 S rRNAs und SRP-RNA (7 S RNA) an definierten Stellen hydrolysiert. Eine weitere Klasse von Ribozymen stellen die selbst-spaltenden Gruppe II Introns dar, die in Bakterien-, niederen Pflanzen- und Organellen-Genomen gefunden wurden und als mobile genetische Elemente fungieren. Desweiteren sind Ribozyme an der Replikation viraler Genome beteiligt (Übersichtsartikel: Doudna & Cech, 2002).

Die meisten Protein-kodierenden Gene in Eukaryoten enthalten nicht-kodierende Sequenzbereiche (Introns). Diese müssen vor der Translation aus den primären Transkripten, den prä-mRNAs, herausgeschnitten werden. Dieser Prozess wird als Splicing bezeichnet und von kleinen, Uracil-reichen, im Kern lokalisierten RNAs (*small nuclear RNAs*, snRNAs) katalysiert. Diese kommen mit Proteinen assoziiert in großen Ribonukleo-Protein-Partikeln (RNP), den Spliceosomen vor (Collins & Guthrie, 2000). Die snRNAs, rRNAs und andere ncRNAs werden selbst durch im Kern lokalisierte *small nucleolar RNAs* (snoRNAs) post-transkriptionell prozessiert oder modifiziert (Brown *et al.*, 2003; Kiss, 2001). Eine solche Modifikation, die auch als RNA-Editierung bezeichnet wird, kann die Insertion

oder Deletion von Nukleotiden oder die Substitution funktioneller Gruppen einer Base durch chemische Modifikation bedeuten. In untranslatierten Regionen (UTRs) von mRNAs können zudem Strukturmotive lokalisiert sein, die post-transkriptionell die Regulation der Expression dieser mRNAs beeinflussen. Beispielsweise erfolgt die Regulation der Expression von Proteinen des Eisenstoffwechsels über kleine Haarnadel-Strukturen, sog. *Iron Responsive Elements* (IRE), die in der 5'-UTR entsprechender mRNAs lokalisiert sind (Hentze & Kuhn, 1996). Histon-mRNAs besitzen in der 3'-UTR eine Haarnadel-Struktur, an die ein 45 kDa Protein (*stem loop binding protein*, SLBP) bindet, das sowohl an der Prozessierung der prä-mRNAs, als auch an der Translation der mRNAs beteiligt ist (Whitfield *et al.*, 2004; Williams & Marzluff, 1995).

Mit der Entdeckung von Fire *et al.* (1998), dass durch die Injektion von doppelsträngiger RNA (dsRNA) in *Caenorhabditis elegans* die Genexpression manipuliert werden kann, ist ein weiteres Feld mit besonders weitreichenden Folgen in der RNA-Forschung eröffnet worden. Die hier zugrundeliegenden Mechanismen werden allgemein unter dem Begriff RNA-Interferenz (RNAi) zusammengefasst. Man versteht darunter die durch endo- oder exogene dsRNA induzierte, post-transkriptionelle Suppression oder Stilllegung der Genexpression. Dabei werden die dsRNAs von dem Protein Dicer oder Dicer-ähnlichen Enzymen, die Homologien zur RNAse III aufweisen, in 21–28 bp lange Stücke prozessiert. Je nach Herkunft oder Funktion werden diese Duplexe als *short interfering* RNAs (siRNAs), microRNAs (miRNAs) oder *repeat-associated siRNAs* (rasiRNAs) bezeichnet. Diese kurzen Doppelstränge werden dann entwunden und mit Proteinen zu speziellen RNPs assembliert, von denen man bislang drei verschiedene kennt:

- den RISC-Komplex (*RNA-induced silencing complex*), der homologe, d. h. zur siRNA komplementäre mRNAs post-transkriptionell degradiert,
- die miRNPs (microRNA RNPs), welche die Translation homologer Ziel-mRNAs reprimieren, und
- den RITS (*RNA-induced transcriptional silencing*) Komplex, der die Modifikation von Chromatin vermittelt.

Bei RNAi handelt es sich um einen konservierten Mechanismus, der in vielen, v. a. höheren Organismen von großer Bedeutung ist. Eine bemerkenswerte Ausnahme stellt die Hefe *Saccharomyces cerevisiae* dar, in der bislang keine entsprechenden Mechanismen nachgewiesen werden konnten. Zudem haben die Erkenntnisse über die zugrundeliegenden Mechanismen RNAi zu einem wichtigen molekularbiologischen Werkzeug zur Erforschung von Genfunktionen gemacht (Übersichtsartikel zum Thema RNAi: Ambros, 2004; Baulcombe, 2004; Hannon & Rossi, 2004; Lippman & Martienssen, 2004; Meister & Tuschl, 2004; Mello & Conte, 2004).

Eine weitere Klasse an regulatorischen RNAs sind die natürlichen antisense-Transkripte (NAT). Auch diese scheinen in Verbindung mit der Regulation der Genexpression in Eukaryoten zu stehen. NATs sind RNAs, die Sequenzbereiche enthalten, die komplementär zu anderen endogenen RNAs sind. Man unterscheidet *cis*-NATs, die auf dem entgegengesetzten (antisense) Strang zu der komplementären RNA am gleichen Genlokus kodiert sind, von *trans*-NATs, die an einer anderen Stelle im Genom kodiert sind (Übersichtsartikel: Lavorgna *et al.*, 2004).

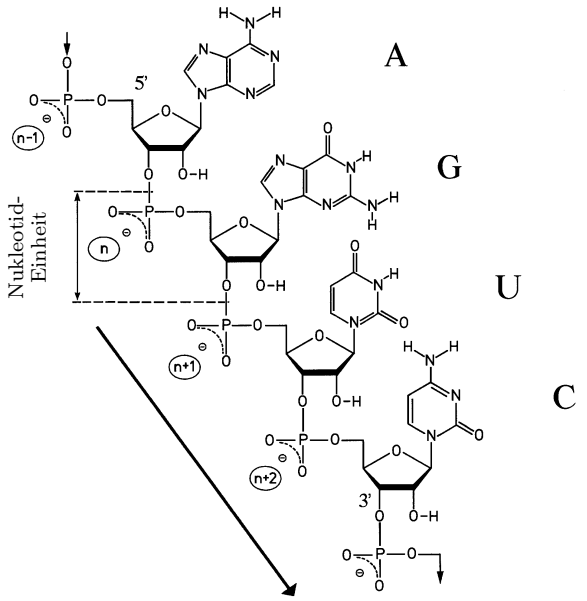


Abbildung 1.1: Bausteine der RNA. Eine RNA-Sequenz setzt sich aus Nukleotiden zusammen, die jeweils aus einer Base (Adenin (A), Guanin (G), Cytosin (C) oder Uracil (U)), einer Ribose und einer Phosphatgruppe bestehen. Dargestellt ist der Aufbau einer RNA mit der Sequenz AGUC in 5'-3'-Richtung, die sich durch die Nummerierung der Kohlenstoffatome in der Ribose ergibt und durch die Pfeile symbolisiert wird. Nach Steger (2003).

Die Fülle an zentralen Aufgaben, die von den verschiedenen Klassen an RNAs in der lebenden Zelle übernommen werden, legt die Vermutung nahe, dass eine „RNA-Welt“ existiert hat, aus der das Leben auf der Erde entstanden ist. In dieser Phase könnte RNA sowohl die Speicherung der genetischen Information als auch die katalytischen Funktionen übernommen haben, die für ein primitives selbst-replizierendes System notwendig sind. Eine solche Phase könnte in der Entwicklungsgeschichte der Erde vor etwa 4 Milliarden Jahren durchlaufen worden sein (Joyce, 2002).

1.2 Primär-, Sekundär- und Tertiär-Struktur von RNA

RNA-Moleküle bestehen aus Strängen an alternierenden Ribose- und Phosphat-Molekülen. Jede Ribose trägt am 1'-Kohlenstoffatom eine von den vier heterocyclischen, Stickstoff-enthaltenden Basen Adenin (A), Guanin (G), Cytosin (C) oder Uracil (U). Die Verbindung aus Base und Ribose wird als Nukleosid bezeichnet (Adenosin, Cytidin, Guanosin und Uridin). Die Nukleoside werden durch Esterbindung jeweils über eine Phosphatgruppe miteinander verknüpft (siehe Abbildung 1.1). Dadurch bleibt bei pH 7 pro Nukleotid (Nukleosid plus Phosphat) eine negative Ladung am Phosphat bestehen. Die Abfolge dieser Bausteine definiert die Primär-Struktur der RNA. RNA unterscheidet sich von DNA nur durch die 2' OH-Gruppe (Ribose vs. Desoxyribose) und durch den Austausch der Base U in RNA durch die Base T in DNA, die sich nur durch die zusätzliche 5-Methylgruppe im Thymin unterscheiden.

Als Sekundär-Struktur wird die zweidimensionale Repräsentation von RNA bezeichnet, die sich, ähnlich wie in der Doppelhelix von DNA (Watson & Crick, 1953), durch Wechselwirkungen zwischen den Basen ergibt. Einerseits können die Basen durch Wasserstoffbrücken miteinander paaren, wobei der daraus resultierende Energiegewinn relativ gering ist, da ebenso gut Wasserstoffbrücken mit dem umgebenden Wasser gemacht wer-

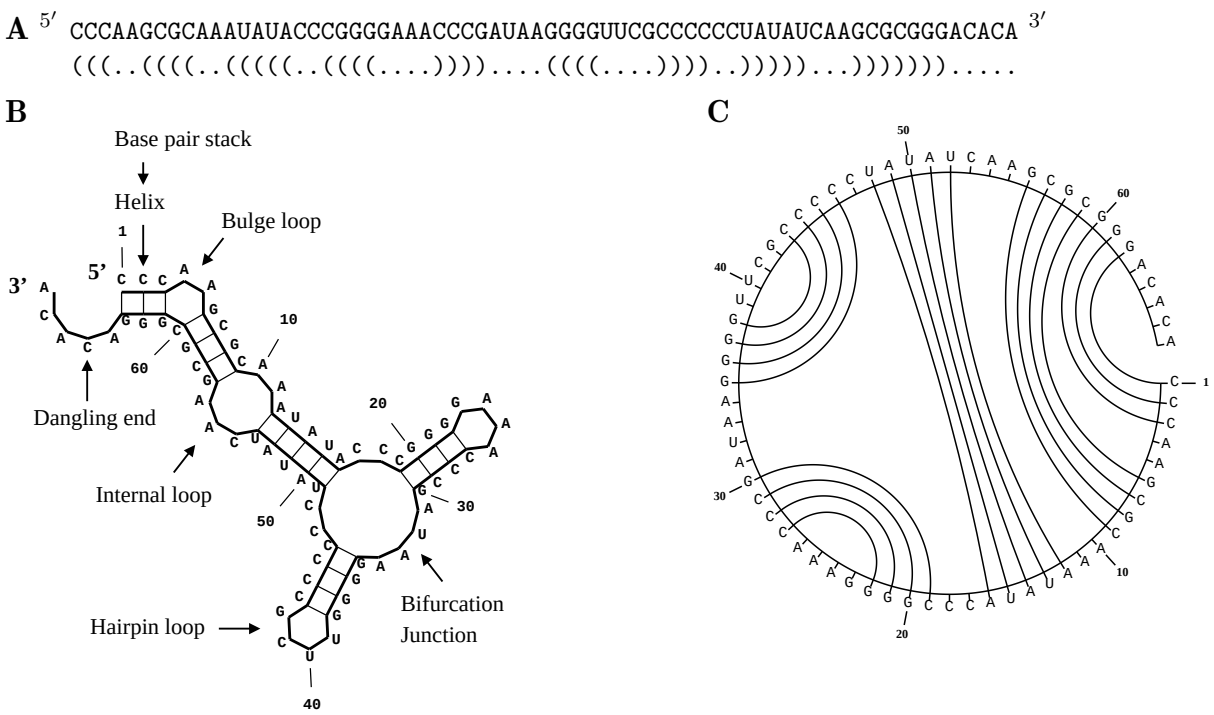


Abbildung 1.2: Graphische Repräsentation von RNA-Sekundärstruktur. Abbgebildet sind drei mögliche Repräsentationformen von RNA-Sekundärstruktur. Die verwendete Sequenz ist oben angegebenen. **A**: Punkt-Klammer-Notation, wie sie insbesondere von RNAfold (Hofacker *et al.*, 1994) benutzt wird; ein Punkt symbolisiert ein ungepaartes Nukleotid; eine öffnende bzw. schließende Klammer stellt eine Base dar, die mit einer 3'- bzw. 5'-lokalisierten Base ein Paar bildet. Mit drei verschiedenen Symbolen ist nur die Darstellung von Sekundärstrukturen möglich. **B**: Eine Darstellung, in der die Loops als gleichwinklige Polygone gezeichnet sind (Hofacker *et al.*, 1994; Lück *et al.*, 1999). Die möglichen Strukturmotive sind durch die Pfeile gekennzeichnet. **C**: "circles plot" (GCG, 2004; Nussinov *et al.*, 1978); nicht beschränkt auf Sekundärstrukturen; praktisch zum Vergleich verschiedener Strukturen einer Sequenz. Nach Steger (2003).

---

den können. Andererseits können die Basen entlang des Phosphat-Rückgrats aufeinander stapeln (*stacking*). Dabei handelt es sich um Dipol-induzierte Dipol-Wechselwirkungen zwischen den aromatischen Ringsystemen der Basen, welche den Hauptanteil des Energiegewinns durch Faltung in eine Sekundärstruktur ausmachen. Basenpaare können standardmäßig zwischen G und C oder A und U (Watson-Crick-Basenpaare) und in RNA auch zwischen G und U (Wobble-Basenpaare) ausgebildet werden. Daneben kommen in der Natur noch weitere nicht-kanonische Basenpaare, die von Leontis & Westhof (2001) klassifiziert und mit einer Nomenklatur versehen wurden.

Durch die Ausbildung von Basenpaaren zwischen einzelsträngigen, entgegengesetzt orientierten Strängen entstehen doppelsträngige Helices (vgl. Abbildung 1.2). Erfolgen die Basenpaarungen intramolekular zwischen zwei Bereichen einer einzelsträngigen RNA, erhält man als einfachste Sekundärstruktur eine Haarnadel-Loop-Struktur (Helix plus *Hairpin Loop*). Dabei muss die einzelsträngige Loopsequenz lang genug sein, um den Helix-Durchmesser zu überbrücken. Sind Helices durch Insertion einzelner oder mehrerer Nukleotide auf einer Seite unterbrochen, dann spricht man von *Bulge-Loops*. Verbinden zwei einzelsträngige Bereiche zwei Helices miteinander, entstehen interne Loops. Werden drei oder

mehr Helices durch einzelsträngige Bereiche verbunden, erhält man sog. Verzweigungsloops. Grundsätzlich ist die Ausbildung von Loops energetisch ungünstiger als die Ausbildung von Helices. Sekundärstruktur von RNA lässt sich auf verschiedene Weise graphisch darstellen. Drei Möglichkeiten sind in Abbildung 1.2 gezeigt. Mit Hilfe von Programmen, wie RNAfold (Hofacker, 2003; Hofacker *et al.*, 1994) und Mfold (Zuker, 2003; Zuker & Stiegler, 1981), kann die Sekundärstruktur einer RNA vorhergesagt werden. Diese Programme berechnen für eine gegebene Sequenz die Struktur mit der optimalen (minimalen) freien Energie (mfe), ΔG^0, oder die thermodynamische Strukturverteilung (McCaskill, 1990) durch Energieminimierung. Die Berechnungen erfolgen mittels dynamischer Programmierung unter Verwendung von experimentell ermittelten oder extrapolierten Energieparametern für verschiedene Looptypen und Basenstapel.

Von Tertiärstruktur spricht man, wenn zusätzliche Wechselwirkungen zwischen Sekundärstrukturelementen ausgebildet werden. Diese tertiären Wechselwirkungen können zwischen Basen, Ribosen und/oder Phosphat stattfinden und sind für die dreidimensionale Struktur und letztendlich für die biologische Funktion von RNA relevant.

Struktur-Motive

Generell sind Loops nicht einfach als ungepaarte Bereiche zu betrachten, sondern es existieren sehr oft Wechselwirkungen zwischen Basen, Ribosen und/oder Phosphat, die aber nicht vom Typ der normalen Watson-Crick-Wechselwirkungen sind. Beispielsweise ist ein in der Natur weit verbreitetes Strukturmotiv die Tetraloop-Hairpin-Struktur, wie sie z. B. in 16 S und 23 S rRNA, in Gruppe I- und Gruppe II-Introns oder in der RNAse P RNA vorkommen. Durch die vier zur Verfügung stehenden Nukleotide gibt es 4^4 verschiedene Möglichkeiten an Loopsequenzen, die sich alle in ihrer thermodynamischen Stabilität unterscheiden, wobei das den Loop flankierende Basenpaar ebenfalls Einfluss auf die Stabilität besitzt. Es konnte gezeigt werden, dass Tetraloops mit den Loopsequenzen UNCG und GNRA energetisch am günstigsten sind (N: A, C, G oder U; R: A oder G; Molinaro & Tinoco, 1995; Tuerk *et al.*, 1988). Der GNRA-Tetraloop kommt in der Natur am häufigsten vor (Woese *et al.*, 1990). Strukturelle Analysen einiger dieser Tetraloops mittels NMR zeigten ein Netzwerk von Wasserstoff-Brücken und Stapel-Wechselwirkungen zwischen den Loopnukleotiden, das diese kompakten Loop-Strukturen stabilisiert (Heus & Pardi, 1991; Jucker *et al.*, 1996).

Ein in der Natur ebenfalls häufig vorkommendes und außerordentlich intensiv untersuchtes Strukturmotiv ist das Hammerhead-Ribozym (HHRz). Es katalysiert beispielsweise die Replikation einer bestimmten Klasse von Pflanzen-pathogenen Viroiden (Flores *et al.*, 1999). Die Sekundärstruktur besteht aus einem Verzweigungsloop aus konservierten Nukleotiden, von dem drei Helices abzweigen, die z. T. durch Loops abgeschlossen werden. HHRz werden auch als biochemische Werkzeuge synthetisch hergestellt und eingesetzt. Diese benötigen im Vergleich zu den natürlich vorkommenden Ribozymen allerdings meistens eine Mg^{2+}-Ionen-Konzentration im mM-Bereich (O'Rear *et al.*, 2001). Für die katalytische Aktivität unter physiologischen Mg^{2+}-Ionen-Konzentrationen werden tertiäre Wechselwirkungen in peripheren Loop-Bereichen diskutiert (De la Peña *et al.*,

2003; Khvorova *et al.*, 2003). Allgemein sind solche Wechselwirkungen für die Ausbildung von Tertiärstruktur-Motiven verantwortlich und in den meisten Fällen für die biologische Aktivität der RNAs von Relevanz. Dabei bilden ungepaarte Loop-Nukleotide eines Sekundärstruktur-Elements zusätzliche Basenpaare mit Loop-Nukleotiden eines anderen Sekundärstruktur-Elements in *cis* oder *trans* aus. Tertiäre Wechselwirkungen führen z. B. in tRNAs dazu, dass die Kleeblatt-förmige Sekundärstruktur in der dritten Dimension eine L-Form zeigt (Kim *et al.*, 1974).

1.3 Suche und Annotation von ncRNAs

Die Suche und Annotation von Genen ist im Zeitalter der Genomsequenzierungen von großer Bedeutung. Für die Suche nach Protein-kodierenden Genen haben sich bereits eine Reihe von Bioinformatik-Werkzeugen bewährt, die mittlerweile standardmäßig zur Annotation sequenzierter Genome eingesetzt werden (Übersichtsartikel: Birney *et al.*, 2004b; Zhang, 2002). Dazu benutzen die zugrunde liegenden Algorithmen die wohl bekannten Strukturen Protein-kodierender Gene (Splicestellen, ORF, UTRs, Promotoren), sowie Ansätze zur vergleichenden Genomanalyse über Homologien zu bereits bekannten Proteinen.

Deutlich komplizierter stellt sich die Situation für ncRNAs dar, da die Struktur von ncRNA-Genen weitaus weniger eindeutig definiert ist. Beispielsweise besitzen ncRNAs in der Regel keinen ORF, über den sie identifiziert werden könnten. Ein möglicher Ansatz könnte demnach darin bestehen, Initiations-, Terminations- und Prozessierungsstellen von RNA-Transkripten vorherzusagen, um dann aus den so gefundenen potentiellen Kandidaten die herauszufiltern, die keinen ORF besitzen. Dieser Ansatz hat allerdings zwei Probleme. Einerseits ist die akkurate Vorhersage dieser Genstrukturelemente nach wie vor ein ungelöstes Problem in der Bioinformatik. Andererseits werden eukaryotische ncRNA-Gene von verschiedenen Polymerasen transkribiert. Während beispielsweise rRNAs von der RNA-Polymerase (Pol) I transkribiert werden, erfolgt die Transkription von 5 S RNA, tRNAs, 7 SL-RNA oder U6-snRNA durch Pol III. Andere ncRNAs werden von Pol II synthetisiert. Dazu kommt, dass z. B. snoRNAs in den meisten Fällen in Introns von Protein-kodierenden Genen enthalten sind und somit nicht unabhängig exprimiert werden (Brown *et al.*, 2003). Dennoch konnte gezeigt werden, dass die Identifikation von Promotoren und Terminatoren für die Annotation von ncRNAs in Prokaryoten durchaus eingesetzt werden kann (Argaman *et al.*, 2001; Wassarman *et al.*, 2001).

Eine weitere Möglichkeit zur Identifikation von ncRNAs besteht in der statistischen Analyse der Basen-Komposition von Sequenzen. Carter *et al.* (2001) haben einen Algorithmus entwickelt, der Ähnlichkeiten aus bekannten Gruppen von RNAs extrahiert und die so gewonnenen Informationen für die Suche von ncRNA-Genen in Prokaryoten und Archaebakterien einsetzt. Allerdings zeigen ncRNAs in den meisten Organismen keine speziellen Basenzusammensetzungen, sodass der Erfolg dieser Methode stark von dem untersuchten Organismus abhängt. In thermophilen Organismen beispielsweise korreliert der GC-Gehalt

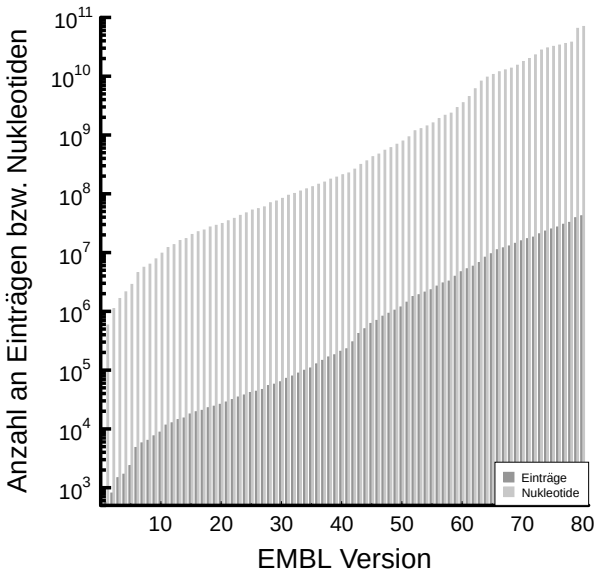


Abbildung 1.3: Wachstum der EMBL-Sequenzdatenbank seit der ersten, im Juni 1982 veröffentlichten Version. Angegeben sind jeweils die Anzahl an Einträgen (dunkelgrau) bzw. Nukleotiden (hellgrau) pro Version der EMBL-Datenbank (Kulikova *et al.*, 2004). Man beachte die logarithmische Skalierung der y-Achse. Der Zuwachs an Sequenzdaten ist in den 22 Jahren (bis Version 80, September 2004) exponentiell verlaufen. Detailliertere Statistiken sind auf der EMBL-Webseite unter `www3.ebi.ac.uk/Services/DBStats` zu finden.

von hoch strukturierten ncRNAs mit der Umgebungstemperatur aufgrund der benötigten thermodynamischen Stabilität (Galtier & Lobry, 1997).

Da ncRNAs nur zu einem relativ geringen Ausmaß in der Sequenz konserviert sind, scheiden Werkzeuge, die ausschließlich auf Methoden wie Blast und FASTA basieren für die Identifikation in Sequenzdaten aus, weil der Anteil an konservierten Nukleotiden in der Regel nicht signifikant genug ist. Im Gegensatz dazu sind ncRNAs aber v. a. in ihrer Sekundärstruktur konserviert, wodurch die Suche auch mit Hilfe von vergleichender Genomanalyse erfolgen kann. Diese Methode basiert auf dem paarweisen Vergleich homologer, strukturierter ncRNAs. Voraussetzung dafür ist, dass die zu vergleichenden Sequenzen ähnlich genug zueinander sind, um ein zuverlässiges Alignment zu erhalten. Andererseits sollten die Sequenzen wiederum so unterschiedlich sein, dass kompensatorische Basenaustausche enthalten sind, welche die Konservierung der Sekundärstruktur unterstützen. Mit Hilfe von statistischen Methoden können dann Schemata von Unterschieden festgestellt werden, die nicht zufällig vorkommen, sondern mit der konservierten Sekundärstruktur korrelieren (Lück *et al.*, 1999). Werden zudem weitere statistische Modelle, z. B. für RNA-Sekundärstruktur mit berücksichtigt, kann die Vorhersage auch ohne vorherige Kenntnis der Sekundärstruktur erfolgen (Rivas & Eddy, 2001). Allerdings werden strukturell wenig konservierte ncRNAs mit diesem Ansatz nicht gefunden.

Neben den allgemeinen Ansätzen zur Identifikation von ncRNAs wurden auch spezialisierte Bioinformatik-Werkzeuge entwickelt, die nur eine bestimmte Familie von homologen ncRNAs suchen können (Fichant & Burks, 1991; Kryukov *et al.*, 1999; Lowe & Eddy, 1997). Entsprechende Programme verwenden z. B. die aus Homologievergleichen und Experimenten gewonnen Eigenschaften einer speziellen RNA-Familie. Der Nachteil ist, dass diese Programme auch nur für die Suche einer einzigen Klasse von ncRNAs eingesetzt werden können. Im Gegensatz dazu stellen andere Werkzeuge Sprachen zur Beschreibung von Sequenz- und Struktur-Eigenschaften beliebiger Klassen von RNAs zur Verfügung. Mit Hilfe dieser Sprachen können Muster formuliert werden, die die Merkmale einer RNA-Familie beschreiben und von dem Programm mehr oder weniger effizient über Sequenzda-

ten gesucht werden können (ANREP (Mehldau & Myers, 1993), Palingol (Billoud B, 1996), PatScan (D'Souza *et al.*, 1997), PatSearch (Pesole *et al.*, 2000), RNAmotif (Macke *et al.*, 2001)). Allerdings unterscheiden sich diese Werkzeuge z. T. erheblich in Sprachumfang und Flexibilität. Bestimmte sequenzielle und/oder strukturelle Eigenschaften können nur umständlich oder gar nicht ausgedrückt werden. Keines der Suchwerkzeuge erlaubt die Angabe von zusätzlichen Bewertungsfunktionen (z. B. thermodynamische Stabilität oder Basenzusammensetzung), was besonders für die Beschreibung von ncRNAs von Nachteil ist. Die Suchwerkzeuge sollten jedes Vorkommen eines Musters in einer Sequenz finden, was in den meisten Fällen allerdings auch nicht der Fall ist. Zudem verlangt die immense Flut an Sequenzdaten (vgl. Abbildung 1.3) eine effiziente und schnelle Suche. Dagegen sind die meisten der Programme, wenn sie zur Verfügung stehen, sehr Rechenzeit-intensiv.

1.4 Aufgabenstellung

Das Ziel der hier vorliegenden Arbeit ist die Entwicklung eines Systems zur Struktur-basierten Beschreibung und Suche von RNA-Familien in genomischen Sequenzdaten. Die Entwicklung sollte in Zusammenarbeit mit der Arbeitsgruppe von Dr. Stefan Kurtz von der Universität Hamburg erfolgen. Der Schwerpunkt der im Rahmen dieser Arbeit zu bearbeitenden Teilgebiete des Projektes liegen in der Biologie und Bioinformatik, während die Kooperationspartner vorrangig informatische Fragestellungen bearbeiten sollten. Zunächst sollte eine Mustersprache entwickelt werden, die die flexible Beschreibung beliebiger Klassen von RNAs erlaubt. Dabei sollte der zu definierende Sprachumfang die folgenden Kriterien erfüllen bzw. Möglichkeiten umfassen:

- Beschreibung von Primär-, Sekundär- und Tertiärstrukturelementen,
- Definition von approximativen Ausdrücken, um homologe und/oder paraloge RNA-Sequenzen beschreiben zu können,
- Definition von positionsspezifischen Gewichtungsmatrizen, die sich aus Sequenz- und Struktur-Alignments ableiten lassen,
- Formulierung von zusätzlichen Bewertungsfunktionen für z. B. thermodynamische Stabilität, Längeneinschränkungen oder Basenzusammensetzung (hybride Muster),
- Modularer Aufbau der Sprache, um einzelne Sprachelemente frei kombinieren und wiederverwenden zu können.

Alle diese Kriterien sollen optimal hinsichtlich der Benutzbarkeit sein, damit der Benutzer intuitiv formulieren kann „Was“ er sucht und nicht „Wie“ gesucht werden soll. Mit dieser Beschreibungssprache sollen dann Modellmuster erstellt werden. Aus diesen und weiteren biologisch relevanten RNA-Mustern soll eine Bibliothek von RNA-Motiven aufgebaut werden, welche dann in Zukunft als Referenz dienen soll. Mit der Fertigstellung des von den Kooperationspartnern zu entwickelnden Suchwerkzeugs kann dann eine Validierung dieser Muster erfolgen. Zudem soll das Suchwerkzeug hinsichtlich Effizienz und Geschwindigkeit mit anderen verfügbaren Suchwerkzeugen verglichen werden.

Die meisten der Mustersuchwerkzeuge sind, wenn sie zur Verfügung stehen, nur als für die meisten Anwender benutzerunfreundliche Kommandozeilen-basierte Suchwerkzeuge

erhältlich. Um diese Hürde zu nehmen soll eine plattformübergeifende Benutzeroberfläche im Internet bereitgestellt werden. Hier sollen neben der Musterbibliothek Benutzerschnittstellen angeboten werden, über die mit hybriden Mustern in zur Verfügung gestellten oder eigenen Sequenzdaten gesucht werden kann. Gleichzeitig soll die Möglichkeit zur Analyse von Suchergebnissen gegeben werden.

Mit dem dann zur Verfügung stehenden Suchwerkzeug und den Möglichkeiten zur Auswertung von Suchergebnissen soll dann die Suche nach neuen, noch nicht annotierten RNAs erfolgen. Interessante Kandidaten sollten ggf. durch molekularbiologische Untersuchungen in Kooperation näher charakterisiert werden.

2

Sequenzdaten und Methoden

2.1 Sequenzdaten

Für die biologischen Anwendungsbeispiele werden standardmäßig die Sequenzdaten aus der EMBL-Datenbank verwendet (Version 78, Kulikova *et al.*, 2004). Hieraus wurden u. a. auch die durchsuchten Sequenzdaten von *Dictyostelium discoideum* extrahiert. Die komplette Datenbank wurde gespiegelt, um sämtliche durchgeführten Mustersuchen lokal ausführen zu können.

2.1.1 *Arabidopsis thaliana*

Die genomischen Sequenzdaten von *Arabidopsis thaliana* waren bei Genbank (Benson *et al.*, 2004) als vollständig annotierte Datensätze erhältlich. Die folgende Tabelle gibt die Größe der einzelnen Chromosomen (Acc.-Nr.: `NC_003070`, `NC_003071`, `NC_003074`, `NC_003075`, `NC_003076`) und des gesamten Genoms in bp sowie die Basenzusammensetzungen an.

Chr.	Version	Σ bp	A	C	G	T	N
I	19-FEB-2004	30432563	9711178	5436538	5422303	9698578	163966
II	25-FEB-2004	19705359	6317430	3544564	3522457	6318395	2513
III	19-FEB-2004	23470805	7487721	4261201	4265346	7451052	5485
IV	19-FEB-2004	18585042	5940445	3371497	3356108	5913959	3033
V	14-MAY-2003	26992728	8626390	4834740	4861257	8656518	13823
	Σ	119186497	38083164	21448540	21427471	38038502	188820

Nukleotidfrequenzen: $f_A = 0.320$, $f_C = 0.180$, $f_G = 0.180$, $f_T = 0.320$, $f_N = 0.002$.

2.1.2 Zufallssequenzen

Zu Testzwecken wurden mit Hilfe des Perl-Programms `gen_nucleic.pl` 10 Zufallsequenzen mit jeweils 10^6 Nukleotiden generiert. Dabei wurde die Gleichverteilung der Nukleotide vorausgesetzt. Die folgende Tabelle listet die jeweiligen Nukleotidfrequenzen (f_B, mit $B = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$) pro Sequenz auf:

Seq.-Nr.	f_A	f_C	f_G	f_T
1 MB Sequenzen (`1MB.fa.gz`)				
1	0.2500	0.2510	0.2496	0.2494
2	0.2506	0.2491	0.2497	0.2506
3	0.2493	0.2506	0.2496	0.2505
4	0.2506	0.2497	0.2503	0.2494
5	0.2494	0.2500	0.2494	0.2512
6	0.2507	0.2501	0.2494	0.2498
7	0.2498	0.2498	0.2497	0.2506
8	0.2505	0.2495	0.2502	0.2497
9	0.2497	0.2499	0.2500	0.2504
10	0.2505	0.2503	0.2505	0.2488

2.2 Entwicklungsumgebung

Die Entwicklung des Systems erfolgte unter Debian GNU/Linux (Version 3.1, testing) mit einem Linux-Betriebssystemkern (Version 2.4.25). Als Server stand ein Dualprozessorsystem (AMD Athlon(TM) MP 1900+) mit 2×1,6GHz und 2 GB RAM zur Verfügung.

2.3 Sprachen

2.3.1 Perl (*Practical Extraction and Report Language*)

Perl (Version 5.8) ist eine Skriptsprache, die sich besonders zur Verarbeitung von Zeichenketten und zur Programmierung von Webseiten eignet. Es existiert eine ganze Reihe von Modulen, die sich von Perl jederzeit in den eigenen Programm-Code einbinden lassen. Eine spezielle Erweiterung, die v. a. Module zur Lösung bioinformatischer Fragestellungen bereitstellt ist BioPerl (Stajich *et al.*, 2002; Stein *et al.*, 2002). Perl und BioPerl bilden die Grundlage für nahezu alle in dieser Arbeit entwickelten Werkzeuge.

2.4 Programmpakete (Software)

2.4.1 Apache-Webserver

Als Webserver wurde ein Apache-Webserver (Version 1.3) verwendet. Dieser wurde so konfiguriert, dass mit jedem Neustart die folgenden Module geladen werden und zur Verfügung stehen:

- mod_ssl (Version 2.8)/*OpenSSL* (Version 0.9) - *secure socket layer* zur Authentifizierung
- *mod_perl* (Version 1.29) - Integration von Perl
- HTML::Mason (Version 1.26) - erlaubt die modulare Programmierung dynamischer Webseiten durch Einbetten von Perl-Code in HTML (***H**yper **T**ext **M**arkup **L**anguage*); für detaillierte Informationen siehe `http://www.masonhq.com`.

2.4.2 PostgreSQL-Datenbank

Mit relationalen Datenbank-Systemen können große Datenmengen in Tabellenform strukturiert und nicht-redundant abgelegt werden. Zur Speicherung der Musterbibliothek, der Suchergebnisse und der zugehörigen Annotationen aus den Sequenzeinträgen wurde eine relationale PostgreSQL-Datenbank (Version 4.7) verwendet. Der Zugriff auf die Tabellenfelder erfolgt mit SQL (***S**tructured **Q**uery **L**anguage*). Um den persistentem Zugriff vom Apache-Server auf die Datenbank zu gewährleisten, wird das Perl-Modul Apache::DBI statisch in der HTML::Mason-Konfiguration geladen.

2.5 Bioinformatik-Werkzeuge

2.5.1 Sequenz-Alignment

Sequenz-Alignment-Methoden werden traditionell zum Auffinden von Homologien in Protein- oder DNA-Sequenzen benutzt. Dabei werden entweder zwei (paarweises Alignment) oder mehrere Sequenzen miteinander verglichen. Im Rahmen dieser Arbeit wurden v. a. die beiden folgenden Alignment-Programme verwendet.

BlastN (*Basic Local Alignment Search Tool*, Altschul *et al.*, 1990) benutzt einen heuristischen Ansatz zur Erstellung von lokalen, paarweisen Alignments und ist die Standardmethode zur Homologiesuche in großen Sequenzdatenbanken. Mit BlastN-Suchen (Alignment von Nukleinsäuresequenzen) wurden z. B. Homologe zu Treffersequenzen gesucht.

ClustAlX (Version 1.82 Jeanmougin *et al.*, 1998) wurde zur Erstellung von multiplen Sequenzalignments verwendet. Diese wurden z. B. bei der Mustererstellung mit dem Werkzeug ConStruct (siehe Abschnitt 2.5.3) benötigt oder für die Auswertung von Suchergebnissen zur Filterung redundanter Treffer eingesetzt.

2.5.2 Sekundärstruktur-Vorhersage

Die Programme RNAfold (Version 1.5, Hofacker, 2003; Hofacker *et al.*, 1994) und Mfold (Version 3.1, Zuker, 2003; Zuker & Stiegler, 1981) basieren auf dem Prinzip der Energieminimierung, mit dem Ziel die thermodynamisch optimale Sekundärstruktur einer gegebenen Nukleinsäuresequenz zu berechnen.

2.5.3 Sequenz-Struktur-Alignment (ConStruct)

ConStruct (Version 3.1) ist ein Werkzeug zur thermodynamisch kontrollierten Vorhersage konservierter Sekundärstruktur von RNA (Lück *et al.*, 1999, 1996). Es kombiniert die Methode der Sekundärstruktur-Vorhersage (RNAfold Hofacker *et al.*, 1994) mit der Methode des multiplen Sequenzalignments (z. B. ClustalX, Jeanmougin *et al.*, 1998) und visualisiert das Ergebnis in einem Basenpaar-Dotplot. Dieses Werkzeug ist für die hier vorliegende Arbeit von großer Bedeutung, da es eine zentrale Rolle bei der Erstellung von Mustern einnimmt. Die folgende Beschreibung fasst die Vorgehensweise bei der Benutzung des Werkzeugs kurz zusammen:

- Für ein Set homologer RNAs wird die thermodynamische Strukturverteilung mit RNAfold für eine sinnvolle Temperatur berechnet. Jede Strukturverteilung wird als Dotplot dargestellt. In diesen ist die Größe eines Punktes an der Position i, j proportional der Wahrscheinlichkeit für das Basenpaar $i{:}j$.
- Unabhängig von der Strukturberechnung wird für das Set homologer RNAs ein multiples Sequenzalignment mit z. B. ClustalX durchgeführt.
- Die Lücken des multiplen Alignments werden in die individuellen Dotplots mit den Strukturverteilungen insertiert. Jetzt haben alle Dotplots die identischen Dimensionen. Homologe Strukturelemente sollten jetzt an identischen Positionen in allen Dotplots zu liegen kommen.
- Summation aller Dotplots resultiert in einem „homologen Dotplot“. Strukturelemente, die allen Sequenzen gemeinsam sind, sollten betont werden, während Strukturelemente, die nur in wenigen Sequenzen möglich sind, unterdrückt werden.
- Im letzten Schritt wird die optimale Konsensus-Struktur aus dem „homologen Dotplot“ per dynamischer Programmierung extrahiert.
- Ein Problem am bisher beschriebenen Algorithmus ist auf das Sequenz-basierte multiple Alignment zurückzuführen. Beispielsweise kann in der Konsensus-Struktur ein thermodynamisch extra-stabiler Hairpin mit der Loopsequenz GNRA oder UNCG auftreten. Dieses Struktur-homologe Element sollte in allen Sequenzen zueinander aligniert werden. Da eine Sequenz-basierte Alignment-Methode darüber aber nichts wissen kann, tendiert diese dazu, an dieser Stelle Lücken einzuführen, um das Sequenz-Alignment zu optimieren, was aber das Struktur-Alignment verschlechtert.
  Zur Beseitigung solcher Alignment-Fehler, die sich per menschlicher Intelligenz meist recht einfach erkennen lassen, bzw. zur generellen Optimierung von Alignment und Struktur stellt das Programm eine grafische Oberfläche zur Verfügung, die eine synchrone Visualisierung von Struktur und Alignment erlaubt.

2.5.4 EMBOSS

Die *European Molecular Biology Open Software Suite* (EMBOSS, Version 2.6.0) ist ein freies *Open Source* Software Analyse Paket, das für die Benutzung im Bereich der Molekularbiologie entwickelt wurde (Rice *et al.*, 2000). Darin enthalten sind etwa 100 Programme die für diverse Fragestellungen eingesetzt werden können. Detaillierte Informationen sind auf der EMBOSS -Webseite zu finden.

3

Ergebnisse

Ziel dieser Arbeit ist es, ein Bioinformatik-Werkzeug zu entwickeln, mit dem es möglich ist, Nukleinsäure- und Protein-Sequenzen und Strukturen zu beschreiben, effizient in großen Sequenzdatensätzen zu suchen und die Ergebnisse benutzerfreundlich auswertbar zu machen. Dabei lag der Schwerpunkt dieser Arbeit in der struktur-basierten Beschreibung, Suche und Annotation von nicht-kodierenden Ribonukleinsäuren (RNA). Der Ergebnisteil gliedert sich in fünf Abschnitte. Einleitend wird die Vorgehensweise bei der Mustersuche dargestellt. Im zweiten Teil werden die entwickelte Musterbeschreibungssprache HyPaL (***Hy****brid* ***Pa****ttern* ***L****anguage*) und die im Rahmen dieser Arbeit häufig verwendete Sprache des Suchwerkzeuges PatScan beschrieben (siehe Abschnitt 3.2). Es folgt die Analyse der verwendeten Muster-Suchwerkzeuge HyPaSearch und PatScan hinsichtlich der zugrundeliegenden Algorithmen (siehe Abschnitt 3.3). In Abschnitt 3.4 werden die erstellte Muster-Datenbank und die zugehörigen, im Internet verfügbaren Benutzerschnittstellen zur Suche mit hybriden Mustern und zur Auswertung von Suchergebnissen vorgestellt. Im letzen Teil wird anhand von einigen biologischen Anwendungsbeispielen demonstriert, wie das hier entwickelte System zur struktur-basierten Beschreibung, Suche und Annotation von nicht-kodierenden RNAs eingesetzt werden kann (siehe Abschnitt 3.5).

3.1 Vorgehensweise bei der Mustersuche

Zur Beschreibung einer RNA-Familie benötigt man Informationen über charakteristische Eigenschaften dieser RNA. Diese Eigenschaften können z. B. aus der Literatur zusammengestellt werden. Alternativ besteht die Möglichkeit sämtliche in Datenbanken zur Verfügung stehende Sequenzen zu sammeln; mit Hilfe von Sequenz-Struktur-Alignments dieser Sequenzen können dann Konsensus-Sequenz- und Sekundärstruktur-Eigenschaften für diese Gruppe von RNAs abgeleitet werden. In beiden Fällen kann aus den so gewonnenen Informationen dann ein entsprechendes Muster mit Hilfe einer Beschreibungssprache

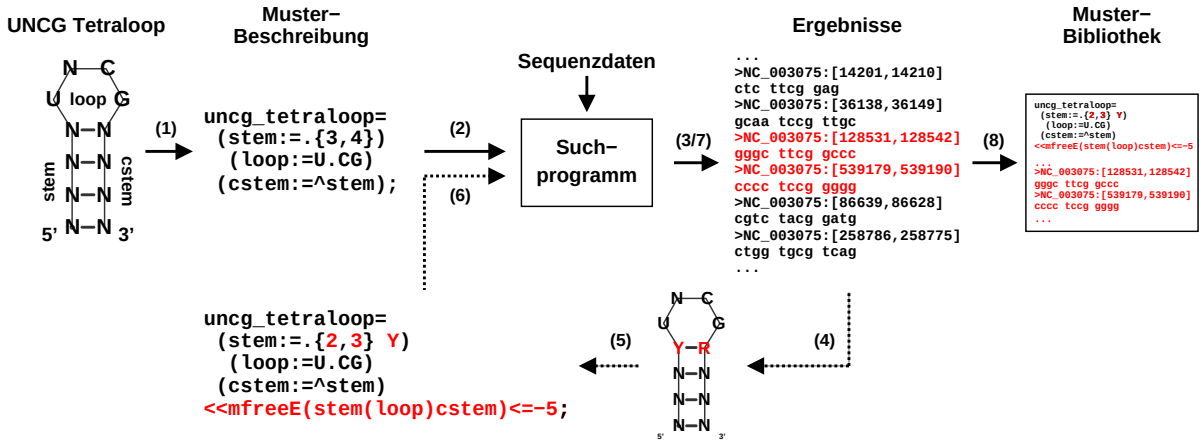


Abbildung 3.1: Schematische Darstellung der Vorgehensweise bei der Suche mit hybriden Mustern. Gesucht werden soll exemplarisch eine Tetraloop-Haarnadel-Struktur. Aus der Literatur sei bekannt, dass die gesuchte RNA-Struktur aus einer Helix von drei oder vier beliebigen Basenpaaren und einer Loopsequenz mit Konsensus UNCG besteht. (1) Diese Eigenschaften können mit Hilfe einer Beschreibungssprache (hier HyPaL) in einem Muster (`uncg_tetraloop`) zusammengefasst werden. (2) Das Suchprogramm (hier HyPaSearch) kann dieses Muster dann verarbeiten und in großen Sequenzdaten suchen. (3) Alle Teilsequenzen, die auf das Muster passen, werden zusammen mit dem Sequenzbezeichner und der Start- und Endposition des Treffers ausgegeben. (4) Angenommen bei der Auswertung der Ergebnisse fällt auf, dass die biologisch interessanten Sequenzen ein Pyrimidin-Purin-Basenpaar besitzen, das den Loop flankiert, und die minimale freie Energie der Gesamtstruktur kleiner oder gleich -5 kcal/mol ist. (5) Dementsprechend muss das Muster angepasst werden. Durch Einfügen der Basenpaar-Einschränkung und der Energie-Parameter erhält man ein spezifischeres, hybrides Muster. (6,7) Mit diesem Muster wird die Suche wiederholt. (8) Die durch die verfeinerte Beschreibung erhaltenen Ergebnisse können dann zusammen mit dem hybriden Muster und weiteren Informationen in einer Bibliothek gespeichert werden.

Tabelle 3.1: Suche des in Abbildung 3.1 verwendeten Musters (`uncg_tetraloop`) auf Chromosom IV von *Arabidopsis thaliana*. Aufgelistet sind die Anzahlen der Treffer vor und nach der Optimierung des Musters jeweils auf dem Vorwärtsstrang und dem reversen Strang.

	vor Optimierung	nach Optimierung
Vorwärtsstrang	11590	41
reverser Strang	11518	0

erstellt werden. Die in dieser Arbeit verwendeten Beschreibungssprachen werden in Abschnitt 3.2 vorgestellt. Ein solches Muster kann dann mit einem Suchwerkzeug, das die Beschreibungssprache verarbeiten kann, in Sequenzdaten gesucht werden. Zur Überprüfung des Musters wird in der Regel initial über die Sequenzen, aus denen die beschriebenen Eigenschaften extrahiert wurden, gesucht. Dabei sollten alle Sequenzen, die zur Erstellung des Musters verwendet wurden, auch tatsächlich gefunden werden. Ist das nicht der Fall, muss das Muster solange korrigiert und verfeinert werden, bis alle Ausgangssequenzen im Ergebnis enthalten sind. Anschließend können dann große Sequenzdaten durchsucht werden. Dabei kann es sich um sequenzierte Genome oder um große Sequenzdatenbanken, wie

EMBL oder Genbank handeln. Letztere enthalten allerdings redundante Sequenzeinträge, was möglicherweise die Auswertung der Ergebnisse stark erschwert. Da die Suche über große Datensätze neben richtig positiven Treffern auch falsch positive Treffer liefern kann, muss das Muster gegebenenfalls erneut optimiert und die Suche wiederholt werden. Ist eine solche Suche ausreichend spezifisch, können das Muster und die gefundenen Treffer zusammen mit weiteren Informationen, wie Literaturzitaten oder Sequenzannotationen, in der dafür entwickelten Datenbank gespeichert werden. Auf die Struktur und den Inhalt der Muster-Datenbank wird in Abschnitt 3.4.3 näher eingegangen. Abbildung 3.1 skizziert die Strategie bei der Mustersuche an einem kleinen Beispiel. Tabelle 3.1 fasst exemplarisch die Suche der beiden in Abbildung 3.1 verwendeten Muster auf Chromosom IV von *Arabidopsis thaliana* zusammen, um zu verdeutlichen, welchen Einfluss zusätzliche Einschränkungen auf das Ergebnis einer Suche haben können. Dabei zeigt die thermodynamische Einschränkung den größten Effekt, während die Basenpaar-Einschränkung alleine die Anzahl der Treffer nur um etwa die Hälfte reduziert (Ergebnis nicht gezeigt), was auf einen rein statistischen Effekt zurückzuführen ist.

3.2 Muster-Beschreibungssprachen und Struktur-basierte RNA-Muster

Die Konservierung von Sequenz- und Struktureigenschaften in RNAs kann zur Klassifizierung herangezogen werden. Eine Klasse von RNAs kann durch ein Muster beschrieben werden. Dafür benötigt man eine Beschreibungssprache, mit der Eigenschaften von RNAs dargestellt werden können. Ein solches Muster kann dann zur Suche in Sequenzdaten verwendet werden, wenn ein Suchwerkzeug zur Verfügung steht, das diese Beschreibungssprache einlesen und verarbeiten kann. Aus der Literatur sind einige Programme bekannt, die sowohl nur spezielle als auch beliebige Klassen von RNAs beschreiben und suchen können (siehe Abschnitt 1.3). Ziel dieser Arbeit war u. a. die Entwicklung eines Systems zur strukturbasierten Beschreibung und Suche nicht-kodierender RNA-Familien. Dazu wurde eine Beschreibungssprache entwickelt, mit der die Sequenz- und Struktureigenschaften von RNAs flexibel und präzise dargestellt werden können.

Bei der entwickelten Musterbeschreibungssprache HyPaL handelt es sich um eine deklarative Sprachen, d. h. der Benutzer formuliert „Was“ und nicht „Wie“ zu suchen ist. Um das zu erreichen, sollten die folgenden Anforderungen von einer Beschreibungssprache erfüllt werden:

- **Flexibilität:** Die Beschreibungssprache soll eine große Anzahl an Möglichkeiten bieten auf lesbare Art und Weise beliebige RNA-Motive zu beschreiben.
- **Approximative Elemente:** Durch die Angabe von nummerischen Toleranzen sollen ähnliche (homologe, paraloge) Sequenzen mit Deletionen, Insertionen oder Ersetzungen bzw. Fehlpaarungen beschreibbar sein.
- **Modularität:** Muster sollen in komplexeren Beschreibungen wiederverwendbar sein.
- **Bewertungsfunktionen und Schwellenwerte:** Der Benutzer soll zusätzliche Angaben machen können, nach denen gefundene Treffer einer zusätzlichen Überprüfung un-

terzogen werden. Dazu gehören beispielsweise thermodynamische Einschränkungen, die Teile der Sekundärstruktur erfüllen müssen, oder die Basenzusammensetzung, der eine gefundene Sequenz genügen muss. Die Angabe solcher Funktionen bzw. Schwellenwerte ist in HyPaL implementiert und stellt einen wesentlichen Unterschied zu anderen aus der Literatur bekannten Sprachen und Suchwerkzeugen dar.

Die beiden letzten Anforderungen an die Sprache werden von PatScan (D'Souza *et al.*, 1997) und anderen Suchwerkzeugen nicht erfüllt. Das Programm HyPaSearch, das die hier entwickelte Beschreibungssprache HyPaL verwendet, war bis zum Ende dieser Arbeit nur in Teilen implementiert, sodass fast ausschließlich PatScan für die Suche mit Strukturbasierten Mustern erfolgte. Deshalb werden im Folgenden die entwickelte Beschreibungssprache HyPaL und die Sprache, die von PatScan verwendet wird, im Detail vorgestellt und verglichen. Tabelle 3.3 gibt einen Überblick über die zur Verfügung stehenden Beschreibungselemente von HyPaL und PatScan. Alle nur in HyPaL verfügbaren Ausdrücke und Bewertungsfunktionen sind in Tabelle 3.4 aufgelistet.

3.2.1 Struktur der Musterbeschreibung

Die Musterbeschreibung wird vom Benutzer erstellt und in eine Datei geschrieben. Diese Datei kann dann vom Programm eingelesen und die Musterbeschreibung verarbeitet werden, um damit in Sequenzdaten zu suchen.

PatScan

Bei PatScan darf eine Datei immer nur ein Muster enthalten, da in der Beschreibungssprache die Vergabe von Musternamen nicht vorgesehen ist. Ein Muster setzt sich definitionsgemäß aus Teilmustern oder Mustereinheiten zusammen, die der Reihe nach abgearbeitet werden. Diese werden durch Leerzeichen oder Zeilenumbrüche getrennt hintereinander geschrieben. Die Treffersequenzen werden den Mustereinheiten entsprechend durch Leerzeichen getrennt ausgegeben (vgl. Abschnitt 3.3.2), was zur Lesbarkeit der Ergebnisse beiträgt.

HyPaL

Ein Muster in HyPaL hat die Form:

```
name=muster;
```

`name` ist der Mustername, dem die Beschreibung `muster` zugeordnet wird. Der Name muss mit einem Kleinbuchstaben beginnen. Jedes Muster endet mit einem Semikolon. Da jedem Muster ein eindeutiger Name zugewiesen wird und jedes Muster eindeutig durch ein Semikolon abgeschlossen wird, können so im Gegensatz zu PatScan in einer Datei mehrere Muster formuliert werden. Darüber hinaus kann ein Muster über den Namen in einem anderen Muster wiederverwendet/referenziert werden, wodurch die Modularität gewährleistet wird (siehe Abschnitt 3.2.12).

Die Musterbeschreibung kann aus beliebigen Ausdrücken bestehen, die im Folgenden noch detailliert erklärt werden. Dem Aufbau der Sprache entsprechend wird zwischen atomaren Sprach-Elementen, regulären Ausdrücken und hybriden Mustern unterschieden. Hybrides

Muster (Hybrid Pattern, kurz HyPa) meint die Gesamtheit des Musters bestehend aus der Beschreibung von Primärsequenz (siehe Abschnitt 3.2.3) und Sekundärstruktur-Elementen (siehe Abschnitt 3.2.7) kombiniert mit approximativen Schwellenwerten (siehe Abschnitt 3.2.8) oder anderen Bewertungsfunktionen (siehe Abschnitt 3.2.11). Reguläre Ausdrücke sind dagegen Teile des Musters, die atomare Sprach-Elemente z. B. logisch miteinander verknüpfen (siehe Abschnitt 3.2.4). Die atomaren Sprach-Elemente sind einzelne Zeichen (z. B. Nukleotide), die auf dem verwendeten Alphabet basieren.

3.2.2 Kommentare/Dokumentation

Durch die Verwendung von speziellen Zeichen kann der Benutzer Kommentare und Beschreibungen mit in die Musterbeschreibung einflechten, die zur Lesbarkeit des Musters beitragen und das Speichern zusätzlicher Informationen ermöglichen. Alle Zeichen, die hinter dem Kommentarzeichen in der aktuellen Zeile stehen, werden vom Programm beim Einlesen der Musterbeschreibung ignoriert.

PatScan
Als Kommentarzeichen wird das „`%`“-Zeichen benutzt. Alle in dieser Arbeit enthaltenen PatScan-Muster enthalten beispielsweise in der ersten Zeile immer den Namen des Musters, der für die weitere Verarbeitung benutzt wird:

```
% muster_name
```

HyPaL
In HyPaL beginnen Kommentare mit doppeltem „`/`“-Zeichen.

3.2.3 Primärsequenz

Die Beschreibung einzelner Nukleotide erfolgt sowohl in PatScan als auch in HyPaL mit Hilfe der in der IUPAC-Nomenklatur (Cornish-Bowden, 1985, vgl. Tabelle 3.2) definierten Buchstaben. Durch Aneinanderreihung der Buchstaben zu einer Zeichenkette kann eine definierte Konsensus-Sequenz beschrieben werden.

HyPaL
Die verwendeten Buchstaben müssen groß geschrieben werden, es sei denn die Zeichenkette wird mit Anführungszeichen quotiert. Alternativ zum Buchstaben `N` für ein beliebiges Nukleotid kann auch als Joker-Symbol „`.`“ verwendet werden. Die mehrdeutigen Buchstaben der IUPAC-Nomenklatur können ebenfalls als Zeichenklassen formuliert werden. Die an einer bestimmten Position erlaubten Buchstaben werden dann in eckigen Klammern zu einer Klasse zusammengefasst. Um das Komplement einer Zeichenklasse zu definieren, kann hinter der öffnenden eckigen Klammer der „`^`“-Operator eingefügt werden.

Die Beschreibung einer RNA-Sequenz bestehend aus einem Adenin, Cytosin oder Uracil, gefolgt von einem Uracil, einem Adenin oder Cytosin, einem Adenin und einem beliebigen Nukleotid kann durch die Zeichenkette `HUMAN` dargestellt werden. In HyPaL kann dieselbe

Tabelle 3.2: IUPAC-Mehrdeutigkeitscode. Nach Cornish-Bowden (1985).

Code	Nucleotide	
A	A	**A**denine
C	C	**C**ytosine
G	G	**G**uanine
U/T	U/T	**U**racil
M	A, C	a**M**ino
R	A, G	pu**R**in
W	A, U/T	**W**eak interaction
S	C, G	**S**trong interaction
Y	C, U/T	p**Y**rimidine
K	G, U/T	**K**etone
V	A, C, G	not U, **V** follows U
H	A, C, U/T	not G, **H** follows G
D	A, G, U/T	not C, **D** follows C
B	C, G, U/T	not A, **B** follows A
N	A, C, G, U/T	a**N**y

Konsensus-Sequenz auch durch `[ACU]U[AC]A[ACGU]` beschrieben werden. Die Zeichenkette `[^A]U[AC]A[ACGU]` würde den Konsensus BUMAN definieren.

3.2.4 Reguläre Ausdrücke

Reguläre Ausdrücke dienen der logischen Verknüpfung, Quantifizierung und Zusammenfassung von Teilmustern. In Anlehnung an reguläre Ausdrücke, die hauptsächlich in Hilfsprogrammen und Texteditoren unter UNIX/Linux Verwendung finden, können entsprechende Formulierungsmöglichkeiten auch in den Beschreibungssprachen verwendet werden.

- **Verkettung:** Mehrere Atome/Teilmuster hintereinander beschreiben Sequenzen, auf die die Atome/Teilmuster in der entsprechenden Reihenfolge passen.

  PatScan

  Die Verknüpfung von Teilmustern durch Leerzeichen hat einen Einfluss auf die Ausgabe und Lesbarkeit der Ergebnisse (vgl. Abschnitt 3.3.2).

  HyPaL

  Um eine entsprechend lesbare Ausgabe von HyPaSearch zu erhalten, müssen in HyPaL Variablen verwendet werden (siehe Abschnitt 3.2.5).

- **Gruppierung:** Einzelne Ausdrücke können durch runde Klammern gruppiert werden (nur in HyPaL möglich, s.u.).

- **Disjunktion (ODER-Verknüpfung):** Durch das Verbinden mehrerer Zeichenketten durch das „`|`“-Symbol können Alternativen beschrieben werden. Durch Gruppierung der Ausdrücke in runden Klammern kann bestimmt werden, welche Teile des Musters alternativ vorkommen dürfen.

  PatScan

  Eine ODER-Verknüpfung in PatScan hat die Syntax `(ausdruck1 | ausdruck2)`. Dabei müssen vor und hinter dem „`|`“-Symbol Leerzeichen stehen und der gesamte Ausdruck muss in runden Klammern gruppiert werden. Es können pro ODER-Verknüpfung nur zwei Alternativen angegeben werden. Mehrere Alternativen können nur durch Verschachtelung von ODER-Verknüpfungen formuliert werden (`((AG | GA) | CU)`).

  HyPaL

  In HyPaL können zwei oder mehr Alternativen gleichzeitig in Verbindung mit dem „`|`“-Symbol angegeben werden. Dabei ist es wichtig, dass die Zuordnung durch Gruppierung in runden Klammern eindeutig angegeben wird, wie man an den folgenden Beispielen sehen kann:

`AA C|G AA`	findet AAC oder GAA
`AA (C|G) AA`	findet AACAA oder AAGAA
`AA (GC|CG) AA`	findet AAGCAA oder AACGAA
`AA (CC)|(GG) AA`	findet AACC oder GGAA
`AA (C|G|U) AA`	findet AACAA oder AAGAA oder AAUAA
`AA (CC|G|UUU) AA`	findet AACCAA oder AAGAA oder AAUUUAA

- **Konjunktion (UND-Verknüpfung):** Durch die Verknüpfung von Teilausdrücken mit dem „`&`“-Symbol können Bedingungen formuliert werden, die gleichzeitig erfüllt sein müssen (nur in HyPaL möglich).

- **Quantifizierung:** Wiederholungen bestimmter Sequenzelemente können mit Hilfe von quantifizierenden Ausdrücken formuliert werden.

  PatScan

  Zur Beschreibung von repetitiven Sequenzelementen muss die Variablen-Notation (siehe Abschnitt 3.2.5) verwendet werden. Eine entsprechende Formulierung mit Hilfe von regulären Ausdrücken ist nicht implementiert. In Ausnahmefällen kann diese Art der Beschreibung in Verbindung mit ODER-Verknüpfungen umgesetzt werden. Beispielsweise beschreibt das Muster

  ```
  (GGG | (GGGG | GGGGG))
  ```

  eine Sequenz aus drei bis fünf aufeinanderfolgenden Gs.

  Für die Definition von Sequenzen mit beliebigen Nukleotiden definierter Länge verfügt PatScan über eine spezielle Notation: `m...n`, mit $0 \leq m \leq n$. Zwei positive ganze Zahlen m und n werden durch drei Punkte ohne Leerzeichen verbunden,

wobei m die minimale und n die maximale Anzahl an erlaubten Nukleotiden bestimmt. Diese Beschreibungsart kann sowohl für einzel- als auch doppelsträngige Bereiche verwendet werden (vgl. Abschnitt 3.2.7).

HyPaL

Die Anzahl an erlaubten Wiederholungen eines Sequenzelements kann in geschwungenen Klammern hinter diesem angegeben werden (Syntax: `ausdruck{n,m}`, mit $0 \leq n \leq m$). Dabei ist zu beachten, dass sich die Quantifizierung nur auf das vorherige Zeichen bezieht. Soll eine Zeichenkette mehrfach vorkommen, muss diese durch runde Klammern gruppiert werden. Beispielweise passt das Muster `(ACU){2,3}` auf die Sequenzen `ACUACU` und `ACUACUACU`. Es wurden folgende Variationen dieser Notation und Abkürzungen implementiert:

`ausdruck{n}`	exakt n Wiederholungen,
`ausdruck{n,}`	n und mehr Wiederholungen,
`ausdruck{,m}`	0 bis m Wiederholungen,
`ausdruck?`	Abk. für `ausdruck{0,1}`,
`ausdruck*`	Abk. für `ausdruck{0,}` und
`ausdruck+`	Abk. für `ausdruck{1,}`.

3.2.5 Variablen

Teilmuster können einer Variablen zugewiesen werden. An einer anderen Stelle im Muster kann dann diese Variable über ihren Namen referenziert und weiterverwendet werden. Variablen werden z. B. zur Beschreibung von repetitiven Sequenzen und Sekundärstruktur-Elementen benutzt.

PatScan

In eingeschränkter Form kann der Benutzer Variablen definieren, wobei der Variablenname nicht frei wählbar ist und einer Variable immer nur eine der zur Verfügung stehenden Beschreibungsarten zugewiesen werden kann. Die Definition von Variablen folgt der folgenden Syntax:

`p`i`=muster`

Der Name der Variablen besteht aus einem kleinen `p` gefolgt von einer ganzen Zahl i zwischen 0 und 50. Variablen in PatScan können zur Beschreibung einer definierten Anzahl repetitiver Elemente verwendet werden, wie z. B.

```
p1=RYRYRY p1 p1
```

Mit diesem Muster wird eine Sequenz beschrieben, die aus 18 Nukleotiden besteht. Dabei wird in der Variable `p1` eine Sechsersequenz bestehend aus alternierenden Purinen und Pyrimidinen gespeichert. Exakt diese Sequenz muss sich dann in der Suchsequenz noch zweimal wiederholen, damit das Muster passt.

HyPaL

Die Syntax für die Definition von Variablen in HyPaL ist:

```
(variable:=muster)
```

Der Variablenname ist `variable`, über den die Referenzierung erfolgt; `muster` steht für die Musterbeschreibung, die eine Sequenz beschreibt. Die gefundene Sequenz wird dann als „Wert“ der Variablen zugewiesen. Ein Variablen-Definition muss von runden Klammern eingeschlossen werden, damit die Zuordnung eindeutig ist. Im Gegensatz zu PatScan können verschiedene Beschreibungsarten bei der Variablen-Definition verwendet werden. Durch sinnvolle Wahl der Variablennamen wird gleichzeitig die Lesbarkeit des Musters gewährleistet. Beispielweise könnte der Deskriptor `(loop:=GNRA)` eine Loop-Sequenz mit dem Konsensus GNRA beschreiben.

Zusätzlich können mit Hilfe der Variablen die Muster in HyPaL unter Verwendung der `where`-Klausel (siehe Abschnitt 3.2.10) übersichtlich strukturiert werden. Hierbei müssen die runden Klammern ausgelassen werden.

3.2.6 Basenpaar-Regeln

Damit basengepaarte Sequenzabschnitte gesucht werden können, muss dem Programm mitgeteilt werden, welche Basenpaarungen erlaubt sind. Dafür können sog. Basenpaar-Regeln definiert werden.

PatScan
Standardmäßig sind in PatScan ausschließlich Watson-Crick-Basenpaare voreingestellt. Davon abweichende Basenpaar-Regeln können mit der folgenden Syntax definiert werden:

```
ri={xy,yx,vw,...}
```

Eine solche Regel erhält einen Namen (`r`i), wobei i eine ganze Zahl ist, über den die Regel weiter unten in der Beschreibung referenziert werden kann. Es folgt ein Gleichheitszeichen und in geschweiften Klammern eine durch Kommas getrennte Liste der in dieser Regel erlaubten Basenpaare. Es dürfen mehrere Regeln definiert werden, die sich dann durch die Nummerierung (i) des Bezeichners unterscheiden lassen. Die Regel

```
r1={au,ua,gc,cg,gu,ug}
```

erlaubt neben Watson-Crick- auch Wobble-Basenpaare, was den Standard-Basenpaarungen für RNA entspricht. Auf die Verwendung der Basenpaar-Regeln bei der Musterbeschreibung wird in Abschnitt 3.2.7 eingegangen.

HyPaL
Standardmäßig sind in HyPaL die Basenpaarungen für RNA (Watson-Crick- und Wobble-Basenpaare) voreingestellt. Zusätzliche, davon abweichende Basenpaar-Regeln können mit

Hilfe von Funktionen vom Benutzer definiert werden. Eine Basenpaar-Regel, die ausschließlich Wobble-Basenpaare erlaubt, würde wie folgt lauten:

```
wobble G := [U];
wobble U := [G];
```

`wobble` ist der Name der Funktion (Regel), über den diese weiter unten in einer Beschreibung aufgerufen wird. Hier werden der Funktion zwei Relationen zugewiesen („G paart mit U und umgekehrt“). Die Verwendung einer solchen Regel in der Musterbeschreibung wird in Abschnitt 3.2.7 erläutert.

3.2.7 Sekundärstruktur

Die Beschreibung von basengepaarten Bereichen erfolgt mit Hilfe der in Abschnitt 3.2.5 eingeführten Variablen. Unter Verwendung der voreingestellten oder Benutzer-definierten Basenpaar-Regeln wird das reverse Komplement zur gefundenen Sequenz, die an die entsprechende Variable gebunden wurde, bestimmt.

PatScan

Das folgende Beispiel beschreibt eine kleine Haarnadel-Struktur in PatScan-Notation:

```
p1=4...7 3...5 ~p1
```

Die erste Mustereinheit beschreibt einen Bereich von 4 bis 7 beliebigen Nukleotiden, der den Namen `p1` erhält. Danach werden 3 bis 5 beliebige Nukleotide im Loop verlangt. In der dritten Mustereinheit wird Bezug auf die erste Einheit genommen. Es wird das reverse Komplement zu der Nukleotidesequenz von `p1`, angedeutet durch die Tilde (~), gesucht. Da nicht explizit eine Basenpaar-Regel angeben ist, sind nur Watson-Crick-Basenpaare erlaubt. Dürfen zusätzlich auch Wobble-Basenpaare auftreten, würde man die in Abschnitt 3.2.6 angegebene Basenpaar-Regel `r1` verwenden und der Tilde den Namen der Regel voranstellen: `r1~p1`.

HyPaL

Steht vor einer Variable der Komplement-Operator (`^`), wird das reverse Komplement der zugewiesenen Sequenz nach Standard-RNA-Basenpaar-Regeln gesucht. Das Muster `(stem:=.{3,4}) GNRA ^stem` definiert eine extrastabile GNRA-Tetraloop-Haarnadel-Struktur. Verlangt werden drei bis vier beliebige Nukleotide, die in der Variable `stem` gespeichert werden. Die Loopsequenz soll den Konsensus GNRA besitzen, gefolgt von dem reversen Komplement (`^stem`) der in `stem` gespeicherten Sequenz. Soll dagegen etwa die in Abschnitt 3.2.6 definierte `wobble`-Funktion zur Ermittlung des reversen Komplements verwendet werden, ersetzt man den Standard-Komplement-Operator „`^`“ einfach durch den Funktionsnamen `wobble`: `... wobble(stem)`.

3.2.8 Approximative Elemente

Um nicht perfekte Treffer finden zu können, kann der Benutzer Ausnahmen in Form von Ersetzungen (replacement) in einzelsträngigen Bereichen oder Fehlpaarungen (mismatch)

A

```
% mismatch
r1={au,ua,gc,cg,gu,ug}
p1=4...4 GNRA r1~p1[1,0,0]

TGAT GTAA AGTA
((.( .... ).))
```

B

```
% deletion
r1={au,ua,gc,cg,gu,ug}
p1=4...4 GNRA r1~p1[0,1,0]

CATG GGAA CTG
((.( .... )))
```

C

```
% insertion
r1={au,ua,gc,cg,gu,ug}
p1=4...4 GNRA r1~p1[0,0,1]

GTCC GTAA GCGGT
((((  .... ).)))
```

Abbildung 3.2: Beschreibung nicht perfekter Treffer durch die Angabe von Approximationen mit PatScan. Dargestellt sind die drei Möglichkeiten (**A**) Fehlpaarungen, (**B**) Deletionen und (**C**) Insertionen in Mustern zu erlauben. Die Angabe erfolgt in eckigen Klammern hinter der entsprechenden Mustereinheit. Alle drei Muster beschreiben eine kleine GNRA-Tetraloop-Haarnadelstruktur, in der jeweils einmal eine entsprechende Ausnahme in dem basengepaarten Bereich `p1...~p1` auftreten darf. Unter jedem Muster ist ein Beispieltreffer angegeben, wobei jeweils in der letzten Zeile die Struktur durch Punkt-Klammer-Notation angedeutet wird.

---

in doppelsträngigen Bereichen, Deletionen und Insertionen formulieren. Die Anzahlen der erlaubten Ausnahmen werden in der folgenden Reihenfolge hinter dem entsprechenden Musterelement in eckigen Klammern durch Kommas getrennt angegeben:

```
[replacement/mismatch,deletion,insertion]
```

Abbildung 3.2 zeigt exemplarisch die drei Arten an möglichen Fehlern am Beispiel einer Tetraloop-Haarnadelstruktur. Nachfolgend werden die zur Verfügung stehenden Approximationen detailliert erklärt.

- **Ersetzungen/Fehlpaarungen**
  In der entsprechenden Treffersequenz dürfen im Bezug auf die Beschreibung Fehler auftreten. Beschreibt die vorhergehende Mustereinheit eine Primärsequenz, spricht man von Ersetzungen. In doppelsträngigen Bereichen dagegen dürften entsprechend viele Fehlpaarungen (nicht A:U, U:A, G:C, C:G, G:U oder U:G, wenn Standard-RNA-Basenpaar-Regeln gelten) vorkommen.
- **Deletion**
  Bei Deletionen dürfen in der Treffersequenz eine entsprechende Anzahl an Nukleotiden fehlen. In doppelsträngigen Bereichen erfolgt die Deletion in der revers-komplementären Sequenz. Das führt zur Reduzierung der Anzahl an verlangten Basenpaaren. Durch die Deletion von Nukleotiden im 3' Teil der Helix entstehen Bulge-Loop-Nukleotide im 5'-Teil.
- **Insertion**
  Die Anzahl an Insertionen gibt an, wieviele Nukleotide zusätzlich zu der Beschreibung in der Treffersequenz insertiert werden dürfen. In doppelsträngigen Bereichen bleibt dabei die Anzahl an geforderten Basenpaaren gleich, aber es entstehen durch das Einfügen von Nukleotiden im 3' Teil der Helix Bulge-Loop-Nukleotide im 3'-Teil.

PatScan

Der zugrunde liegende Algorithmus liefert nur einen, den möglichen Ausnahmen entsprechenden Treffer zurück. Passt eine Sequenz ohne Verwendung einer angegebenen Ausnahme, wird z. B. nur der exakte Treffer ausgegeben, auch wenn alternative Lösungen

existieren. Damit muss der gefundene Treffer nicht zwangsläufig der optimale bzw. vom Benutzer gewollte Treffer sein (vgl. Abschnitt 3.3.2).

HyPaL
Im Gegensatz zu PatScan liefert eine approximative Suche mit HyPaSearch alle möglichen Treffer, die auf eine Beschreibung mit Ausnahmen passen.

3.2.9 Gewichtungs-/Profil-Matrizen

Mit Hilfe von Profil-Matrizen lassen sich gewichtete Konsensus-Sequenzen konstanter Länge beschreiben. In der Regel können diese Matrizen aus Sequenz-Alignments abgeleitet werden, indem man in einem Ausschnitt des Alignments in jeder Spalte das Auftreten der einzelnen Nukleotide zählt. Die so gewonnenen Frequenzen der einzelnen Nukleotide können dann als 4-Tupel pro Spalte zusammengefasst werden. Durch Aneinanderreihung der 4-Tupel erhält man dann die Beschreibung einer gewichteten Konsensus-Sequenz konstanter Länge. Durch die Angabe eines Schwellenwertes wird dem Suchprogramm mitgeteilt, ab wann eine Sequenz auf eine solches Muster passt.

Die Syntax und Verwendung soll an einem kleinen Beispiel verdeutlicht werden. Angenommen der Benutzer möchte eine konservierte Sequenz aus vier Nukleotiden mit dem Konsensus GNRA suchen. Aus einem korrekten Sequenz-Alignment sei bekannt, dass an Position 2 des Konsensus zu je 30% C oder U und zu je 20% A oder G vorkommen, und dass an Position 3 zu 75% A und zu 25% G auftreten. Diese Informationen lassen sich nicht einfach mit Hilfe der IUPAC-Nomenklatur beschreiben, da so die Gewichtungen verloren gingen. Durch die Verwendung der Matrix-Schreibweise lassen sich diese Informationen allerdings ausdrücken. Die allgemeine Syntax (PatScan) ist

$$\{(f_{\mathrm{A}_1}, f_{\mathrm{C}_1}, f_{\mathrm{G}_1}, f_{\mathrm{U}_1}), (f_{\mathrm{A}_2}, f_{\mathrm{C}_2}, f_{\mathrm{G}_2}, f_{\mathrm{U}_2}), \ldots, (f_{\mathrm{A}_n}, f_{\mathrm{C}_n}, f_{\mathrm{G}_n}, f_{\mathrm{U}_n})\} > S,$$

wobei die Frequenzen der Nukleotide (f_{N_i}) pro Spalte im Alignment in runden Klammern durch Kommata getrennt aufgelistet werden und diese 4-Tupel wiederum durch Kommas getrennt in geschweiften Klammern zusammengefasst werden. Um zu überprüfen, ob eine gegebene Sequenz eine solche Profil-Matrix erfüllt, werden die jeweiligen Werte der Suchsequenz entsprechend aufsummiert. Ist die Summe größer als der angegebene Schwellenwert S, ist die Bedingung erfüllt und ein Treffer gefunden.

PatScan
Für das angegebene Beispiel mit dem Konsensus GNRA ergibt sich somit nach PatScan-Notation

```
{(0,0,100,0),(20,30,30,20),(75,0,25,0),(100,0,0,0)} > 244.
```

Als Vergleichs-Operator für die Angabe des Schwellenwertes ist nur die Echt-größer-Relation (>) erlaubt.

HyPaL
In HyPaL werden sowohl die 4-Tupel als auch der gesamte Ausdruck mit eckigen Klammern

gruppiert. Für das angegebene Beispiel mit dem Konsensus GNRA ergibt sich somit nach HyPaL-Notation

```
[[0,0,100,0],[20,30,30,20],[75,0,25,0],[100,0,0,0]] >= 245.
```

Als Vergleichs-Operatoren für die Angabe des Schwellenwertes sind hier `<`, `>`, `==`, `<=`, `>=` erlaubt.

3.2.10 `where`-Klausel

Ein weitere Möglichkeit die Muster übersichtlich zu strukturieren besteht mit der `where`-Klausel (nur HyPaL). Dabei wird das Muster in einer Art Vorspann mit Hilfe der Variablennamen „vordefiniert“. Nach dem Schlüsselwort „`where`“ folgt dann die eigentliche Variablenzuweisung. Das folgende Beispiel verdeutlicht die Verwendung der `where`-Klausel:

```
gnra_tetraloop = stem loop ^stem
where stem:=.{3,4}
      loop:=GNRA;
```

3.2.11 Bewertungsfunktionen

Hybride Muster zeichnen sich dadurch aus, dass neben Primärsequenz- und Sekundärstruktur-Eigenschaften weitere Merkmale wie z. B. Nukleotidgehalt oder thermodynamische Stabilität überprüft werden können. Solche Funktionen stehen in den bisherigen Beschreibungssprachen nicht zur Verfügung. Hierfür wurden in HyPaL Bewertungsfunktionen implementiert, die auf Sequenzen angewendet werden können, wenn diese an Variablen gebunden sind. Die Angabe dieser Funktionen erfolgt hinter dem „`<<`“-Operator entweder am Ende des Musters, wenn keine `where`-Klausel (vgl. Abschnitt 3.2.10) verwendet wird, oder zwischen der allgemeinen Musterbeschreibung und der `where`-Klausel. Die Funktionen geben definierte Werte zurück, die dann mit Vergleichsoperatoren in Relation zu einem gegebenen Schwellenwert gesetzt werden können. Gleichzeitig können die Funktionen mit nummerischen Operatoren verknüpft werden (z. B. `+` (Addition), `-` (Subtraktion), `*` (Multiplikation), `/` (Division)). Durch die Verknüpfung mit logischen Operatoren (`&&` (und), `||` (oder), `!` (nicht)) können mehrere Funktionsrelationen gleichzeitig verwendet werden, wobei diese nicht mit denen verwechselt werden dürfen, die in regulären Ausdrücken Verwendung finden. Sequenzen, die auf die Beschreibung passen und gleichzeitig die angegebenen Relationen erfüllen, werden als Treffer ausgegeben. Auf die implementierten Bewertungsfunktionen wird im Folgenden näher eingegangen.

Längeneinschränkungen

Die Funktion `length(var)` ermittelt die Länge der Sequenz, die an die Variable `var` gebundenen ist. Mit Hilfe dieser Funktion können Längeneinschränkungen festgelegt werden. In Verbindung mit einem Vergleichsoperator (`<`, `>`, `==`, `<=`, `>=`), gefolgt von einer positiven

ganzen Zahl, wird die Bedingung, die zu erfüllen ist, definiert. Angenommen der Benutzer möchte eine Haarnadel-Struktur beschreiben, die aus zwei Helices besteht, die durch einen Bulge-Loop mit Konsensus AAA verbunden sind, und im Hairpin-Loop drei bis vier beliebige Nukleotide enthält. Insgesamt soll die Haarnadel-Struktur mindestens acht Basenpaare besitzen. Das folgende Muster beschreibt eine solche Haarnadel-Struktur:

```
hbh_motif=(stem1:=.{3,5}) AAA (stem2:=.{3,5})
          (loop:=.{3,4})
          ^stem2 ^stem1
<< length(stem1)+length(stem2) >= 8;
```

Nukleotidgehalt

Mit den Funktionen `gccontent(var)`, `xcontent(b,var)` und `absxcontent(b,var)` können Bedingungen an den Nukleotidgehalt der Sequenz gestellt werden, die an die Variable `var` gebundenen ist. Während `gccontent` den relativen GC-Gehalt der Sequenz bestimmt, kann mit `xcontent` der relative Anteil der Base(n) `b` an der Sequenz ermittelt werden, die über die Variable `var` referenziert wird. Dabei kann `b` eine beliebige, auf dem verwendeten Alphabet basierende Kombination an Basen als Wert annehmen; `xcontent(AU,var)` beispielsweise würde den relativen AU-Gehalt der Sequenz bestimmen, die in `var` gespeichert ist, während `xcontent(G,var)` nur den relativen Anteil an Gs in der Sequenz ermittelt. Zur Berechnung des absoluten Anteils einer Nukleotid-Kombination `b` an einer Sequenz, kann die Funktion `absxcontent(b,var)` benutzt werden. Im folgenden Beispiel werden zwei der Funktionen auf verschiedene Teile des Musters `hbh_motif` unter Verwendung der Und-Verknüpfung (`&&`) angewendet.

```
hbh_motif=(stem1:=.{3,5}) AAA (stem2:=.{3,5})
          (loop:=.{3,4}) ^stem2 ^stem1
<< gccontent(stem1) > 0.5 && xcontent(U,stem2) > 0.4;
```

Thermodynamische Stabilität (freie Energie)

Zur Berechnung der thermodynamischen Stabilität von Sekundärstruktur-Elementen stehen die folgenden drei Funktionen zur Verfügung:

- `mfreeE(var_construct,T)`
  Berechnet wird bei der Temperatur `T` (in °C) die minimale freie Energie des Sequenzabschnittes, der durch `var_construct` definiert wird. Diese Funktion ist z. B. von Relevanz, wenn ein als Loop formulierter Musterbereich eine beliebige Struktur mit definierter Stabilität ausbilden soll.
- `freeE(var_construct,T)`
  Berechnet wird bei der Temperatur `T` (in °C) die freie Energie des Sekundärstruktur-Elements, das durch `var_construct` definiert wird. Die Berechnung erfolgt für eine Punkt-Klammer-Notation (vgl. Abbildung 1.2), die aus der Musterbeschreibung abgeleitet wird. Dafür muss die Angabe von `var_construct` die folgende Syntax haben:

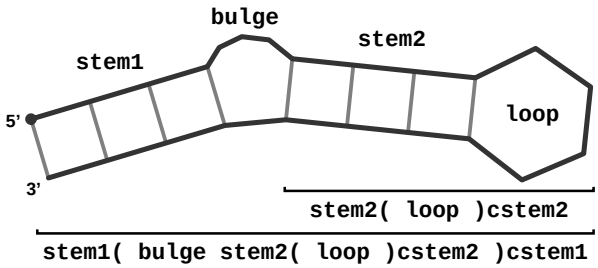


Abbildung 3.3: Sekundärstruktur, die durch das Muster `hbh_motif` beschrieben wird. Unter der Struktur angegeben sind die Variablen-Konstrukte, die im Muster zur Berechnung der freien Energien verwendet werden.

  - An Variablen gebundene basengepaarte Bereiche werden durch das Anhängen bzw. Voranstellen von runden Klammern an bzw. vor den Variablennamen definiert. Beispielsweise könnte der 5'-Teil eines basengepaarten Bereichs durch „`5prime(`“ und der 3'-Teil durch „`)3prime`“ definiert werden. Entsprechend der Längen der Sequenzen, die an die Variablen gebundenen sind, werden daraus die Anzahlen an öffnenden und schließenden Klammern für die Punkt-Klammer-Notation abgeleitet. Es ist darauf zu achten, dass der Ausdruck genausoviele öffnende wie schließende Klammern besitzt.
  - Einzelsträngige Bereiche werden einfach durch den entsprechenden Variablennamen definiert, woraus sich die Punkte für die Punkt-Klammer-Notation ergeben.
- `mfeDiff(var_construct,T)`
  Berechnet wird die Differenz zwischen freier Energie und minimaler freier Energie des Struktur-Elements, das durch `var_construct` definiert wird, bei der Temperatur `T` in °C.

Die Berechnung der Energiewerte erfolgt in kcal/mol. Wird die Angabe der Temperatur `T` beim Funktionsaufruf weggelassen, erfolgt die Berechnung der Energie standardmäßig für 37 °C. Exemplarisch wird die Verwendung der Funktion `freeE` am Beispiel des Musters `hbh_motif` demonstriert (vgl. Abbildung 3.3):

```
hbh_motif=(stem1:=.{3,5}) (bulge:=AAA) (stem2:=.{3,5})
          (loop:=.{3,4})
          (cstem2:=^stem2) (cstem1:=^stem1)
<< freeE(stem1( bulge stem2( loop )cstem2 )cstem1)
 - freeE(stem2( loop )cstem2) < -8;
```

Berechnet werden hier separat die Energien der gesamten Haarnadelstruktur (`stem1( bulge stem2( loop )cstem2 )cstem1`) und der Helix 2 inkl. des apikalen Loops (`stem2( loop )cstem2`). Durch Subtraktion der Energie von Helix 2 mit Loop von der der Gesamtstruktur wird der Energiewert für Helix 1 mit Bulge-Loop bestimmt. Das Ergebnis wird dann zur Definition der Bedingung verwendet; im Beispiel muss Helix 1 inkl. dem benachbarten Bulge-Loop einen günstigen Energiebeitrag liefern. Bei der Mustererstellung ist zu beachten, dass der revers komplementäre 3' Teil eines basengepaarten Bereiches einer Variable zugeordnet sein muss (vgl. `cstem1` und `cstem2` im Beispiel), damit die Funktion richtig aufgerufen werden kann.

3.2.12 Wiederverwendbarkeit von Mustern

Die Beschreibungssprache HyPaL wurde so angelegt, dass in einer Datei bereits definierte Muster in anderen wiederverwendet werden können. Die Referenzierung der Muster erfolgt über den Musternamen. Eine bestimmte Reihenfolge ist bei der Definition der Muster nicht einzuhalten. Das folgende Beispiel zeigt eine solche Formulierung.

```
loop=GNRA;
gnra_tetraloop = (stem:=.{3,4}) loop ^stem;
```

Zunächst wird die Konsensus-Sequenz GNRA durch das Muster `loop` definiert. Dieses Muster wird dann in der Beschreibung des `gnra_tetraloop`-Motivs über den Musternamen `loop` referenziert.

3.2.13 Muster mit Parametern

In HyPaL hat der Benutzer die Möglichkeit verallgemeinerte Muster zu formulieren. Die Definition des Musters erfolgt hierbei mit einem oder mehreren Platzhaltern, die in runden Klammern hinter dem Musternamen durch Kommas getrennt angegeben werden. Diese Platzhalter setzen sich definitionsgemäß aus einem %-Zeichen und dem Namen des Platzhalters (in Kleinbuchstaben) zusammen. In der Beschreibung können diese dann wie Variablen benutzt werden. Bei Verwendung eines solchen Musters können so flexibel andere Muster als Parameter übergeben werden. Ein Muster, das Parameter enthält, kann selber nicht gesucht werden. Diese Art der Formulierung trägt ebenfalls zur Modularität und Flexibilität von HyPaL bei. Im Folgenden wird am Beispiel einer Tetraloop-Haarnadel-Struktur die Definition und Verwendung eines solchen Motivs demonstriert.

```
hairpin(%loop)=(stem:=.{3,4}) %loop ^stem;
gnra=GNRA;
gnra_tetraloop=hairpin(gnra);
```

Das Muster `hairpin` beschreibt allgemein eine Haarnadel-Struktur mit drei bis vier beliebigen Nukleotiden in der Helix (`stem` und `^stem`) und einer mit dem Parameter `%loop` nicht weiter definierten Loop-Sequenz. `gnra` definiert die Konsensus-Sequenz GNRA. Diese beiden Muster werden dann benutzt, um die GNRA-Tetraloop-Haarnadel-Struktur (`gnra_tetraloop`) zu beschreiben, indem dem Muster `hairpin` das Muster `gnra` als Parameter übergeben wird.

3.2.14 Vergleich der Beschreibungssprachen

Ziel dieser Arbeit ist es eine Beschreibungssprache zu entwickeln, die modular aufgebaut ist und dem Benutzer die Möglichkeit bietet flexibel und in lesbarer Form Sequenz- und Sekundärstruktur-Eigenschaften von RNA in Kombination mit zusätzlichen Einschränkungen zu formulieren. Der Benutzer soll beschreiben „Was“ gesucht werden soll und nicht „Wie“ gesucht werden soll. In den vorangegangenen Abschnitten wurde die entwickelte Sprache HyPaL neben der von PatScan benutzten Sprache vorgestellt. In Tabelle 3.3

Tabelle 3.3: Überblick über HyPaL und die in PatScan verwendete Beschreibungssprache. Aufgelistet sind die grundlegenden Formulierungsmöglichkeiten, die in beiden Werkzeugen zur Verfügung stehen. n.i.: nicht implementiert.

HyPaL	PatScan	Bemerkungen
Allgemeines		
Muster-Struktur		
`name=muster;`	`muster`	HyPaL: die Definition mehrerer Muster in einer Datei ist möglich
Kommentare		
`//`	`%`	Programm ignoriert alle bis zum Ende der Zeile folgenden Zeichen
Primärsequenz		
Nukleotide (Atome)		
`HUMA.`	`HuMAn`	alle Zeichen der IUPAC-Nomenklatur sind erlaubt; HyPaL: nur Großbuchstaben erlaubt; „`.`“ alternativ für `N` benutzbar
Zeichenklassen		
`[ACU]`, `[^CU]`	n.i.	Alternative zur IUPAC-Nomenklatur; `^`: Komplement-Operator
Reguläre Ausdrücke		
Verkettung		
`GC GN[GA]A GC`	`GC GNRA GC`	Aneinanderreihung einzelner Mustereinheiten bestimmt die Reihenfolge, in der diese gefunden werden sollen
Disjunktion		
`G|A`	`(G | A)`	Definition von Alternativen
Quantifizierung		
`G{2,3}`, `.{3,5}`, `A*`, `G?`, `U+`	`(GG | GGG)`, `3...5`	Definition von Wiederholungen; in HyPaL sind zusätzliche Abkürzungen implementiert
Gruppierung		
`(G|A)U(A|C)`, `U(ACU){3,5}`	n.i.	Zuordnung von logischen Verknüpfungen oder Quantifizierungen
Variablen		
`(loop:=GNRA)`	`p1=GNRA`	Speicherung von Zeichenketten; HyPaL: Name beliebig wählbar, aber Kleinbuchstaben
Sekundärstruktur		
Struktur		
`(stem:=.{3,4})` ... `^stem`	`p1=3...4` ... `~p1`	Verwendung von Variablen zur Bildung des rev. Komplements mit Hilfe entsprechender Funktionen; HyPaL: „`^`“=Watson-Crick- und Wobble-Bp.; PatScan: „~“=Watson-Crick-Bp.
Basenpaar-Regeln		
`wobble G:=[U];` `wobble U:=[G];`	`r1={gu,ug}`	Definition von spez. Komplement-Funktionen (hier Wobble-Bp.: `wobble(stem)` bzw. `r1~p1`)
Approximative Elemente		
`UNCG[2,0,1]`, `^stem[0,1,1]`	`UNCG[2,0,1]`, `~p1[0,1,1]`	ermöglicht die Angabe von Fehlern in einzel- und doppelsträngigen Bereichen (`[mis,del,ins]`)
Profil-Matrizen		
`[[0,0,100,0], [...],[...]]>=245`	`{(0,0,100,0), (...),(...)}>244`	Beschreibung gewichteter Konsensus-Sequenzen konstanter Länge; Gew.-Reihenfolge: f_A,f_C,f_G,f_U

Tabelle 3.4: Überblick über HyPaL-spezifische Sprachelemente und Bewertungsfunktionen. In der linken Spalte ist immer ein z. T. verallgemeinertes Beispiel angegeben, das in der rechten Spalte kurz erklärt wird. Eine detaillierte Beschreibung der aufgeführten Sprachelemente ist im Text zu finden.

HyPaL	**Erklärungen**
`where`-Klausel	

`tetraloop = stem loop ^stem`
`where stem:=.{3,4}`
`loop:=GNRA;`
	Erlaubt die übersichtliche Gestaltung des Musters. (hier am Beispiel einer Haarnadel-Struktur); die Variablen-Definition erfolgt erst nach der `where`-Klausel.

Muster mit Parametern	

`hp(%loop)=(stem:=.{3,4})`
`%loop ^stem;`
`gnra=GNRA;`
`gnra_tetraloop=hp(gnra);`
	Erlaubt die Definition verallgemeinerter Muster (hier `hp(%loop)`), denen Parameter übergeben werden, die selber Muster sind (hier `gnra`). Erstere können dann zur Definition spezieller Muster verwendet werden.

Bewertungsfunktionen[a]	
Längeneinschränkungen	
`length(var)`	Funktion zur Berechnung der Länge der Sequenz, die in der Variable `var` gespeichert ist.
Nukleotidgehalt[b]	
`gccontent(var)`	Funktion zur Berechnung des relativen GC-Gehalts der Sequenz, die in der Variable `var` gespeichert ist.
`xcontent(b,var)`	Funktion zur Berechnung des relativen Gehalts der Base(n) `b` der Sequenz, die in der Variable `var` gespeichert ist.
`absxcontent(b,var)`	Funktion zur Berechnung des absoluten Gehalts der Base(n) `b` der Sequenz, die in der Variable `var` gespeichert ist.
Thermodynamische Stabilität (freie Energie)	
`mfreeE(var_construct,T)`	Funktion zur Berechnung der minimalen freien Energie des Variablen-Konstrukts `var_construct` bei der Temp. `T` in °C.
`freeE(var_construct,T)`	Funktion zur Berechnung des Sekundärstruktur-Konstrukts `var_construct` bei der Temp. `T` in °C.
`mfeDiff(var_construct,T)`	Funktion zur Berechnung der Differenz zwischen freier und minimaler freier Energie des Sekundärstruktur-Konstrukts bei der Temp. `T` in °C `var_construct`.

[a] Angabe der Bewertungsfunktionen erfolgt hinter dem „`<<`“-Operator entweder am Ende des Musters, wenn keine `where`-Klausel enthalten ist, oder zwischen der allgemeinen Musterbeschreibung und der `where`-Klausel. Diese Funktionen dürfen immer nur auf Sequenzen angewendet werden, die an Variablen gebunden sind.

[b] `b` kann eine beliebige, auf dem verwendeten Alphabet basierende Nukleotid-Kombination als Wert annehmen.

und Tabelle 3.4 sind die zur Verfügung stehenden Sprachelemente nochmals im Überblick zusammengestellt. Die Beschreibung von Primärsequenz und Sekundärstruktur ist in beiden Sprachen in nahezu dem gleichen Umfang möglich. Allerdings zeichnet sich HyPaL schon hier durch wesentlich größere Flexibilität und Vielfalt aus, da z. B. reguläre Ausdrücke verwendet werden können. Die Formulierungen einer variablen Anzahl an konservierten Nukleotiden beispielsweise ist in der PatScan-Notation nur umständlich oder gar nicht möglich (`((GG | ( GGG | GGGG))` vs. `G{2,4}` oder `(W | 0...0)` vs. `W?`). Ein weiterer Vorteil gegenüber PatScan steckt in der Art und Weise wie Variablen eingesetzt werden können. Eine Variablen-Definition darf verschiedene Beschreibungstypen enthalten, was in PatScan nicht möglich ist (z. B. `(stem:=.{3,4} U C{2,1}) ...`). Zudem kann die sinnvolle Wahl von Variablennamen entscheidend zur Lesbarkeit eines Musters beitragen. Im Vergleich dazu stehen in PatScan nur 51 vordefinierte Variablennamen zur Verfügung (`p0` bis `p50`). Eine übersichtliche Strukturierung des Musters kann in HyPaL u. a. durch Verwendung der `where`-Klausel erreicht werden. Durch den modularen Aufbau der Sprache ist es möglich bereits definierte Muster wiederzuverwenden und somit komplexere Beschreibungen aus einfacheren Mustern aufzubauen. Alternativ dazu können verallgemeinerte Muster erstellt werden, die wiederum andere Muster als Parameter akzeptieren. Alle diese modularen Eigenschaften von HyPaL sind weder in PatScan noch in anderen Muster-Suchwerkzeugen zu finden.

Der größte Vorteil von HyPaL allerdings besteht in der Möglichkeit hybride Muster zu formulieren, die zusätzliche Bewertungsfunktionen enthalten können, wodurch die Spezifität und Sensitivität entscheidend erhöht und die Anzahl an falsch positiven Treffern deutlich verringert werden kann. Eine entsprechende Funktionalität ist ebenfalls weder in PatScan noch in anderen Suchwerkzeugen auf diese Art und Weise implementiert. Somit sind die Eingangs erwähnten Anforderungen an HyPaL alle erfüllt und umgesetzt worden.

3.3 Muster-Suchwerkzeuge

Für die Mustersuche wurden die Suchwerkzeuge PatScan und das in Kooperation entwickelte HyPaSearch verwendet. In Abschnitt 3.1 wurde bereits allgemein die Vorgehensweise bei der Mustersuche skizziert. Im Folgenden wird zunächst allgemein auf Algorithmen zur Suche von Mustern in Zeichenketten eingegangen, um dann die verwendeten Werkzeuge detailliert beschreiben und vergleichen zu können.

3.3.1 Algorithmen zur Suche in Zeichenfolgen

Die Problemstellung

Das grundsätzliche Ziel ist es, eine Zeichenkette, ein sog. Muster, in einer längeren Zeichenkette zu finden. Angenommen P sei das Muster und S die Zeichenkette, die durchsucht werden soll. Dann gilt es jedes Vorkommen von P in S zu finden.

Allgemeine Definitionen: Eine Zeichenkette S (String; z. B. eine DNA-Sequenz) ist eine kontinuierliche Aneinanderreihung von n Buchstaben (z. B. Nukleotiden) eines definierten, endlichen Alphabets $\mathcal{A}$. In einem beliebigen String S ist $S[i]$, mit $1 \leq i \leq n$, der i-te Buchstabe von S. Jeder String $S[i \dots j]$, mit $1 \leq i < j \leq n$, ist ein zusammenhängender Teil-String von S, der an der i-ten Position von S anfängt und an der j-ten Position von S endet. Jeder Teil-String $S[1 \dots j]$, mit $1 < j \leq n$, der an Position 1 von S beginnt, wird definitionsgemäß als **Präfix** von S bezeichnet. Umgekehrt ist jeder Teil-String $S[i \dots n]$, mit $1 \leq i < n$, der mit dem letzten Buchstaben endet, ein **Suffix** von S. Ist $i > j$, ist der Teil-String $S[i...j]$ von S eine leere Zeichenkette.

Der naive Ansatz

Ein naiver, gieriger (*greedy*) Algorithmus legt P am Beginn von S an und vergleicht dann der Reihe nach jeden Buchstaben in P mit dem entsprechenden Buchstaben in S. Eine Übereinstimmung zwischen P und S ist gefunden, wenn eine Folge von positiven Vergleichen das Ende von P erreicht hat. Dann, oder wenn bei einem Vergleich an einer Position unterschiedliche Buchstaben gefunden werden, wird P relativ zu S um eine Position weiter geschoben und der Vergleich von P mit S wird erneut der Reihe nach durchgeführt. Dieser Prozess wird solange wiederholt, bis das Ende von P um einen Buchstaben über das Ende von S hinausragt und damit die gesamte Zeichenkette S nach Übereinstimmungen mit P durchsucht wurde.

Angenommen $P[1 \dots m]$ sei das in $S[1 \dots n]$ gesuchte Muster. Dann kann das Muster P genau $(n-m+1)$-mal über S weitergeschoben werden. Dabei wären im ungünstigsten Fall (*worst case*) insgesamt $m(n-m+1)$ Vergleiche notwendig, um die gesamte Zeichenkette S zu durchsuchen. Der *worst case* tritt beispielsweise ein, wenn S und P ausschließlich aus ein und demselben Buchstaben bestehen. Angenommen $P = \text{CCC}$ und $S = \text{CCCCCCC}$, dann ist $n = 3$ und $m = 7$. Demnach sind mit diesem Ansatz insgesamt 15 Vergleiche notwendig, um alle Übereinstimmungen zwischen P und S zu finden. Damit ergibt sich für diesen *greedy*-Algorithmus eine *worst-case*-Laufzeit von $\mathcal{O}(nm)$. In Abbildung 3.4 ist die Strategie nochmals an einem anderen Beispiel dargestellt. Man kann sich leicht ausrechnen, dass beispielsweise das Durchsuchen von ganzen Genomen mit diesem Algorithmus sehr aufwendig ist und dementsprechend lange dauern würde.

Ein solcher *greedy*-Algorithmus lässt sich optimieren, indem man versucht die Anzahl an notwendigen Vergleichen zu reduzieren. Man kann beispielsweise versuchen, das Muster P nicht immer nur um eine, sondern gleich um mehrere Positionen weiterzuschieben. Dabei muss natürlich sicher gestellt werden, dass keine Übereinstimmungen zwischen P und S übersehen werden. Das kann erreicht werden, indem durch Vorverarbeitung des Musters P oder der Zeichenkette S gewonnene Information genutzt wird, um zu entscheiden, wie weit das Muster weitergeschoben werden kann. In Abbildung 3.4 sind zwei weitere Strategien skizziert, die von Informationen Gebrauch machen, die durch Vorverarbeitung des Musters P gewonnen wurden. In dem einen Fall (**B**) weiß der Algorithmus nach dem neunten Vergleich, dass die ersten 7 Buchstaben von P mit den Buchstaben 2 bis 8 von S übereinstimmen. Zudem kommt der erste Buchstabe von P, ein `A`, erst wieder an Position 5

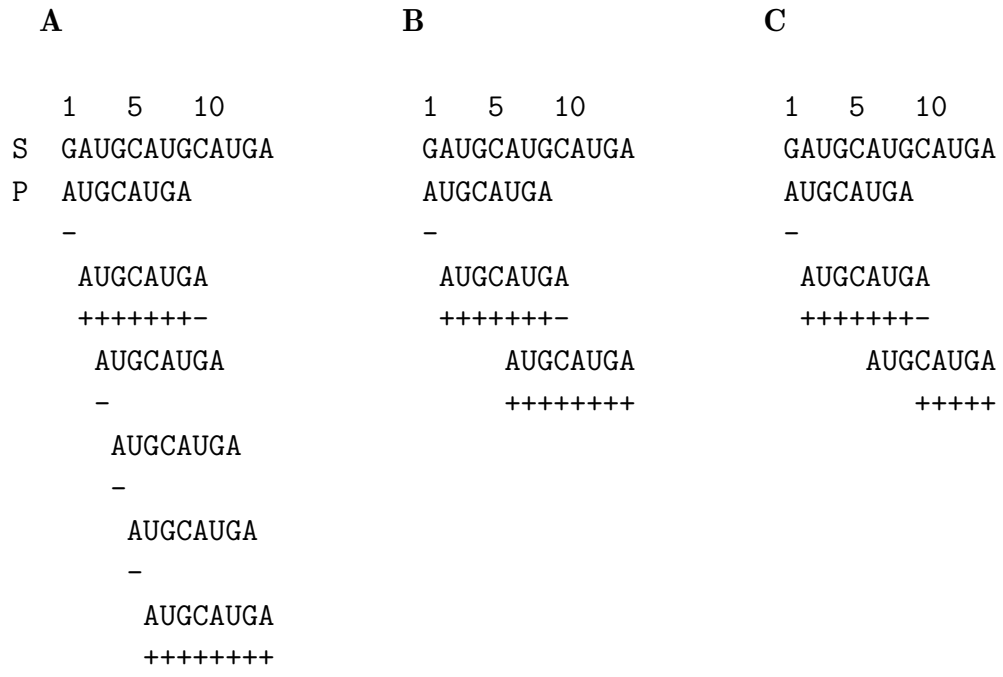


Abbildung 3.4: Drei Strategien, mit denen eine Sequenz S mit einem Muster P mehr oder weniger effizient durchsucht werden kann. Während der naive Ansatz (**A**) 20 Vergleiche benötigt, kommen die beiden anderen Strategien (**B** und **C**; Erklärungen siehe Text) mit nur 17 bzw. 14 Vergleichen aus, um alle Übereinstimmungen zwischen P und S zu ermitteln. Alle notwendigen Vergleiche sind entweder durch ein „+“ (Übereinstimmung) oder ein „−“ (Unterschied) gekennzeichnet (Nach Gusfield, 1997, verändert).

---

von P vor. Daraus kann der Algorithmus ableiten, dass bis Position 6 von S keine weiteren Übereinstimmungen mit P vorkommen und somit P gleich um drei Positionen relativ zu S weitergeschoben werden kann. Bei diesem Beispiel müssen im Vergleich zum naiven Ansatz drei Vergleiche weniger durchgeführt werden. In dem anderen Beispiel (**C**) können gegenüber dem naiven Ansatz sogar sechs Vergleiche eingespart werden. Bei der hier verwendeten Strategie ist dem Algorithmus bekannt, dass die ersten drei Buchstaben von P (`AUG`) ab Position 5 in P nochmal vorkommen. Zudem stimmen die ersten 7 Buchstaben von P mit den Buchstaben 2 bis 8 von S überein. Damit kann der Algorithmus sowohl P um drei Positionen weiterschieben, als auch die nächsten drei Vergleiche überspringen.

Entsprechende oder ähnliche Strategien sind z. B. im Knuth-Morris-Pratt(KMP)-Algorithmus (Knuth *et al.*, 1977) oder im Boyer-Moore-Algorithmus (Boyer & Moore, 1977) umgesetzt. Hier wird vor der eigentlichen Suche das Muster P analysiert, wobei der Algorithmus zu diesem Zeitpunkt noch keine Informationen über den zu durchsuchenden String S hat. Alle diese Algorithmen konnten so implementiert werden, dass sie in linearer Zeit mit Aufwand $\mathcal{O}(n + m)$ laufen.

Suffix-Bäume und Suffix-Listen

Die zuvor erwähnten Algorithmen lassen sich nur für die Suche von exakt übereinstimmenden Zeichenketten in langen Strings einsetzen. Beim Durchsuchen von genomischen Sequenzdaten sollen dagegen auch homologe oder paraloge Sequenzen gefunden werden können, die sich in einigen Positionen unterscheiden. Zu diesem Zweck können Suffix-Bäume verwendet werden, mit deren Hilfe sowohl das Auffinden von exakten als auch von nicht-exakten Mustern effizient möglich ist. Der Suffix-Baum ist eine Datenstruktur, die die zu durchsuchende Sequenz repräsentiert. Dabei werden alle möglichen Suffixe einer Zeichenkette gemeinsam in einer Baumstruktur zusammengefasst. Im Unterschied zu den zuvor eingeführten Algorithmen findet hier also eine Vorverarbeitung der zu durchsuchenden Zeichenkette und nicht des Musters statt. Zunächst wird im Folgenden die Datenstruktur des Suffix-Baums erläutert, um dann an einem kleinen Beispiel zu verdeutlichen, wie die Suche in einer solchen Datenstruktur vollzogen wird. Abschließend wird die Datenstruktur der Suffix-Listen vorgestellt, die von Suffix-Bäumen abgeleitet werden kann und einige Vorteile gegenüber der Baumstruktur mit sich bringt.

Allgemeine Definitionen: Vorrausgesetzt wird, dass das verwendete Alphabet $\mathcal{A}$ endlich und bekannt ist. Ein Suffix-Baum $\mathcal{T}$, der einen auf dem verwendeten Alphabet $\mathcal{A}$ basierenden String S der Längen n repräsentiert, ist ein gewurzelter, gerichteter Baum (*rooted directed tree*) mit exakt n (Länge des Strings S) Blättern. Alle zu einem Suffix von S gehörenden Teil-Strings werden an entsprechende Kanten geschrieben. Ein solcher Baum zeichnet sich dadurch aus, dass der Weg von der Wurzel bis in ein beliebiges Blatt i (L_i, mit $1 \leq i \leq n$) den Suffix von S wiedergibt, der an Position i von S beginnt, d. h. alle Suffixe enden in einem Blatt. Jeder interne Knoten (mit Ausnahme der Wurzel) hat mindestens zwei nachfolgende Kanten, von denen jede einen nicht-leeren Teil-String von S repräsentiert. Kanten, die einem Knoten entspringen, dürfen nicht mit dem gleichen Buchstaben beginnen.

Abbildung 3.5 zeigt exemplarisch den Suffix-Baum der Sequenz S = CUGAUGAA, die auf dem Alphabet $\mathcal{A} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{U}\}$ für RNA basiert. Folgt man in dem Baum dem Weg von der Wurzel bis in das Blatt, das mit 1 beschriftet ist, findet man den kompletten Sequenz-String CUGAUGAA. Dieser Suffix beginnt an Position 1 von S. Geht man den Weg von der Wurzel bis in das Blatt, das mit der 6 markiert ist, erhält man den Suffix GAA, der an Position 6 von S beginnt.

Welche Bedeutung hat das in der Darstellung verwendete „$“-Zeichen? Man denke sich in Abbildung 3.5 alle „$“-Zeichen, sowie alle Kanten, die nur mit einem „$“-Zeichen beschriftet sind, weggelassen. Da der Ein-Buchstaben-Suffix A gleichzeitig auch Präfix der Suffixe AGUAA (4) und AA (7) ist, würde so der Ein-Buchstaben-Suffix A nicht in einem Blatt enden. Das ist allerdings nach der o. a. Definition nicht erlaubt. Um dieses Problem zu umgehen, wird für die Konstruktion eines Suffix-Baums das verwendete Alphabet $\mathcal{A}$ um ein terminales Zeichen, meistens das „$“-Zeichen erweitert, das nicht im Alphabet und damit auch nicht in der Zeichenkette vorkommt. Somit wird verhindert, dass ein Suffix gleichzeitig Präfix eines anderen Suffix ist.

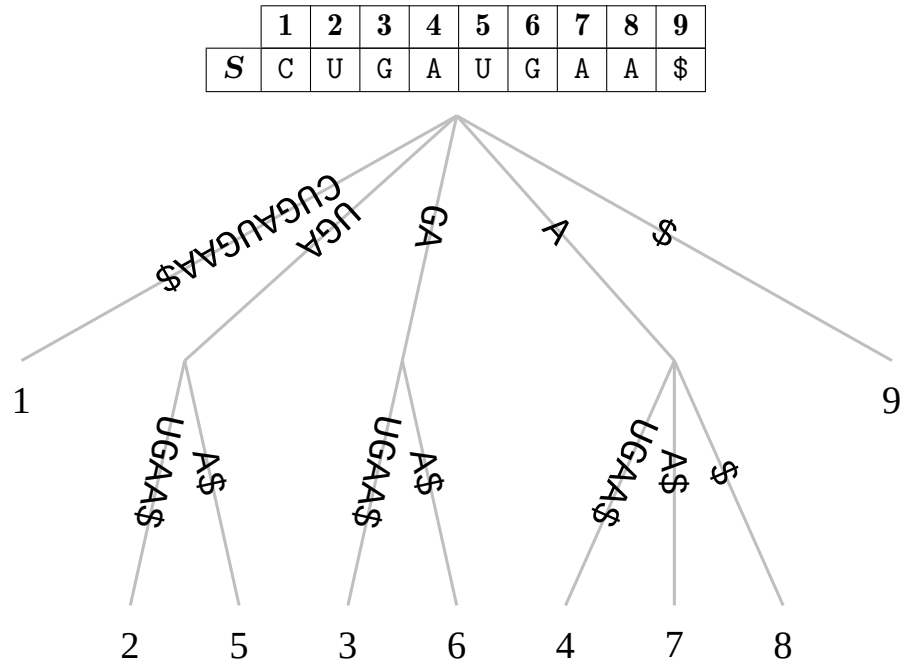


Abbildung 3.5: Repräsentation der Zeichenkette `CUGAUGAA` als Suffix-Baum.

Die Suche in einem solchen Baum beginnt an der Wurzel. Der Algorithmus vergleicht das Muster Zeichen für Zeichen von links nach rechts mit den Buchstaben (ausgehend von der Wurzel), die an den Kanten stehen, und bewegt sich dabei immer weiter in den Baum hinein in Richtung der Blätter. Ist der gesuchte String P gefunden, bezeichnen alle an der entsprechenden Kante hängenden Blätter die Indizes, an denen das Muster P in der Sequenz S beginnt. Ein Muster P kommt in der durchsuchten Zeichenkette S an Position j nur vor, wenn P ein Präfix eines im Baum enthaltenen Suffix $S[j \dots n]$ ist. Können ab einer bestimmten Position im Muster keine Übereinstimmungen mehr mit den noch an entsprechenden Kanten zur Verfügung stehenden Buchstaben gefunden werden, ist der gesuchte String P nicht in der Zeichenkette S enthalten, da er aufgrund der Datenstruktur auch nicht in einem anderen Teil des Baums vorkommen kann.

Angenommen es soll das Muster P = GA in dem String S aus Abbildung 3.5 gesucht werden. Diesen String findet der Algorithmus an der dritten Kante von links als Präfix der Suffixe GAUGAA$ und GAA$. Die in dem darunter liegenden Teil-Baum enthaltenen Blätter besitzen die Indizes 3 und 6, welche die Startpositionen des Musters P in der Zeichenkette S markieren. Soll dagegen der String P = UGAG in S gesucht werden, findet der Algorithmus die ersten drei Buchstaben der Suchsequenz an der zweiten Kante von links (ausgehend von der Wurzel). Die nächsten zwei Vergleiche mit dem U, das zu dem Blatt mit Index 2 führt, und mit dem A der benachbarten Kante ergeben keine Übereinstimmung mit dem noch übrigen G von P. Damit folgt, dass dieses Muster nicht in der Sequenz enthalten ist.

Die Erstellung eines Suffix-Baums kann mit Hilfe eines naiven Ansatzes erfolgen. Dabei wird der Baum mit dem längesten Suffix $S[1 \dots n]$\$ initialisiert, so dass dieser aus der Wurzel, einer Kante, an der der gesamte Sting S\$ steht, und einem Blatt mit dem Index 1

besteht. Anschließend werden sukzessive alle anderen Suffixe ($S[i \dots n]\$, \dots, S[n \dots n]\$, \$$, mit $2 \leq i \leq n$) in den wachsenden Baum eingehängt. Dazu muss durch Zeichenvergleiche der längstmögliche Weg in dem bis dahin bestehenden Baum gefunden werden, der mit dem Präfix des einzubindenden Suffix übereinstimmt. Hinter dem Punkt, an dem die letzte Übereinstimmung gefunden wurde, wird die bestehende Kante durch einen neuen Knoten unterbrochen. An diesem Knoten wird dann eine neue Kante eingehängt, an die der verbleibende Rest-String des einzubindenden Suffix geschrieben wird. Im zugehörigen Blatt wird die Startposition des Suffix in S gespeichert. Dieser naive Ansatz ist mit einem Aufwand von $\mathcal{O}(n^2)$ verbunden, wobei n die Länge der zu indizierenden Sequenz ist. Dieser Aufwand lässt sich auf $\mathcal{O}(n)$ reduzieren, wenn man für die die Konstruktion des Suffix-Baums die von Weiner (1973), McCreight (1976) oder Ukkonen (1995) entwickelten Algorithmen verwendet (Übersichtsartikel: Giegerich & Kurtz, 1997).

Die Suche setzt sich zusammen aus den notwendigen Vergleichen, um das Muster im Baum zu identifizieren, und dem „Einsammeln“ aller zugehörigen Indizes. Dieser Schritt kann durch Traversierung durch den verbleibenden Teil-Baum in linearer Zeit mit Aufwand $\mathcal{O}(k)$ erfolgen, wobei k die Anzahl der im Baum enthaltenen Treffer ist. Damit ergibt sich ein Aufwand von $\mathcal{O}(m + k)$ für die Suche, wobei m die Länge des Musters ist. Im Gegensatz dazu erfolgt bei den Algorithmen, die auf dem naiven Ansatz basieren (vgl. Abschnitt 3.3.1), eine Vorverarbeitung des Muster P mit Aufwand $\mathcal{O}(m)$ und nicht der Sequenz. Die Suche mit einem vorverarbeiten Muster kann in linearer Zeit mit Aufwand $\mathcal{O}(n)$ durchgeführt werden. Man beachte, dass das Musters im Normalfall erheblich kürzer ist als die zu durchsuchende Sequenz, zumal das hier zu entwickelde Werkeug für die Suche in großen genomischen Sequenzdaten eingesetzt werden soll.

Eine alternative, vom Suffix-Baum abgeleitete Datenstruktur ist die Suffix-Liste, die meist als Suffix-Array bezeichnet wird (Manber & Myers, 1993). Diese zeichnet sich im Vergleich zum Suffix-Baum v. a. durch den drei bis fünf mal geringeren Speicherbedarf aus. Während ein Suffix-Baum im *worst-case* $20n$ an Speicher bedarf, benötigt ein Suffix-Array in seiner grundlegenden Form nur 4 Bytes pro Eingabe-Zeichen. Die Konstruktion lässt sich im *worst-case* mit Aufwand $\mathcal{O}(n)$ durchführen, wenn zuerst der entsprechende Suffix-Baum erstellt wird. Dazu kommt, dass Algorithmen, die Suffix-Arrays benutzen ein besseres Laufzeitverhalten bei der Suche zeigen ($\mathcal{O}(m \log n)$ im *worst-case*) und sich einfacher implementieren lassen (Abouelhoda *et al.*, 2002a). Zudem konnte gezeigt werden, dass jeder Algorithmus, der Suffix-Bäume benutzt, durch einen äquivalenten auf Suffix-Arrays basierenden Algorithmus unter Verwendung zusätzlicher Informationen ersetzt werden kann (Abouelhoda *et al.*, 2004).

Allgemeine Definitionen: Gegeben sei eine zu durchsuchende Zeichenkette S mit n Buchstaben auf einem definierten, endlichen Alphabet $\mathcal{A}$. Zudem enthält $\mathcal{A}$ ein weiteres Zeichen \$, das nicht in S vorkommt und lexikalisch größer ist, als alle anderen Zeichen in $\mathcal{A}$. Dann ist $S[i]$, mit $0 \leq i \leq n$, ein Zeichen an der Position i von S. Der String $S[i \dots j]$ ist demnach ein Teil-String von S, der an Position i startet und an Position j endet, wenn $i \leq j$ ist. Ein Suffix-Array suf einer Zeichenkette S ist eine Liste (Array) von ganzen Zahlen (Integer) mit Werten zwischen 0 und n, in der die Indizes der $n+1$ Suffixe von $S\$$ in lexikographischer Reihenfolge gespeichert werden. Diese Liste ($S_{\mathsf{suf}[0]}, S_{\mathsf{suf}[1]}, \dots, S_{\mathsf{suf}[n]}$)

suf-Index	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
S	C	U	G	A	U	G	A	A	$

i	$\mathsf{suf}[i]$	$\mathsf{lcp}[i]$	$S_{\mathsf{suf}[i]}$
0	7	0	A$
1	6	1	AA$
2	3	1	AUGAA$
3	0	0	CUGAUGAA$
4	5	0	GAA$
5	2	2	GAUGAA$
6	4	0	UGAA$
7	1	3	UGAUGAA$
8	8	0	$

Abbildung 3.6: Suffix-Array der Zeichenkette S = CUGAUGAA**.** Links ist die Zeichenkette S mit den Indizes der Suffixe dargestellt. Rechts sind die im Suffix-Array enthaltenen Werte in lexikalischer Reihenfolge der Suffixe aufgelistet, die für die Suche nach einem Muster P in der Zeichenkette S benötigt werden.

enthält also der lexikographischen Sortierung entsprechend die Startpositionen der Suffixe von $S\$$ in aufsteigender Reihenfolge, wobei $S_i = S[1 \ldots n-1]\$$, mit $0 \leq i \leq n$, den i-ten, nicht leeren Suffix von $S\$$ markiert (vgl. Abbildung 3.6). Zur Speicherung werden hierfür die bereits erwähnten $4n$ Bytes benötigt, vorausgesetzt $n < 2^{32}$. Die Spalte mit dem längsten gemeinsamen Präfix lcp (*longest common prefix*) ist ebenfalls ein Array von Integers mit Werten zwischen 0 und n, in der jeweils die Länge des längsten gemeinsamen Präfix von $S_{\mathsf{suf}[i-1]}$ und $S_{\mathsf{suf}[i]}$ gespeichert wird, mit $1 \leq i \leq n$. Per Definition ist $\mathsf{lcp}[0] = 0$ und $\mathsf{lcp}[n] = 0$ (vgl. Abbildung 3.6). Diese Liste wird parallel zur Konstruktion des Suffix-Arrays erstellt oder kann mit linearem Aufwand aus dem Suffix-Arrays berechnet werden. Zur Speicherung dieser Liste werden ebenfalls $4n$ Bytes benötigt.

Auf dieser Datenstruktur kann nun mit einem einfachen Algorithmus gesucht werden, um alle Teilsequenzen zu finden, die auf ein Muster P passen. Ist das Muster P in der Zeichenkette S enthalten, so liegen alle diese Teilstrings aufgrund der lexikalischen Sortierung gruppiert in dem Suffix-Array vor. Soll also beispielsweise der String P = UGA in S = CUGAUGAA gesucht werden (vgl. Abbildung 3.6), kann das mit einer binären Suche erfolgen. Dabei wird als erstes das mittlere Element des Arrays mit Index $i = \lceil m/2 \rceil$ ausgewählt und mit dem dort enthaltenen Suffix mit P verglichen. Der Vergleich im Beispiel ergibt, dass P lexikalisch größer ist als $S_{\mathsf{suf}[4]}$ = GAA$, wodurch der Algorithmus weiß, dass das gesuchte Muster, wenn es in S enthalten ist, nur noch in einem Element mit Index $i > 4$ vorkommen kann. Aus diesem Teil des Arrays wird dann wieder das mittlere Element ($\lceil (i + 1 + m)/2 \rceil$) für die nächsten Vergleiche gewählt. Im Beispiel würde so als nächstes der Suffix $S_{\mathsf{suf}[7]}$ = UGAUGAA mit P verglichen, womit der erste Treffer gefunden wäre und der aktuelle Index auf der gesuchten Gruppe von Suffixen im Array steht. Diese Iteration wird solange durchgeführt, bis die untere und die obere Grenze der gesuchten Gruppe gefunden ist. Im Beispiel sind das die Suffixe $S_{\mathsf{suf}[6]}$ und $S_{\mathsf{suf}[7]}$. Die zugehörigen Startpositionen ergeben sich aus der suf-Spalte, hier 1 und 4. Durch zusätzliche Berücksichtigung der lcp-Werte bei der binären Suche, können zudem viele redundante

Zeichen-Vergleiche eingespart werden, auch wenn dies auf den ersten Blick einen zusätzlichen Aufwand bei der Durchführung bedeutet (für Details siehe Gusfield, 1997, S. 152ff).

3.3.2 PatScan/PatSearch

Das Suchwerkzeug PatScan wurde von D'Souza *et al.* (1997) entwickelt und ist frei als Internet-Service verfügbar. Zusätzlich konnte man das Programm inklusive des Programm-Codes herunterladen und lokal übersetzen, installieren und benutzen. Eine Weiterentwicklung dieses Programms steht jetzt unter dem Namen PatSearch (Grillo *et al.*, 2003; Pesole *et al.*, 2000) zur Verfügung. Lange Zeit konnte letzteres nur nach Registrierung über das Internet benutzt werden. Erst seit kurzem steht eine übersetzte, binäre Version zur lokalen Benutzung zur Verfügung. Da in beiden Fällen die im Internet angebotenen Sequenzdaten für die in dieser Arbeit durchgeführten Analysen nicht ausreichend waren, wurde ausschließlich die lokal installierbare Version von PatScan benutzt. Diese Version wurde zudem im Rahmen dieser Arbeit von Stephan Zanger im Rahmen eines Praktikums in Teilen erweitert und weiterentwickelt, worauf weiter unten in diesem Abschnitt noch näher eingegangen wird.

Kommandozeilen-Parameter und Benutzung

Das ausführbare Programm, das die Suche in Sequenzdaten durchführt, heißt `scan_for_matches`. Mit der Option `-h` wird die in Abbildung 3.7 dargestellte Hilfe zur Benutzung und zu den Kommandozeilen-Parametern ausgegeben. Diese übersichtliche Hilfe wurde neu formuliert, um dem Benutzer das Arbeiten mit dem Programm zu erleichtern. Nachfolgend sind die wichtigsten Optionen im Einzelnen aufgeführt. Standardmäßig sucht das Programm nur auf dem Vorwärts-Strang der angegebenen Sequenzen. Soll ebenfalls der revers-komplementäre Strang durchsucht werden, kann der Benutzer das durch Angabe der Option `-c` bewirken. Die folgenden Optionen wurden neu implementiert. Normalerweise sucht PatScan nicht-überlappend, d. h. es werden nur Treffer ausgegeben die sich gegenseitig nicht überschneiden. Will der Benutzer alle Treffer, d. h. inkl. überlappender Treffer, kann er die Option `-o` angeben. Der ebenfalls neu implementierte Schalter `-s` bewirkt, dass unter jeder Treffersequenz auch die zugehörige Sekundär-Struktur in Form der Punkt-Klammer-Notation (vgl. Abbildung 1.2) mit ausgegeben wird. Diese Darstellung erleichtert v. a. bei großen Mustern das Wiederfinden der einzelnen Musterelemente und kann zudem zur Berechnung der thermodynamischen Stabilität der Struktur verwendet werden. Die Ausgabe der Treffersequenzen kann auch ohne Leerzeichen erfolgen, wenn die Option `-W` angegeben wird.

Die zu durchsuchenden Sequenzdaten können auf zwei verschiedene Arten dem Programm übergeben werden. Ursprünglich verarbeitete `scan_for_matches` ausschließlich FASTA-formatierte Sequenzdaten. Diese können dem Programm über die Standard-Eingabe zur Verfügung gestellt werden. Um ebenfalls Sequenzdaten im EMBL- oder Genbank-Format verarbeiten zu können, wurden zwei kleine Programme in der Programmiersprache C (EMBL2FASTA und GB2FASTA) geschrieben, die die Konvertierung ins FASTA-Format

```
NAME
   scan_for_matches - PatScan

USAGE
   scan_for_matches [options] patternfile < fasta_input
                    > hitsfile
   scan_for_matches [options] -f sequence_input patternfile
                    > hitsfile

OPTIONS
   -h       help; print this message
   -c       search also complementary strand
   -o       perform overlapping search (slow)
   -p       for Protein sequences
   -s       print out bracket-dot notation (only [DR]NA)
   -W       to get results without separating whitespaces
   -n int   stop_after_int_misses
   -m int   max_hits
   -i file  file_of_ids_to_ignore
   -f file  sequence_input

            File specified with -f can be in the following
            formats (autodetection, using SQUID-library):
            * Unaligned - fasta, embl, genbank, gcg, gcgdata, pir, raw
            * Aligned - stockholm, msf, a2m, phylip, clustal, selex
```

Abbildung 3.7: Ausgabe der Hilfe zum Suchwerkzeug PatScan. Diese Ausgabe wird beim Aufruf des Suchprogramms `scan_for_matches` mit der Option `-h` angezeigt und liefert alle Informationen zur Benutzung und zu den verfügbaren Kommandozeilen-Parametern des Programms.

durchführen. Diese beiden Programme können wahlweise dem Suchwerkzeug zur Vorverarbeitung vorangestellt werden. Beide lesen die Sequenzdaten über die Standard-Eingabe ein und geben die konvertierten Sequenzen über die Standard-Ausgabe aus. Alternativ können die Sequenzdaten jetzt auch mit der neu implementierten Option `-f` übergeben werden. Hierbei wird zum Einlesen der Sequenzdaten die SQUID-Bibliothek von Eddy (2002) verwendet, die automatisch entscheidet, welches Format die Sequenzdaten haben. Die Sequenzformate, die erkannt werden können, sind in der Hilfe in Abbildung 3.7 aufgelistet.

Repräsentation der Ergebnisse

Gefundene Treffer werden in der folgenden FASTA-ähnlichen Formatierung ausgegeben:

```
>S3QNAM3:[3617,3634]
cgatgtg cgt cgcactcg
((((((( ... )))).)))
```

Jeder Treffer setzt sich aus zwei bzw. drei Zeilen zusammen, je nachdem ob die Punkt-Klammer-Notation mit ausgegeben wird oder nicht. Die erste Zeile beginnt mit einem „>“-Zeichen gefolgt von dem Bezeichner der Sequenz, in der der Treffer gefunden wurde. Durch Doppelpunkt getrennt folgt in eckigen Klammern Komma-separiert die Angabe der absoluten Start- und Endposition des Treffers in der Sequenz. Ist der Wert für die Startposition größer als der für die Endposition, liegt der Treffer auf dem reversen Strang. Die zweite Zeile enthält die Nukleotidsequenz des Treffers. Standardmäßig wird die Treffersequenz entsprechend den einzelnen Mustereinheiten durch Leerzeichen getrennt ausgegeben. Eine Mustereinheit besteht definitionsgemäß aus einer Abfolge von Zeichen und Zahlen ohne Leerzeichen. Diese Mustereinheiten ergeben sich zum einen aus den verschiedenen Musterbeschreibungsmöglichkeiten und andererseits aus den vom Benutzer bei der Beschreibung verwendeten Leerzeichen und Zeilenumbrüchen. Die Sprachelemente der Beschreibungssprache von PatScan wurden bereits in Abschnitt 3.2 erläutert. In der dritten Zeile folgt ggf. die zugehörige Punkt-Klammer-Notation.

Algorithmus

Ein Muster besteht aus verschiedenen Mustereinheiten, die sich durch die verschiedenen Beschreibungsmöglichkeiten ergeben (vgl. 3.2). Im Allgemeinen sind einzelne Mustereinheiten durch Leerzeichen oder Zeilenumbrüche getrennt. Bei dem hier verwendeten Algorithmus handelt es sich um einen sog. naiven oder gefräßigen (*greedy*) Ansatz, der die Sequenz linear vom Anfang bis zum Ende durchsucht. Die Suche beginnt an Position 1 der zu durchsuchenden Sequenz, d.h. der Zähler für die aktuelle Position in der Sequenz wird auf 1 gesetzt. Passt die erste Mustereinheit auf die an Position 1 beginnende Subsequenz, wird anschließend die nächste Mustereinheit gesucht. Passt eine Mustereinheit nicht auf die aktuelle Subsequenz, springt der Algorithmus zurück zur unmittelbar vorangegangenen Einheit (hier als *Backtrack* bezeichnet) und versucht einen alternativen Treffer zu finden. Ist ein solcher vorhanden, wird erneut die nächste Mustereinheit gesucht. Ist keine vorangegangene Einheit vorhanden oder wurde das Gesamtmuster gefunden, wird der Zähler für die aktuelle Position in der Sequenz um eins erhöht und alles beginnt von vorne. Die Treffer werden im oben beschriebenen Format ausgegeben, wenn sie auf das Muster passen. Der Suchprozeß wird beendet, sobald die aktuelle Sequenzposition das Ende der Sequenz erreicht hat.

Die überlappende Suche, die mit der Option `-o` eingeschaltet werden kann, wird vom Algorithmus nur für Bereiche von Nukleotiden (`n...m`) und Alternativen durchgeführt (`(ausdruck1 | ausdruck2)`). Hierbei versucht der Algorithmus alle möglichen Kombinationen der einzelnen Mustereinheiten zu finden.

Da die approximative Suche basengepaarter Bereiche (vgl. Abschnitt 3.2.8) z. T. unerwartete Ergebnisse lieferte, wird dieser Teil des Algorithmus im Detail besprochen. Der Algorithmus vergleicht den bereits gefundenen, potentiellen 5'-Sequenzteil in reverser Orientierung mit dem möglichen 3'-Sequenzteil beginnend an dessen 5'-Ende wie folgt:

- Sind in der Beschreibung **nur** Basenpaarungen oder Fehlpaarungen erlaubt?
  1. Basenpaarung machen, wenn möglich, ansonsten
  2. Fehlpaarung machen, wenn *noch* erlaubt und kein Basenpaar gemacht werden kann, ansonsten
  3. Abbruch, da nicht die nötige Anzahl von Basenpaarungen erreicht wird.
- Sind Fehlpaarungen und/oder Insertionen und/oder Deletionen erlaubt?
  Die Implementierung erfolgt hier mit Hilfe eines Stapelspeichers als Datenstruktur, in der die noch zur Verfügung stehenden Approximationen gespeichert werden. Parallel dazu werden die maximalen Anzahlen an Fehlpaarungen, Deletionen und Insertionen in separaten Variablen mitgeführt, die dann nach Verbrauch der jeweiligen Operation reduziert werden. Es wird solange eine Schleife durchlaufen, wie alternative Approximationen im Stapelspeicher enthalten sind oder bis kein Symbol mehr in der revers-komplementären Zeichenkette, dem 3'-Sequenzteil, vorhanden ist. Im Prinzip handelt es sich hierbei um eine Rekursion, was hier im Allgemeinen einen hohen Aufwand darstellt. Innerhalb der Schleife werden dann immer die folgenden Bedingungen nacheinander abgearbeitet:

1. Basenpaarung annehmen, wenn möglich;
   Abbruch, wenn kein Symbol mehr in der 5'-Zeichenkette vorhanden.
2. Fehlpaarung annehmen, wenn erlaubt;
   Aktuelle Position auf den Stapelspeicher für Insertion schieben, wenn *noch* erlaubt; oder
   Aktuelle Position auf den Stapelspeicher für Deletion schieben, wenn *noch* erlaubt;
   maximale Anzahl an erlaubten Fehlpaarungen um eins erniedrigen;
   Abbruch, wenn kein Symbol mehr in der 5' Zeichenkette vorhanden.
3. Insertion machen, wenn erlaubt;
   Aktuelle Position auf den Stapelspeicher für Deletion schieben, wenn *noch* erlaubt;
   maximale Anzahl an erlaubten Insertionen um eins erniedrigen;
   Abbruch, wenn kein Symbol mehr in der 5' Zeichenkette vorhanden.
4. Deletion machen, wenn erlaubt;
   maximale Anzahl an erlaubten Deletionen um eins erniedrigen;
   Abbruch, wenn kein Symbol mehr in der 5' Zeichenkette vorhanden.
5. Erniedrige Stapelspeicher-Zähler um eins, wenn *noch* möglich;
   versuche alternative, noch zur Verfügung stehende Approximation wie zuvor auf den Stapelspeichern abgelegt anzuwenden, wenn vorhanden.
6. Abbruch der Schleife, da kein Treffer gefunden werden konnte.

Es werden also wenn möglich immer erst Fehlpaarungen vor Insertionen und Insertionen vor Deletionen eingebaut. Aus diesem *greedy*-Ansatz folgt, dass nicht zwangsläufig die minimale Anzahl an Ausnahmen bei der Suche verwendet werden. Abbildung 3.8 skizziert diese Strategie an zwei kleinen Beispielen. Dieses Problem könnte durch Maximierung der Anzahl an Basenpaaren unter Beachtung der erlaubten Approximationen für entsprechende Subsequenzen mittels dynamischer Programmierung gelöst werden (Lück *et al.*, 1999; Nussinov *et al.*, 1978). Zudem könnten mit diesem Ansatz auch alternative, suboptimale Lösungen gefunden werden (Riks, 2001). Im Gegensatz dazu liefert die aktuell in PatScan implementierte Strategie immer nur den ersten möglichen Treffer zurück, was gleichzeitig bedeutet, dass in diesem Fall keine „überlappende" Suche stattfindet.

3.3.3 HyPaSearch

Das Programm HyPaSearch (für ***Hy****brid* ***Pa****ttern* ***Search***) wird von den Kooperationspartnern der Universitäten Hamburg und Bielefeld entwickelt und ist bis zum Ende dieser Arbeit nicht vollständig und fehlerfrei implementiert gewesen. Der dort verwendete Algorithmus wird detailliert in Strothmann (2005) beschrieben werden. Im Folgenden wird der Vollständigkeit halber die Theorie des zugrunde liegenden Algorithmus skizziert und auf die wichtigsten Kommandozeilen-Parameter und die Benutzung des Programms eingegangen.

A

```
>seq1
gggccgg GAAA ccaggaccc

r1=au,ua,gc,cg
p1=7...7 GNRA r1~p1[1,0,2]

>seq1:[1,20]
gggccgg GAAA ccaggaccc
((((.(( .... )).)..)))
(((((((( .... )).)).)))
```

B

```
>seq2
gaccagg GAAA ccggc

r1=au,ua,gc,cg
p1=7...7 GNRA r1~p1[1,2,0]

>seq2:[1,16]
gaccagg GAAA ccggc
(..(.(( .... )).))
(.((.(( .... )))))
```

Abbildung 3.8: Fehler in der PatScan-Strategie für die approximative Suche. Als Beispielsequenzen wurden `seq1` und `seq2` verwendet. Die angegebenen Muster beschreiben eine kleine Haarnadelschleife mit vier Nukleotiden im Loop (Konsensus: GNRA) und einer Helix der Länge sieben. In der Helix sind jeweils eine Fehlpaarung und in **(A)** zwei Insertionen bzw. in **(B)** zwei Deletionen erlaubt. Die gefundenen Treffer sind jeweils unter dem Muster inklusive Punkt-Klammer-Notation angegeben. In der letzten Zeile sind jeweils die Punkt-Klammer-Notationen der optimalen Lösungen mit minimaler Anzahl an verwendeten Operationen dargestellt. PatScan benötigt in beiden Fällen eine Fehlpaarung und zwei Insertionen bzw. Deletionen. Im Vergleich dazu enthalten die optimalen Lösungen nur zwei Insertionen bzw. Deletionen und dadurch ein Basenpaar mehr.

Algorithmus

Das Ziel ist es, eine Suchmaschine zu entwickeln, die effizient auf großen genomischen Datensätzen suchen kann. Dazu werden Algorithmen benötigt, die mit möglichst geringem Aufwand Sequenzdaten schnell durchsuchen können. Andererseits kann durch geschickte Wahl der Suchstrategie der Aufwand minimiert werden. Der vom Programm HyPaSearch verwendete Algorithmus kombiniert beides zu einem effizienten Suchwerkzeug.

Der entwickelte Algorithmus baut auf den Index-Datenstrukturen auf, die bereits in Abschnitt 3.3.1 vorgestellt wurden. Da große genomische Sequenzdaten durchsucht werden sollen und Suffix-Bäume einen sehr hohen Bedarf an Speicherplatz haben, werden die wesentlich speichereffizienteren, sog. erweiterten Suffix-Arrays (Abouelhoda *et al.*, 2002b) für die Indizierung der Sequenzdaten verwendet. Abouelhoda *et al.* (2004) konnten zeigen, dass Suffix-Baum-basierte Algorithmen auf sog. erweiterten Suffix-Arrays mit etwa 60 % weniger Speicherbedarf und reduzierter Laufzeit simuliert werden können. In Suffix-Arrays können Muster allerdings nur in einer Richtung gesucht werden. Um bei der Suche jederzeit die Suchrichtung wechseln zu können, wurden Suffix-Arrays für die Vorwärts- und Rückwärtsrichtung miteinander kombiniert. Dazu werden zusätzliche Informationen gespeichert, die korrespondierende Teile der Arrays miteinander verbinden. Diese erweiterten Datenstrukturen werden als Affix-Arrays bezeichnet und bilden die Grundlage des hier entwickelten Algorithmus (Strothmann, 2005). Durch die Vorverarbeitung der Sequenzdaten ist somit die Suchzeit nur noch abhängig von dem zu suchenden Muster.

Um eine geschickte Suchstrategie zu ermöglichen, wird das zu suchende Muster in nicht weiter zerlegbare Teil-Muster aufgespalten. Für diese können dann spezielle Algorithmen, sog. atomare Matcher, implentiert werden, die explizit für die Suche der Teil-Muster optimiert sind. Die meisten der atomaren Matcher basieren auf den verwendeten Index-Datenstrukturen zur Suche von sequenziellen Mustern, wie z. B. regulären Ausdrücken oder Sequenz-Profilen (vgl. Abschnitt 3.2ff). Für die Suche mit Sequenz-Profilen wurde beispielsweise das Programm PoSSuMsearch entwickelt (Beckstette *et al.*, 2004), dessen Funktionalität auch in HyPaSearch integriert ist. Die Zerlegung des Gesamt-Musters in Teil-Muster bringt einen weiteren Vorteil für die Suchstrategie mit sich. Durch die Berechnung der Signifikanz der einzelnen Teilmuster kann die globale Suche des Gesamt-Musters optimiert werden, in dem der Algorithmus mit der Suche des signifikantesten Teil-Musters beginnt. Da die atomaren Matcher auf Affix-Arrays basieren und somit nicht nur in Vorwärtsrichtung gesucht werden kann, wird diese Strategie überhaupt erst ermöglicht. Ein Beispiel dafür ist der integrierte Algorithmus für die Suche von doppelsträngigen Bereichen. Angenommen ein Muster beschreibt eine Stem-Loop-Struktur mit einer bestimmten Anzahl an beliebigen Basenpaaren in der Helix und einer konservierten Sequenz im Loop. Durch die Signifikanzanalyse des Musters sucht der Algorithmus zunächst nach der Loop-Sequenz, wodurch der Suchraum drastisch eingeschränkt wird. Der zugrunde liegende Affix-Array erlaubt nun zur Suche des basengepaarten Bereichs das Hin- und Herspringen zwischen den 5'- und 3'-Teilen, die die Loopsequenz flankieren. Beginnend mit dem nächsten Nukleotid hinter dem 3'-Ende der Loop-Sequenz, wird dieses mit dem Nukleotid vor dem 5'-Ende der Loop-Sequenz verglichen. Können diese beiden Nukleotide ein Basenpaar bilden, geht der Algorithmus ein weiteres Nukleotid in der Sequenz in 5'-Richtung vorne und vergleicht dieses wiederum mit dem nächsten Nukleotid im 3'-Teil. Diese Prozedur wird solange wiederholt, bis die im Muster geforderten Bedingungen für den basengepaarten Bereich erfüllt sind. Durch die geschickte Wahl der Suchreihenfolge und die Möglichkeit zur bidirektionellen Suche, können also schnell und effizient komplexe Beschreibungen verarbeitet werden. Wieder andere atomare Matcher wurden implementiert, um dem Benutzer die Möglichkeit zu geben definierte Bewertungsfunktionen (z. B. Nukleotidgehalt oder thermodynamische Stabilität) auf gefundene Treffer anzuwenden, durch die deutlich spezifischere Ergebnisse erhalten werden können.

Neben den indexbasierten atomaren Matchern gibt es aber auch immer einen entsprechenden sequenziellen Matcher, die direkt auf der Sequenz arbeiten. Diese sind in der Regel einfacher zu implementieren und werden zur Überprüfung der atomaren Matcher benutzt.

Demnach ergibt sich die folgende Reihenfolge für die Vorgehensweise bei der Suche von hybriden Mustern in großen Sequenzdaten mit HyPaSearch:

- Laden der zu einem Affix-Array vorverarbeiteten Sequenzdaten
- Zerlegung des Musters in nicht weiter zerlegbare Teil-Muster
- Signifikanzanalyse der Teil-Muster zur Bestimmung der Suchreihenfolge
- Mustersuche, ggf. Anwendung von Bewertungsfunktionen und Ausgabe der Treffer inkl. zusätzlicher Informationen

Die Analyse der Effizienz der Suche mit HyPaSearch im Vergleich zu anderen Suchwerkzeugen erfolgt zur Zeit im Rahmen der Diplomarbeit von Zanger (2005). Erste Ergebnisse

```
-q              specify name of the query file
-index          specify name of the indexed database file
-search         specify names of search patterns
-o              specify name of the output file (if this option is
                not set no output in matchfile-format is exported)
-so             specify name of the output file
                do not report further match information
-fwd            search the forward strand
                (only for DNA/RNA patterns and DNA/RNA databases).
                If option rev is not set this option is default.
-rev            search the reverse complementary strand
                (only for DNA/RNA patterns and DNA/RNA databases)
-nopos          no output of the match positions
-noseq          no output of identical instances
                of the same matching sequence.
-seqnum         extra output of the sequence identity number
-absolute       show absolute instead of relative match positions
-nodesc         no output of sequence description
-fulldesc       output of the complete description
-nohitcount     do not count the hits and do not report the title
                matchstring
-nodetails      only output matching strings once
                without all matching positions
-onlyhitnumber  only report the number of matches
-noscore        do not report the score of a match
-allowduplicates allow for duplicate matches (same start position
                and length may yield different matches)
-showbest       allow for k best duplicates(default is 1)
-novalues       do not output the values of variable expressions of
                matching constraints
-showprotein    (only for DNA/RNA database) show protein sequence
                (first reading frame starts at first match position)
-showleft       specify number of characters that are shown
                on the left side of a match
-showright      specify number of characters that are shown
                on the right side of a match
-cutoff         terminate the program if the expected number of hits
                exceeds the cutoff value
-desclen        limit the output of the sequence description
                to the specified number
-columns        set the number of columns (for output) to the
                specified number (standard (80)
-showvariables  additionallay output variable bindings
-showformat     additionally format the output according to the
                pattern definition
-delimiter      choose delimiter symbol.
                If no delimiter is chosen match informations are
                separated by new lines
-showcomplement show the reverse complement of the matching sequences
                (incl. additional left and right pieces of
                showleft/showright options and variable bindings)
-notopt         do not optimze the search order
-online         check the results with a test-algorithm
-verbose        verbose mode
-help           this option
```

Abbildung 3.9: Ausgabe der Hilfe zum Suchwerkzeug HyPaSearch**.** Diese Ausgabe wird beim Aufruf des Suchprogramms `hypa` mit der Option `-help` angezeigt und liefert alle Informationen zu den verfügbaren Kommandozeilen-Parametern des Programms.

zum Laufverhalten von HyPaSearch im Vergleich zu PatScan werden in Abschnitt 3.3.4 dagestellt.

Vorverarbeitung der Sequenzdaten

Die Sequenzdaten müssen aufgrund des verwendeten Algorithmus, der auf der Basis der bereits beschriebenen Affix-Arrays implementiert ist, vorverarbeitet werden. Diese Vorverarbeitung wird mit dem Programm `mkaffix` durchgeführt, welches selber die Programme `mkvtree`, `mkcfr` und `mkstatistics` für die Indizierung der Daten benutzt (siehe Strothmann, 2005). Die zu verarbeitenden Sequenzdaten können sowohl komprimiert als auch unkomprimiert vorliegen und müssen entweder EMBL-, Genbank oder FASTA-Format haben. Die allgemeine Syntax von `mkaffix` für die Indizierung von Sequenzdaten ist:

```
mkaffix <file> <index> -dna
```

Dabei wird mit `<file>` die einzulesende Sequenzdatei und mit `<index>` der Pfad und Präfix, den die zu erstellenden Index-Dateien tragen sollen, angegeben. Mit `-dna` wird dem Programm mitgeteilt, dass es sich bei den angegebenen Sequenzdaten um Nukleinsäuresequenzen handelt. Der Index, der erstellt wird, besteht aus mehreren verschiedenen Dateien, die insgesamt etwa einen um den Faktor 30 höheren Speicherplatz benötigen als die ursprüngliche Sequenzdatei im FASTA-Format.

Kommandozeilen-Parameter und Benutzung des Suchprogramms `hypa`

Die Option `-help` liefert die in Abbildung 3.9 angegebene Ausgabe der Hilfe zur Benutzung des Programms. Im Folgenden werden nur die wichtigsten Optionen kurz erläutert.

Für die Suche über indizierte Sequenzdaten müssen immer die Optionen `-q` und `-index` angegeben werden. Mit `-index` gefolgt von dem Pfad und Präfix zum Sequenzdatenindex, der mit `mkaffix` erstellt wurde, wird definiert, in welchen Sequenzdaten gesucht werden soll. Die Option `-q` gefolgt von einem Dateinamen spezifiziert die Musterdatei. Da bei HyPaSearch in einer Beschreibungsdatei mehrere Muster gleichzeitig enthalten sein dürfen, kann der Benutzer mit der Option `-search` gefolgt von einem oder mehreren Musternamen angeben, welches bzw. welche Muster gesucht werden sollen. Die Angabe mehrerer Muster erfolgt durch Leerzeichen getrennt ohne Kommas. Die Ergebnisse werden normalerweiser in die Standard-Ausgabe geschrieben, es sei denn der Benutzer gibt die Option `-o` gefolgt von dem Namen der Datei an, in der die Ergebnisse gespeichert werden sollen. Standardmäßig wird auf dem Vorwärts-Strang gesucht. Durch Verwendung der Optionen `-fwd` bzw. `-rev` können wahlweise nur der eine oder andere Strang oder beide Stränge durchsucht werden. Die meisten Optionen haben nur Einfluss auf die Formatierung der Ausgabe und können in der Hilfe zum Programm oder der zum Programm gehörenden Dokumentation nachgelesen werden. Um eine FASTA-ähnliche Repräsentation der Ergebnisse zu erzeugen, wie es bei PatScan der Fall ist (vgl. Abschnitt 3.3.2), müssen die Optionen `-showformat` und `-nohitcount` verwendet werden, vorausgesetzt es werden Variablen für die Musterbeschreibung verwendet (vgl. Abschnitt 3.2.5).

3.3.4 Vergleich der Laufzeiten

Ein Ziel der Entwicklung des neuen Suchwerkzeuges HyPaSearch ist es die Beschleunigung der Suche gegenüber den herkömmlichen Suchwerkzeugen. Daher wurden die in dieser Arbeit verwendeten Werkzeuge diesbezüglich miteinander verglichen. Durchgeführt wurden die Tests auf einem Computer mit AMD Athlon(TM) XP 2400+ Prozessor (1,9 GHz) und 768 MB Arbeitsspeicher (RAM). Als Test-Sequenzdatensatz diente das Chromosom IV von *Arabidopsis thaliana* mit $1{,}86 \times 10^7$ Nukleotiden. Die ausgewählten Muster wurden jeweils 100 Mal hintereinander auf Chromosom IV gesucht und die dafür benötigte Zeit wurde gemessen. Zudem wurde die Anzahl gefundenen Treffer pro Muster bestimmt und die Treffer verglichen, um zu überprüfen, ob beide Suchwerkzeuge das gleiche Ergebnis liefern. Tabelle 3.5 fasst diese Ergebnisse im Überblick zusammen.

Der Laufzeit-Vergleich mit PatScan zeigt den erwarteten, z. T. erheblichen Geschwindigkeitsgewinn bei der Suche mit HyPaSearch aufgrund der verwendeten Index-Datenstruktur und der optimierten Suchreihenfolge. Die erzielte Beschleunigung variiert dennoch stark mit der Art des gesuchten Musters. Das Paradebeispiel ist die Suche mit Profil-Matrizen (`profile`), die etwa 100-mal schneller mit HyPaSearch durchgeführt werden kann. Der Grund dafür liegt in dem eigens dafür entwickelten atomaren Matcher (Beckstette *et al.*, 2004). Dagegen ist die Suche mit sequenziellen Mustern (`sequence1` und `sequence2`) nur geringfügig schneller als mit PatScan. Dennoch ergibt sich auch hier schon ein leichter Vorteil durch die optimierte Suchstrategie, der sich im Fall des Musters `sequence1` weniger deutlich zeigt, da PatScan aufgrund der konservierten Sequenz am 5'-Ende ebenfalls als erstes den signifikantesten Teil des Musters sucht. Das Muster `range_ds` beschreibt einen langen Hairpin-Loop mit 29 bis 30 beliebigen Basenpaaren

Tabelle 3.5: Laufzeit-Vergleich von verschiedenen Mustern zwischen HyPaSearch und PatScan. Durchgeführt wurden die Tests auf einem Computer mit AMD Athlon(TM) XP 2400+ Prozessor (2,0 GHz) und 768 MB Arbeitsspeicher. Als Test-Sequenzdatensatz diente das Chromosom IV von *Arabidopsis thaliana* mit $1{,}86 \times 10^7$ Nukleotiden. Die gemessenen Zeiten gelten für eine 100 malige Suche der angegebenen Muster auf Chromosom IV. Die Musterbeschreibungen sind in HyPaL-Notation angegebenen; äquivalente Muster wurden für die Suche mit PatScan definiert.

HyPaSearch [a]		PatScan		Faktor
Zeit [s]	Treffer [#]	Zeit [s]	Treffer [#]	
`sequence1=CUGANGAW NNNNNNN GAAA;`				
110,6	58	159,94	58	1,45
`sequence2=N RY NNNN CUGANGAG NNN RY NNN;`				
111,41	130	456,39	130	4,10
`range_ds=(stem:=.{29,30}) .{4} ^(stem);`				
2683,39	49	2711,46	49	1,01
`uncg_r=(stem:=.{7} R) (loop:=UNCG) ^(stem);`				
115,18	89	354,23	89	3,08
`approx_ss_rep=AAAAAAAAA[1,0,0] CUGANGAW;`				
5,59	13	341,24	13	61,04
`approx_ss_del=AAAAAAAAA[0,1,0] CUGANGAW;`				
4,45	16	326,83	12	73,44
`approx_ss_ins=AAAAAAAAA[0,0,1] CUGANGAW;`				
5,55	14	367,12	9	66,15
`approx_ss=AAAAAAAAA[1,1,1] CUGANGAWGA;`				
130,53	15	1407,14	9	10,78
`approx_ds_mis=(stem:=G{7}) .{4} ^stem[1,0,0];`				
4,73	29	156,44	29	33,07
`approx_ds_del=(stem:=G{7}) .{4} ^stem[0,1,0];`				
4,63	13	156,6	9	33,82
`approx_ds_ins=(stem:=G{7}) .{4} ^stem[0,0,1];`				
5	26	155,86	22	31,17
`approx_ds=(stem:=G{12}) .{4} ^stem[1,1,1];`				
7,43	10	155,77	6	20,97
`profile` [b]				
4,46	67	572,99	67	128,47
`hhrz_III_gc` [c]				
182,35	4	10826,43	12	59,37

[a] Der Laufzeit-Vergleich wurde mit der Version vom 19.11.2004 durchgeführt.

[b]
```
profile=[[1,2,3,4],[1,2,3,4],[4,2,3,1],[1,4,3,1],[1,2,3,4],
         [1,2,3,4],[4,2,3,1],[1,4,3,1],[4,2,3,1],[1,4,3,1]]>=40;
```

[c]
```
hhrz_III_gc=(stem3:=.{4,5}) U (csite:=H) (stem1:=.{5,7}) .{3,13} ^(stem1)[1,0,0]
            G (stem2:=.{3,4}) .{3,6} ^(stem2) C GAA A ^(stem3);
```

in der Helix und 4 beliebigen Nukleotiden im Loop. In diesem Fall hat die HyPaSearch-Suchstrategie keinen Einfluss auf die Laufzeit, da im Muster kein signifikanter Teil enthalten ist, der zuerst gesucht werden könnte. Im Vergleich dazu ist die Suche mit dem Tetraloop-Muster (`uncg_r`) wiederum leicht beschleunigt, da aufgrund der verwendeten

```
>NC_003075:[15027889,15027835]
CTGG T C GTGAT CTGAAACTCG ATCAC CTGATGA G CTC AAGGCA GAG C GAA A CCAG
ctgg t c gtgat ctgaaactcg atcac ctgatga g ctc aaggca gag c gaa a ccag
CTGG T C GTGATC TGAAACTC GATCAC CTGATGA G CTC AAGGCA GAG C GAA A CCAG
CTGG T C GTGATCT GAAACT CGATCAC CTGATGA G CTC AAGGCA GAG C GAA A CCAG

>NC_003075:[15027890,15027834]
ACTGG T C GTGAT CTGAAACTCG ATCAC CTGATGA G CTC AAGGCA GAG C GAA A CCAGT
actgg t c gtgat ctgaaactcg atcac ctgatga g ctc aaggca gag c gaa a ccagt
ACTGG T C GTGATC TGAAACTC GATCAC CTGATGA G CTC AAGGCA GAG C GAA A CCAGT
ACTGG T C GTGATCT GAAACT CGATCAC CTGATGA G CTC AAGGCA GAG C GAA A CCAGT

>NC_003075:[15032902,15032848]
CTGG T C GTGAT CTGAAACCCG ATCAC CTGATGA G CTC AAGGTA GAG C GAA A CCAG
ctgg t c gtgat ctgaaacccg atcac ctgatga g ctc aaggta gag c gaa a ccag
CTGG T C GTGATC TGAAACCC GATCAC CTGATGA G CTC AAGGTA GAG C GAA A CCAG
CTGG T C GTGATCT GAAACC CGATCAC CTGATGA G CTC AAGGTA GAG C GAA A CCAG

>NC_003075:[15032903,15032847]
ACTGG T C GTGAT CTGAAACCCG ATCAC CTGATGA G CTC AAGGTA GAG C GAA A CCAGT
actgg t c gtgat ctgaaacccg atcac ctgatga g ctc aaggta gag c gaa a ccagt
ACTGG T C GTGATC TGAAACCC GATCAC CTGATGA G CTC AAGGTA GAG C GAA A CCAGT
ACTGG T C GTGATCT GAAACC CGATCAC CTGATGA G CTC AAGGTA GAG C GAA A CCAGT
```

Abbildung 3.10: Auflistung der mit dem Muster `hhrz_III_gc` auf Chromosom IV von *Arabidopsis thaliana* mit HyPaSearch und PatScan gefundenen Treffer. Angegeben sind jeweils die Start- und Endpositionen auf Chromosom IV (`NC_003075`) in den Zeilen, die mit dem „>“-Zeichen beginnen. HyPaSearch-Treffersequenzen sind in Kleinbustaben geschrieben, PatScan-Treffer in Großbuchstaben. Die Basenaustausche sind schwarz markiert. Die unterschiedlich getroffenen Teilsequenzen sind dunkelgrau hervorgehoben.

Affix-Array-Datenstruktur erst die Loopsequenz und dann die Helix-Basenpaare gesucht werden können. Ein erheblicher Gewinn an Geschwindigkeit ist bei der approximativen Suche zu verzeichnen, was sich mit dem hohen Aufwand durch die rekursive Implementierung im PatScan-Algorithmus erklären lässt. Zudem findet HyPaSearch bei der Suche von approximativen Mustern, in denen Deletionen und/oder Insertionen erlaubt sind, mehr Treffer als PatScan, was sich mit dem von PatScan verwendeten Algorithmus begründen lässt (vgl. Abschnitt 3.3.2).

Das Muster `hhrz_III_gc` beschreibt Hammerhead-Ribozyme, auf die in Abschnitt 3.5.2 noch detailliert eingegangen wird. In der Beschreibung enthalten sind konservierte Sequenzen, reguläre Ausdrücke und basengepaarte Bereiche mit und ohne Approximationen. Der von HyPaSearch verwendete Algorithmus führt hier ebenfalls zu einer erheblichen Beschleunigung der Suche, findet allerdings 8 Treffer weniger als PatScan. In Abbildung 3.10 sind die gefundenen Treffer aufgelistet. Insgesamt handelt es sich bei den 4 bzw. 12 Treffern um zwei verschiedene Treffer, die nur durch zwei Basenaustausche unterscheiden.

Die 12 von PatScan gefundenen Treffer resultieren aus der überlappenden Suche. In der Musterbeschreibung (siehe Fußnote in Tabelle 3.5) sind in `stem3` 4 oder 5 Basenpaare erlaubt, sodass dieser Teil des Musters in der Sequenz zweimal passt. In `stem1` werden 5 bis 7 Basenpaare verlangt, wobei eine Fehlpaarung erlaubt ist, was in der Sequenz dreimal gefunden werden kann. Zusammen mit den zwei Basenaustauschen ergeben sich daraus 12 Kombinationen, die alle von PatScan gefunden werden. Da HyPaSearch vor der Ausgabe redundante Treffer herausfiltert, die identische Start- und Endpositionen besitzen, werden nur 4 Treffer ausgegeben. Dabei handelt es sich allerdings nicht um die optimale Lösung. Wünschenswert wäre es, wenn bei der Filterung zusätzlich die Anzahl an Basenpaaren berücksichtigt würde, sodass die Treffer mit den meisten Basenpaaren ausgegeben würden.

Insgesamt konnte mit diesem ersten Laufzeit-Vergleich demonstriert werden, dass HyPaSearch aufgrund des Affix-Array-basierten Algorithmus und der optimierten Suchstrategie den erwarteten Geschwindigkeitsgewinn gegenüber PatScan zeigt.

3.3.5 Statistik

Für die Auswertung und Beurteilung einer durchgeführten Suche ist von Interesse, wieviele Treffer rein statistisch zu erwarten sind. Durch den Vergleich der Anzahl an zu erwartenden Treffern mit der Anzahl an tatsächlich gefundenen Treffern lassen sich Aussagen über die Signifikanz eines Suchergebnisses machen. Werden beispielsweise bei einer Suche in der gleichen Größenordnung Treffer wie erwartet gefunden, kann man davon ausgehen, dass diese rein statistisch verteilt vorkommen und damit sehr wahrscheinlich nicht von biologischer Relevanz sind. Findet man dagegen eine Anzahl von Treffern weit über oder unter dem Erwartungswert, handelt es sich dabei in aller Regel nicht um Zufallstreffer. Da im Rahmen dieser Arbeit fast ausschließlich PatScan für die Suche mit struktur-basierten Mustern eingesetzt wurde, soll im Folgenden abgeschätzt werden, wie hoch die Anzahl der zu erwartenden Treffer für ein beliebiges PatScan-Muster ist. Diese Berechungen lassen sich aber teilweise auch auf hybride Muster übertragen. Am Ende dieses Abschnitts werden die gemachten Abschätzungen an einigen Beispielen überprüft.

Abschätzung der Anzahl an zu erwartenden Treffern

Die Berechnung soll für die überlappende Suche (vgl. Abschnitt 3.3.2) erfolgen. In die Berechnung fließt die jeweilige Nukleotidzusammensetzung der zu durchsuchenden Sequenz mit ein. Zudem sollen sämtliche Kombinationen oder Permutationen von approximativen Mustern berücksichtigt werden.

Angenommen wird eine zu durchsuchende Sequenz S, die auf einem definierten, endlichen Alphabet $\mathcal{A}$ mit n Buchstaben basiert. Für DNA gilt beispielsweise:

$$\mathcal{A} = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\};\ n = 4$$

Die Nukleotidfrequenz f_B einer beliebigen Base B in einer Sequenz S der Länge l ist definiert als der Quotient aus der Anzahl an Nukleotiden der Base B durch die Länge der Sequenz,

$$f_B = \frac{n_B}{l}, \tag{3.1}$$

was der Wahrscheinlichkeit p_{B_i} für das Auftreten eines Nukleotids B an Position i in S entspricht, vorrausgesetzt das Auftreten eines Nukleotids wird als unabhängiges Ereignis betrachtet. Sind die Nukleotide einer Sequenz gleichverteilt gilt:

$$f_{\mathrm{A}} = f_{\mathrm{C}} = f_{\mathrm{G}} = f_{\mathrm{U}} = \frac{1}{4}$$

Allgemein ergibt sich die Gesamtwahrscheinlichkeit für gemeinsam auftretende Ereignisse aus dem Produkt der Einzelwahrscheinlichkeiten der Ereignisse:

$$p_{\mathrm{X}_1 \ldots \mathrm{X}_n} = \prod_{i=1}^{n} p_{\mathrm{X}_i} \tag{3.2}$$

Für die Gesamtwahrscheinlichkeit alternativ auftretender Ereignisse gilt:

$$p_{\mathrm{X}_1 \ldots \mathrm{X}_n} = \sum_{i=1}^{n} p_{\mathrm{X}_i} \tag{3.3}$$

Danach ergibt sich die Gesamtwahrscheinlichkeit für n einzelsträngige Nukleotide:

$$p_{\mathrm{B}_1 \ldots \mathrm{B}_n} = \prod_{i=1}^{n} p_{\mathrm{B}_i}$$

Wird für die Beschreibung einzelsträngiger Bereiche die IUPAC-Nomenklatur (vgl. Tabelle 3.2) verwendet, müssen die entsprechenden Alternativen berücksichtigt werden. Exemplarisch soll die zu erwartende Trefferanzahl für die Nukleotidsequenz HUMAN (H: A,C oder U; H: A oder C; N: A, C, G oder U) unter der Voraussetzung, dass die Nukleotide gleichverteilt in der Sequenz vorkommen, berechnet werden.

$$\begin{aligned} p_{\mathrm{HUMAN}} &= p_{\mathrm{H}} \cdot p_{\mathrm{U}} \cdot p_{\mathrm{M}} \cdot p_{\mathrm{A}} \cdot p_{\mathrm{N}} \\ &= (p_{\mathrm{A}} + p_{\mathrm{C}} + p_{\mathrm{U}}) \cdot p_{\mathrm{U}} \cdot (p_{\mathrm{A}} + p_{\mathrm{C}}) \cdot p_{\mathrm{A}} \cdot (p_{\mathrm{A}} + p_{\mathrm{C}} + p_{\mathrm{G}} + p_{\mathrm{U}}) \\ &= \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} \cdot \frac{1}{4} \cdot 1 = \frac{3}{128} \end{aligned}$$

Es ist also etwa alle 43 Nukleotide ein Treffer zu erwarten.

Die Wahrscheinlichkeit für das Auftreten eines Basenpaares Bp_{ij} zwischen den Positionen i und j lässt sich berechnen nach

$$p_{\mathrm{Bp}_{ij}} = p_{\mathrm{B}_i} \cdot p_{\mathrm{B}_j}, \tag{3.4}$$

da das Auftreten der Base B_i an Position i unabhängig von dem Auftreten der Base B_j an Position j ist.

	A	C	G	U
A				wc
C			wc	
G		wc		wo
U	wc		wo	

Tabelle 3.6: Basenpaar-Kombinationen. Auflistung aller Watson-Crick- (wc) und Wobble- (wo) Basenpaare.

Auch hierbei müssen bei Verwendung der IUPAC-Nomenklatur die Alternativen beachtet werden. Zudem fließt die angegebene Basenpaar-Regel mit in die Berechnung ein. Tabelle 3.6 fasst alle möglichen Watson-Crick- und Wobble-Basenpaare zusammen. Die Berechnung zu erwartender Treffer einer Helix mit Konsensus HUMAN nach Standard-RNA-Basenpaar-Regeln (`r1=au,ua,gc,cg,gu,ug`), d.h. Watson-Crick- und Wobble-Basenpaare sind erlaubt, ergibt

$$\begin{aligned} p_{\texttt{p1=HUMAN}\cdots\texttt{r1}\sim\texttt{p1}} &= p_{\mathrm{BP_H}} \cdot p_{\mathrm{BP_U}} \cdot p_{\mathrm{BP_M}} \cdot p_{\mathrm{BP_A}} \cdot p_{\mathrm{BP_N}} \\ &= (p_{\mathrm{AU}} + p_{\mathrm{CG}} + p_{\mathrm{UG}} + p_{\mathrm{UA}}) \cdot (p_{\mathrm{UG}} + p_{\mathrm{UA}}) \cdot (p_{\mathrm{AU}} + p_{\mathrm{CG}}) \cdot p_{\mathrm{AU}} \\ &\quad \cdot (p_{\mathrm{AU}} + p_{\mathrm{UA}} + p_{\mathrm{GC}} + p_{\mathrm{CG}} + p_{\mathrm{GU}} + p_{\mathrm{UG}}) \\ &= \frac{1}{4} \cdot \frac{1}{8} \cdot \frac{1}{8} \cdot \frac{1}{16} \cdot \frac{3}{8} = \frac{3}{32768}. \end{aligned}$$

Es ist also etwa alle 10923 Nukleotide ein Treffer zu erwarten.

Dürfen Muster-Einheiten alternativ auftreten (`(X | Y)`), werden die Einzelwahrscheinlichkeiten addiert.

In einzelsträngigen Bereichen, die mit Hilfe der IUPAC-Nomenklatur definiert werden, und in doppelsträngigen Bereichen können Ausnahmen (Approximationen) angegeben werden (vgl. Abschnitt 3.2.8). Sind Ersetzungen bzw. Fehlpaarungen, Deletionen oder Insertionen erlaubt, müssen die Einzelwahrscheinlichkeiten sämtlicher Alternativen, die sich durch die angegeben Ausnahmen ergeben, aufsummiert werden, da es sich jeweils um verschiedene Lösungen handelt. Dabei müssen zwei Fälle unterschieden werden:

- ist nur eine Art der Approximation angegeben oder
- wird eine Kombination von Ausnahmen erlaubt.

Im ersten Fall handelt es sich im kombinatorischen Sinne um die Auswahl von k aus n Elementen ohne Wiederholungen und ohne Berücksichtigung der Reihenfolge. Die Anzahl der zu betrachtenden Möglichkeiten läßt sich somit mit Hilfe des Binomialkoeffizienten berechnen. Danach ergeben sich

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{3.5}$$

Möglichkeiten, wobei n die Länge der betrachteten Sequenz bzw. Helix und k die Anzahl der erlaubten Ersetzungen bzw. Fehlpaarungen oder Deletionen sind. Für Insertionen müssen i Insertionen auf insgesamt n Basenpaare und maximal i Insertionen verteilt werden; also ergibt sich

$$\binom{n+i}{i} = \frac{(n+i)!}{i!(n+i-i)!} = \frac{(n+i)!}{i!\ n!} \tag{3.6}$$

Möglichkeiten, wobei i die Anzahl der erlaubten Insertionen ist. Da im PatScan-Algorithmus keine Insertionen am 3' Ende des reversen Komplements vorgesehen sind, muss der Term für die Berechnung der Kombinationen für Insertionen auf $\binom{n-1+i}{i}$ korrigiert werden.

Sind im Gegensatz dazu beliebige Kombinationen verschiedener Typen von Ausnahmen erlaubt, spielt die Reihenfolge, in der diese Vorkommen können eine Rolle. Im kombinatorischen Sinne handelt es sich hierbei um die Auswahl von k aus n Elementen ohne Wiederholungen, aber mit Berücksichtigung der Reihenfolge. Dementsprechend ergeben sich

$$\frac{(n+i)!}{(n-k)!} \tag{3.7}$$

Möglichkeiten, wobei n die Länge der betrachteten Sequenz bzw. Helix, i die Anzahl an erlaubten Insertionen und k die Summe der Anzahlen aller erlaubten Ausnahmen ist.

Da es sich bei der Angabe von Approximationen jeweils um die Anzahl an maximal erlaubten Ausnahmen handelt, müssen bei der Berechnung die Kombinationen für alle Werte, die kleiner oder gleich dem angegebenen Maximalwert sind, berücksichtigt werden. Sind beispielsweise zwei Fehlpaarungen und eine Deletion (`[2,1,0]`) erlaubt, so gehen alle Kombinationen für `[0,0,0]`, `[1,0,0]`, `[2,0,0]`, `[0,1,0]` , `[1,1,0]` und `[2,1,0]` mit in die Berechnung ein. Abbildung 3.11 zeigt exemplarisch die Berechnung für das Muster `ADAM[2,0,0]` bei Gleichverteilung der Nukleotide in der zu durchsuchenden Sequenz.

Überprüfung der statistischen Abschätzung

Um die angegebenen Abschätzungen zu überprüfen wurden 10 Zufallssequenzen mit je 10^7 Nukleotiden generiert, wobei die einzelnen Nukleotide gleichverteilt vorliegen. Diese Sequenzen wurden dann mit ausgewählten Mustern überlappend durchsucht, der Mittelwert und die Standardabweichung berechnet und mit dem abgeschätzten Erwartungswert für das Muster verglichen. In Mustern mit basengepaarten Bereichen wurden immer die Standard RNA-Basenpaar-Regeln benutzt (`r1={au,ua,gc,cg,gu,ug}`).

Abbildung 3.12 fasst das Ergebnis zusammen. Die abgeschätzten Erwartungswerte für die Muster 1 bis 5 stimmen mit der Anzahl an gefundenen Treffern überein. Mit den Mustern 6 und 7 findet PatScan in beiden Fällen deutlich weniger Treffer als abgeschätzt wurden. Diese Muster beschreiben Tetraloop-Hairpin-Strukturen mit 3 bis 4 beliebigen Basenpaaren in der Helix, wobei eine Deletion (Muster 6) bzw. Insertion (Muster 7) erlaubt ist. Daraufhin wurden äquivalente Muster mit HyPaL formuliert und ebenfalls gesucht. Die Anzahl der mit HyPaSearch gefundenen Treffer lag in beiden Fällen erheblich über der abgeschätzten Trefferanzahl (Muster 6: 163152 Treffer, Muster 7 75430 Treffer). Der Vergleich der Treffer zeigte, dass PatScan zum einen grundsätzlich keine Insertionen am 3'-Ende des revers-komplementären Sequenzteils findet. Andererseits vergleicht PatScan bei der approximativen Suche nur solange die Teilstrings miteinander, bis die erste Möglichkeit gefunden ist. Wird eine Übereinstimmung ohne Verwendung einer Ausnahme gefunden, wird auch bei der überlappenden Suche nicht nach einem weiteren, alternativen Treffer gesucht (vgl. Abschnitt 3.3.2 und Abbildung 3.8). Im Gegensatz dazu

ADAM $(m = 0)$		$\Sigma = \frac{3}{128}$
ADAM		$\frac{1}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{3}{128}$
ADAM $(m = 1)$		$\Sigma = \frac{22}{128}$
xDAM	B...	$\frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{9}{128}$
AxAM	.C..	$\frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{128}$
ADxM	..B.	$\frac{1}{4} \cdot \frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{2} = \frac{9}{128}$
ADAx	...K	$\frac{1}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{3}{128}$

ADAM $(m = 2)$		$\Sigma = \frac{52}{128}$
xxAM	BC..	$\frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{3}{128}$
xDxM	B.B.	$\frac{3}{4} \cdot \frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{2} = \frac{27}{128}$
xDAx	B..K	$\frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{9}{128}$
AxxM	.CB.	$\frac{1}{4} \cdot \frac{1}{4} \cdot \frac{3}{4} \cdot \frac{1}{2} = \frac{3}{128}$
AxAx	.C.K	$\frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{4} \cdot \frac{1}{2} = \frac{1}{128}$
ADxx	..BK	$\frac{1}{4} \cdot \frac{3}{4} \cdot \frac{3}{4} \cdot \frac{1}{2} = \frac{9}{128}$

$$
\begin{aligned}
p_{\texttt{ADAM[2,0,0]}} &= p_{\texttt{ADAM}} + p_{\texttt{BDAM}} + p_{\texttt{ACAM}} + \cdots + p_{\texttt{ACAK}} + p_{\texttt{ADBK}} \\
&= p_{\texttt{ADAM}\,(m=0)} + p_{\texttt{ADAM}\,(m=1)} + p_{\texttt{ADAM}\,(\texttt{m=2})} \\
&= \frac{3}{128} + \frac{22}{128} + \frac{52}{128} = \frac{77}{128}
\end{aligned}
$$

Abbildung 3.11: Abschätzung der Anzahl an zu erwartenden Treffern für das Muster `ADAM[2,0,0]`. Das Muster beschreibt eine Sequenz aus vier Nukleotiden unter Verwendung der IUPAC-Nomenklatur. Dabei sind bis zu zwei Austausche erlaubt sind. Die Abschätzung erfolgt für Sequenzen, in denen die Nukleotide gleichverteilt vorkommen. Die Anzahl an zu berücksichtigenden Kombinationen wird durch den Binomialkoeffizienten $\binom{n}{m}$, mit $n = 4$ (Länge der Sequenz) und $m = \{0, 1, 2\}$ (Anzahl an erlaubten *mismatches*) bestimmt (`ADAM` $(m = 0)$: 1 Mögl., `ADAM` $(m = 1)$: 4 Mögl., `ADAM` $(m = 0)$: 6 Mögl.). In der ersten Spalte sind die Alternativen aufgelistet, wobei `x` jeweils die *mismatch*-Positonen markiert. In der zweiten Spalte sind die Nukleotide angegeben, die an der entsprechenden *mismatch*-Positionen erlaubt sind (`B` (nicht `A`), `K` (nicht `M`; `G` oder `T`)). Die Punkte symbolisieren die ursprüngliche Sequenz. Die dritte Spalte enthält die Produkte der Einzel-Wahrscheinlichkeiten der jeweiligen Alternative, die in der Summe die Gesamt-Wahrscheinlichkeit für das Auftreten einer Sequenz, die auf das Muster passt, ergeben. Die Formel fasst den Rechenweg noch einmal zusammen. Daraus ergibt sich, dass das Muster `ADAM[0,0,0]` etwa alle 43 nt, das Muster `ADAM[1,0,0]` etwa alle 5 nt und das Muster `ADAM[2,0,0]` etwa alle 2 nt vorkommt.

findet HyPaSearch alle Treffer sowohl mit als auch ohne erlaubte Fehler. Die im Vergleich zu den Erwartungswerten erheblich höhere Anzahl an Treffern, die HyPaSearch mit diesen beiden Mustern gefunden hat, kann mit der zugrundeliegenden Suchstrategie für basengepaarte Bereiche erklärt werden. HyPaSearch beginnt zunächst nach der konservierten Loopsequenz zu suchen und kann dann, wie in Abschnitt 3.3.3 beschrieben, zwischen den flankierenden 3'- und 5'-Sequenzteilen hin und her springen, um die geforderten Basenpaare zu finden. Dabei können wahlweise sowohl im 3'- als auch im 5'-Teil die erlaubten Ausnahmen verwendet werden, wodurch erheblich mehr Treffer gefunden werden können, wie das Ergebnis zeigt.

erwartet	beobachtet	σ
1: `HUMANS`		
117188	117101	±361
2: `ADAM[1,0,0]`[a]		
195313	195559	±551
3: `p1=RYR 3...4 r1~p1`		
131836	131979	±297
4: `p1=2...4 GNRA r1~p1`		
66604	66610	±259
5: `p1=3...4 GNRA r1~p1[1,0,0]`		
146255	146261	±310
6: `p1=3...4 GNRA r1~p1[0,1,0]`		
146255	140521	±330
7: `p1=3...4 GNRA r1~p1[0,0,1]`		
69008	65959	±204

[a] Die Suche erfolgte hier über 10 Zufallssequenzen mit je 10^6 nt.

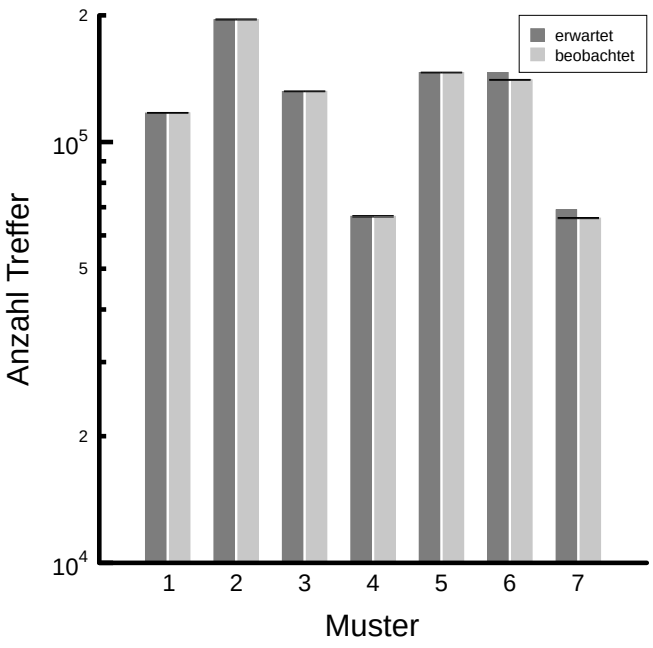


Abbildung 3.12: Vergleich der abgeschätzten mit der tatsächlich gefundenen Anzahl an Treffern von ausgewählten PatScan-Mustern. Die Suchen erfolgten überlappend über 10 Zufallssequenzen mit je 10^7 nt. In der Tabelle sind die erwartete Trefferanzahl (erwartet), der Mittelwert (beobachtet) über die 10 Zufallssequenzen und die Standardabweichung (σ) für jedes Muster aufgelistet. Das Histogramm fasst das Ergebnis graphisch zusammen.

3.4 Muster-Datenbank und Webservice

Ein Ziel dieser Arbeit ist es, eine Bibliothek von hybriden Mustern zu erstellen. Desweiteren sollte das in der Gruppe von Dr. Stefan Kurtz entwickelte Mustersuch-Programm HyPaSearch via Internet zugänglich gemacht werden. Den Benutzern soll mit dieser Schnittstelle die Möglichkeit gegeben werden

- eigene Musterbeschreibungen über zur Verfügung gestellte Sequenzdaten zu suchen, sowie
- in eigenen Sequenzdaten mit in der Musterbibliothek vorhandenen oder eigenen hybriden Mustern zu suchen.

Dazu wurde eine Internetpräsenz des Projektes entwickelt, die im Internet online erreichbar ist. Abbildung 3.13 zeigt die Startseite des Webservers. Auf einige Komponenten, die über das Menü verfügbar sind, wird im Folgenden näher eingegangen. Zunächst wird der allgemeine Aufbau des Webservers (siehe Abschnitt 3.4.1) und das zugrunde liegende Datenbankschema (siehe Abschnitt 3.4.2) erläutert. Anschließend wird das allgemeine Format und der Inhalt der Musterbibliothek vorgestellt (siehe Abschnitt 3.4.3), die Benutzung der Oberfläche zum Muster-Suchprogramm (siehe Abschnitt 3.4.4) und zur Auswertung von Mustersuchen beschrieben (siehe Abschnitt 3.4.5). Eine Weiterentwicklung des Servers ist bei Teune (2004) zu finden.

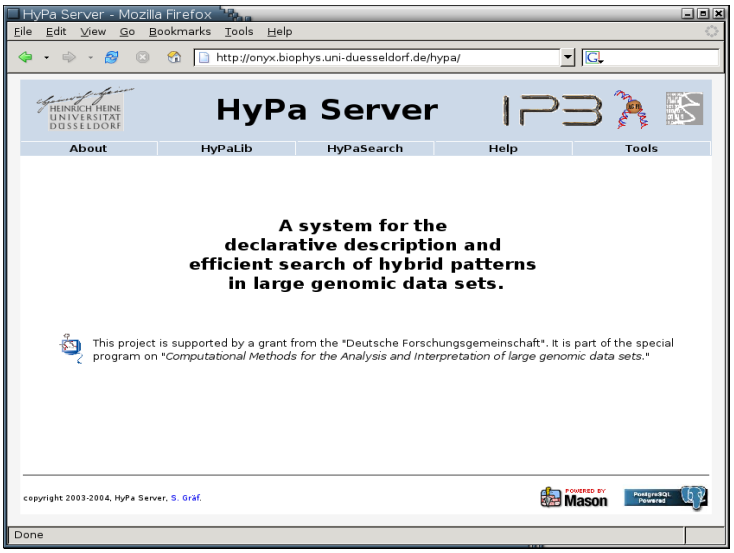


Abbildung 3.13: HyPaServer. Dargestellt ist die Startseite, über die die einzelnen Komponenten des Webserver im Menü angewählt werden können.

3.4.1 Aufbau des Webservers

Zur Repräsentation des Projektes, der entwickelten Werkzeuge zur Struktur-basierten Beschreibung, Suche und Annotation von nicht-kodierenden RNAs und der Ergebnisse wurde die Klient-Webserver-Technologie verwendet. Damit ist die Benutzung plattformunabhängig und dem Benutzer werden die benötigten Ressourcen für die Suche und Auswertung via Internet zur Verfügung gestellt. Abbildung 3.14 zeigt schematisch den Aufbau des Webservers. Dabei wurde darauf Wert gelegt, dass ausschließlich offene, unter der GNU General Public Lizenz (GNU-GPL) stehende Software verwendet wird. Kern des Systems ist ein Apache-Webserver, der über das Internet erreichbar ist, Anfragen eines Benutzers entgegennimmt und die angeforderten Informationen an den Klienten zurückschickt. Um Inhalte dynamisch generieren zu können und ein einheitliches Aussehen der Internetseiten zu gewährleisten, wurde das Perl-Paket HTML::Mason verwendet. Dieses mächtige Paket integriert die Skriptsprache Perl in den Apache-Webserver, sodass Perl-Code in HTML eingebettet werden kann. Zusätzlich können Komponenten programmiert werden, die an anderer Stelle wiederbenutzt werden können. Gleichzeitig sind sämtliche andere Perl- und BioPerl-Pakete integrierbar. Sowohl für die Speicherung der Daten, als auch für die Auswertung von Mustersuchen hat es sich als sinnvoll erwiesen, eine relationale Datenbank aufzubauen, in der sich die großen Mengen an Informationen nicht-redundant speichern und verwalten und mit Hilfe von beliebig komplexen Abfragen nach verschiedensten Aspekten sortieren lassen. Die Struktur der relationalen Datenbank wird in Abschnitt 3.4.2 beschrieben. Um persistenten Zugriff auf die Datenbank zu ermöglichen und aus dem Webserver heraus komplexe Abfragen formulieren zu können, wurde das Perl-Paket Apache::DBI verwendet.

3.4.2 Struktur der relationalen Datenbank

Die hier verwendete PostgreSQL-Datenbank basiert, wie die meisten Datenbank-Management-Systeme (DBMS), auf dem von Codd (1970) entwickelten Relationen-Modell, d. h.

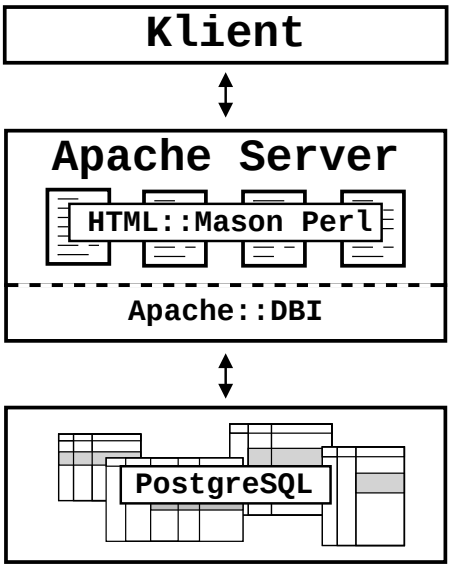


Abbildung 3.14: Schematische Darstellung des Webserveraufbaus. Skizziert sind hier die Softwarekomponenten, die zur Implementierung des Klienten-Server-Systems verwendet wurden.

alle Informationen können in Tabellen abgespeichert werden. Durch die Benutzung eines DBMS soll sichergestellt werden, dass

- die Daten konsistent gespeichert werden und manipulierbar bleiben und
- Redundanzen verhindert werden, da jede Information nur einmal gespeichert wird; gezielte Redundanz ist allerdings in einigen Fällen notwendig und unabdingbar.

In Abbildung 3.15 ist die entwickelte Struktur der Datenbank, die zur Speicherung der Informationen, die bei der Mustersuche anfallen, mit Hilfe des Entität-Relationen-Modells dargestellt, in dem Entitäten und ihre Beziehungen zueinander definiert werden. Eine Entität ist ein Objekt, das sich von anderen Objekten unterscheidet. Alle Entitäten mit gleichen Merkmalen (Attributen) werden in einer Tabelle zusammengefasst. Diese Objekte besitzen den gleichen Entitätstyp. Jede Zeile einer Tabelle entspricht also einer Entität des gleichen Typs. Die Spalten der Tabelle enthalten die Attribute des Entitätstyps. Durch die Definition von Relationen können verschiedene Entitätstypen in Beziehung gebracht werden. Es gibt drei Typen von Relationen, die sich in der Komplexität unterscheiden. Angenommen A und B seien Entitätstypen, die durch die Relation $R \subseteq A \text{x} B$ in Beziehung stehen, dann definieren sich die verschiedenen Relationstypen wie folgt:

- **1:1-Beziehung:** Hier wird eine Entität von A genau einer Entität von B durch die Relation R zugeordnet und umgekehrt.
- **1:n-Beziehung:** Jeder Entität aus B wird durch die Relation R höchstens eine Entität aus A zugeordnet. Umgekehrt können Entitäten aus A mehreren Entitäten aus B zugeordnet sein.
- **m:n-Beziehung:** Beliebige Entitäten aus A können durch die Relation R mehreren Entitäten aus B zugeordnet werden und umgekehrt. Diese Art der Relation wird allerdings aufgrund der Komplexität nach Möglichkeit vermieden.

Im Folgenden soll anhand der angegebenen Definitionen die hier entwickelte Datenbank-Struktur und damit die Modellierung der bei der Mustersuche anfallenden Informationen beschrieben werden. Ein Struktur-basiertes RNA-Muster besitzt als Merkmale einen Namen (`name`), die Muster-Beschreibung (`pattern`), sowie weitere Informationen, wie sie in

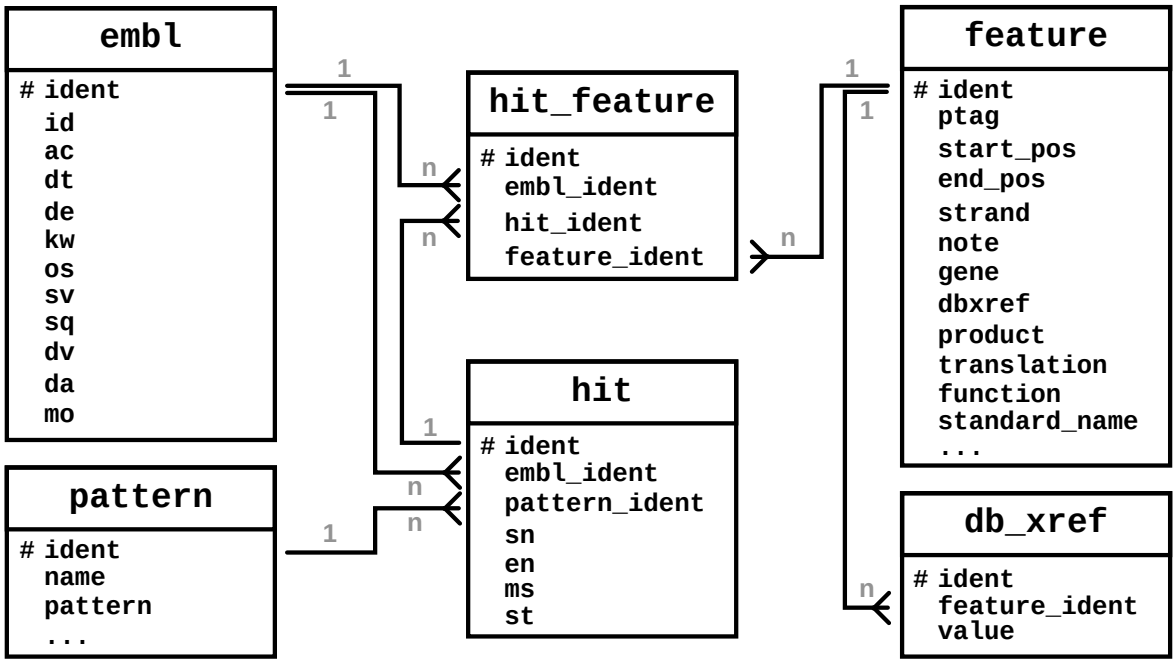


Abbildung 3.15: Entität-Relationen-Modell der entwickelten Datenbank-Struktur. Dargestellt ist das Schema, das die Entitätstypen und ihre Beziehungen untereinander definiert. Der Tabellenname ist jeweils im oberen Teil eines Kastens angegeben. Im unteren Teil sind jeweils die Attribute, die den Entitätstyp definieren, aufgelistet. Die Beziehungen der Tabellen untereinander sind durch die Krähenfuß-Notation symbolisiert. Die detaillierte Beschreibung dieses Schemas ist im Text zu finden.

Abschnitt 3.4.3 vorgestellt wurden. Diese Attribute werden pro Muster (Entität) in der Tabelle `pattern` (Entitätstyp) zusammengefasst. Für jedes Muster ergibt sich somit eine Zeile in der Tabelle. Diese Tabelle enthält somit die eigentliche Musterbibliothek HyPaLib (vgl.. Abschnitt 3.4.3). Als Ergebnis der Suche erhält man Treffer, die in den durchsuchten Sequenzdaten auf das Muster passen. Jeder Treffer besitzt als Information mindestens den Bezeichner der Sequenz (`embl_ident`), in der der Treffer gefunden wurde, die Start- und Endposition des Treffers (`sn` und `en`), die Treffersequenz (`ms`) und ggf. die Punkt-Klammer-Notation der Sekundär-Struktur (`st`). Diese Merkmale werden in der Tabelle `hit` zusammengefasst. Damit lässt sich auch die erste Relation des Schemas beschreiben. Jedem Muster aus der `pattern`-Tabelle wird eine beliebige Anzahl an Treffern in der `hit`-Tabelle zugeordnet. Umgekehrt gehört jeder Treffer zu genau einem Muster. Bei dieser Relation handelt es sich um eine 1:n-Beziehung, die in Abbildung 3.15 durch die Krähenfuß-Notation symbolisiert wird. Dafür wird in der Tabelle `pattern` das zusätzliche Attribut `ident` als Primärschlüssel eingeführt. Dieser Primärschlüssel besitzt als Wert eine ganze Zahl, die aufgrund des Relationstyps für diese Tabelle eindeutig ist. In der Tabelle `hit` wird gleichzeitig für jeden Treffer unter dem Attribut `pattern_ident` der Wert des Primärschlüssels des zugeordneten Musters abgespeichert. Dementsprechend erfolgte auch die Wahl der Attributnamen. Alle verwendeten Primärschlüssel heißen `ident`. Attribute die sich auf Primärschlüssel einer anderen Tabelle beziehen, setzen sich zusammen aus dem Namen der referenzierten Tabelle gefolgt von `_ident`. Ein Teil der allgemeinen Informationen,

die in den durchsuchten EMBL-Sequenzeinträgen zur Verfügung stehen, wurden in der `embl`-Tabelle gespeichert. Dabei wurden die Attribut-Namen dem EMBL-Format entsprechend gewählt. Die Sequenz des Eintrags beispielsweise wird in dieser Tabelle aufgrund des zu erwartenden Speicherplatzbedarfs nicht mit gespeichert. Das gleiche gilt für die Annotationen eines Eintrags. Diese werden, wie weiter unten beschrieben, separat verwaltet. Als zusätzliche Information, die nicht aus EMBL-Einträgen stammt, wird im `da`-Feld der Dateiname des Archivs gespeichert, das den Eintrag enthält. Die Relation zwischen der `hit`- und der `embl`-Tabelle erfolgt über das Attribut `embl_ident`. Dabei wird jedem Treffer in der `hit`-Tabelle genau ein EMBL-Eintrag zugeordnet, während umgekehrt in einem EMBL-Eintrag mehrere Treffer auftreten können (1:n-Beziehung). Da für die Auswertung einer Mustersuche nur Annotationen von Interesse sind, die im Bereich eines Treffers liegen, werden nicht alle Annotationen eines Eintrags gespeichert. Dafür wurden zwei weitere Tabellen (`feature` und `db_xref`) in das Schema integriert. Die Eigenschaften einer Annotationseinheit werden mit Ausnahme der Verweise (`db_xref`), die zu anderen Internet-Datenbanken zeigen, in der `feature`-Tabelle verwaltet. Auch hier wurden die Attributnamen dem EMBL-Format entsprechend übernommen. Da die Verweise zu anderen Datenbanken innerhalb einer Annotationseinheit mehrfach vorkommen können und verschiedene Einheiten die identischen Verweise besitzen können, wurden beide Tabellen durch eine 1:n-Relation verbunden, um Redundanzen zu vermeiden. In Anbetracht der Tatsache, dass in der Umgebung eines Treffers mehrere Annotationen vorhanden sein können, aber auch im Bereich einer Annotation verschiedene Treffer liegen können, wurde die Tabelle `hit_feature` zwischengeschaltet. Dadurch kann diese m:n-Beziehung in weniger komplexere 1:n-Relationen aufgeteilt werden. Außerdem wird auf diese Weise ebenfalls die redundante Speicherung von Annotationen verhindert. Gleichzeitig wird mit dem `embl_ident`-Attribut der `hit_feature`-Tabelle diese mit der `embl`-Tabelle in Relation gesetzt. Dieses Schema wurde dann in die PostgreSQL-Datenbank übertragen, sodass mit Hilfe der strukturierten Suchsprache (*Stuctured Query Language*, kurz SQL) gezielt die gewünschten Informationen selektiert werden können.

3.4.3 Bibliothek hybrider Muster

Die Bibliothek von hybriden Mustern (***Hy**brid **Pa**ttern **Lib**rary*, kurz HyPaLib, Gräf *et al.*, 2001) enthält die erstellten Musterbeschreibungen zusammen mit weiteren Informationen. Diese ist im Internet online als ASCII-Textdatei oder im HTML-Format verfügbar. Abbildung 3.16 zeigt exemplarisch den Eintrag für die Selenocystein-Insertions-Sequenz (SECIS), eine Haarnadel-Struktur, die in 3' UTRs von Selenoprotein-mRNAs vorkommt (Kryukov *et al.*, 1999).

Jede Zeile eines Eintrags beginnt mit einer Zwei-Zeichenkombination gefolgt von einem Leerzeichen und der zeilenspezifischen Information. Tabelle 3.7 fasst alle erlaubten Symbolkombinationen zusammen und beschreibt deren Bedeutung und Format. In jedem Eintrag müssen die Zeilen mit dem eindeutigen Bezeichner (`ID`), dem Datum der Erstellung und ggf. dem Datum der letzten Aktualisierung (`DT`), der Kurzbeschreibung (`DE`) und der Musterbeschreibung (`HP`) enthalten sein. Die letzte Zeile eines jeden Eintrags beginnt

```
ID secis
DT 28/08/2000, Update:03/10/2004
DE Selenocyteine Insertion Sequence (SECIS), a common stem-loop structure in
DE the 3' untranslated region of selenoprotein mRNAs.
KW SECIS, selenocysteine insertion sequence
HP secis=stem1 iloop1 quartet stem2 aloop cstem2 iloop2 cquartet cstem1
HP <<mfreeE(stem1( iloop1 iloop2 )cstem1)<-7 && mfreeE(stem2( aloop )cstem2)<-5
HP where stem1:=.{7}
HP       iloop1:=.{2,6} A
HP       quartet:=UGA.
HP       stem2:=.{11,12}
HP       aloop:=AA .{5,15}
HP       cstem2:=^stem2[3,1,1]
HP       iloop2:=.{3,7}
HP       cquartet:=.GA.
HP       cstem1:=^stem1[1,1,1];
RE Kryukov GV, Kryukov VM, Gladyshev VN. New mammalian selenocysteine-containing
RE proteins identified with an algorithm that searches for selenocysteine
RE insertion sequence elements. J Biol Chem. 1999 Nov 26;274(48):33888-97.
RE PMID:10567350.
RE Kryukov GV, Gladyshev VN. The prokaryotic selenoproteome. EMBO Rep. 2004
RE May;5(5):538-43. Epub 2004 Apr 23. PMID:15105824.
AU Graef S.
//
```

Abbildung 3.16: Format der Einträge in der Musterbibliothek HyPaLib. Exemplarisch ist hier der Mustereintrag für die Selenocystein-Insertions-Sequenz (SECIS) abgebildet. Jede Zeile eines Eintrags beginnt mit eine definierten Zweibuchstaben-Kombination gefolgt von einem Leerzeichen. Die Bedeutung dieser Symbole ist in Tabelle 3.7 aufgelistet.

---

mit der Zeichenkombination „`//`“. Alle anderen in Tabelle 3.7 aufgelisteten Zeilen-Typen sind optional und können ausgelassen werden. Einige der Zeilen-Typen dürfen nur genau einmal pro Eintrag vorkommen. In der HTML-Version sind die Referenzen (`RE`) über ihre Identifikationsnummer (`PMID`) mit dem Pubmed-Service (Wheeler *et al.*, 2004) verlinkt (vgl. Abbildung 3.17).

In der Bibliothek enthalten sind aktuell rund 60 Einträge. Diese enthalten einfache Muster, die z. B. ausschließlich konservierte Primärsequenzen beschreiben, sowie weitaus komplexere hybride Muster, die Beschreibungen von Primärsequenz- und Sekundärstruktur-Elementen kombiniert mit zusätzlichen Einschränkungen enthalten (vgl. Abbildung 3.16). Exemplarisch werden einige dieser Muster kurz vorgestellt; auf drei komplexere Muster wird in Abschnitt 3.5 eingegangen.

Durch den modularen Aufbau der Beschreibungssprache können einfache, weniger spezifische Muster zu komplexeren Mustern kombiniert werden. Beispiele dafür sind die Muster für pro- und eukaryotische Promotoren (`promoter_pro` und `promoter_euk`), die selber wieder aus Mustern bestehen, die z. B. die konservierte -35- und die -10-Region (`m35_region`

Tabelle 3.7: Spezifikation der in der Musterbibliothek HyPaLib zur Verfügung stehenden Bezeichner. Jede Zeile eines HyPaLib-Eintrags muss mit einer der aufgelisteten Zwei-Zeichen-Kombinationen beginnen. Einige davon müssen in jedem Eintrag enthalten sein, während andere je nach Bedarf verwendet werden können.

Symbole	Beschreibung	Eigenschaft[a]	Bemerkung
`ID`	Bezeichner	oe	Eindeutiger Name des hybriden Musters
`DT`	Datum	oe	Daten der Erstellung und ggf. letzten Aktualisierung im Format `DD-MM-YYYY[, Update: DD-MM-YYYY]`
`DE`	Beschreibung	o	Kurze Beschreibung oder Kommentar zum hybriden Muster
`KW`	Schlüsselworte		Liste von Schlüsselwörtern, die zur Suche in der Datenbank dienen
`HP`	Hybrides Muster	o	Musterbeschreibung in HyPaL-Syntax inkl. Kommentaren
`NO`	EMBL-Version		Version der EMBL Datenbank, mit der das Muster erstellt wurde
`AL`	Alignment	e	Alignierte Sequenzen und andere Informationen, die die Herkunft dokumentieren
`CS`	Konsensus-Sequenz	e	Konsensus-Sequenz der alignierten Sequenzen (Zur Ausrichtung von Konsensus-Sequenz und Konsensus-Sekundärstruktur werden „-“-Zeichen verwendet)
`ST`	Konsensus-Struktur	e	Konsensus-Sekundärstruktur des Musters im Vienna-Format (Punkt-Klammer-Notation)
`PM`	Parameter		Spezielle Parameter (z. B. $T/\,^{\circ}\mathrm{C}$)
`RE`	Referenzen		Liste von Referenzen im Pubmed-Format
`AU`	Autor	oe	Autor des Eintrags
`//`	Eintrag-Ende	oe	Letzte Zeile eines jeden Eintrags

[a] o: obligatorisch (nicht-Null); e: Zeile darf nur einmal pro Eintrag vorkommen

und `pribnow_box`) oder Initiator-Elemente (`inr`) beschreiben (Bucher, 1990; Butler & Kadonaga, 2002; Mehldau & Myers, 1993).

Andere Muster beschreiben Sekundärstruktur-Elemente, die z. T. konservierte einzelsträngige Bereiche enthalten, wie z. B. Tetraloop-Harrnadelstrukturen, Pseudoknoten oder Kleeblatt-Strukturen (Searls, 1993). Diese sind zwar nicht besonders spezifisch, können aber ebenfalls zu komplexeren Deskriptoren kombiniert werden. Ein weiteres Beispiel ist die 3’ nicht-kodierende Region der genomischen Tabak-Mosaik-Virus-RNA (`tmv_3prime`), das aus drei hintereinander liegenden Pseudoknoten, einem weiteren Pseudoknoten, der eine Harrnadelstruktur einschließt, einer weiteren Harrnadelstruktur und einem weiteren Pseudoknoten besteht (Takamatsu *et al.*, 1990).

Desweiteren enthalten sind Muster, die RNA-Protein-Bindemotive beschreiben, wie die Sekundärstrukturen von vier Rev-Binde-Elementen (RBE) oder das Tat-Binde-Element (TBE) von Viren, die mit der menschlichen Immunkrankheit in Verbindung gebracht werden (Bourdeau *et al.*, 1999; Ferbeyre *et al.*, 1997; Leclerc *et al.*, 1994). Beispiele für RNAs, die kleine Moleküle binden können, sind die Aptamer-Motive für Valin (`valine_binding_motif`) oder Neomycin (`neomycin_binding_motif`). Diese Muster wurden von Bourdeau *et al.* (1999) übernommen.

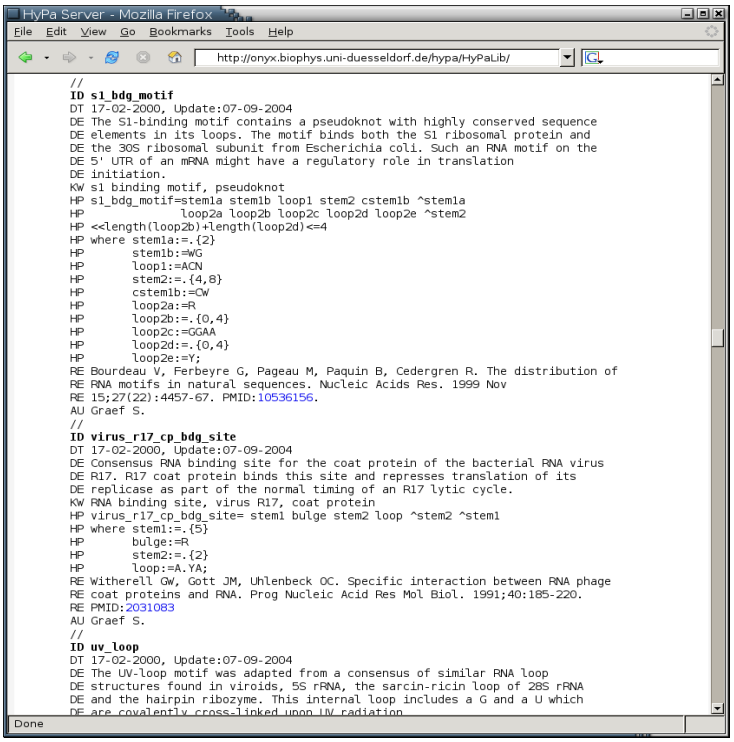


Abbildung 3.17: HyPaLib. Ausschnitt aus der Bibliothek hybrider Muster (HTML-Version).

Eine weitere Klasse von RNAs beschreibt biochemisch und katalytisch aktive Motive, wie den UV-sensitiven Loop E (`uv_loop`), das Leadzym- (`leadzyme`) und DNA-Enzym-Muster (`dna_enzyme` und `dna_enzyme_target`) oder Ribozym-Motive (Bourdeau *et al.*, 1999). Auf Typ III- und Typ I-Ribozyme wird in Abschnitt 3.5.2 noch detailliert eingegangen.

Muster, die z. B. die 5' Enden von U3 und U5 snRNAs (`u2_5prime` und `u3_5prime_vert`), die Domänen 6 und 8 von pflanzlichen 7 SL RNAs (`7SL_RNA_d6_d8_plant`) oder ribosomale 5 S RNAs (z. B. `5S_rRNA_animals` oder `5S_rRNA_archea`) beschreiben, verwenden überwiegend Profil-Matrizen (vgl. Abschnitt 3.2.9). Diese Deskriptoren wurden auf Grundlage von Sequenzen und Strukturen aus der U RNA-Datenbank (Zwieb, 1997), der Signal-Erkennungs-Partikel-Datenbank (SRPDB, Rosenblad *et al.*, 2003) oder der 5 S RNA-Datenbank (Szymanski *et al.*, 2002) erstellt und mit Hilfe von ClustAlX (Jeanmougin *et al.*, 1998) und ConStruct (Lück *et al.*, 1999) soweit optimiert, bis Suchen in großen Sequenz-Datenbanken, wie EMBL oder Genbank, abgesehen von nicht-annotierten Sequenzen keine falsch positiven Treffer mehr zurücklieferten. Auf die Vorgehensweise bei der Mustererstellung wird detailliert am Beispiel von Typ I-Hammerhead-Ribozymen in Abschnitt 3.5.2 eingegangen.

Abschließend seien die hybriden Muster erwähnt, die neben Primärsequenz- und Sekundärstruktur-Elementen weitere Einschränkungen enthalten, wie die Haarnadel-Struktur C (`hairpinC`), die in mRNAs des Prion-Proteins vorkommt (Lück *et al.*, 1996), oder die Selenocystein-Insertions-Sequenz (SECIS), die in 3' UTRs von Selenoprotein-mRNAs zu finden ist (Kryukov *et al.*, 1999, siehe Abbildung 3.16). Beide zuvor erwähnten Motive enthalten zusätzliche thermodynamische Einschränkungen, um die Anzahl an falsch positiven Treffern zu reduzieren.

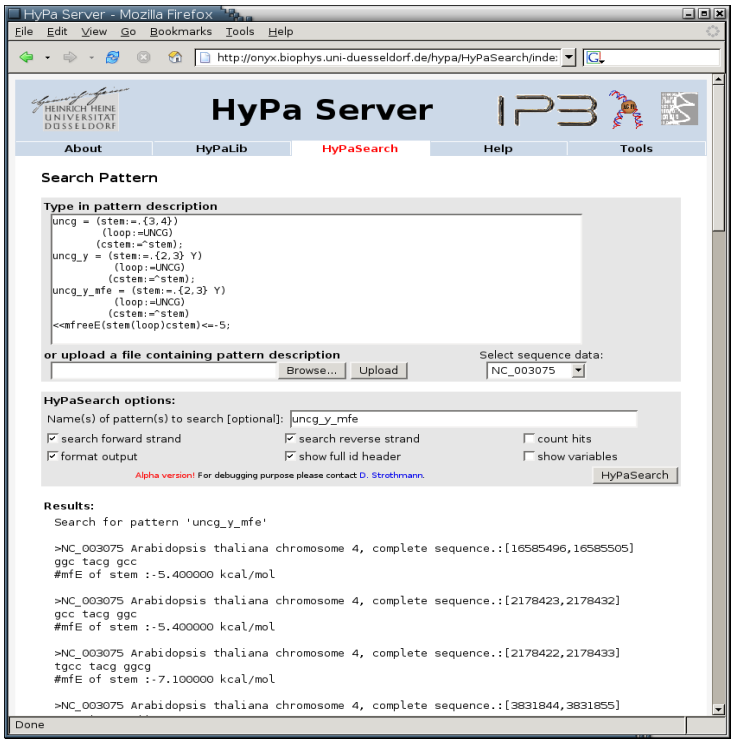


Abbildung 3.18: HyPaSearch. Dargestellt ist die Benutzerschnittstelle zum Suchprogramm HyPaSearch. Hier kann der Benutzer seine Musterbeschreibung in HyPaL eingeben oder hochladen, diverse Programmoptionen selektieren und dann über zur Verfügung gestellte Sequenzdaten suchen. Die Ergebnisse werden dann unterhalb der Eingabeeinheit angezeigt.

An diese Stelle sei noch darauf hingewiesen, dass es sich bei der Erstellung von RNA-Mustern um einen sehr zeitintensiven Prozess handelt. Konservierte Sequenzen und Strukturen aus der Literatur sind in Anbetracht der rapide anwachsenden Mengen an Sequenzdaten (vgl. Abbildung 1.3) meist nicht spezifisch genug, sodass daher fast immer aufwendige Muster-Optimierungen sowohl zur Erstellung als auch zur Aktualisierung notwendig sind. Dieser Prozess lässt sich leider nur in teilweise automatisieren.

3.4.4 Benutzerschnittstelle zur Mustersuche

Unter der Rubrik HyPaSearch auf der Webseite hat der Benutzer die Möglichkeit Muster über Sequenzdaten zu suchen (vgl. Abbildung 3.18). Die Beschreibung kann entweder in das vorgesehene Textfeld eingetippt werden oder aus einer lokal gespeicherten Textdatei hochgeladen und in dem Textfenster editiert werden. Die zu durchsuchenden Sequenzdaten und diverse Programmoptionen können ausgewählt werden. Durch Klicken auf den Knopf HyPaSearch wird dann die Suche auf dem Server gestartet. Vor der eigentlichen Suche wird vom Programm eine Abschätzung durchgeführt, wieviele Treffer zu erwarten sind. Liegt diese Zahl oberhalb eines intern definierten Schwellenwertes, wird die Suche nicht gestartet. Der Benutzer erhält einen Hinweis, dass mit seinem Muster zu viele Treffer zu erwarten sind und deshalb aus Klient-Server-technischen Gründen die Suche nicht durchgeführt wird. Andernfalls werden die Ergebnisse unter dem Steuerelement, den selektierten Programmoptionen entsprechend, angezeigt.

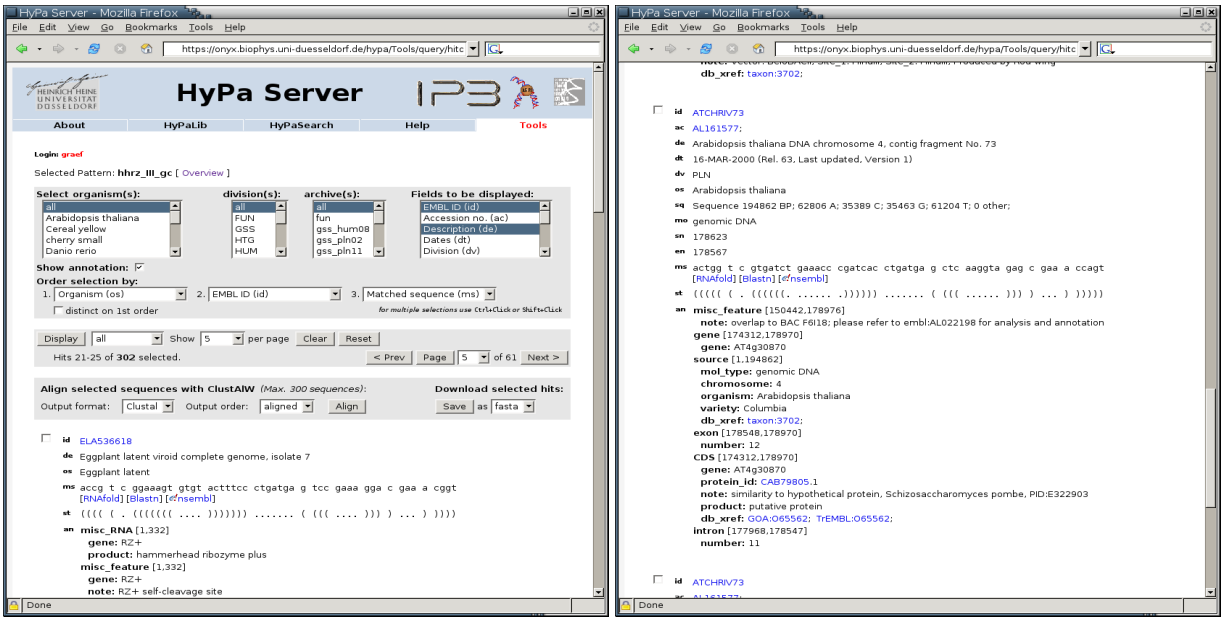


Abbildung 3.19: Treffer-Browser. Bildschirmfotos der Seite zur Auswertung von Mustersuchen. Links: Steuereinheit zur Selektion der gewünschten Informationen einer Mustersuche; rechts: Anzeige der selektierten Informationen eines Treffers inkl. Annotation.

3.4.5 Benutzerschnittstelle zur Analyse der Treffer

Für die Analyse von Mustersuchen hinsichtlich der verfügbaren Informationen und Annotationen aus den Sequenzdaten wurde eine separate, bislang lang nur intern zugreifbare Seite aufgesetzt. Da bis zum Ende dieser Arbeit das HyPaSearch-Programm nur in Teilen benutzbar war, wurde hier als Suchwerkzeug PatScan verwendet. Für die aufwendige Suche mit PatScan über große Datensätze wurde das BioPerl-Skript `SearchPattern.pl` implementiert, das

- die Mustersuche parallel, auf Rechnern im Netzwerk verteilt sucht,
- die Ergebnisse pro Sequenzarchiv auf der Festplatte speichert,
- sämtliche Informationen und Annotationen, die im Bereich der Treffer liegen, aus den Sequenzdaten extrahiert und
- in die in Abschnitt 3.4.2 beschriebene Datenbank importiert.

Danach stehen die Ergebnisse zur Analyse der Treffer einer Suche im Webserver zur Verfügung, wie es in Abbildung 3.19 dargestellt ist. Die obere, grau hinterlegte Steuerungseinheit dient der Selektion der Informationen, die angezeigt werden sollen. Hier kann der Benutzer

- einzelne Organismen (linkes Feld), Abteilungen (zweites Feld v. l.) und/oder Sequenzarchive (drittes Feld v. l.) auswählen,
- selektieren, welche Informationen angezeigt werden sollen (rechtes Feld)
- die Annotationsinformation ein- und ausschalten (Klickbox), und
- die Sortier-Reihenfolge festlegen, in der die Daten angezeigt werden sollen.

In der darunterliegenden Navigationsleiste kann ausgewählt werden, wieviele Treffer pro Seite angezeigt werden sollen. Durch Klicken auf den Knopf Display werden dann die Tref-

fer der Auswahl entsprechend angezeigt bzw. die Ansicht aktualisiert, wenn die Selektion verändert wurde. Mit den Knöpfen Prev, Page und Next kann man durch die einzelnen Seiten blättern. Vor jedem Treffer wird ein kleines Kästchen angezeigt; darüber können einzelne Treffer markiert werden. Je nachdem, welche Option in dem Feld hinter dem Display-Knopf in der Navigationsleiste gewählt ist (`all`, `selected` oder `unselected`), werden alle, nur die markierten oder nur die unmarkierten Treffer angezeigt. Dadurch kann der Benutzer z. B. redundante Treffer aus der Ansicht ausschließen.

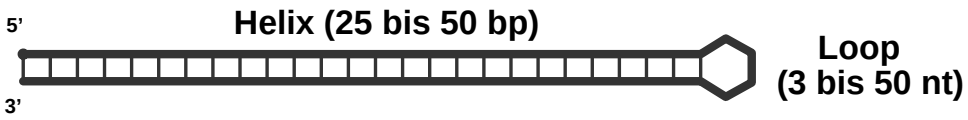
Die angezeigte Information und Annotation der Treffer enthält Verweise zu weiteren Datenbanken und anderen Web-Analyse-Werkzeugen, wie RNAfold, Blast oder Ensembl. Desweiteren besteht die Möglichkeit, ausgewählte Treffer zu alignieren. Durch Klicken auf den Knopf Align in der untersten grau hinterlegten Leiste können bis zu 300 Treffersequenzen mit ClustAlW verarbeitet werden. Das erstellte Alignment und zugehörige Dendrogramme kann sich der Benutzer im ASCII-Textformat, als Pdf- oder Postscript-Datei herunterladen. Außerdem können ausgewählte Treffer durch Klicken auf den Knopf Save in verschiedenen Formaten zur Weiterverarbeitung heruntergeladen werden.

3.5 Anwendungsbeispiele

3.5.1 Doppelsträngige RNA in *Dictyostelium discoideum*

Bei *in-silico*-Untersuchungen an verschiedenen eukaryotischen Genomen konnte eine Abundanz an antisense-Transkripten gezeigt werden (Lehner *et al.*, 2002). Bei einigen dieser Transkripte stellte sich dann heraus, dass sie Einfluss auf die Regulation der Genexpression haben. Es wird vermutet, dass es sich dabei um einen allgemeinen Mechanismus zur Regulation der Genexpression handelt. In diesem Zusammenhang spricht man allgemein von natürlichen antisense-Transkripten, sog. NATs, die komplementär zu anderen endogenen RNAs sind. Diese können sowohl in *cis*, d. h. an der gleichen Position auf dem Gegenstrang des Zielmoleküls, als auch in *trans*, sprich an einer anderen Stelle im Genom, kodiert sein. In diesem Zusammenhang werden verschiedene Mechanismen diskutiert, wie diese Art der Genregulation erfolgen kann (Übersichtsartikel: Lavorgna *et al.*, 2004). Im Rahmen dieser Arbeit wurden ähnliche Untersuchungen an genomischen Sequenzdaten von *Dictyostelium discoideum* in Zusammenarbeit mit Dr. Christian Hammann von der Universität Kassel durchgeführt (Gräf *et al.*, 2004).

Lange doppelsträngige RNA (dsRNA) wird in einer Vielzahl von höheren Organismen zu ungefähr 21 nt langen *small interfering RNAs* (siRNAs) prozessiert. Dieser Schritt wird durch ein RNAse III-ähnliches Protein namens Dicer vermittelt (Bernstein *et al.*, 2001a; Billy *et al.*, 2001; Novotny *et al.*, 2001; Zamore *et al.*, 2000). Diese siRNAs werden in Säugetieren, *Drosophila melanogaster* und wahrscheinlich auch in *Caenorhabditis elegans* in einen Endoribonuklease-enthaltenden Komplex eingebaut, der als RISC (*RNA-induced silencing complex*) bezeichnet wird. Ein solcher Komplex kann mRNAs degradieren, die zu der eingebauten siRNAs komplementär sind, wodurch die Translation entsprechender mRNAs post-transkriptionell verhindert wird (Tomari *et al.*, 2004). Dieser




```
% dd_lhp
r1={au,ua,gc,cg,gu,ug}
p1=25...50 3...50 r1~p1
```

Abbildung 3.20: Musterbeschreibung `dd_lhp` für die Suche mit PatScan nach langen dsRNAs in *Dictyostelium discoideum*. Dargestellt ist die Sekundärstruktur (oben), die durch die Muster (unten) beschrieben wird. Die Beschreibung beginnt mit dem Namen des Muster, gefolgt von der Basenpaar-Regel (Watson-Crick- und Wobble-Basenpaare sind erlaubt) und der eigentlichen Beschreibung.

Mechanismus wird allgemein als *RNA interference* (RNAi) bezeichnet (Übersichtsartikel: Bernstein *et al.*, 2001b). RISC konnte bislang nicht in Pilzen, Pflanzen oder Amöben nachgewiesen werden, obwohl auch hier RNAi stattfindet. In *Dictyostelium discoideum* beispielsweise konnte gezeigt werden, dass ein alternativer Mechanismus existiert, für den die RNA-abhängige RNA-Polymerase (RdRP) benötigt wird (Martens *et al.*, 2002). Dabei können die siRNAs als Primer dienen, die gegen entsprechende mRNAs hybridisieren. Durch Verlängerung der Primer durch die RdRP entstehen lange dsRNAs, die wieder von Dicer degradiert werden können. Diese Tatsache schließt allerdings auch nicht aus, dass ein RISC-ähnlicher Komplex in *Dictyostelium discoideum* existiert, er wird nur nicht zwingend benötigt.

Um die Funktionalität eines RNAi-Systems zu garantieren, sollten endogene RNAs in der Zelle keine langen doppelsträngigen Bereiche ausbilden können, da diese ansonsten degradiert werden könnten. Denkbar wäre allerdings auch, dass doppelsträngige RNAs in natürliche Mechanismen zur Genregulation involviert sein könnten, insbesondere wenn die Expression von dsRNAs kontrolliert erfolgt. Um der Fragestellung nachzugehen, ob in *Dictyostelium discoideum* RNAs kodiert sind, die intramolekular lange Doppelstränge ausbilden können, wurden Struktur-basierte Muster erstellt, um damit über ausgewählte Sequenzdaten zu suchen. Zu diesem Zweck wurden sämtliche Einträge aus der EMBL-Datenbank (Version 78) mit Hilfe eines dafür geschriebenen BioPerl-Skripts herausgefiltert, die in der OS (*Organism species*)-Zeile „*Dictyostelium discoideum*“ enthalten. Insgesamt waren 156560 Einträge mit insgesamt 91077278 nt enthalten, die sich auf die Abteilungen EST_INV (Invertebraten ESTs), HTG_INV (Hoch-Durchsatz-Genomsequenzen), INV (Invertebraten) und ORG (Organellen) verteilen und genomische Sequenzdaten von Chromosom 2 sowie Sequenzen von publizierten Genen und ESTs umfassen. Das zur Suche verwendete Muster beschreibt eine lange Helix mit 25 bis 50 Basenpaaren, die durch einen Loop mit variabler Größe (3 bis 50 nt) abgeschlossen wird (vgl. Abbildung 3.20). Mit diesem Muster ist alle 3,2 $\times 10^7$ nt ein Treffer zu erwarten, wenn man die Nukleotid-Frequenzen in *Dictyostelium discoideum* (A: 0,4, C: 0,1, G: 0,1, T: 0,4) zugrunde legt (Glockner *et al.*, 2002). Die Suche wurde mit PatScan überlappend auf beiden Strängen durchgeführt. Anschließend wurden alle Treffer, die vollständig in einem anderen enthal-

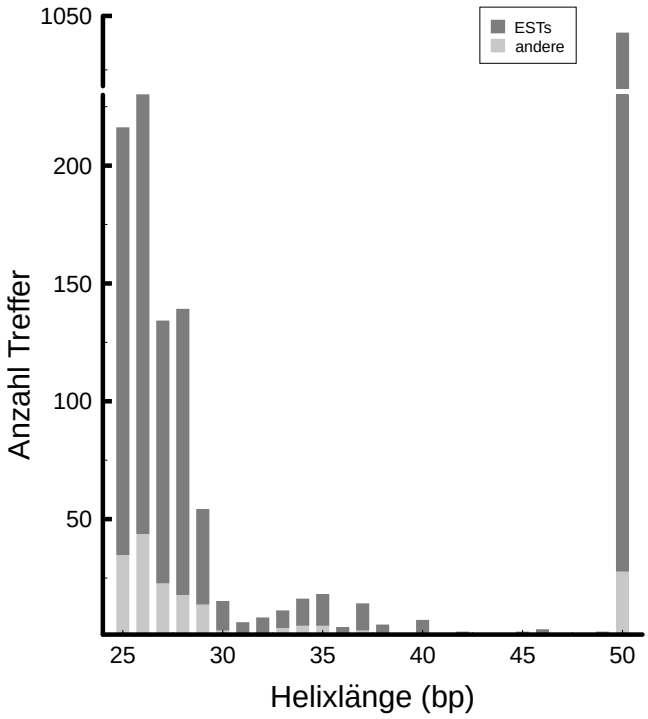


Abbildung 3.21: Verteilung der Länge der doppelsträngigen Bereiche in den gefundenen Treffern. Aufgetragen ist die Anzahl der gefundenen Treffer über der Helixlänge in bp. Der Anteil der EST-Datensätze ist dunkelgrau hinterlegt, alle anderen Treffersequenzen hellgrau.

ten waren mit dem Programm `sort_olapping.pl` herausgefiltert, womit insgesamt 3170 Treffer übrigblieben; dies sind erheblich mehr Treffer als statistisch zu erwarten waren. Das *Dictyostelium discoideum*-Genom besitzt einen AT-Gehalt von etwa 80 %. Besonders nicht-transkribierte und nicht-kodierende Bereiche weisen einen hohen Anteil an A und T auf. Auch wenn nicht ausgeschlossen werden kann, dass solche Sequenzen regulatorisch aktiv sind, kann davon ausgegangen werden, dass die meisten nicht transkribiert werden. Um also solche Treffer mit geringer oder keiner Signifikanz auszuschließen, wurden weitere Filterschritte vorgenommen. Dabei wurden alle Sequenzen mit

- AT-Gehalt größer als 86 %,
- 10 oder mehr aufeinanderfolgenden As oder Ts, sowie mit
- ausschließlich A:U- oder G:U-Basenpaaren in der Helix

herausgefiltert. Dadurch konnte die Anzahl der Treffer auf 1932 reduziert werden. Davon wurden 1746 (451 verschiedene) in EST-Sequenzen und 186 (136 verschiedene) in den übrigen *Dictyostelium discoideum*-Datensätzen gefunden. Schaut man sich die Verteilung der Helixlängen in den gefundenen Treffern an, ergibt sich das in Abbildung 3.21 gezeigte Bild. Im Prinzip verteilen sich die Treffer bzgl. der Anzahl an Basenpaaren in der Helix auf zwei Gruppen. Die eine Gruppe von Treffern weist Helixlängen zwischen 25 und 29 bp auf, was sehr wahrscheinlich der minimalen Länge entspricht, die für die Wechselwirkung mit Dicer und anderen dsRNA-bindenden Proteinen benötigt wird. Die überwiegende Mehrzahl der Treffer besitzt 50 und mehr Basenpaare im Doppelstrang, wobei hier die gefundenen ESTs mit 1014 Treffern den größten Anteil ausmachen. Längere doppelsträngige Bereiche wurden von dem hier beschriebenen Muster zwar nicht erfasst, allerdings konnten in hier nicht dargestellten Suchen noch zwei weitere Gruppen von Helixlängen um 53 und 96 bp gefunden werden.

Rund 90 % der Treffer wurden in EST-Sequenzdaten gefunden. Die Orientierung von ESTs ist in der Regel nicht bekannt (Lehner *et al.*, 2002). Um die Relevanz der gefundenen Treffer beurteilen zu können, muss allerdings die Orientierung der Sequenz bekannt sein. Wird ein langer doppelsträngiger Bereich nur aufgrund von G:U- und U:G-Basenpaaren ausgebildet, kann an den entsprechenden Stellen in der revers-komplementären Orientierung mit A und C keine Basenpaarung stattfinden. Dementsprechend wurden alle EST-Sequenzen, die mit der kodierenden Sequenz von vorhergesagten oder bekannten Proteinen übereinstimmen, von der weiteren Betrachtung ausgeschlossen. Dieser Vergleich wurde auf der *Dictyostelium discoideum*-Genom-Webseite dictyBase durchgeführt. Interessanterweise blieben einige Treffersequenzen über, die nicht direkt zugeordnet werden konnten. Diese EST-Sequenzen stimmen nur in Teilen mit mRNAs, die im Genom annotiert sind, überein und bilden doppelsträngige Bereiche aus, die nicht genomisch kodiert sind. Dieser Sachverhalt könnte durch die Tatsache erklärt werden, dass es sich bei diesen RNA-Hybriden um Sequenzen handelt, bei denen die reverse Transkriptase bei der Herstellung der ESTs auf den neu synthetisierten cDNA-Strang gewechselt und zurückgelesen hat, was allgemein als *template swapping* bezeichnet wird. Allerdings zeigte sich bei der Analyse der Sekundärstruktur entsprechender Sequenzen auch, dass diese in den interessanten Bereichen nicht hoch strukturiert sind, was nicht unbedingt für *template swapping* spricht. In diesen Zusammenhang sei erwähnt, dass erst kürzlich sense-antisense RNA-Hybride im Menschen nachgewiesen werden konnten (Bartsch *et al.*, 2004).

Als nächstes wurden die 186 nicht-EST-Sequenzen untersucht, die zum Teil redundant gefunden wurden. Dabei handelt es sich beispielsweise um DNA-Transposons, die identische Doppelstrang-bildende Sequenzen besitzen und nicht herausgefiltert wurden, da sie an verschiedenen Positionen oder in verschiedenen Einträgen gefunden wurden. Alle 186 Sequenzen bilden fast ausschließlich Doppelstränge mit 25 bis 29 bp aus (vgl. Abbildung 3.21). Die 82 nicht-redundant vorkommenden Treffer wurden auf dem Gegenstrang von annotierten kodierenden Regionen gefunden. Davon liegen 45 in antisense-Richtung zu Protein-kodierenden Regionen, 16 im Bereich von 5'- oder 3'-UTRs und 5 im Bereich von Introns oder Intron-Exon-Grenzen. Alle übrigen Treffer wurden in extrachromosomalen, palindromischen rRNA-Gen-Sequenzen, in DNA-Transposons und in chromosomalen, nicht-kodierenden Regionen gefunden. Für eine der 45 antisense-RNAs konnte bereits gezeigt werden, dass sie exprimiert wird und in der Zelle regulatorische Funktion hat (Hildebrandt & Nellen, 1992). Durch Sequenz-Alignment der in antisense zu mRNAs passenden Treffer mit ClustalX konnte kein Konsensus gefunden werden. Auch waren die Treffer nicht in bestimmten Bereichen der jeweiligen in antisense annotierten mRNAs zu finden, sondern kommen eher willkürlich in verschiedenen Bereichen der mRNAs vor. Einundzwanzig der 45 auf dem Gegenstrang annotierten mRNAs kodieren für bekannte Proteine (Gräf *et al.*, 2004). Dabei handelt es sich bei der Mehrzahl um regulatorische Proteine, wie Kinasen. Andere kodieren strukturelle Komponenten, die in der Entwicklung von *Dictyostelium discoideum* eine Rolle spielen. Darunter ist auch ein Protein (DrnA), das Homologien zu Dicer aufweist. Die 24 übrigen, unbekannten ORFs wurden mit Hilfe von InterProScan (Zdobnov & Apweiler, 2001) analysiert. Während sich 10 der putativen ORFs nicht näher verifizieren ließen, konnten die restlichen 14 ebenfalls als regulatorische

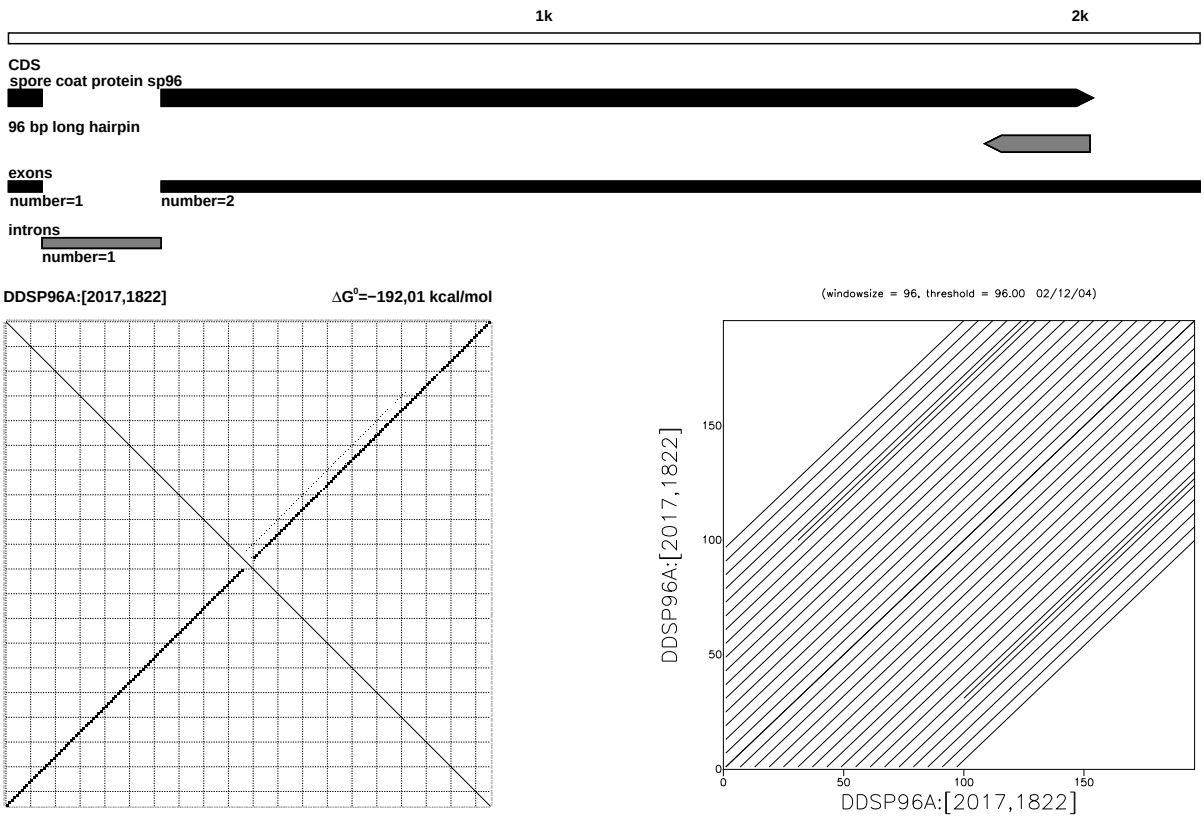


Abbildung 3.22: Lokalisation und Analyse des 96 bp langen Doppelstangs im EMBL-Eintrag `DDSP96A`. Oben: Graphische Darstellung der enthaltenen Sequenz-Annotation. Die kodierende Sequenz des annotierten Sporen-Mantel-Proteins und der Bereich, der den 96 bp langen Doppelstang enthält, sind durch die dicken Balken markiert. Die Orientierung wird durch die Pfeilspitzen symbolisiert. Unten links: Dotplot der mit RNAfold berechneten Sekundärstruktur. Unten rechts: Graphische Darstellung der repetitiven Sequenzbereiche in dem langen Doppelstang. Der Dotplot wurde mit dem Programm dotmatcher erstellt, mit dem hier die Sequenz mit sich selber verglichen wurde. Dabei wird ein Fenster definierter Größe (hier 96 nt) über alle Diagonalen in der Matrix geschoben, in dem jeweils beide Sequenzen miteinander auf Ähnlichkeiten untersucht werden. Ist die Anzahl an Übereinstimmungen in einem Fenster größer oder gleich dem angegebenen Schwellenwert (hier 96 nt), werden an den Positionen, die zwischen den Sequenzen übereinstimmen Punkte (Dots) gemacht. Bereiche in denen aufeinanderfolgende Nukleotide identisch sind, ergeben Diagonalen. Parallele Diagonalen zeigen repetitive Bereiche an.

Proteine identifiziert werden. Zudem konnte festgestellt werden, dass diese mRNAs selber keine Doppelstränge mit mehr als 24 bp ausbilden können. Das unterstreicht die Vermutung, dass die Ausbildung von langen Doppelsträngen in mRNAs während der Evolution unterdrückt wurde, da sonst eine Degradation durch RNAi möglich wäre.

Die Sequenzen von 28 Treffern können Doppelstränge mit 50 bp ausbilden (vgl. Abbildung 3.21). Diese wurden in nur drei verschiedenen Einträgen mit genomischer DNA gefunden (EMBL-IDs `AC117075` (13), `DDSP96A` (13) und `AY171066`(2)). Bei den zwei Treffern im Eintrag `AY171066` handelt es sich um Teilsequenzen des bereits erwähnten extrachromosomalen Palindroms, das die rRNA-Gene in *Dictyostelium discoideum* kodiert (Sucgang *et al.*, 2003). Die beiden anderen Einträge enthalten den identischen Bereich an genomischer DNA von Chromosom 2. Diese 13 Treffer erstrecken sich über einen Be-

reich von 196 nt und sind jeweils um 3, 6 oder 9 nt gegeneinander verschoben. In diesem Bereich ist in antisense zu den Treffern ein Gen für das Sporen-Mantel-Protein SP96 annotiert, wobei die Treffer am 3'-Ende der kodierenden Sequenz liegen (vgl. Abbildung 3.22). Schneidet man den 196 nt langen Bereich aus, über den sich die 13 Treffer erstrecken (`DDSP96A:[2017,1822]`) und berechnet davon die Sekundärstruktur mit RNAfold, erhält man einen vollständig durchgepaarten Doppelstrang mit 96 bp, der von einem Tetraloop abgeschlossen wird. In Abbildung 3.22 ist der zugehörige Dotplot dargestellt. Die Analyse der Sequenz mit dem Programm dotmatcher (Rice *et al.*, 2000) ergibt, dass die Sequenz aus mehreren repetitiven Abschnitten besteht, wie in dem Dotplot in Abbildung 3.22 deutlich an den parallelen Diagonalen zu sehen ist. Die repetitive Natur der Sequenz zeigt sich auch in der resultierenden Aminosäure-Sequenz, die mit Ausnahme von zwei Isoleucinen ausschließlich aus Alanin und Threonin besteht. Bei der Suche der Nukleotidsequenz mit BlastN findet man einige Treffer mit bis zu 75 % Identität in diversen Organismen. Darunter befindet sich auch ein Treffer (74 % Ähnlichkeit) in antisense-Orientierung zu dem Protein DrnA aus *Dictyostelium discoideum*, das Homologien zu Dicer aufweist.

Zusammenfassend kann also festgehalten werden, dass potentielle antisense-Transkripte verschiedener Gene in *Dictyostelium discoideum* lange doppelsträngige Bereiche ausbilden können. Der überwiegende Teil der Kandidaten, die bei der hier beschriebenen Mustersuche identifiziert werden konnten, scheint in Verbindung mit regulatorischen Proteinen zu stehen. Ein RNAi-Mechanismus zur Regulation der Genexpression durch in *cis* kodierte natürliche antisense-Transkripte (NATs) scheint demnach auch in *Dictyostelium discoideum* möglich. Zur Bestätigung dieser Hypothese könnte in Northern-Blot-Analysen von entsprechenden Kandidaten überprüft werden, ob diese *cis*-antisense-Transkripte tatsächlich exprimiert werden. Desweiteren müsste untersucht werden, ob diese Transkripte einen Einfluss auf die Regulation der Expression entsprechender Gene haben. Die hier vorgestellten Ergebnisse stehen in jedem Fall in guter Übereinstimmung mit entsprechenden Erkenntnissen, die aktuell in der Literatur auch für andere eukaryotische Organismen bzgl. der post-transkriptionellen Genregulation durch antisense-Transkripte diskutiert werden (Übersichtsartikel: Lavorgna *et al.*, 2004).

3.5.2 Hammerhead-Ribozyme

Ribozyme sind kleine, katalytische RNA-Motive, die in *cis* oder *trans* das RNA-Phosphatrückgrat an definierter Stelle spalten und ligieren können (Übersichtsartikel: Doudna & Cech, 2002). Die ersten natürlich vorkommenden Ribozyme wurden in einem Gruppe I-Intron aus *Tetrahymena thermophila* (Kruger *et al.*, 1982) und in der Ribonuklease P (RNAse P) RNA von *Bacillus subtilis* (Guerrier-Takada *et al.*, 1983) gefunden. Eine weitere Klasse von natürlich vorkommenden Ribozym-Transkripten entdeckte man in subviralen Pathogenen wie Viroiden oder Virus-Satelliten-RNAs (Birikh *et al.*, 1997; Hutchins *et al.*, 1986; Miller *et al.*, 1991) und in Satelliten-DNA von Amphibien, Saugwürmern und Grillen (Ferbeyre *et al.*, 1998; Pabón-Peña *et al.*, 1991; Rojas *et al.*, 2000; Zhang & Epstein, 1996). Aufgrund ihrer konservierten, einem Hammerkopf ähnelnden Sekundärstruktur werden diese als Hammerhead-Ribozyme (HHRz) bezeichnet (vgl. Abbildung 3.23).

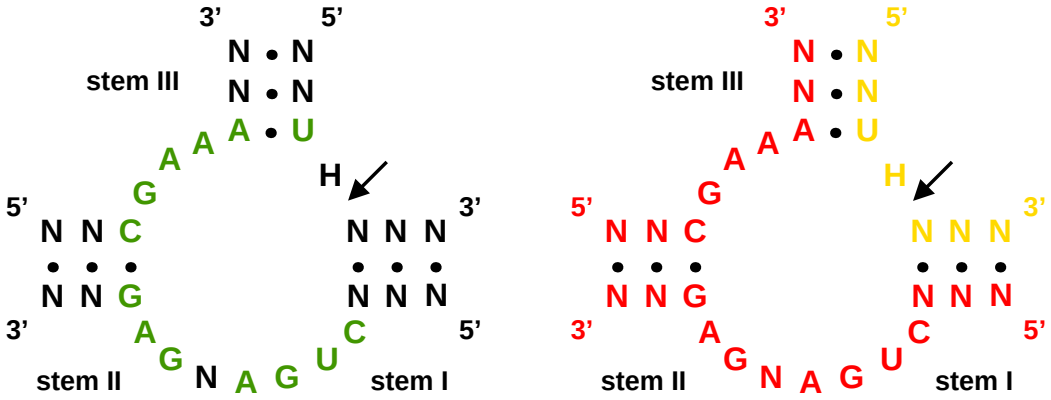


Abbildung 3.23: Sekundärstruktur von Hammerhead-Ribozymen. Konservierte einzelsträngige Bereiche verbinden drei Helices, die im Uhrzeigersinn von I bis III durchnummeriert werden, zu einem Verzweigungsloop. Die Spaltstelle ist mit dem Pfeil markiert. Links sind in gründie konservierten Nukleotide gekennzeichnet. Die rechte Abbildung skizziert die Einteilung des Moleküls in Ribo-„Enzym“ (rot) und Substrat (gelb). H: A, C oder U (nicht G); N: A, C, G oder U.

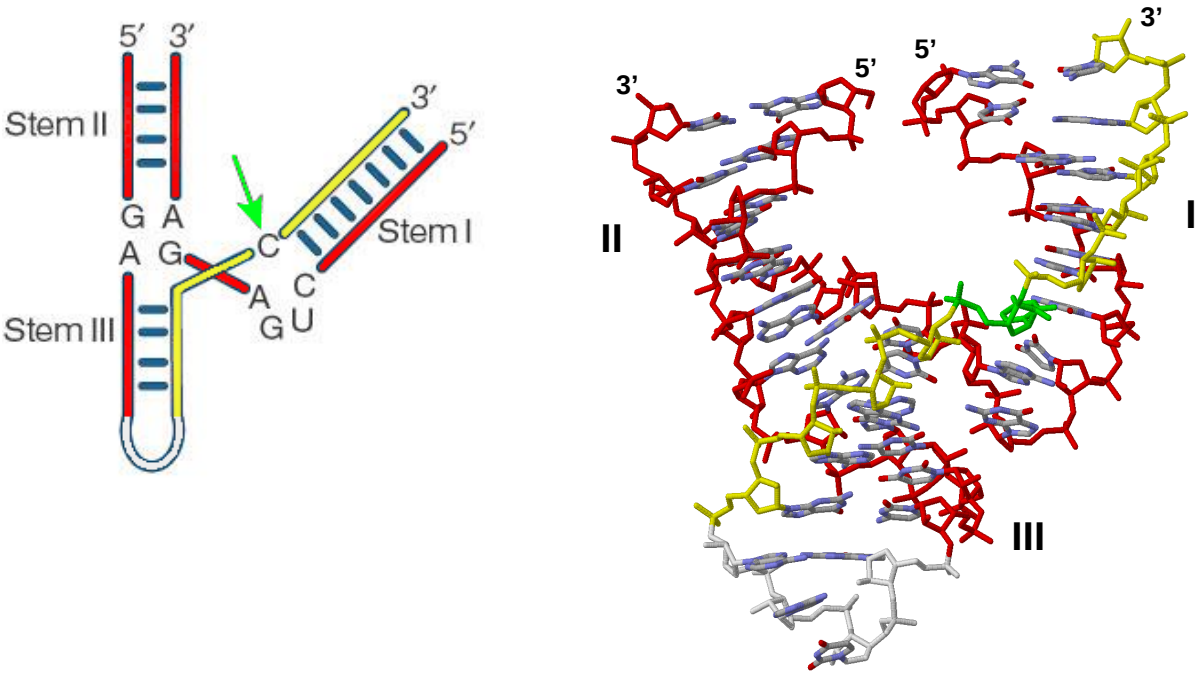


Abbildung 3.24: Dreidimensionale Struktur des Hammerhead-Ribozyms. In dieser Struktur sind die 5'- und 3'-Enden von Helix III durch einen extrastabilen GNRA Tetraloop verbunden. Das Phosphatrückgrat des Ribo-„Enzym“ Teils ist rot eingefärbt, das des Substratstrangs gelb. Die Spaltstelle ist grün markiert. Links: Schematische Darstellung der 3D-Struktur nach Doudna & Cech (2002); verändert. Rechts: Röntgen-Kristallstruktur von Scott *et al.* (1996); verändert.

Die minimale Konsensus-Struktur besteht aus drei Helices, die durch hoch-konservierte, einzelsträngige Bereiche miteinander verbunden sind. Ein Teil des Moleküls bildet das katalytische Zentrum, das mit dem Substratstrang, der die Spaltstelle enthält, basenpaart.

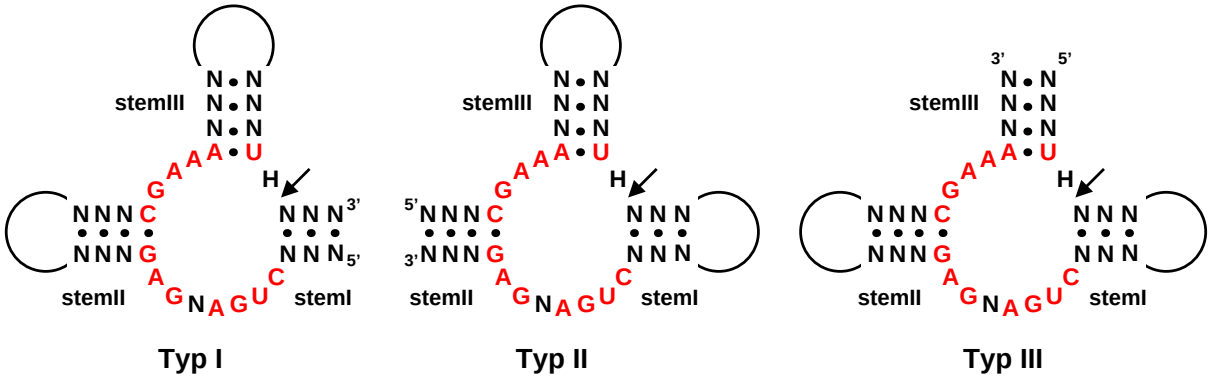


Abbildung 3.25: Sekundärstrukturen der drei *cis*-aktiven Hammerhead-Ribozym-Typen. Definitionsgemäß werden die Helices im Uhrzeigersinn von I bis III durchnummeriert. Die einzelnen Typen werden nach der Helix benannt, die das 5'- und 3'-Ende enthält. H: A, C oder U (nicht G); N: A, C, G oder U.

Die dreidimensionale Kristallstruktur von Scott *et al.* (1996) zeigt eine Y-förmige Konformation, in der vier der sieben konservierten Nukleotide zwischen Helix I und II die Spaltstelle auf dem Substratstrang umgeben, während die anderen drei Nukleotide Basenpaarungen mit dem konservierten einzelsträngigen Bereich zwischen Helix II und III eingehen (vgl. Abbildung 3.24). Des weiteren fällt auf, dass Helix I und II in räumliche Nähe gelangen.

Hammerhead-Ribozyme können ihre katalytische Aktivität sowohl in *trans* als auch in *cis* ausführen. Bei dem von Scott *et al.* (1996) untersuchten bimolekularen Komplex handelt es sich um ein *trans*-aktives HHRz aus einem „enzymatischen" Teil und der „Substrat"-Sequenz. Natürlich vorkommende HHRz, bei denen das gesamte Ribozym aus einem Strang besteht, schneiden dagegen ausschließlich in *cis*. Bei diesen *cis*-aktiven Varianten sind zwei der drei Helices durch Loops mit variablen Längen geschlossen. Abhängig davon, welche Helix das 5'- und 3'-Ende enthält, werden drei Typen von HHRz unterschieden (vgl. Abbildung 3.25), wobei die natürlich vorkommenden HHRz vom Typ III oder Typ I sind.

Die seit nun über 10 Jahren an HHRz durchgeführten Strukturuntersuchungen haben sich fast ausschließlich auf die *trans*-aktive Form dieses RNA-Motivs konzentriert. Neuere Ergebnisse lassen vermuten, dass die wesentlich höhere Effizienz der katalytischen Aktivität der natürlichen HHRz auf zusätzliche Wechselwirkungen zwischen den Loopbereichen von Helix I und II zurückzuführen ist. Während natürlich vorkommende Ribozyme bei physiologischen Mg^{2+}-Ionen-Konzentrationen im sub-mM Bereich katalytisch aktiv sind, werden bei künstlich hergestellten Ribozymen in der Regel deutlich höhere Konzentrationen an Magnesium benötigt. Khvorova *et al.* (2003) und De la Peña *et al.* (2003) konnten zeigen, dass für die zusätzlichen Wechselwirkungen in Typ III-HHRz die folgenden Voraussetzungen gegeben sein müssen:

- Länge von Helix I: 5 bis 7 Basenpaare,
- Größe von Loop I: 3 bis 7 Nukleotide,
- Länge von Helix II: 4 bis 5 Basenpaare,

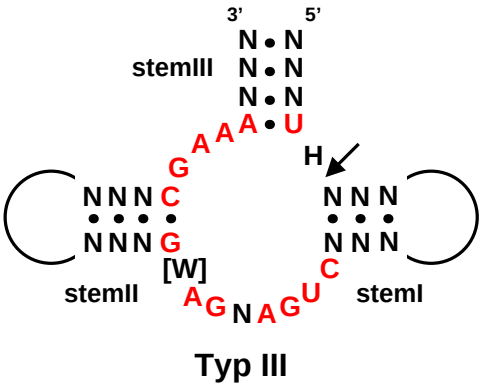
- Größe von Loop II: 4 bis 8 Nukleotide.

Gleichzeitig spielen die Sequenzen in den Loopbereichen, wo die Wechselwirkungen stattfinden sollen, eine Rolle. Garrett *et al.* (1996) vermuteten ähnliche Wechselwirkung bei Typ I-HHRz von Salamandern. Hier allerdings soll die Wechselwirkung zwischen einem internen Loop von Helix I mit Loop II stattfinden. In einigen Fällen sind zur Stabilisierung der Ribozym-Struktur zusätzlich Protein-Komponenten involviert (Denti *et al.*, 2000; Luzi *et al.*, 1997; Ohno *et al.*, 2000).

Auf Basis von Struktur-Alignments und den oben angegebenen Sekundärstruktur-Parametern wurde in dieser Arbeit für jeden HHRz-Typ ein Muster erstellt. Die Alignments enthielten Sequenzen von bekannten HHRz des entsprechenden Typs, die in Sequenzdatenbanken erhältlich waren. Die Optimierung der mit ClustAlX generierten Alignments wurde mit Hilfe von ConStruct durchgeführt. Im Folgenden werden die einzelnen Muster und die bei der Suche über die EMBL-Datenbank (Version 78) erhaltenen Ergebnisse vorgestellt. Von der Suche ausgeschlossen wurden die EST-Datensätze, da darin keine Treffer zu erwarten waren und somit rund 25 % weniger Sequenzdaten durchsucht werden mussten. Als Suchwerkzeug wurde PatScan verwendet, da HyPaSearch bis zum Ende dieser Arbeit nicht mit HHRz-Mustern suchen konnte. Gesucht wurde jeweils auf dem Vorwärtsstrang sowie auf dem reversen Strang.

Typ III-Hammerhead-Ribozyme

Die Mehrheit der natürlich vorkommenden HHRz vom Typ III enthält ein konserviertes, den internen Loop flankierendes G:C-Basenpaar in Helix II, das in der Minderheit in ein A:U-Basenpaar ausgetauscht ist. Abbildung 3.26 zeigt das für die Suche mit PatScan formulierte Muster für Typ III-HHRz mit G:C-Basenpaar in Helix II. Alle Zeilen, die mit einem „`%`“-Zeichen beginnen, enthalten Kommentare, um das Muster lesbarer zu gestalten. Das HHRz-Motiv wird von 5’ nach 3’ formuliert. Definitionsgemäß beginnt die Beschreibung mit dem Musternamen (`hhrz_III_gc`). Es folgt die Definition von drei Basenpaar-Regeln, die weiter unten verwendet werden. Regel `r1` erlaubt Watson-Crick- und Wobble-Basenpaare, Regel `r2` nur U:A-Basenpaare (nicht A:U) und Regel `r3` nur G:C-Basenpaare (nicht C:G). Als erstes werden 4 bis 5 Nukleotide gefolgt von einem U im 5’-Teil von Helix III verlangt (`p31` und `p32`). Dann wird die Spaltstelle definiert, die ein A, C, oder U (`H`: nicht G) sein darf. Es folgt die Definition der Haarnadel-Schleife I (Stemloop I). Diese darf aus 5 bis 7 Basenpaaren in der Helix (`p1`) und 3 bis 13 Nukleotiden im Loop. Als Basenpaar-Regel wird `r1` verwendet, wobei in der Helix eine Fehlpaarung erlaubt ist (`[1,0,0]`). Als nächstes wird ein einzelsträngiger Bereich mit dem Konsensus CUGANGA verlangt. Abweichend vom minimalen Konsensus ist in einigen Ribozymen ein weiteres A oder U (`W`) am 3’-Ende dieses einzelsträngigen Bereichs zu finden, was durch die Formulierung `(W | 0...0)` ausgedrückt wird. Es folgt die Definition von Stemloop II mit einem G (`p21`) gefolgt von 3 bis 4 Nukleotiden im 5’-Teil der Helix (`p22`) und 3 bis 6 Nukleotiden im Loop und dem reversen Komplement zu `p22` und `p21`, wobei die Basenpaar-Regeln `r1` und `r2` angewendet werden. Für die Definition des G:C-Basenpaares hätten die Nukleotide auch einzeln angegeben werden können. Da allerdings die Sekundärstruktur der Treffer in Form




```
% hhrz_III_gc
r1={au,ua,gc,cg,gu,ug} r2={ua} r3={gc}
% stem III
p31=4...5 p32=U
% cleavage site
H
% stemloop I
p1=5...7 3...13 r1~p1[1,0,0]
CUGANGA (W | 0...0)
% stemloop II
p21=G p22=3...4 3...6 r1~p22 r3~p21
GAA
% stem III (rev. compl.)
r2~p32 r1~p31
```

Abbildung 3.26: PatScan**-Muster für Hammerhead-Ribozyme vom Typ III.** Links ist schematisch die Sekundärstruktur von Typ III(G:C)-Hammerhead-Ribozymen dargestellt, die durch das rechts abgebildete PatScan-Muster beschrieben werden. Kommentarzeilen beginnen mit „%“-Zeichen. Die eigentliche Musterbeschreibung ist fett gedruckt.

der Punkt-Klammer-Notation von PatScan mit ausgegeben werden sollte, wurde hier die Komplementfunktion verwendet. In dem Muster für die Klasse mit dem A:U-Basenpaar ist an dieser Stelle das G durch ein A ausgetauscht. Als nächstes wird wieder ein einzelsträngiger Bereich mit dem Konsensus GAA verlangt. Die Musterbeschreibung endet mit dem reversen Komplement zu Helix III unter Verwendung der Basenpaar-Regeln `r2` und `r1`. Mit den in Abschnitt 3.3.5 angegebenen Abschätzungen ist dieses Muster einmal in 6×10^9 Nukleotiden zu finden.

Tabelle 3.8 fasst das Ergebnis der Suche über die EMBL-Datenbank zusammen. Insgesamt wurden 302 Treffer gefunden. Erwartungsgemäß sind rund 60 % der Treffer (180) annotierte HHRz aus viralen Satelliten-RNAs und zwei Viroiden. Die zwei anderen Viroide aus der Gruppe der *Avsunviroidae* (Avocado Sunblotch Viroid (ASBVd) und Chrysanthemum Chlorotic Mottle viroid (CChMVd); Einteilung nach Flores *et al.* (1998)), die ebenfalls HHRz enthalten, konnten nicht gefunden werden. Viroide sind zirkulär geschlossene RNAs. Im Fall von (+)-CChMVd erstrecken sich die enthaltenen HHRz über die Sequenzgrenzen der Einträge hinaus. Verdoppelt man monomere (+)-CChMVd-Sequenzen, werden auch diese HHRz mit dem Muster gefunden. In (-)-Polarität bilden CChMVd-RNAs zwar HHRz vom Typ III aus, allerdings werden diese nicht von dem Muster erfasst, da Helix II 15 Basenpaare, sowie drei Bulgeloops besitzt. Die in ASBVd enthaltenen HHRz sind weder vom Typ III noch vom Typ I. Monomere (+)-ASBVd-RNAs beispielsweise können nur thermodynamisch instabile HHRz ausbilden, die in Helix III nur zwei Basenpaare und im Loop III nur drei Nukleotide besitzen. Dagegen bilden oligomere (+)-ASBVd-RNAs sog. doppelte, hintereinander geschaltete HHRz, die ebenfalls nicht von dem Muster erfasst werden (Flores *et al.*, 1999).

Eine weitere Gruppe an Treffern bilden die synthetischen (SYN), nicht-klassifizierten (UNC) und prokaryotischen Sequenzen (PRO), die etwa 32 % der insgesamt gefundenen Treffer ausmachen. Bei diesen 98 Treffern handelt es sich zum überwiegenden Teil um synthetische Konstrukte, die HHRz als biochemische Werkzeuge enthalten, um z. B. die Expression von rekombinanten Proteinen in prokaryotischen Zellen zu kontrollieren (Che-

Tabelle 3.8: Verteilung der mit dem Muster `hhrz_III_gc` in der EMBL-Datenbank (Version 78) gefundenen Treffer. Link: gruppiert nach Organismen. Rechts: gruppiert nach Abteilungen (FUN: Pilze, GSS: Genome Survey Sequenzen, HTG: Hoch-Durchsatz-Genomsequenzen, HUM: *Homo sapiens*, MUS: *Mus musculus*, PLN: Pflanzen, PRO: Prokaryoten, SYN: synthetische Sequenzen, UNC: nicht klassifizierte Sequenzen, VRL: virale Sequenzen, VRT: Vertebraten).

Organismus	**Anzahl**
Arabidopsis thaliana	11
Cereal yellow dwarf virus satellite RNA	1
Danio rerio	2
Eggplant latent viroid	8
Enterobacteria phage	1
Escherichia coli	4
Expression vector	13
Homo sapiens	7
Lucerne transient streak virus satellite RNA	6
Mus musculus	1
Pea early browning virus (Patent WO02059335)	1
Peach latent mosaic viroid	160
Subterranean clover mottle virus satellite RNA	2
Thanatephorus cucumeris	1
Tobacco rattle virus RNA2-based VIGS vector (SYN)	1
Tobacco ringspot virus	1
Xenopus tropicalis	2
Cherry small circular viroid-like RNA, satellite RNA	1
synthetic construct	46
unclassified sp.	7
unidentified sp.	26
Σ	**302**

Abteilung	**Anzahl**
FUN	1
GSS	6
HTG	3
HUM	6
MUS	1
PLN	6
PRO	4
SYN	61
UNC	33
VRL	180
VRT	1
Σ	**302**

valet *et al.*, 2000). Zu den 83 Treffern in patentierten Sequenzen waren keine weiteren Informationen in den Einträgen verfügbar. In den meisten Fällen sind jedoch zumindest HHRz annotiert.

Die verbleibenden 24 Treffer wurden ausschließlich in Sequenzen von höheren Eukaryoten gefunden. Diese wurden näher untersucht, um festzustellen, ob es sich tatsächlich um HHRz handelt. Um redundante Treffer auszuschließen, wurden die Treffer, die mehrfach in einem Organismus vorkamen, miteinander aligniert und die nichtidentischen Sequenzen mit BlastN über das entsprechende Genom auf der Ensembl-Webseite gesucht (Birney *et al.*, 2004a), wenn nicht explizit eine andere Ressource angegeben ist. Somit konnte die Anzahl der Treffer nochmals reduziert werden und z. T. die Lokalisation im Genom festgestellt werden (+ und − geben die Orientierung von Such-/Treffersequenz des lokalen BlastN-Alignments an):

- *Arabidopsis thaliana* (2 Treffer):
  beide auf Chr. 4 (15027834-15027890 (+/−) und 15032847-15032903 (+/−))[1]

[1] TAIR (Rhee *et al.*, 2003)

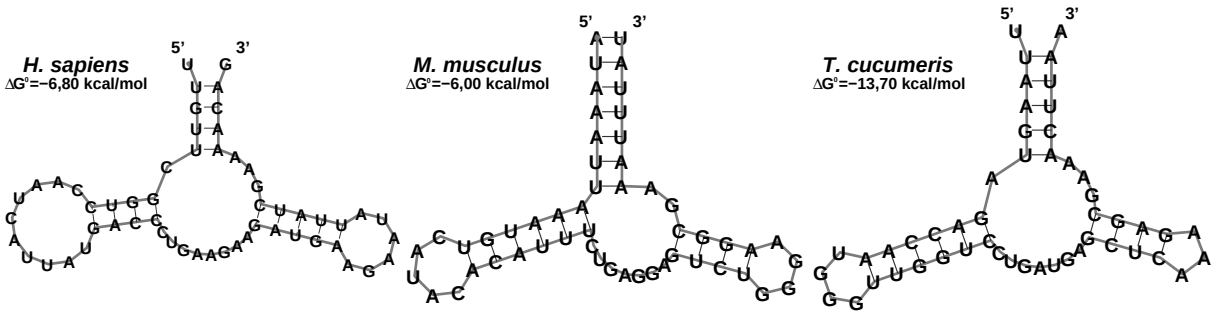


Abbildung 3.27: Sekundärstruktur-Vorhersage der Hammerhead-Ribozyme aus *Homo sapiens* (Chr. 4), *Mus musculus* und *Thanatephorus cucumeris*. Dargestellt sind die mit Hilfe von RNAfold berechneten Strukturen mit minimaler freier Energie. Unter den Organismusnamen ist das ΔG^0 der jeweiligen Struktur angeben.

- *Danio rerio* (2 Treffer):
  Chr. 2 (30421519-30421568 (+/+)) und Chr. 20 (41246587-41246641 (+/+))
- *Homo sapiens* (2 Treffer):
  Chr. 10 (128855253-128855312 (−/+)) und Chr. 15 (27724999-27725054 (−/+))
- *Mus musculus* (1 Treffer):
  Chromosom 15 (84547363-84547414 (+/+))
- *Thanatephorus cucumeris* (1 Treffer):
  Plasmid pRS224; pRS-RT, Gen für reverse Transkriptase (Gegenstrang)
- *Xenopus tropicalis* (2 Treffer mit identischer Sequenz):
  BAC-Klon CH216-90N23 9546-9605 (+/+) und 172896-172955 (+/−)

Um zu überprüfen, ob die Treffersequenzen HHRz-Motive ausbilden können, wurden mit Hilfe von RNAfold die Sekundärstrukturen mit der minimalen freien Energie vorhergesagt. Die Abbildungen 3.27 und 3.28 zeigen die Sekundärstrukturen der Sequenzen von *Homo sapiens* (Chr. 15), *Mus musculus*, *Thanatephorus cucumeris* und *Arabidopsis thaliana*. Alle anderen Sequenzen falteten nicht in Hammerhead-ähnliche Strukturen und wurden als falsch positive Treffer deklariert und somit nicht weiter betrachtet.

Die verbleibenden Sequenzen wurden daraufhin in Kooperation mit Dr. Christian Hammann von der Universität Kassel *in vitro* transkribiert. Er konnte nachweisen, dass mit Ausnahme des menschlichen HHRz alle anderen Ribozyme *in vitro* mit hoher Effizienz katalytisch aktiv sind. Diese Ergebnisse legen die Vermutung nahe, dass es sich bei diesen vier Treffern um neue, noch nicht beschriebene HHRz handelt. Vor allem die beiden Ribozyme aus *Arabidopsis thaliana* wurden daraufhin näher untersucht und charakterisiert, worauf im nächsten Abschnitt detailliert eingegangen wird.

In Tabelle 3.9 sind die Treffer aufgelistet, die mit dem Muster `hhrz_III_au`, das ein A:U- anstatt eines G:C-Basenpaares in Helix II verlangt. Auch hier sind redundante Treffer enthalten, die durch BlastN-Suchen eliminiert und z. T. in den entsprechenden Genomen lokalisiert werden konnten:

- *Bacillus subtilis* (1 Treffer):
  1232981-1232927 (+/−)[2]

Tabelle 3.9: Verteilung der mit dem Muster `hhrz_III_au` in der EMBL-Datenbank (Version 78) gefundenen Treffer. Link: Gruppiert nach Organismen. Rechts: gruppiert nach Abteilungen (GSS: Genome Survey Sequenzen, HTG: Hoch-Durchsatz-Genomsequenzen, HUM: *Homo sapiens*, INV: Invertebraten, PRO: Prokaryoten, UNC: nicht klassifizierte Sequenzen).

Organismus	Anzahl
Bacillus subtilis	2
Brassica oleracea	1
Caenorhabditis elegans	1
Homo sapiens	5
Mus musculus	2
Rattus norvegicus	3
unidentified sp.	1
Σ	**15**

Abteilung	Anzahl
GSS	1
HTG	7
HUM	3
INV	1
PRO	2
UNC	1
Σ	**15**

- *Brassica oleracea* (1 Treffer):
  keine exakte Lokalisation im Genom möglich
- *Caenorhabditis elegans* (1 Treffer):
  Chr. 5 (4836554-4836610 (+/+))
- *Homo sapiens* (3 Treffer):
  Chr. 3 (71921996-71922052 (+/+), EMBL-IDs: `AC105938`, `AC107623` und `AC116352`),
  Chr. 6 (119527516-119527578 (+/+), EMBL-ID: `HS354N19`) und
  Chr. X (111236535-111236585 (−/+), EMBL-ID: `AL772262`)
- *Mus musculus* (1 Treffer):
  Chr. 14 (90176058-90176113 (+/−))
- *Rattus norvegicus*: (1 Treffer):
  Chr. 14 (109101454-109101507 (−/+))
- unidentified sp.: (1 Treffer):
  Bacillus subtilis (1232981-1232927 (+/−)))[2]

Auch diese Treffer wurden hinsichtlich ihrer Fähigkeit HHRz-Motive auszubilden mit RNAfold untersucht. Dabei faltete allerdings keine der Sequenzen in eine HHRz-ähnliche Struktur. Somit wurden diese Sequenzen nicht weiter analysiert und als falsch positive Treffer deklariert, da es sich sehr wahrscheinlich bei den mit diesem Muster gefundenen Sequenzen nicht um genomisch kodierte HHRz handelt.

Typ III-Hammerhead-Ribozyme in *Arabidopsis thaliana*

Die beiden in *Arabidopsis thaliana* gefundenen Sequenzen zeigten in der Sekundärstruktur-Vorhersage eindeutig die Ausbildung von HHRz (vgl. Abbildungen 3.28). Im Vergleich zu den anderen, oben erwähnten Ribozymen besitzen diese beiden HHRz mit einem $\Delta G^0 = -25,73$ kcal/mol die weitaus niedrigste Energie. Die beiden Sequenzen unterscheiden sich nur in zwei Positionen. Während in der siebten Position von Loop I das C in der anderen Sequenz durch ein U ausgetauscht ist, ist in der fünften Position von Loop II

[2] SubtiList (Moszer *et al.*, 2002)

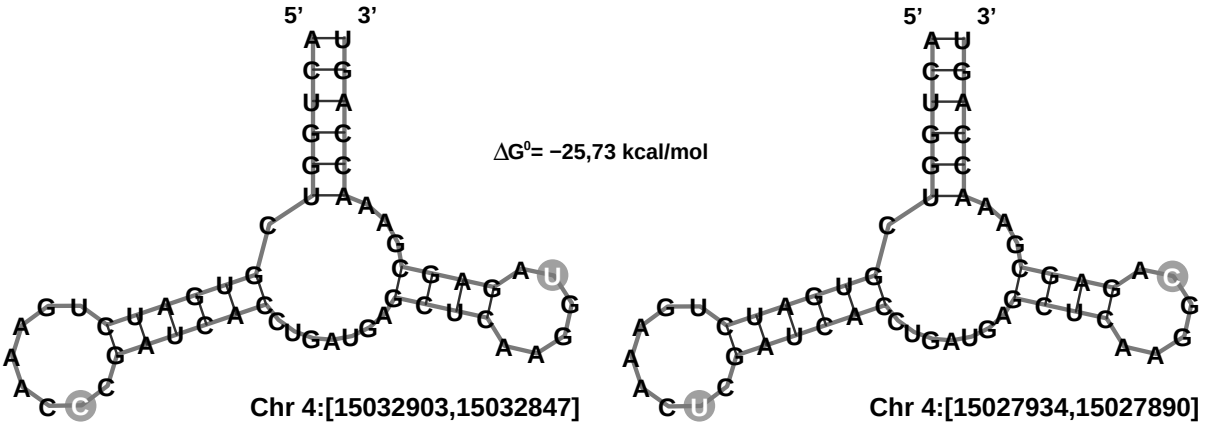


Abbildung 3.28: Sekundärstruktur-Vorhersage der Hammerhead-Ribozyme aus *Arabidopsis thaliana*. Dargestellt sind die mit Hilfe von RNAfold berechneten Strukturen mit minimaler freier Energie. Unter jeder Struktur ist die genaue Lokalisation der Sequenzen auf Chromosom 4 angeben. Die Basenaustausche in den Loops sind grau hinterlegt.

Abbildung 3.29: TAIR SeqViewer: Ausschnitt des Chromosom 4 von *Arabidopsis thaliana*. Dargestellt ist der Bereich des reversen Strangs, in dem die beiden Hammerhead-Ribozyme (rot hinterlegt) gefunden wurden. Annotationen sind nur auf dem Vorwärtsstrang zu finden; orange: Exons; lila: Introns; blau hinterlegt: Start-/Stopp-Kodons; rot: UTR (Rhee *et al.*, 2003, http://arabidopsis.org).

---

das U durch ein C in der zweiten Sequenz ersetzt. Auf dem Hintergrund der Erkenntnisse von Khvorova *et al.* (2003) und De la Peña *et al.* (2003) über Wechselwirkungen zwischen Loop I und II könnte es sich dabei um kompensatorische Basenaustausche handeln. Beide Treffersequenzen liegen auf dem reversen Strang von Chromosom 4 in einem Abstand von etwa 5 kB. Abbildung 3.29 zeigt die Umgebung der Treffer auf Chromo-

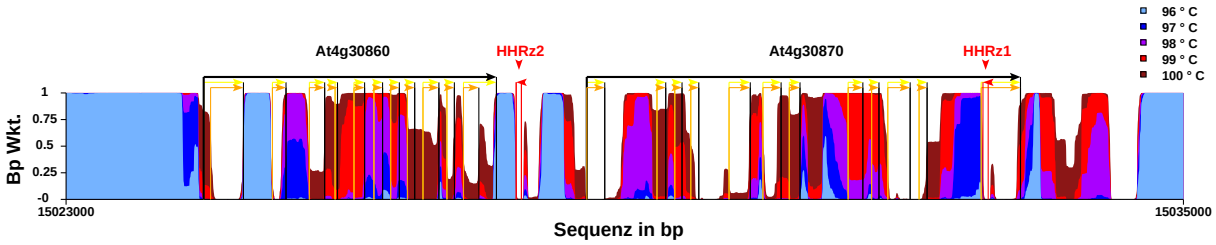


Abbildung 3.30: DNA-Denaturierungsplot des Ausschnitts um die Fundstellen der Hammerhead-Ribozyme. Dargestellt ist der Vorwärtsstrang von Chromosom 4. Die Pfeile geben an auf welchem Strang sich die vorhandene Annotation befindet (vorwärts: →, revers: ←). Die Hammerhead-Ribozyme sind rot markiert, die in diesem Bereich annotierten Proteine schwarz (gelb: CDS, orange: mRNA).

som 4. Auf dem reversen Strang sind in diesem Bereich keine Annotationen vorhanden. Auf dem Gegenstrang, sprich dem Vorwärtsstrang von Chromosom 4, sind in diesem Bereich zwei Protein-kodierende Leseraster annotiert. In dem einen Fall handelt es sich um ein Protein (`AT4G30860`), das eine SET-Domäne[3] besitzt und geringe Ähnlichkeit zum IL-5 Promotor REII-Region-bindenden Protein aus *Homo sapiens* aufweist. Bei dem anderen handelt es sich um ein Protein der Reparatur-Endonuklease-Familie[4] (`AT4G30870`), das in die Reparatur von DNA involviert ist. Beide Annotationen basieren auf multiplen Protein-Sequenz-Alignments und Hidden-Markov-Modellen der Protein-Familien-Datenbank Pfam (Bateman *et al.*, 2004) und wurden automatisch annotiert.

Ein weitere Möglichkeit nicht-kodierende RNAs in genomischer DNA zu finden besteht in der Berechnung von Denaturierungsplots. Das Prinzip, das dahintersteht, beruht auf der Vermutung, dass thermodynamisch stabile Bereiche in DNA mit kodierenden Regionen korrelieren (Yeramian, 2000). Die Berechnung von DNA-Stabilitäts-Plot erfolgte mit einer von Akin (2003) erweiterten Implementation des Poland-Algorithmus (Poland, 1974; Steger, 1994). Abbildung 3.30 zeigt einen solchen Denaturierungsplot. Es fällt auf, dass der Bereich um die Fundstellen der HHRz thermodynamisch deutlich stabiler als ihre Umgebung ist und wirft die Frage auf, wie die Bereiche aussehen, die die HHRz-Sequenzen umgeben.

Daraufhin wurde ein Vergleich der an die HHRz angrenzenden Bereiche via Sequenz-Alignment mit ClustAlX durchgeführt. Wie in Abbildung 3.31 deutlich zu erkennen ist, sind 220 nt vor den Treffern und 25 nt nach den Treffern hoch konserviert. Auf einer Länge von insgesamt 301 nt sind nur 12 Basenaustausche, sowie eine Insertion und eine Deletion zu finden (95 % Identität). Wie bereits erwähnt sind jeweils zwei Austausche in den Loops der Ribozyme zu finden.

In Kooperation mit der Arbeitsgruppe von Dr. Christian Hammann von der Universität Kassel wurden diese Sequenzen hinsichtlich der katalytischen Aktivität sowie der Expression in der Pflanze molekularbiologisch näher charakterisiert (Przybilski *et al.*, 2005). Zunächst wurden die Wildtyp-Sequenzen untersucht. Es konnte nachgewiesen werden,

[3] SET Domänen kommen in Protein-Lysin-Methyltransferase-Enzymen vor.

[4] Besitzt Domänen mit Endonuklease-Aktivität und für Nukleinsäure-Bindung.

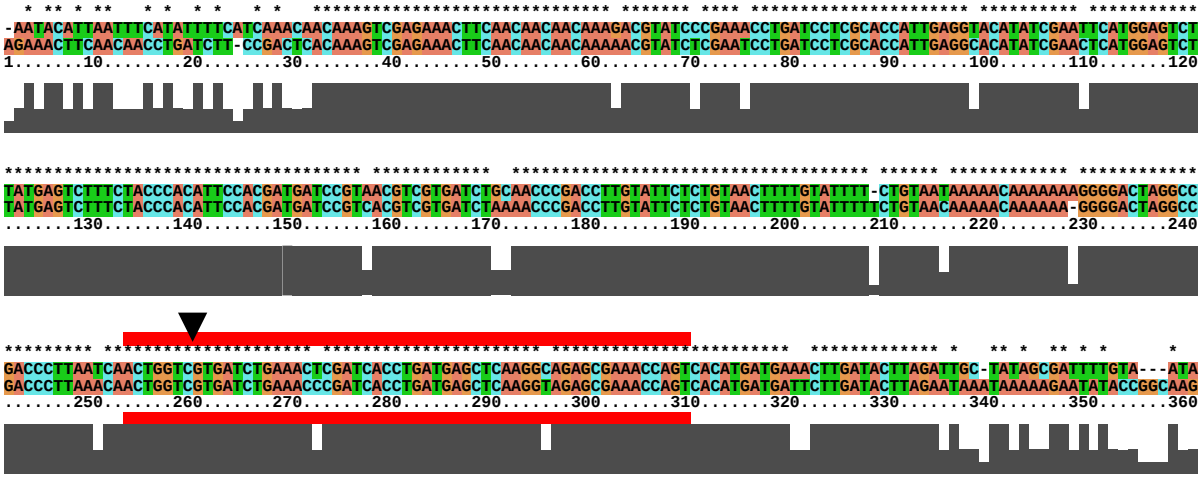


Abbildung 3.31: Sequenz-Alignment der Hammerhead-Ribozyme inkl. angrenzender Bereiche. Das Alignment wurde mit ClustAlX generiert. Die Sequenzen sind in 5'-3'-Richtung (reverser Strang von Chr. 4) abgebildet. Der Grad der Identität wird durch den schwarzen Balken unter dem Alignment angezeigt. Die Hammerhead-Ribozyme sind rot eingerahmt. Die Spaltstellen sind mit dem schwarzen Dreieck markiert.

dass bereits während der *in vitro*-Transkription der Sequenzen die Ribozyme katalytisch aktiv sind. Etwa 95 % der Transkripte wurden bereits während der Transkription prozessiert. Durch die Zugabe von antisense-DNA-Oligonukleotiden kann die Faltung der HHRz-Motive verhindert werden, wodurch Volllängentranskripte hergestellt werden können. Diese Transkripte wurden dann bei 25 °C in 0,6 mM $MgCl_2$ inkubiert. Aus einer Zeitreihe konnte eine Geschwindigkeitskonstante von $k_{obs} = 2,2\ \text{min}^{-1}$ abgeschätzt werden. Diese hohe Spaltreaktionsrate bei niedriger (physiologischer) Mg^{2+}-Ionen-Konzentration lässt vermuten, dass auch in diesen beiden Ribozymen Loop-Loop-Wechselwirkungen eine Rolle spielen. Um den Einfluss der Loopsequenzen auf die Spaltreaktion zu untersuchen wurden eine Reihe von Loop-Varianten hergestellt, die Mutationen oder Deletionen in einem oder beiden Loops besitzen. Während in einigen Varianten katalytische Aktivität vollständig verloren ging, zeigten andere im Vergleich zu den Wildtyp-Sequenzen geringere Spaltaktivität unter nahezu physiologischen Mg^{2+}-Konzentration. Andere Veränderungen dagegen ergaben eine erhöhte Spaltreaktionsrate. Tauscht man beispielsweise die variablen Positionen (CU und UC, vgl. Abbildung 3.28) in den Loops in UU bzw. CC, wird die Spaltungsrate nur geringfügig vermindert. Somit handelt es sich bei den variablen Positionen nicht um kompensatorische Basenaustausche, wie ursprünglich vermutet. Andererseits konnte eine erhöhte Aktivität durch Deletion eines A in Loop I erreicht werden, was dafür spricht, dass die Ausbildung eines stabilen GNRA Tetraloops in Loop I nicht für die Loop-Loop-Wechselwirkung notwendig ist.

Um nun zu überprüfen, ob diese Sequenzen *in vivo* transkribiert werden oder nur auf DNA-Ebene in der Pflanze vorkommen, wurde RNA aus Blättern, Blüten, Wurzeln und Stängeln isoliert. Mit Ausnahme der Wurzeln konnten in allen anderen Geweben mittels RT-PCR entsprechende RNA-Transkripte nachgewiesen werden. Diese Experimente ließen allerdings noch keine Aussage zu, ob die Ribozyme *in vivo* auch katalytisch aktiv sind. Um diese Frage zu beantworten wurde die S1-Nuklease-Methode verwendet. Dazu

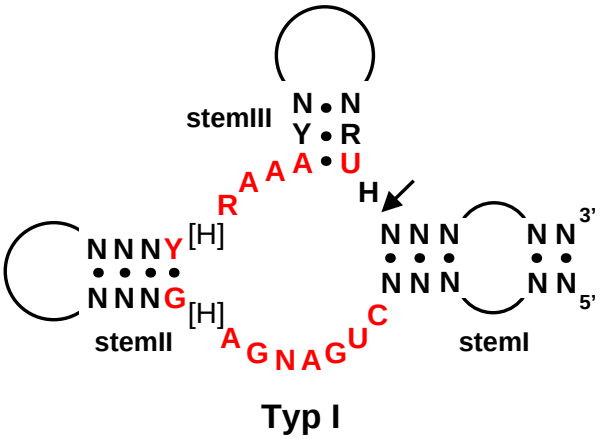


Abbildung 3.32: Sekundärstruktur von erweiterten Typ I-Hammerhead-Ribozymen. Die schematische Darstellung der Sekundärstruktur fasst die in der Literatur erhältlichen Sequenz- und Struktur-Eigenschaften zusammen. Die konservierten Nukleotide sind rot eingefärbt. Einige Nukleotid-Positionen werden mit Hilfe der IUPAC-Nomenklatur (vgl. Tabelle 3.2) beschrieben. Die Spaltstelle ist durch den Pfeil gekennzeichnet.

wurden radioaktiv markierte antisense-RNAs hergestellt, die gegen die Ribozymsequenzen hybridisiert wurden. Findet die Spaltungsreaktion *in vivo* statt, so sind nach der Behandlung des Ansatzes mit der S1-Nuklease verkürzte antisense-RNAs zu erwarten, da einzelsträngige Bereiche von der Nuklease degradiert werden. Mit dieser Methode konnten sowohl geschnittene als auch ungeschnittene Ribozym-Sequenzen in Blättern, Blüten und Stängeln eindeutig nachgewiesen werden. Summa summarum belegen diese Ergebnisse, dass in *Arabidopsis thaliana* genomisch kodierte RNAs mit HHRz-Motiven gewebespezifisch exprimiert werden und dass diese in der Pflanze sowohl in der prozessierten also auch in der unprozessierten Form vorkommen.

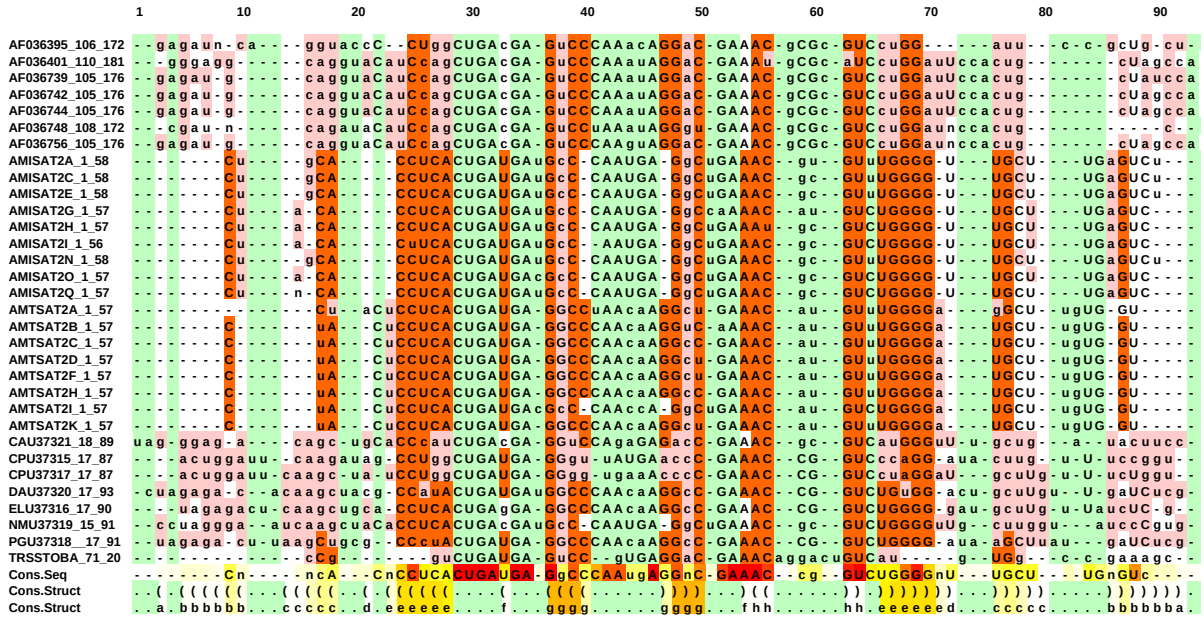
Typ I-Hammerhead-Ribozyme

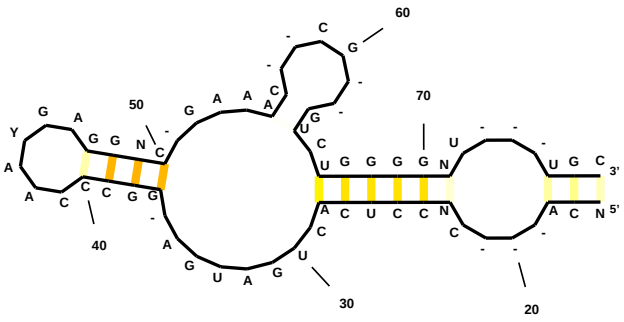
HHRz vom Typ I wurden bislang in ASBVd (Hutchins *et al.*, 1986), in Satelliten-RNAs des *Tobacco ringspot*-Virus (sTobRV, Haseloff & Gerlach (1989)) und in Transkripten repetitiver Satelliten-DNA von Amphibien (Green *et al.*, 1993; Pabón-Peña *et al.*, 1991; Zhang & Epstein, 1996) und Saugwürmern (Trematoden, Ferbeyre *et al.*, 1998) beschrieben. Die Sekundärstruktur dieser Ribozyme ist in Abbildung 3.32 dargestellt. Im Gegensatz zu Typ III-HHRz enthält Helix I das 5'- und 3'-Ende der Sequenz. Die in der Literatur beschriebenen Ribozyme dieses Typs weisen zudem Unterschiede in dem konservierten Verzweigungsloop auf. So darf das an den Verzweigungsloop angrenzende Basenpaar in Helix II ein G:C- oder C:G-Basenpaar sein. In einigen Ribozymen ist ein zusätzliches Nukleotid (H: nicht G) am 3'-Ende von Helix II insertiert. In dem einzelsträngigen Bereich zwischen Helix II und III kann anstelle des G auch ein A (R: A oder G) vorkommen. In den meisten Fällen besteht die Helix III nur aus zwei Basenpaaren und der zugehörige Loop enthält nur zwei Nukleotide, wodurch die Sekundärstruktur in diesem Bereich thermodynamisch sehr instabil ist. In diesen Fällen spielen sehr wahrscheinlich RNA-bindende Proteine eine Rolle, die die aktive Konformation stabilisieren (Denti *et al.*, 2000; Luzi *et al.*, 1997) oder diese Ribozyme sind als Doppel-HHRz-Struktur katalytisch aktiv (Davies *et al.*, 1991; Forster *et al.*, 1988). Desweiteren ist die Helix I durch einen internen Loop erweitert. Auch in dieser Konformation scheinen Wechselwirkungen zwischen Loopbereichen für die katalytische Aktivität von Bedeutung zu sein. Garrett *et al.* (1996) konnten mit Mutationsanalysen an HHRz aus Salamandern zeigen, dass Sequenzbereiche des inter-

nen Loops von Helix I die Spaltaktivität beeinflussen. Damit wäre vorstellbar, dass auch hier Wechselwirkungen zwischen den Loopbereichen von Helix I und II stattfinden, wie es für Typ III HHRz gezeigt wurde (De la Peña *et al.*, 2003; Khvorova *et al.*, 2003).

Exemplarisch soll an dieser Stelle die Vorgehensweise bei der Erstellung von strukturbasierten Mustern am Beispiel der Typ I-HHRz beschrieben werden. Aus der o. a. Literatur konnten erste Informationen über charakteristische Sekundärstruktur-Eigenschaften dieser Ribozyme entnommen werden. Zudem standen 32 zugehörige Sequenzen von Amphibien (*Ambystoma talpoideum* (8), *Ambystoma tridactylum* (9), *Plethodon glutinosus* (1), *Necturus maculosus* (1), *Eurycea longicauda* (1), *Desmognathus apalachicolae* (1), *Cynops pyrrhogaster* (2), *Cryptobranchus alleganiensis* (1)), Trematoden (*Schistosoma mansoni* (5), *Schistosoma haematobium* (1), *Schistosomatium douthitti* (1)) und sTobRV(1) aus der Sequenzdatenbank Genbank (Benson *et al.*, 2004) zur Verfügung. Über diese Sequenzen wurde dann mit einem initial erstellten Muster gesucht, das die Sequenz- und Struktur-Eigenschaften, die in der Literatur beschrieben sind, zusammenfasst. Zur Verfeinerung der Musterbeschreibung wurden diese HHRz-Sequenzen als nächstes in das Werkzeug ConStruct (Lück *et al.*, 1999) importiert. Mit diesem Programm kann man interaktiv Struktur-Sequenz-Alignments erstellen. Dabei werden das multiple Sequenz-Alignment, das z. B. mit ClustalX generiert wird, und die Basenpaarungsmatrizen aus der Sekundärstruktur-Vorhersagen der Sequenzen mit RNAfold gemeinsam in einer Basenpaarungsmatrix visualisiert. Der Benutzer kann dann mit Hilfe des integrierten Alignment-Editors das optimale Struktur-Alignment der Sequenzen herstellen, in dem sowohl Sequenz- als auch Struktur-Eigenschaften der betrachteten RNAs berücksichtigt werden. Da die Ausgabe der Treffer, die bei einer Mustersuche gefunden werden, den Mustereinheiten entsprechend in Blöcken ausgegeben werden (vgl. Abschnitt 3.3.2), wurde das Programm `PatScan2Aln.pl` geschrieben, das die Treffer einliest und ein den Sequenz-Blöcken entsprechendes Alignment der Treffersequenzen ausgibt. Dieses kann dann zusammen mit den Basenpaarungsmatrizen aus der Sekundärstruktur-Vorhersage vom ConStruct-Programm eingelesen werden. Im nächsten Schritt folgt die Optimierung des Struktur-Sequenz-Alignments durch Schieben einzelner Nukleotide oder ganzer Nukleotid-Blöcke im Alignment-Editor.

In Abbildung 3.33 ist das optimierte Alignment zusammen mit der Konsensus-Sekundärstruktur für die 32 natürlich vorkommenden Typ I-HHRz dargestellt. Da die Helix III wie bereits beschrieben thermodynamisch sehr instabil ist, wurden hier zur Berechnung der Matrizen die entsprechenden zwei Basenpaare vorgegeben (Helix b im Alignment). Wie dem Alignment deutlich zu entnehmen ist, können alle Sequenzen die HHRz-Struktur ausbilden. Aus diesem Alignment konnte somit die finale Musterbeschreibung für die Suche mit PatScan abgeleitet werden, die ebenfalls in Abbildung 3.32 gezeigt ist. Beginnend mit Helix c (1. Teil von Helix I) haben alle Sequenzen zwei bis sechs Basenpaare. Der 5'-Teil des internen Loops enthält null bis vier Nukleotide; das einzelne Basenpaar d wird aufgrund der geringen Basenpaarungswahrscheinlichkeit nicht berücksichtigt. Im zweiten Teil von Helix I sind zwei bis sechs Nukleotide gepaart. Aus der Kombination beider basengepaarter Teile könnten Treffer gefunden werden, die nur aus insgesamt vier Basenpaaren bestehen. Solche Treffer hätten allerdings mit großer Wahrscheinlichkeit eine sehr geringe






```
% hhrz_I_ext
r1={au,ua,gc,cg,gu,ug}
% ext. stem I
(p11=2...6 0...4 p12=4...6 |
 p13=4...6 0...4 p14=2...6
CUGANGA (H | 0...0)
% stemloop II
p21=G p22=2...3 3...6 r1~p22 r1~p21
(H | 0...0) RAA
% stemloop III
p31=A p32=Y 2...6 r1~p32 r1~p31
% cleavage site
H
% ext. stem I (rev. compl.)
(r1~p12 0...3 r1~p11 |
 r1~p14 0...3 r1~p13)
```

Abbildung 3.33: Struktur-Sequenz-Alignment und Konsensus-Sekundärstruktur von 32 natürlich vorkommenden Typ I-Hammerhead-Ribozymen und die daraus abgeleitete Musterbeschreibung. Das Alignment und die Sekundärstruktur wurden mit dem ConStruct-Programm (Lück *et al.*, 1999) generiert. Links neben den Alignment sind die Bezeichner der Sequenzen mit Start- und Endposition in dem jeweiligen Genbank-Eintrag angegeben. Einzelsträngige Bereiche sind grün oder weiß hinterlegt, wobei letztere in ansonsten basengepaarten Bereichen liegen. Nach Konsensus basengepaarte Bereiche sind rot bzw. rosa markiert, wobei hier letztere nach dem Alignment-Konsensus kompensatorische Basenpaaraustausche sind. In der drittletzten Zeile ist die Konsensus-Sequenz angegeben. Die letzten beiden Zeilen deuten die Konsensus-Struktur mittels Punkt-Klammer-Notation und in alphabetischer Kodierung an. Der Grad der Konservierung (Konsensus-Sequenz) bzw. die Basenpaarungswahrscheinlichkeit (Konsensus-Struktur) ist in allen drei Zeilen farblich von weiß (0 %) über gelb und orange nach rot (100 %) gekennzeichnet. Diese Farbkodierung ist in der Konsensus-Struktur, die links unter dem Alignment dargestellt ist, ebenfalls wiederzufinden. Die hier angegebene Sequenz entspricht der Konsensus-Sequenz. Daraus wurde das unten rechts abgebildete PatScan-Muster erstellt. Kommentarzeilen beginnen mit „%“-Zeichen. Die eigentliche Musterbeschreibung ist fett gedruckt.

thermodynamische Stabilität und sollen daher bei der Suche nicht gefunden werden. Aus diesem Grund wurde die Definition von Helix I mit der Oder-Verknüpfung formuliert. Als

unteres Limit wurde die in sTobRV gefundenen Helixlängen zugrunde gelegt, d. h. wenn in einem Teil nur zwei Basenpaare vorkommen muss der andere Teil mindestens drei Basenpaare enthalten und umgekehrt. Es folgt der konservierte, einzelsträngige Bereich mit dem Konsensus CUGANGA, der das katalytische Zentrum der Ribozyme bildet. In einigen Sequenzen ist hinter diesem einzelsträngigen Bereich ein zusätzliches Nukleotid insertiert. Bei den im Alignment verglichenen Sequenzen handelt es sich dabei ausschließlich um ein Pyrimidin. In der Literatur ist allerdings auch zu finden, dass an dieser Position ein A vorkommen kann, wie es z. B. bei Typ III-HHRz der Fall ist. Somit wurde hier ein H (nicht G) in die Musterbeschreibung eingesetzt. Es folgt Helix II, die in jeder Sequenz mir einem G:C- oder G:U-Basenpaar beginnt, gefolgt von zwei oder drei beliebigen Basenpaaren. Der Loop II besteht in allen Fällen aus vier bis sechs Nukleotiden, wobei z. B. bei den Sequenzen aus *Cynops pyrrhogaster* (`CPU37315_17_87` und `CPU37317_17_87`) die Randnukleotide ein weiteres Basenpaar bilden können. In dem Muster werden von daher drei bis sechs Nukleotide in Loop II erlaubt. Am 3'-Ende von Helix II ist wiederum in einigen Sequenzen des Alignment ein zusätzliches Pyrimidin vorhanden. In Sequenzen aus Molchen bildet an dieser Stelle ein A ein zusätzliches Basenpaar mit einem U vor dem 5'-Ende von Helix II. Deshalb wird in der Beschreibung an dieser Stelle ebenfalls ein H (nicht G) erlaubt. Es folgt der Konsensus GAA, wobei in einer Sequenz von *Ambystoma tridactylum* (`AMISAT2G_1_57`), das G durch ein A ersetzt ist. In dem Muster wird deshalb hier der Konsensus RAA definiert. Dem Alignment kann außerdem entnommen werden, dass das zweite A ein Basenpaar mit dem fünften Nukleotid des einzelsträngigen Konsensus CUGANGA eingehen kann (Basenpaar f im Alignment). Dieses Basenpaar ist auch in der Röntgenstruktur von Scott *et al.* (1996) enthalten. Die zwischen diesem Basenpaar und Helix II vorhandenen Purine bilden in der Röntgenstruktur ebenfalls nicht-kanonischen G:A- und A:G-Basenpaare aus, die allerdings mittels Sekundärstruktur-Vorhersage aufgrund von fehlenden thermodynamischen Parametern nicht vorhergesagt werden können und von daher auch nicht im Struktur-Sequenz-Alignment erscheinen. Helix III besteht mindestens aus zwei Basenpaaren, wobei das erste immer ein A:U- und das zweite ein C:G-Basenpaar ist, wobei letzteres kompensatorisch durch ein U:A-Paar ersetzt sein kann (`p31=A p32=Y ... r1~p32 r1~p31` in der Musterbeschreibung). Weitere Loopnukleotide werden durch die Definition des Loop III (`2...6`) erfasst. Die nächste Position nach Helix III ist die Spaltstelle (Position 65 im Alignment). In der Mehrzahl der Sequenzen findet man an dieser Stelle ein C, etwas seltener ist hier ein U vorhanden. Aus der Literatur ist außerdem bekannt, dass wie z. B. in der Satelliten-RNA des Barley yellow dwarf Virus (Miller *et al.*, 1991) oder im Lucerne transient streak Virus (ein Virusoid, Forster & Symons, 1987) ein A die Spaltstelle markiert. Damit ist an dieser Position jedes Nukleotid mit Ausnahme eines G (`H`) erlaubt. Es folgen die komplementären 3'-Teile von Helix I, die durch null bis drei einzelsträngige Nukleotide verbunden werden. Mit diesem Muster wurde dann erneut über die im Alignment enthaltenen Sequenzen gesucht, um sicherzustellen, dass alle Sequenzen gefunden werden. Dann wurde damit über die EMBL-Datenbank (Version 78) exklusive der EST-Sequenzdaten überlappend in beiden Polaritäten gesucht. Bei statistischer Gleichverteilung der Nukleotide in den zu durchsuchenden Sequenzdaten ist alle $3{,}8 \times 10^7$ Nukleotide ein Treffer mit diesem Muster zu erwarten. Bei der überlap-

penden Suche mit PatScan kann es vorkommen, dass einige Treffer mehrfach gefunden werden. Daher wurden die gefundenen Treffer anschließend gefiltert. Das dafür geschriebene Programm `sort_olapping.pl` sortiert alle die Treffer aus, die vollständig in einem anderen Treffer enthalten sind. Aufgrund der variablen Bereiche, die für die Beschreibung von Helix I verwendet wurden, lieferte PatScan weitere überlappende Treffer, von denen ebenfalls jeweils die kürzeren aussortiert wurden.

Die Verteilung der 1413 in der EMBL-Datenbank gefundenen Treffer ist in Tabelle 3.10 und Tabelle 3.11 aufgelistet. Rund 62 % der Treffer wurden in Einträgen der Abteilungen GSS, HTC, HTG. Dabei handelt es sich um Sequenzierungsdatensätze, die sich meist nicht eindeutig zuzuordnen lassen und in der Regel keinerlei Annotation enthalten. Etwa 70 % der Treffer verteilen sich auf die Organismen *Aedes aegypti* (13), *Arabidopsis thaliana* (10), *Drosophila melanogaster* (14), *Danio rerio* (228), *Homo sapiens* (274), *Mus musculus* (131), *Rattus norvegicus* (113) und *Xenopus tropicalis* (209), welche in dieser Datenbank-Version z. T. vollständig sequenziert vorlagen. Demnach ist zu erwarten, dass in diesen Gruppen etliche redundante Treffer enthalten sind. Um eine Vorstellung von der enthaltene Redundanz zu bekommen, wurden mit Hilfe des Treffer-Browsers (vgl. Abschnitt 3.4.5) entsprechende Treffer nach der Sequenzen geordnet distinkt selektiert. Im Fall des Zebrafische ergab sich z. B., dass es sich bei den insgesamt 228 gefundenen Treffer (GSS: 8, HTG: 110, VRT: 110) nur um 111 verschiedene Sequenzen handelt. Dabei ist allerdings nicht berücksichtigt, dass identische Sequenzen auch mehrfach im Genom enthalten sein können.

Wie erwartet wurden alle im Alignment enthaltenen Satelliten-DNA-Sequenzen aus den Amphibien (*Ambystoma talpoideum*, *Ambystoma tridactylum*, *Cryptobranchus alleganiensis*, *Cynops pyrrhogaster*, *Desmognathus apalachicolae*, *Eurycea longicauda* und *Plethodon glutinosus*) gefunden, sowie drei weitere (*Ambystoma maculatum*, *Necturus maculosus* und *Notophthalmus viridescens*), die ebenfalls in der Literatur beschrieben sind. Desweiteren wurden die Satelliten-DNAs aus den Trematoden (*Schistosomatium douthitti*, *Schistosoma haematobium* und *Schistosoma mansoni*), sowie die Satelliten-RNA des TobRV gefunden. Auffällig ist, dass viele Treffer in Repeat-Regionen höherer Organismen, wie *Homo sapiens*, *Mus musculus*, *Brassica oleracea*, oder *Danio rerio* gefunden werden konnten.

Da sich die Analyse dieser Menge an gefundenen Treffern aufgrund der enthaltenen Redundanz und der z. T. noch nicht verfügbaren Annotation schwierig gestaltet, wurde versucht mittels Sekundärstruktur-Vorhersage weitere interessante Treffer heraus zu filtern. Dazu wurde RNAfold verwendet, um abzuschätzen, ob eine Treffersequenz ein HHRz-Motiv mit entsprechender thermodynamischer Stabilität ausbilden kann. Zu diesem Zweck wurde das Programm (`rnafold.pl`) geschrieben, das automatisch alle übergebenen Treffersequenzen mit RNAfold unter Verwendung der angegebenen Parameter faltet. Als Ausgabe erhält der Benutzer die Verteilung der berechneten Energien. Die berechneten Sekundärstrukturen und Basenpaarungsmatrizen werden in einer Acrobat-Reader-Datei zusammengefasst, wodurch eine Vielzahl von Treffern im Überblick analysiert werden kann. Der Benutzer kann zudem Grenzen angeben, in denen der Wert für die freie Energie einer Struktur liegen soll, um angezeigt (in die PDF-Datei übernommen) zu werden.

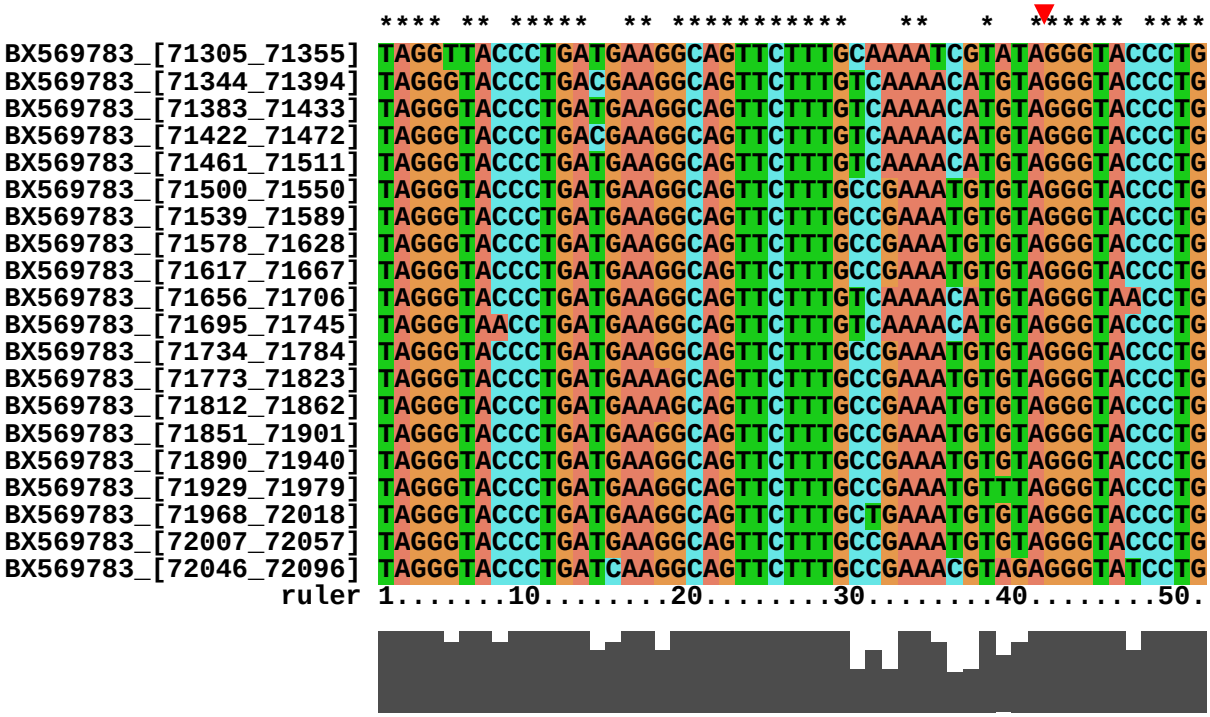


Abbildung 3.34: Sequenz-Alignment der 20 in Tandem angeordneten Hammerhead-Ribozyme auf Chr. 20 von *Danio rerio*. Das Alignment wurde mit ClustAlX erstellt. Alle hier dargestellten Sequenzen wurden im EMBL-Eintrag `BX569783` gefunden. Die jeweiligen Start- und End-Positionen sind in eckigen Klammern angeben. Alle Sequenzen überlappen sich um exakt 11 Nukleotide mit der unmittelbar benachbarten Sequenz. Die vermeintliche Spaltstelle ist durch das rote Dreieck markiert. Der Konsensus CUGANGA liegt zwischen Position 10 und 16, das Purin-Quartett RAAA an Position 32 bis 35.

Im Fall der Typ I-HHRz wurden aus den bereits beschiebenen Gründen die zwei Basenpaare in Helix III für die Berechnung vorgegeben. Die Berechnungen wurden bei 25 °C (mit den Parametern `-p -d3 -C`) durchgeführt. Die vorhergesagten Strukturen besitzen Energien zwischen −0.63 und −31.07 kcal/mol. Unter diesen fielen v. a. 14 Treffer auf, die im Zebrafisch gefunden werden konnten und in nur einem Eintrag (ID: `BX569783`) enthalten waren. Im Mittel hatten diese Treffer ein ΔG^0 von etwa −30 kcal/mol. Alle diese Treffer zeigten die Ausbildung eines HHRz-Motivs und wiesen nur geringe Sequenzunterschiede auf. Ebenfalls auffällig war, dass sich diese Treffer z. T. sowohl am 5'- als auch am 3'-Ende um jeweils 11 Nukleotide überlappen. Daraufhin wurden diese Treffer auf der Ensembl-Webseite (Birney *et al.*, 2004a) mit BlastN im Zebrafisch-Genom gesucht. In Tabelle 3.12 ist das Ergebnis dieser BlastN-Suche für einem Treffer auf Chr. 20 aufgelistet. Es handelt sich hier um 20 in Tandem angeordnete HHRz vom Typ I, von denen nur 6 nicht durch die Musterbeschreibung erfasst wurden. Abbildung 3.34 zeigt das Alignment dieser 20 Sequenzen, die durch die BlastN-Suche gefunden wurden. Daraus wird deutlich, dass jede Sequenz um exakt 11 Nukleotide mit der jeweils benachbarten Sequenz überlappt. Aufgrund der Position der Schnittstelle ergäbe sich also für die katalytische Aktivität, dass das abgespaltene 3'-Ende gleichzeitig das 5'-Ende des nächsten Ribozyms bildet. Es könnten somit bei der Transkription dieses repetitiven Bereichs 19 definierte Fragmen-

te mit 30 Nukleotiden Länge entstehen. Auch in diesem Fall stehen die gefunden HHRz anscheinend mit repetitiven Sequenzen in Verbindung.

Weitere interessante Treffer konnten bis zum Ende dieser Arbeit in dieser Datenmenge nicht gefunden werden. Diese Suche zeigt ganz deutlich, welche Probleme bei der Auswertung auftreten können. Ein komplexes Muster, wie das hier verwendete, kann eine große Menge an Treffern zurückliefern. Die Analyse der Treffer wird stark durch die enthaltene Redundanz in der Sequenzdatenbank erschwert. Diese kann nur aufwendig verringert werden. Liegt z. B. das entsprechende Genom sequenziert vor, könnte mit dem Muster oder mit BlastN erneut darüber gesucht werden, um die genaue Lokalisation sowie die im Bereich der Treffer vorhandene Annotation auszuwerten.

Tabelle 3.10: Verteilung der mit dem Muster `hhrz_I_ext` in der EMBL-Datenbank (Version 78) gefundenen Treffer gruppiert nach Organismen.

Organismus	Anzahl
Acipenser fulvescens	1
Aedes aegypti	13
Ambystoma maculatum	1
Ambystoma talpoideum	8
Amphiuma tridactylum	9
Anomalospiza imberbis	1
Anopheles gambiae	4
Arabidopsis thaliana	10
Bacillus sp. BSG40	1
Bacillus subtilis	2
Bacillus subtilis subsp.	1
Bacteroides thetaiotaomicron	1
Barbatia reeveana	1
Bos taurus	3
Brassica oleracea	3
Bungarus multicinctus	1
Caenorhabditis briggsae	3
Caenorhabditis elegans	8
Candida albicans	1
Canis familiaris	10
Capra hircus	1
Cryptobranchus alleganiensis	1
Cryptosporidium parvum	1
Cynops pyrrhogaster	2
Danio rerio	228
Debaryomyces hansenii	4
Desmognathus apalachicolae	1
Dictyostelium discoideum	1
Didelphis virginiana	1
Diploptera punctata	1
Drosophila melanogaster	14
Drosophila sp.	4
Entamoeba histolytica	3
Enterococcus faecalis V583	1
Eurycea longicauda	1
Felis catus	1
Gallus gallus	10
Glycine max	2
Haemophilus influenzae	5
Homo sapiens	274
Hordeum vulgare	2
Human rhinovirus 9	1
Hypocrea jecorina	14
Hypocrea koningii	2
Legionella pneumophila	1
Lotus corniculatus var.	1
Macaca fascicularis	1
Medicago truncatula	4
Mus musculus	131
Mytilus edulis	1
Necturus maculosus	1
Neisseria meningitidis	2
Neisseria meningitidis MC58	1
Neisseria meningitidis Z2491	1
Nicotiana tabacum	1
Nostoc sp. PCC	1
Notophthalmus viridescens	1
Oikopleura dioica	1
Ornithorhynchus anatinus	1
Oryza sativa	13
Otolemur garnettii	4
Pan troglodytes	14
Papio anubis	4
Plethodon glutinosus	1
Poncirus trifoliata	1
Pristionchus pacificus	2
Rattus norvegicus	113
Saccharomyces cerevisiae	2
Saccharomyces servazzii	2
Schistosoma haematobium	1
Schistosoma mansoni	180
Schistosomatium douthitti	1
Shewanella oneidensis MR-1	1
Siphoviridae Bacteriophage SPBc2	1
Sorghum bicolor	1
Strongylocentrotus purpuratus	2
Sus scrofa	3
Tetraodon nigroviridis	5
Tobacco ringspot virus	1
Tribolium castaneum	1
Xenopus laevis	1
Xenopus tropicalis	209
Zea mays	8
synthetic construct	10
unclassified sp.	6
unidentified sp.	20
Σ	**1413**

Abteilung	Anzahl
FUN	19
GSS	462
HTC	3
HTG	417
HUM	182
INV	35
MAM	14
MUS	43
ORG	1
PHG	1
PLN	29
PRO	18
ROD	2
STS	5
SYN	10
UNC	26
VRL	2
VRT	144
Σ	**1413**

Tabelle 3.11: Verteilung der mit dem Muster `hhrz_I_ext` in der EMBL-Datenbank (Version 78) gefundenen Treffer gruppiert nach Abteilungen. FUN: Pilze, GSS: Genome Survey Sequenzen, HTC: Hoch-Durchsatz cDNAs, HTG: Hoch-Durchsatz-Genomsequenzen, HUM: *Homo sapiens*, INV: Invertebraten, MAM: andere Säugetiere, MUS: *Mus musculus*, ORG: Organellen, PHG: Phagen, PLN: Pflanzen, PRO: Prokaryoten, ROD: Nagetiere, STS: *Sequence Tagged Sites*, SYN: synthetische Sequenzen, UNC: nicht klassifizierte Sequenzen, VRL: virale Sequenzen, VRT: Vertebraten.

Tabelle 3.12: BlastN-Suche mit dem Treffer `BX569783:[71500,71550]` über das Zebrafisch-Genom. Die Suche wurde auf der Ensembl-Webseite (Birney *et al.*, 2004a) durchgeführt. Aufgelistet sind die 20 Treffer auf Chromosom 20, die jeweils um exakt 11 Nukleotide überlappen. Start: erstes Nukleotid der Such- bzw. Treffersequenz; End: letztes Nukleotid der Such- bzw. Treffersequenz; Ori: Orientierung der Such- bzw. Treffersequenz; Score: erreichte Punktzahl; E-val: Erwartungswert; %ID: Identität in %; Länge: Alignmentlänge.

Start	End	Ori	Start	End	Ori	Score	E-val	%ID	Länge
1	51	+	48501390	48501440	+	32	3.0e-09	90.38	52
1	51	+	48501429	48501479	+	31	1.3e-08	90.20	51
1	51	+	48501468	48501518	+	35	3.5e-11	92.16	51
1	51	+	48501507	48501557	+	31	1.3e-08	90.20	51
1	51	+	48501546	48501596	+	35	3.5e-11	92.16	51
1	51	+	48501585	48501635	+	51	1.0e-21	100.00	51
1	51	+	48501624	48501674	+	51	1.0e-21	100.00	51
1	51	+	48501663	48501713	+	51	1.0e-21	100.00	51
1	51	+	48501702	48501752	+	51	1.0e-21	100.00	51
1	51	+	48501741	48501791	+	31	1.3e-08	90.20	51
1	51	+	48501780	48501830	+	31	1.3e-08	90.20	51
1	51	+	48501819	48501869	+	51	1.0e-21	100.00	51
1	51	+	48501858	48501908	+	47	2.5e-19	98.04	51
1	51	+	48501897	48501947	+	47	2.5e-19	98.04	51
1	51	+	48501936	48501986	+	51	1.0e-21	100.00	51
1	51	+	48501975	48502025	+	51	1.0e-21	100.00	51
1	51	+	48502014	48502064	+	47	2.5e-19	98.04	51
1	51	+	48502053	48502103	+	47	2.5e-19	98.04	51
1	51	+	48502092	48502142	+	51	1.0e-21	100.00	51
1	51	+	48502131	48502181	+	32	3.0e-09	90.38	52

4

Diskussion

In den letzten Jahren konnte gezeigt werden, dass RNA nicht wie ursprünglich angenommen nur in den Prozess der Protein-Biosynthese involviert ist, sondern darüber hinaus strukturelle, katalytische und regulatorische Funktionen in der Zelle erfüllt. Daher ist im Zeitalter der Genomsequenzierung die zuverlässige Suche und Annotation von nicht für Proteine kodierenden RNAs (ncRNAs) und anderen RNA-Familien, wie z. B. *Riboswitches*, die in nicht-kodierenden Bereichen von mRNAs vorkommen (Mandal & Breaker, 2004), von großer Bedeutung. Die diversen Funktionen von RNA werden neben der Sequenz v. a. durch ihre Sekundär- und Tertiärstruktur bestimmt. Bislang stehen allerdings keine Bioinformatik-Werkzeuge zur Suche beliebiger RNA-Familien zur Verfügung, die die Sequenz- und Struktur-basierte Beschreibung mit der gewünschten Flexibilität erlauben. Zudem können in keinem der existierenden Programme Nebenbedingungen, wie z. B. thermodynamische Stabilität oder Basenzusammensetzung definiert werden, was besonders für die Beschreibung von RNAs von Nachteil ist. Aufgrund des großen Umfangs verfügbarer Sequenzdaten ist es darüberhinaus wichtig, dass die Suche effizient und schnell erfolgen kann, was bislang ebenfalls nicht für die vorhandenen Suchwerkzeuge zutrifft. Aus diesen Gründen sollte in der hier vorliegenden Arbeit in Zusammenarbeit mit der Arbeitsgruppe von Dr. Stefan Kurtz von der Universität Hamburg ein neues System zur Beschreibung und Suche von ncRNAs implementiert werden. Um dieses Ziel zu erreichen, wurde zunächst eine mächtige Beschreibungsprache entwickelt, die die Formulierung von hybriden Mustern (***Hy****brid* ***Pa****tterns*, kurz HyPa) erlaubt. Ein hybrides Muster enthält neben der Definition von Sequenz- und Struktureigenschaften auch approximative Elemente, die eine vom Benutzer definierte Anzahl an Fehlern zulassen, und Bewertungsfunktionen z. B. für die thermodynamische Stabilität oder die Basenzusammensetzung. Ein entsprechendes Suchwerkzeug, das diese hybriden Muster verarbeiten und effizient und schnell suchen kann, wird von Strothmann (2005) entwickelt. Obwohl das Suchwerkzeug HyPaSearch bis zum Ende dieser Arbeit nur in Teilen und bei weitem nicht fehlerfrei implementiert war, konnten dennoch erste Laufzeitvergleiche mit PatScan durchgeführt werden. Darüber hinaus wurde eine Bibliothek von hybriden Mustern (HyPaLib, Gräf *et al.*, 2001), sowie

eine Datenbank-gestützte, benutzerfreundliche Schnittstelle zur Suche der Muster und zur Auswertung der Suchergebnisse erstellt. Diese Komponenten des Systems sind über den HyPaServer im Internet frei verfügbar. Der Einsatz dieses Systems konnte erfolgreich an ersten Anwendungsbeispielen demonstriert werden. So wurden einerseits Hinweise auf eine neue Art der Genregulation in *Dictyostelium discoideum* gefunden (Gräf *et al.*, 2004) und andererseits erstmals funktionelle Hammerhead-Ribozyme im Genom von Pflanzen identifiziert (Przybilski *et al.*, 2005).

4.1 Beschreibungssprachen und Suchwerkzeuge

Für die Suche nach ncRNAs in Sequenzdaten stehen verschiedene Werkzeuge zur Verfügung. Einerseits sind Spezialprogramme verfügbar, mit denen nur ganz bestimmte Klassen von RNAs gesucht werden können (Fichant & Burks, 1991; Kryukov *et al.*, 1999; Laslett *et al.*, 2002; Lowe & Eddy, 1997). Andererseits wurden verschiedene Ansätze benutzt, um beliebige ncRNAs in Sequenzdaten zu finden (Carter *et al.*, 2001; Griffiths-Jones *et al.*, 2003; Lambert *et al.*, 2004; Rivas & Eddy, 2001). Eine Möglichkeit besteht in der Verwendung einer allgemeingültigen, deklarativen Mustersprache für RNA, mit der beliebige RNA-Familien beschrieben und mit dem zugehörigen Suchprogramm identifiziert werden können (Billoud B, 1996; D'Souza *et al.*, 1997; Grillo *et al.*, 2003; Macke *et al.*, 2001; Mehldau & Myers, 1993). Das im Rahmen dieser Arbeit entwickelte Werkzeug HyPaSearch basiert ebenfalls auf der Struktur-basierten Beschreibung von ncRNAs und wird im Folgenden mit den existierenden Suchwerkzeugen verglichen, die einen entsprechenden Ansatz verfolgen.

RNAs bilden charakteristische Sekundär- und Tertiärstrukturen aus, woraus die diversen strukturellen, katalytischen und regulatorischen Funktionen resultieren. Diese Strukturen lassen sich in grundlegende Strukturelemente wie z. B. einzelsträngige und basengepaarte Bereiche zerlegen, die sich mit Hilfe einer Mustersprache beschreiben und mit einem entsprechenden Suchwerkzeug suchen lassen. Alle verfügbaren Suchwerkzeuge für RNA, die eine Mustersprache verwenden, erlauben die Beschreibung von Primär- und Sekundärstruktur. Allerdings unterscheiden sie sich z. T. erheblich in der Flexibilität der Sprachen, die in einigen Fällen zudem extrem kompliziert sind. Keines dieser Werkzeuge, mit Ausnahme von RNAmotif, erlaubt die Angabe von zusätzlichen Bewertungsfunktionen.

Bei der Entwicklung der von HyPaSearch verwendeten Sprache HyPaL (siehe Abschnitt 3.2) wurde Wert darauf gelegt, dass die Sprache einfach und benutzerfreundlich gestaltet ist. Der Benutzer soll in der Lage sein, intuitiv das beschreiben zu können, was er sucht. Dazu stehen diverse Sprachelemente zur Verfügung, mit denen Sequenz- und Struktur-Motive flexibel beschrieben werden können (vgl. Tabelle 3.3). So können beispielsweise durch die Verwendung von IUPAC-Symbolen in Kombination mit regulären Ausdrücken basengepaarte Bereiche variabler Länge definiert werden, was mit den Mustersprachen der anderen Werkzeuge gar nicht oder nur sehr umständlich möglich ist. Zudem können

Basenaustausche bzw. Fehlpaarungen, Insertionen und Deletionen durch sog. approximative Ausdrücke erlaubt werden, was ebenfalls nicht in allen der anderen Werkzeuge möglich ist. Ein weiterer großer Vorteil von HyPaL ist der modulare Aufbau der Sprache. Dadurch können Teile eines Muster bzw. komplette Muster durch Referenzierung über Variablen bzw. Musternamen wiederbenutzt werden. Die freie Wahl von Variablen- und Musternamen, die Möglichkeit Kommentare in die Beschreibung mit einzubinden und die Verwendung der `where`-Klausel erlauben eine übersichtliche Strukturierung des Musters und tragen im Vergleich zu anderen Programmen erheblich zur Lesbarkeit der Muster bei. Darüber hinaus erlaubt HyPaL die Definition von hybriden Mustern, d. h. neben der Beschreibung von Sequenz- und Struktureigenschaften können weitere, frei definierbare Nebenbedingungen angegeben werden, die entscheidend die Sensitivität und Spezifität von Mustern erhöhen können. So lassen sich beispielsweise verschiedene thermodynamische Einschränkungen oder Bedingungen zur Basenzusammensetzung für bestimmte Teilmuster oder das gesamte Muster definieren, die erfüllt sein müssen. Insgesamt vereinigt HyPaL sämtliche in anderen Beschreibungssprachen zur Verfügung stehenden Formulierungsmöglichkeiten in einer Sprache und bietet darüber hinaus einige neue Sprachelemente. Es handelt sich um eine umfangreiche Mustersprache, mit der flexibel übersichtliche und gut lesbare RNA-Muster beliebiger Komplexität formuliert werden können.

Zur Suche von Mustern in Sequenzdaten können verschiedene Verfahren eingesetzt werden, die sich u. a. in der Art der Vorverarbeitung unterscheiden. Einerseits kann vor der Suche die Vorverarbeitung des zu suchenden Musters erfolgen, wie es beispielsweise bei den Suchwerkzeugen PatScan und RNAmotif der Fall ist. Dadurch ist die eigentliche Suche „nur“ von der Länge der zu durchsuchenden Sequenz abhängig. Im Gegensatz dazu kann die zu durchsuchende Sequenz vorverarbeitet (indiziert) werden (vgl. Abschnitt 3.3.3), eine Strategie, die z. Z. nur von HyPaSearch verfolgt wird. Damit ist die eigentliche Suche nur noch von der Länge des Musters abhängig. Im Hinblick auf die Suche in großen Sequenzdatensätzen ergibt sich daraus ein entscheidender Vorteil, weil die zu durchsuchenden Sequenzen stets erheblich länger als das Muster sind. Zudem müssen die Sequenzdaten nur einmal indiziert werden und stehen dann für die Suche verschiedener Muster zur Verfügung. Ein Nachteil ergibt sich allerdings durch den benötigten Speicherbedarf, der etwa um den Faktor 30 größer ist als für die Ausgangssequenzen. Das hat zur Folge, dass entsprechende Voraussetzungen an die Hardware für die Suche in großen Datensätzen gestellt werden müssen. So muss auf der einen Seite genügend Speicherplatz auf der Festplatte vorhanden sein, um die indizierten Sequenzdaten ablegen zu können, und andererseits ausreichend Arbeitsspeicher (RAM) verfügbar sein, damit die indizierten Sequenzdaten für die Suche geladen werden können.

Nach erfolgter Vorverarbeitung kann die eigentliche Suche durchgeführt werden. Die herkömmlichen Suchwerkzeuge beginnen dazu am Anfang der zu durchsuchenden Sequenz, vergleichen Nukleotid für Nukleotid mit dem Muster und arbeiten sich so linear bis zum Ende der Sequenz durch. Das Suchwerkzeug PatSearch (Grillo *et al.*, 2003), eine Weiterentwicklung von PatScan, erlaubt dem Benutzer die schrittweise Suche von Mustern, wodurch z. B. erst der signifikanteste Teil eines Musters gesucht werden kann. In dem daraus resultierenden, verkleinerten Suchraum erfolgt dann eine weitere Suche des ei-

gentlichen Musters. Eine solche Suchreihenfolge muss allerdings vom Benutzer angegeben werden. HyPaSearch verwendet dagegen eine entsprechende Strategie standardmäßig, ohne dass der Benutzer eingreifen muss (vgl. Abschnitt 3.3.3). Nach der Zerlegung der hybriden Muster wird eine Signifikanzanalyse der Teilmuster durchgeführt und daraus eine optimale Suchreihenfolge komponiert. Die zugrundeliegende Affix-Array-Datenstruktur erlaubt zu jedem Zeitpunkt des Suchprozesses die Suche in Vorwärts- und Rückwärtsrichtung. Dadurch kann zunächst das signifikanteste Teilmuster gesucht werden, was den verbleibenden Suchraum z. T. erheblich einschränkt und somit die Anzahl an durchzuführenden Zeichenvergleichen verringert. Zudem wird die Effizienz des Algorithmus durch die sog. atomaren Matcher bestimmt. Dabei handelt es sich um Algorithmen, die nach speziellen, nicht weiter zerlegbaren Mustern effizient suchen. Die ersten Laufzeitvergleiche (siehe Abschnitt 3.3.4) bestätigen den Geschwindigkeitsgewinn gegenüber den herkömmlichen Werkzeugen. HyPaSearch ist im Vergleich zu PatScan bei der Suche bestimmter Musterklassen erheblich schneller (bis Faktor > 100), was auf die automatische Optimierung der Suchreihenfolge und die effizienten atomaren Matcher zurückzuführen ist (siehe Abschnitt 3.3.3).

4.2 Mustererstellung und Mustersuche

Voraussetzung für die Erstellung eines Musters ist die Kenntnis von Konsensus-Sequenz und -Struktur, die eine bestimmte Familie von RNAs charakterisieren. Diese charakteristischen Eigenschaften einer RNA-Familie können durch den Vergleich homologer oder paraloger Sequenzen gewonnen werden. Dazu müssen allerdings Sequenzen dieses Typs entweder in Sequenzdatenbanken verfügbar oder aus molekularbiologischen Experimenten bekannt sein. Da ncRNAs in der Regel weniger in ihrer Sequenz als in ihrer Struktur konserviert sind, hat es sich bewährt, Sequenz-Struktur-Alignments von den zu vergleichenden Sequenzen z. B. mit dem Werkzeug ConStruct (Lück *et al.*, 1999) anzufertigen. Dieses interaktive Programm kombiniert die Methode des multiplen Sequenzalignments mit der Sekundärstruktur-Vorhersage von RNA und visualisiert das Ergebnis in einem Basenpaar-Dotplot. Mit Hilfe des Alignmenteditors kann dann die Benutzer-gesteuerte Optimierung des Alignments hinsichtlich Konsensus-Sequenz und -Struktur vorgenommen werden. Dabei handelt es sich allerdings um einen sehr aufwendigen Schritt bei der Musterstellung, wenn beispielsweise hunderte von Sequenzen verglichen werden müssen. Andererseits ist eine zu geringe Zahl an Sequenzen oftmals statistisch nicht ausreichend signifikant, um ein korrektes Alignment zu erhalten. Aus einem solchen Sequenz-Struktur-Alignment können dann die charakteristischen Eigenschaften der betrachteten RNA-Familie extrahiert werden. Dazu wurden im Rahmen dieser Arbeit kleine Werkzeuge mit Perl und BioPerl programmiert, die die gewünschten Sequenz- und Strukturinformationen aus dem Alignment ableiten und übersichtlich darstellen. Die so gewonnenen charakteristischen Merkmale können dann mit Hilfe der Beschreibungssprache in Form eines Musters zusammengefasst werden. Mit diesem Muster wird dann initial über die Sequenzen des Alignments (Trainingssequenzen) gesucht, um sicherzustellen, dass die Beschreibung kor-

rekt ist und alle Trainingssequenzen gefunden werden. Gegebenenfalls muss das Muster korrigiert und erneut über den Trainingsdatensatz gesucht werden.

Durch geschickte Auswahl der Sequenzdaten kann der Aufwand für die eigentliche Mustersuche in großen Datenmengen und die anschließende Auswertung des Suchergebnisses z. T. erheblich minimiert werden. Sind beispielweise nur Sequenzen eines Organismus von Interesse, können entsprechende Sequenzeinträge aus den großen Datenbanken, wie EMBL oder Genbank, herausgefiltert und ausschließlich diese durchsucht werden. Im Idealfall liegt das sequenzierte Genom des Organismus vor, sodass redundante Treffer vermieden werden können, die normalerweise aufgrund der in den großen Datenbanken enthaltenen Redundanz mehrfach gefunden werden. Bei der Suche der Hammerhead-Ribozyme (HHRz) vom Typ III wurden beispielsweise in *Arabidopsis thaliana* 11 Treffer in der gesamten EMBL-Datenbank gefunden, wobei die anschließende Analyse nur zwei verschiedene Treffer im Genom ergab (vgl. Abschnitt 3.5.2). Allerdings kann sich aus der Fragestellung, die zu der Mustersuche geführt hat, auch ergeben, dass gesamte Sequenzdatenbanken durchsucht werden müssen. So war es im Falle der HHRz von Interesse, sämtliche HHRz-Motive zu finden, die in allen verfügbaren Sequenzdaten vorhanden waren.

Beurteilung und Auswertung des Suchergebnisses

Nach der durchgeführten Mustersuche muss nun festgestellt werden, welche der gefundenen Treffer *de facto* richtige bzw. falsche Treffer sind, um die Qualität der Suche und damit die Zuverlässigkeit des verwendeten Musters abzuschätzen. Außerdem ist man an neuen Kandidaten, die sich ggf. unter den Treffern befinden, interessiert, die bislang noch nicht der entsprechenden RNA-Familie zugeordnet wurden.

In der Bioinformatik stehen verschiedene Methoden zur Abschätzung der Qualität einer Vorhersage zur Verfügung. Eine Möglichkeit ist die Bestimmung von Sensitivität und Spezifität. Dazu lassen sich vier Kategorien von gefundenen bzw. nicht-gefundenen Treffern unterscheiden (Baldi *et al.*, 2000; Lathrop *et al.*, 1993):

	richtig	falsch
Muster passt	TP (*true positives*)	FP (*false positives*)
Muster passt nicht	TN (*true negatives*)	FN (*false negatives*)

Bei den *true positives* handelt es sich um die Treffer, auf die das Muster passt und die auch tatsächlich zu der gesuchten RNA-Familie gehören. Im Fall der *true negatives* passt zwar das Muster, aber diese Sequenzen gehören nicht zu der RNA-Familie. Für die hier anzustellenden Betrachtungen ist die dritte wichtige Kategorie die der *false negatives*. Dabei handelt es sich um Sequenzen, die nicht von dem Muster erfasst worden sind, aber zu der gesuchten RNA-Familie gehören. Diese Kategorien können für die Definition der Sensitivität und Spezifität verwendet werden. Die Sensitivität ist definiert als der Quotient aus der Anzahl an gefundenen, richtigen Treffern (TP) und der Anzahl aller im durchsuchten Datensatz vorhandenen Treffer (TP und FN),

$$\text{Sensitivität} = \frac{TP}{TP + FN}, \tag{4.1}$$

und ist ein Maß für die Wahrscheinlichkeit, dass ein richtiger Treffer korrekt vorhergesagt wird. Die Spezifität ist definiert als der Quotient aus der Anzahl an gefundenen, richtigen Treffern (TP) und der Anzahl an insgesamt in der durchsuchten Sequenz gefundenen richtigen und falschen Treffer (TP und FP),

$$\text{Spezifität} = \frac{TP}{TP + FP}, \tag{4.2}$$

und ist ein Maß für die Wahrscheinlichkeit, dass ein als richtig vorhergesagter Treffer auch korrekt ist. Ein Muster besitzt die maximale Sensitivität, wenn bei der Suche mit dem Muster keine falsch-negativen Treffer auftreten, und die maximale Spezifität, wenn keine falsch-positiven Treffer im Suchergebnis enthalten sind.

Zunächst ist folglich zu klären, in welche Kategorie die gefundenen Treffer gehören. Dazu können die Treffer auf verschiedene Art und Weise analysiert werden, um sie einer der beiden Kategorien (TP und FP) zuzuordnen. So kann beispielsweise die in den zugehörigen Sequenzeinträgen enthaltene Annotation zur Beurteilung ausgewertet werden. Ist eine entsprechende Annotation am Fundort des Treffers vorhanden, kann dieser als TP kategorisiert werden. Ein solcher Treffer bestätigt die Richtigkeit des Musters, ist allerdings in den meisten Fällen nicht besonders interessant, da er schon bekannt sind. Allerdings ist die Annotation in den Sequenzeinträgen häufig weder lückenlos noch vollständig, sodass diese Art der Auswertung nicht immer möglich ist. Alternativ kann versucht werden die Treffersequenz durch BlastN-Suche näher zu charakterisieren. Bewährt haben sich in diesem Zusammenhang auch die im Internet verfügbaren Genom-Browser, wie z. B. Ensembl (Birney *et al.*, 2004b) oder der TAIR SeqViewer (Rhee *et al.*, 2003). Hier sind die vorhandenen, aktuellen Annotationen erhältlich, die Treffer können in dem entsprechenden Genom lokalisiert werden, wenn nicht über genomische Sequenzdaten gesucht wurde, und eine Fülle von weiteren Informationen und Verknüpfungen zu anderen Internet-Ressourcen sind verfügbar. Durch Sekundärstruktur-Vorhersage mit RNAfold oder Mfold kann zudem überprüft werden, ob die Treffersequenz die gewünschte Struktur ausbildet. Dabei ist allerdings auch immer zu beachten, dass beide Programme die optimale Struktur mit der minimalen freien Energie berechnen. Im Gegensatz dazu können aber auch suboptimale Strukturen von biologischer Relevanz sein. Beliebige weitere Bioinformatik-Werkzeuge, die z. B. von Programmpaketen wie EMBOSS zur Verfügung gestellt werden, können zur weiteren Charakterisierung der Treffersequenzen beitragen. Eine Vielzahl der hier angeführten Möglichkeiten wurden direkt in den HyPaServer (vgl. Abschnitt 3.4) integriert, um die Auswertung von Mustersuchen für den Benutzer erheblich zu erleichtern. Durch die so vorgenommenen Analysen der Sequenzen können die interessanten Treffer herausgefunden werden, die potentiell in die Kategorie TP gehören. Diese sollten dann nach Möglichkeit durch anschließende molekularbiologische Experimente näher charakterisiert und bestätigt werden, bevor man sie in entsprechenden Sequenzeinträgen annotiert. Zudem erlaubt die Analyse der Treffer z. B. die Formulierung zusätzlicher Bewertungsfunktionen, wie es in der hier entwickelten Beschreibungssprache HyPaL möglich ist, um die Muster spezifischer zu gestalten (vgl. Abschnitt 3.2.11). Das ist insbesondere dann notwendig, wenn eine zu hohe Anzahl an falsch-positiven Treffern (FP) gefunden wurde.

Problematischer stellt sich die Situation mit der Kategorie der *false negatives* (FN) dar. Diese Treffer sind nicht in den Suchergebnissen enthalten, obwohl sie in der durchsuchten Sequenz existieren. Das bedeutet, dass das Muster nicht sensitiv genug formuliert wurde, weil z. B. nicht ausreichend viele homologe Trainingssequenzen für die Mustererstellung vorhanden waren. Dies kann eintreten, obwohl sämtliche in der Literatur verfügbaren Informationen, sowie alle in Datenbanken erhältlichen Sequenzen bei der Mustererstellung verarbeitet wurden. Dann bleibt zur Vermeidung von falsch-negativen Treffern nur die Möglichkeit, zu jedem Zeitpunkt, an dem neue Informationen zu einer Klasse von RNAs erhältlich werden, diese zu nutzen und die entsprechende Musterbeschreibung dahingehend zu aktualisieren. Molekularbiologische Experimente, wie Mutationsanalysen oder SELEX-Experimente (Evolution im Reagenzglas) zur Untersuchung der Funktionalität oder Methoden zur Strukturaufklärung (NMR, Röntgenstruktur-Analyse), könnten diesen Prozess zielgerichtet unterstützen. Grundsätzlich wird aus diesem Sachverhalt auch klar, dass es sich bei der Suche mit Struktur-basierten Mustern um eine Methode handelt, bei der biologisches *know-how* unbedingt notwendig ist, um korrekte Musterbeschreibungen zu erstellen. Mit Hilfe von deklarativen Beschreibungssprachen soll der Benutzer deshalb formulieren können, „Was“ er sucht und nicht „Wie“ gesucht werden soll, eine grundlegende Voraussetzung, die im Rahmen dieser Arbeit zu der mächtigen, flexiblen und benutzerfreundlichen Beschreibungssprache HyPaL (vgl. Abschnitt 3.2) geführt hat. Die Sensitivität eines Muster kann allerdings nur bestimmt werden, wenn bekannt ist, wieviele Sequenzen tatsächlich in dem durchsuchten Datensatz enthalten sind. Dieses Qualitätsmaß eignet sich daher v. a. für den Vergleich von alternativen Mustern, die unter Verwendung verschiedener Sprachelemente die gleiche RNA-Familie beschreiben.

Bei falsch-negativen Treffern könnte es sich allerdings auch um Sequenzen handeln, die an Sequenzeintragsgrenzen liegen, und von denen nur der 5'- oder 3'-Teil der gesuchten Sequenz in dem Eintrag enthalten ist. Diese Treffer sind unabhängig von Sensitivität des Musters. Um solche Fälle auszuschließen, müssten die Sequenzen dieser Einträge vor der Mustersuche aneinandergehängt werden. Liegen zirkulären RNA-Sequenzen vor, wie es beispielsweise bei Viroiden der Fall ist, würde man diese Sequenzen vor der Suche verdoppeln (vgl. Abschnitt 3.5.2).

Mit der in Abschnitt 3.3.5 angegebenen statistischen Abschätzung der Anzahl an zu erwartenden Treffern steht eine alternative Möglichkeit zur Beurteilung des Suchergebnisses zur Verfügung. Weicht die Anzahl an gefundenen Treffern erheblich von dem berechneten Erwartungswert ab, ist das ein Hinweis darauf, dass diese Sequenzen nicht zufällig in dem durchsuchten Datensatz vorkommen. Mit dem Muster `hhrz_III_gc` (vgl. Abschnitt 3.5.2) wurde unter Berücksichtigung der Nukleotidfrequenzen im *Arabidopsis thaliana*-Genom (Größe: $1{,}2\times10^8$ Bp, vgl. Abschnitt 2.1.1) alle $4{,}8\times10^9$ nt ein Treffer erwartet. Bei der Suche auf beiden Strängen des Genoms von *Arabidopsis thaliana* wurden aber zwei Treffer auf einem Chromosom gefunden, was einer Rate von 1 Treffer pro $1{,}2\times10^8$ nt entspräche! Durch bioinformatische und molekularbiologische Untersuchungen konnte bestätigt werden, dass es sich bei den Treffern um die ersten funktionellen Ribozyme handelt, die in einem Pflanzengenom beschrieben wurden (siehe Abschnitt 3.5.2). Die vorgenannte Abschätzung basiert allerdings auf den Mononukleotidfrequenzen, da in diesem Modell

angenommen wird, dass die Nukleotide unabhängig voneinander in der Sequenz vorkommen. Die Basen-Komposition von Genomen variiert jedoch in den Di- und Trinukleotidfrequenzen sowohl zwischen verschiedenen Genomen, als auch lokal innerhalb eines Genoms (Karlin *et al.*, 1998; Schattner, 2002). Deshalb handelt sich bei der hier berechneten Abschätzung lediglich um eine Näherung, da sich die Berücksichtigung von lokalen Unterschieden in der Basen-Komposition als schwierig erwies.

Zusammenfassend kann also festgehalten werden, dass die Erstellung von Mustern stark von den zur Verfügung stehenden Informationen zu einer Klasse von RNAs, sowie den in Datenbanken enthaltenen Sequenzen und damit von der Qualität des initialen Sequenz-Struktur-Alignment abhängt. Grundsätzlich ist der Prozess der Mustererstellung und die Auswertung von Suchergebnissen mit einiger Handarbeit verbunden und kann nur in Teilen automatisiert werden, da in Abhängigkeit von der Fragestellung und der Datenlage die unterschiedlichsten Bioinformatik-Methoden angewendet werden müssen. Die stetig anwachsende Menge an Sequenzdaten und immer neue wissenschaftliche Erkenntnisse erlauben auf der einen Seite zwar eine stringentere Suche, haben andererseits zur Folge, dass einmal erstellte Muster einer regelmäßigen Revision bedürfen. Zudem sollte eine Mustersuche nach Möglichkeit durch molekularbiologische Analyse der Treffer unterstützt werden.

4.3 Datenbank hybrider Muster und Webservice

Im Rahmen dieser Arbeit wurde mit Hilfe der entwickelten Beschreibungssprache HyPaL und den in der Literatur verfügbaren Informationen zu diversen RNA-Familien sowie in Datenbanken erhältlichen Sequenzen eine Bibliothek von hybriden Mustern aufgebaut (siehe Abschnitt 3.4.3; Gräf *et al.*, 2001). Aktuell sind darin rund 60 verschiedene Muster enthalten, die sowohl einfache, nur auf Konsensus-Sequenzen basierende Motive, als auch weitaus komplexere, strukturelle Motive beschreiben. In einigen Fällen werden zudem die in HyPaL erlaubten zusätzlichen Bewertungsfunktionen verwendet. Neben den Musterbeschreibungen werden zusätzliche Informationen, wie eine kurze Beschreibung des Motivs oder Literaturverweise, abgespeichert (vgl. Abbildung 3.16). Diese zusätzlichen Informationen sollen dem Benutzer die Möglichkeit geben, das Muster zu beurteilen und es z. B. entsprechend der Verfügbarkeit aktueller Sequenzdaten ergänzen und aktualisieren zu können. Da das Suchwerkzeug HyPaSearch bis zum Ende der hier vorliegenden Arbeit nicht vollständig von den Kooperationspartnern implementiert war, steht eine Validierung dieser Muster noch aus.

Aus diesem Grund ist das vorrangige Ziel der Diplomarbeit von Zanger (2005) die Untersuchung des Suchwerkzeugs HyPaSearch auf Fehler, um so schnell wie möglich eine vollständig funktionsfähige und fehlerfreie Version zu erhalten. Dazu sollen Referenzmuster zur Kontrolle der Ergebnisse und zum vollständigen Vergleich hinsichtlich des Laufzeitverhaltens und der Suchergebnisse mit bereits bestehenden Werkzeugen definiert werden. Daran anschließen soll sich die Überarbeitung, Überprüfung und Erweiterung der bestehenden Musterdatenbank HyPaLib.

Die Musterdatenbank sowie die Benutzerschnittstellen zur Suche mit hybriden Mustern in Sequenzdaten und zur Datenbank-basierten Auswertung von Suchergebnissen werden via Internet zur Verfügung gestellt (vgl. Abschnitt 3.4.3). Dabei erfolgte die Entwicklung aus den bekannten Gründen auf Basis des Suchwerkzeugs PatScan. Durch den modular gehaltenen Aufbau des Webserversystems kann das darunterliegende Suchwerkzeug problemlos durch eine funktionierende HyPaSearch-Version ausgetauscht werden. Die Wahl, einen Webserver als Plattform zur Veröffentlichung des entwickelten HyPa-Systems zu verwenden, beruht u. a. auf den bereits beschriebenen Hardware-Voraussetzungen, die für die Suche mit HyPaSearch benötigt werden. Zudem kann die Software plattformübergreifend von jedem Computer aus benutzt werden, der über eine Anbindung an das Internet verfügt, ohne dass eine spezielle Software installiert werden muss.

Parallel zu der hier vorliegenden Arbeit wurde der HyPa-Webservice im Rahmen der Diplomarbeit von Teune (2004) in Teilen erweitert und weiterentwickelt. Diese Teile müssen nun noch in den bereits öffentlich zur Verfügung stehenden Webservice integriert werden. Dazu gehören u. a. das Session-Management zur Verwaltung der Benutzer und die Warteschlangen-Funktion für zu rechnende Mustersuchen, um einen reibungslosen Mehrbenutzerbetrieb zu gewährleisten, sowie einige der Visualisierungskomponenten. Zudem wäre beispielsweise eine Anbindung an das Genom-Annotationsprojekt Ensembl (Birney *et al.*, 2004b) wünschenswert, da so immer aktuelle Annotationen zur Verfügung stünden. Gleichzeitig könnte eine Erweiterung der Visualisierungskomponenten unter Verwendung der Ensembl-Programmierschnittstelle (Stabenau *et al.*, 2004) erfolgen. Mit Hilfe des Annotationssystems DAS (*Distributed Annotation System* Dowell *et al.*, 2001) könnte zudem transparent die lokal vorgenommene Annotation zur Verfügung gestellt werden.

4.4 Einsatz des entwickelten Systems in der Biologie

Die automatische Suche und Annotation von Genen, die für Proteine kodieren, ist aufgrund der gut bekannten Genstrukturen und von Homologien zu bereits bekannten Proteinen standardmäßig möglich (Übersichtsartikel: Zhang, 2002). Diese Werkzeuge lassen sich allerdings nicht für die Suche von ncRNAs einsetzen, da die Genstruktur von ncRNAs (noch) nicht eindeutig definiert ist. In den letzten Jahren sind immer wieder Spezialprogramme entwickelt worden, die nur eine spezielle Klasse von RNAs suchen können. Mit dem hier entwickelten System steht ein flexibles und vielseitig einsetzbares Werkzeug für die Suche nach ncRNAs zur Verfügung. So könnte z. B. die automatische Annotation von ncRNAs in sequenzierten Genomen unter Verwendung der in der Muster-Bibliothek gesammelten Deskriptoren erfolgen. Andererseits können, wie das Anwendungsbeispiel zur Suche von dsRNAs in *Dictyostelium discoideum* zeigt, Hypothesen getestet werden, um zu überprüfen, ob sich ggf. Experimente zur Untersuchung der Hypothese lohnen würden (vgl. Abschnitt 3.5.1).

Die exponentiell anwachsende Menge an Sequenzdaten, sowie neue Erkenntnisse über die charakteristischen Eigenschaften einer RNA-Familie machen es notwendig, bereits durchgeführte Mustersuchen zu wiederholen. So haben z. B. Ferbeyre *et al.* (2000) bereits

vor einigen Jahren die Verteilung von HHRz-Motiven in Genbank analysiert. Die erst jüngst von Khvorova *et al.* (2003) und De la Peña *et al.* (2003) beschriebenen, zusätzlichen Einschränkungen für die Sekundärstruktur ließen im Rahmen der vorliegenden Arbeit jedoch eine wesentlich stringentere Suche zu (vgl. Abschnitt 3.5.2).

Durch molekularbiologische Analysen unterstützt, konnten mit der Suche nach Typ III-HHRz die ersten natürlich im Genom einer Pflanze vorkommenden, funktionellen HHRz nachgewiesen werden (Przybilski *et al.*, 2005). Bislang sind HHRz in höheren Organismen nur in Satelliten-DNA von Amphibien, Schistosomen und Grillen beschrieben worden (Ferbeyre *et al.*, 1998; Rojas *et al.*, 2000; Zhang & Epstein, 1996). Hierbei wird vermutet, dass es sich um mobile genetische Elemente handelt, die im Laufe der Evolution aufgenommen und ins Genom integriert wurden. Im Fall der beiden neu entdeckten HHRz in *Arabidopsis thaliana* sprechen einige Gründe gegen eine solche Theorie. Die Tatsache, dass im gesamten Genom von *Arabidopsis thaliana* nur zwei HHRz-Sequenzen gefunden werden konnten, spricht nicht für eine Aufnahme von Satelliten-DNAs, die ins Genom integriert wurden, da ansonsten mehr Treffer hätten gefunden werden müssen. Es konnten auch keine Ähnlichkeiten zu HHRz enthaltenden Viroiden (*Avsunviroidae*; Flores *et al.*, 1998) festgestellt werden. In antisense-Orientierung zu den gefundenen Sequenzen waren keine weiteren Hammerhead- oder Hairpin-Ribozym-Motive zu finden. Zudem gibt es Hinweise darauf, dass *Avsunviroidae* in *Arabidopsis thaliana* nicht vollständig replizieren können (Daros & Flores, 2004; Matousek *et al.*, 2004). Damit ist auch eine viroidale Herkunft sehr unwahrscheinlich. In Datenbanken konnten auch keine weiteren Sequenzen gefunden werden, die Ähnlichkeiten mit den HHRz-Sequenzen oder mit dem 300 nt langen Konsensus, der beide Ribozyme umgibt, aufwiesen, was für eine unabhängige Entwicklung im Genom von *Arabidopsis thaliana* spricht. Die Sequenzkonservierung, die Gewebe-spezifische Expression und die *in vivo* nachgewiesene, katalytische Aktivität der neu entdeckten HHRz sprechen für eine bislang noch nicht bekannte biologische Funktion in *Arabidopsis thaliana*.

5

Zusammenfassung

Im Zeitalter der Genomsequenzierungen wächst die Menge an Sequenzdaten exponentiell an. Eine Herausforderung ist es, in diesen großen Datenmengen funktionelle, jedoch nicht für Proteine kodierende Ribonukleinsäuren, sog. *non-coding RNAs* (ncRNAs), zu identifizieren. Die Suche nach ncRNA-Genen ist im Gegensatz zu jener nach Protein-kodierenden Genen deutlich schwieriger, da ihre Genstruktur nicht signifikant durch z. B. Splicestellen oder Leserahmen definiert ist. Jedoch bilden ncRNAs charakteristische Sekundär- und Tertiärstrukturen aus, auf denen die diversen strukturellen und funktionellen Aufgaben von ncRNAs basieren. Mit Hilfe einer Beschreibungssprache können diese für eine bestimmte RNA-Familie spezifischen Eigenschaften in einem Muster zusammengefasst werden, das dann mit dem zugehörigen Software-Werkzeug in Sequenzdaten gesucht werden kann.

Im Rahmen der hier vorliegenden Arbeit wurde ein solches System zur Sequenz- und Struktur-basierten Beschreibung und effizienten Suche von RNAs in großen Sequenzdatensätzen entwickelt und angewendet. Dazu wurde zunächst eine umfangreiche Musterbeschreibungssprache HyPaL (für ***Hy****brid* ***Pa****ttern* ***L****anguage*) definiert, mit der flexibel sog. hybride Muster (***Hy****brid* ***Pa****tterns*, kurz HyPa) formuliert werden können. Hybride Muster kombinieren die Beschreibung von Konsensus-Sequenz- und Konsensus-Struktur-Motiven mit zusätzlichen Bewertungsfunktionen, wie z. B. für die thermodynamische Stabilität. Mit Hilfe von HyPaL wurde eine Bibliothek von hybriden Mustern (***Hy****brid* ***Pa****ttern* ***Lib****rary*, kurz HyPaLib) erstellt, die eine Vielzahl von einfachen und komplexen RNA-Motiven enthält (Gräf *et al.*, 2001). Für die Suche mit hybriden Mustern wurde und wird in Zusammenarbeit mit der Arbeitsgruppe von Dr. Stefan Kurtz (Universität Hamburg) ein Suchwerkzeug namens HyPaSearch entwickelt, dass diese Muster effizient und schnell in großen Sequenzdatensätzen suchen kann. Durch erste Laufzeitvergleiche mit dem bereits existierenden Suchwerkzeug PatScan (D'Souza *et al.*, 1997) konnte die Effizienz von HyPaSearch bestätigt werden. Zudem konnte im Rahmen dieser Arbeit der Einsatz des entwickelten Systems an einigen biologischen Anwendungsbeispielen erfolgreich demonstriert werden. So wurden beispielsweise genomisch kodierte RNAs in *Dictyostelium discoide-*

um gefunden, die intramolekular lange doppelsträngige Bereiche ausbilden können. Die Lokalisation einer überwiegenden Zahl dieser RNAs in antisense-Orientierung zu Exon-Bereichen lässt vermuten, dass diese RNAs die Expression der Proteine über einen RNA-Interferenz-Mechanismus post-transkriptionell regulieren (Gräf *et al.*, 2004). Außerdem wurden im Genom von *Arabidopsis thaliana* erstmals natürlich vorkommende, funktionelle Hammerhead-Ribozyme gefunden. Diese werden in der Pflanze organspezifisch exprimiert und zeigen *in vivo* und *in vitro* katalytische Aktivität (Przybilski *et al.*, 2005). Um das entwickelte HyPa-System der Allgemeinheit zur Verfügung zu stellen, wurde ein Datenbank-gestützter Webserver eingerichtet, der die Musterbibliothek HyPaLib umfasst sowie Benutzerschnittstellen, welche die Suche mit hybriden Mustern in Sequenzdaten und die Auswertung von Mustersuchen ermöglichen.

Literaturverzeichnis

Caenorhabditis elegans Sequencing Consortium, The (1998). Genome sequence of the nematode *Caenorhabditis elegans*: A platform for investigating biology. *Science*, **282**, 2012–2018, PMID: 9851916. 1

Abouelhoda, M.I., Kurtz, S. & Ohlebusch, E. (2002). The enhanced suffix array and its applications to genome analysis. In *Proceedings of the Second Workshop on Algorithms in Bioinformatics*. Lecture Notes in Computer Science 2452, Springer-Verlag, S. 449–463. Pdf. 3.3.1

Abouelhoda, M.I., Kurtz, S. & Ohlebusch, E. (2004). Replacing suffix trees with enhanced suffix arrays. *J. Discrete Algorithms*, 2, 53–86, Pdf. 3.3.1, 3.3.3

Abouelhoda, M.I., Ohlebusch, E. & Kurtz, S. (2002). Optimal exact string matching based on suffix arrays. In *Proceedings of the Ninth International Symposium on String Processing and Information Retrieval*. Lecture Notes in Computer Science 2476, Springer-Verlag, S. 31–43. Pdf. 3.3.3

Adams, M.D., Celniker, S.E., Holt, R.A., Evans, C.A., Gocayne, J.D., Amanatides, P.G., Scherer, S.E., Li, P.W., Hoskins, R.A., Galle, R.F., George, R.A., Lewis, S.E., Richards, S., Ashburner, M., Henderson, S.N., Sutton, G.G., Wortman, J.R., Yandell, M.D., Zhang, Q., Chen, L.X., Brandon, R.C., Rogers, Y.H., Blazej, R.G., Champe, M., Pfeiffer, B.D., Wan, K.H., Doyle, C., Baxter, E.G., Helt, G., Nelson, C.R., Gabor, G.L., Abril, J.F., Agbayani, A., An, H.J., Andrews-Pfannkoch, C., Baldwin, D., Ballew, R.M., Basu, A., Baxendale, J., Bayraktaroglu, L., Beasley, E.M., Beeson, K.Y., Benos, P.V., Berman, B.P., Bhandari, D., Bolshakov, S., Borkova, D., Botchan, M.R., Bouck, J., Brokstein, P., Brottier, P., Burtis, K.C., Busam, D.A., Butler, H., Cadieu, E., Center, A., Chandra, I., Cherry, J.M., Cawley, S., Dahlke, C., Davenport, L.B., Davies, P., de Pablos, B., Delcher, A., Deng, Z., Mays, A.D., Dew, I., Dietz, S.M., Dodson, K., Doup, L.E., Downes, M., Dugan-Rocha, S., Dunkov, B.C., Dunn, P., Durbin, K.J., Evangelista, C.C., Ferraz, C., Ferriera, S., Fleischmann, W., Fosler, C., Gabrielian, A.E., Garg, N.S., Gelbart, W.M., Glasser, K., Glodek, A., Gong, F., Gorrell, J.H., Gu, Z., Guan, P., Harris, M., Harris, N.L., Harvey, D., Heiman, T.J., Hernandez, J.R., Houck, J., Hostin, D., Houston, K.A., Howland, T.J., Wei, M.H., Ibegwam, C., Jalali, M., Kalush, F., Karpen, G.H., Ke, Z., Kennison, J.A., Ketchum, K.A., Kimmel, B.E., Kodira, C.D., Kraft, C., Kravitz, S., Kulp, D., Lai, Z., Lasko, P., Lei, Y., Levitsky, A.A., Li, J., Li, Z., Liang, Y., Lin, X., Liu, X., Mattei, B., McIntosh, T.C., McLeod, M.P., McPherson, D., Merkulov, G.,

Milshina, N.V., Mobarry, C., Morris, J., Moshrefi, A., Mount, S.M., Moy, M., Murphy, B., Murphy, L., Muzny, D.M., Nelson, D.L., Nelson, D.R., Nelson, K.A., Nixon, K., Nusskern, D.R., Pacleb, J.M., Palazzolo, M., Pittman, G.S., Pan, S., Pollard, J., Puri, V., Reese, M.G., Reinert, K., Remington, K., Saunders, R.D., Scheeler, F., Shen, H., Shue, B.C., Siden-Kiamos, I., Simpson, M., Skupski, M.P., Smith, T., Spier, E., Spradling, A.C., Stapleton, M., Strong, R., Sun, E., Svirskas, R., Tector, C., Turner, R., Venter, E., Wang, A.H., Wang, X., Wang, Z.Y., Wassarman, D.A., Weinstock, G.M., Weissenbach, J., Williams, S.M., WoodageT, Worley, K.C., Wu, D., Yang, S., Yao, Q.A., Ye, J., Yeh, R.F., Zaveri, J.S., Zhan, M., Zhang, G., Zhao, Q., Zheng, L., Zheng, X.H., Zhong, F.N., Zhong, W., Zhou, X., Zhu, S., Zhu, X., Smith, H.O., Gibbs, R.A., Myers, E.W., Rubin, G.M. & Venter, J.C. (2000). The genome sequence of drosophila melanogaster. *Science*, **287**, 2185–2195, PMID: 10731132. 1

Akin, A. (2003). Thermodynamische Analyse genomischer Sequenzen. Diplomarbeit, Heinrich-Heine-Universität Düsseldorf. 3.5.2

Altschul, S.F., Gish, W., Miller, W., Myers, E.W. & Lipman, D.J. (1990). Basic local alignment search tool. *J. Mol. Biol.*, **215**, 403–410, PMID: 2231712, http://www.ncbi.nlm.nih.gov/BLAST/. 2.5.1

Ambros, V. (2004). The functions of animal microRNAs. *Nature*, **431**, 350–355, PMID: 15372042. 1.1

Argaman, L., Hershberg, R., Vogel, J., Bejerano, G., Wagner, E.G., Margalit, H. & S., Altuvia (2001). Novel small RNA-encoding genes in the intergenic regions of *Escherichia coli*. *Curr. Biol.*, **11**, 941–950, PMID: 11448770. 1.3

Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A.F. & Nielsen, H. (2000). Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics*, **16**, 412–424, PMID: 10871264. 4.2

Bartsch, H., Voigtsberger, S., Baumann, G., Morano, I. & Luther, H.P. (2004). Detection of a novel sense-antisense RNA-hybrid structure by RACE experiments on endogenous troponin I antisens RNA. *RNA*, **10**, 1215–1224, PMID: 15272119. 3.5.1

Bateman, A., Coin, L., Durbin, R., Finn, R.D., Hollich, V., Griffiths-Jones, S., Khanna, A., Marshall, M., Moxon, S., Sonnhammer, E.L., Studholme, D.J, Yeats, C. & Eddy, S.R. (2004). The pfam protein families database. *Nucleic Acids Res.*, **32**, D 138–141, PMID: 14681378. 3.5.2

Baulcombe, D. (2004). RNA silencing in plants. *Nature*, **431**, 356–363, PMID: 15372043. 1.1

Beckstette, M., Strothmann, D., Homann, R., Giegerich, R. & Kurtz, S. (2004). PoSSuMsearch: Fast and Sensitive Matching of Position Specific Scoring Matrices using Enhanced Suffix Arrays. In *Proceedings of the German Conference on Bioinformaticsb (GCB) 2004*. S. 53–64. Pdf. 3.3.3, 3.3.4

Benson, D.A., Karsch-Mizrachi, I., Lipman, D.J., Ostell, J. & Wheeler, D.L. (2004). GenBank: update. *Nucleic Acids Res.*, **32**, 23–26, http://www.ncbi.nlm.nih.gov, PMID: 14681350. 2.1.1, 3.5.2

Bernstein, E., Caudy, A.A., Hammond, S.M. & Hannon, G.J. (2001). Role for a bidentate ribonuclease in the initiation step of rna interference. *Nature*, **409**, 363–366, PMID: 11201747. 3.5.1

Bernstein, E., Denli, A.M. & Hannon, G.J. (2001). The rest is silence. *RNA*, **7**, 509–521, PMID: 11720281. 3.5.1

Billoud B, Kontic M, Viari A. (1996). Palingol: a declarative programming language to describe nucleic acids' secondary structures and to scan sequence database. *Nucleic Acids Res.*, **24**, 1395–1403, PMID: 8628670. 1.3, 4.1

Billy, E., Brondani, V., Zhang, H., Muller, U. & W., Filipowicz (2001). Specific interference with gene expression induced by long, double-stranded RNA in mouse embryonal teratocarcinoma cell lines. *Proc. Nat. Acad. Sci. U.S.A.*, **98**, 14428–14433, PMID: 11724966. 3.5.1

Birikh, K.R., Heaton, P.A. & Eckstein, F. (1997). The structure, function and application of the hammerhead ribozyme. *Eur. J. Biochem.*, **245**, 1–16, PMID: 9128718. 3.5.2

Birney, E., Andrews, D., Bevan, P., Caccamo, M., Cameron, G., Chen, Y., Clarke, L., Coates, G., Cox, T., Cuff, J., Curwen, V., Cutts, T., Down, T., Durbin, R., Eyras, E., Fernandez-Suarez, XM., Gane, P., Gibbins, B., Gilbert, J., Hammond, M., Hotz, H., Iyer, V., Kahari, A., Jekosch, K., Kasprzyk, A., Keefe, D., Keenan, S., Lehvaslaiho, H., McVicker, G., Melsopp, C., Meidl, P., Mongin, E., Pettett, R., Potter, S., Proctor, G., Rae, M., Searle, S., Slater, G., Smedley, D., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Storey, R., Ureta-Vidal, A., Woodwark, C., Clamp, M. & Hubbard, T. (2004). Ensembl 2004. *Nucleic Acids Res.*, **32**, D468–470, PMID: 14681459. 3.5.2, 3.5.2, 3.12

Birney, E., Andrews, T.D., Bevan, P., Caccamo, M., Chen, Y., Clarke, L., Coates, G., Cuff, J., Curwen, V., Cutts, T., Down, T., Eyras, E., Fernandez-Suarez, X.M., Gane, P., Gibbins, B., Gilbert, J., Hammond, M., Hotz, H.R., Iyer, V., Jekosch, K., Kahari, A., Kasprzyk, A., Keefe, D., Keenan, S., Lehvaslaiho, H., McVicker, G., Melsopp, C., Meidl, P., Mongin, E., Pettett, R., Potter, S., Proctor, G., Rae, M., Searle, S., Slater, G., Smedley, D., Smith, J., Spooner, W., Stabenau, A., Stalker, J., Storey, R., Ureta-Vidal, A., Woodwark, K.C., Cameron, G., Durbin, R., Cox, A., Hubbard, T. & Clamp, M. (2004). An overview of Ensembl. *Genome Res.*, **14**, 925–928, PMID: 15078858. 1.3, 4.2, 4.3

Birney, E., Clamp, M. & Durbin, R. (2004). GeneWise and Genomewise. *Genome Res.*, **14**, 988–995, PMID: 15123596. 1

Birney, E. & Durbin, R. (2000). Using GeneWise in the *Drosophila* annotation experiment. *Genome Res.*, **10**, 547–548, PMID: 10779496. 1

Bourdeau, V., Ferbeyre, G., Pageau, M., Paquin, B. & Cedergren, R. (1999). The distribution of RNA motifs in natural sequences. *Nucleic Acids Res.*, **27**, 4457–4467, PMID: 10536156. 3.4.3

Boyer, R.S. & Moore, J.S. (1977). A fast string searching algorithm. *Comm. ACM*, **20**, 762–772. 3.3.1

Brown, J.W., Echeverria, M. & Qu, L.H. (2003). Plant snoRNAs: functional evolution and new modes of gene expression. *Trends Plant. Sci.*, **8**, 42–49, PMID: 12523999. 1.1, 1.3

Bucher, P. (1990). Weight matrix descriptions of four eukaryotic RNA polymerase II promoter elements derived from 502 unrelated promoter sequences. *J. Mol. Biol.*, **212**, 563–578, PMID: 2329577. 3.4.3

Burge, C. & Karlin, S. (1997). Prediction of complete gene structures in human genomic dna. *J. Mol. Biol.*, **268**, 78–94, PMID: 9149143. 1

Butler, J.E. & Kadonaga, J.T. (2002). The RNA polymerase II core promoter: a key component in the regulation of gene expression. *Genes Dev.*, **16**, 2583–2592, PMID: 12381658. 3.4.3

Carter, R.J., Dubchak, I. & Holbrook, S.R. (2001). A computational approach to identify genes for functional RNAs in genomic sequences. *Nucleic Acids Res.*, **29**, 3928–3938, PMID: 11574674. 1.3, 4.1

Chevalet, L., Robert, A., Gueneau, F., Bonnefoy, J.Y. & Nguyen, T. (2000). Recombinant protein production driven by the tryptophan promoter is tightly controlled in icone 200, a new genetically engineered e. coli mutant. *Biotechnol. Bioeng.*, **20**, 351–358, PMID: 10862673. 3.5.2

Codd, E.F. (1970). A relational model of data for large shared data banks. In *Communications of the ACM.*, volume **13**. Association for Computing Machinery, Inc., S. 377–387. http://www.acm.org/classics/nov95/toc.html. 3.4.2

Collins, C.A. & Guthrie, C. (2000). The question remains: is the spliceosome a ribozyme? *Nat. Struct. Biol.*, **7**, 850–854, PMID: 11017191. 1.1

Cornish-Bowden, A. (1985). Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984. *Nucleic Acids Res.*, **13**, 3021–3030, PMID: 2582368. 3.2.3, 3.2

Crick, F. (1958). On protein synthesis. *Symp. Soc. Exp. Biol.*, **12**, 138–163, PMID: 13580867. 1

Crick, F. (1970). Central dogma of molecular biology. *Nature*, **227**, 561–563, PMID: 4913914. 1

Daros, J.A. & Flores, R. (2004). Arabidopsis thaliana has the enzymatic machinery for replicating representative viroid species of the family Pospiviroidae. *Proc. Nat. Acad. Sci. U.S.A.*, **101**, 6792–6797, PMID: 15096616. 4.4

Davies, C., Sheldon, C.C. & Symons, R.H. (1991). Alternative hammerhead structures in the self-cleavage of avocado sunblotch viroid RNAs. *Nucleic Acids Res.*, **19**, 1893–1898, PMID: 2030968. 3.5.2

De la Peña, M., Gago, S. & Flores, R. (2003). Peripheral regions of natural hammerhead ribozymes greatly increase their self-cleavage activity. *EMBO J.*, **22**, 5561–5570, PMID: 14532128. 1.2, 3.5.2, 3.5.2, 3.5.2, 4.4

Denti, M.A., Martinez de Alba, A.E., Sagesser, R., Tsagris, M. & Tabler, M. (2000). A novel RNA-binding protein from Triturus carnifex identified by RNA-ligand screening with the newt hammerhead ribozyme. *Nucleic Acids Res.*, **28**, 1045–1052, PMID: 10666442. 3.5.2, 3.5.2

Doudna, J.A. & Cech, T.R. (2002). The chemical repertoire of natural ribozymes. *Nature*, **41**, 222–228, PMID: 12110898. 1.1, 3.5.2, 3.24

Dowell, R.D., Jokerst, R.M., Day, A., Eddy, S.R. & Stein, L. (2001). The distributed annotation system. *BMC Bioinformatics*, **2**, 7, PMID: 11667947. 4.3

D'Souza, M., Larsen, N. & Overbeek, R. (1997). Searching for patterns in genomic data. *Trends Genet.*, **13**, 497–498, PMID: 9433140. 1.3, 3.2, 3.3.2, 4.1, 5

Eddy, Sean (2002). SQUID – C function library for sequence analysis (v1.9g). http://www.genetics.wustl.edu/eddy/software/#squid. 3.3.2

Ferbeyre, G., Bourdeau, V. & Cedergren, R. (1997). Does HIV tat protein also regulate genes of other viruses present in HIV infection? *Trends Biochem. Sci.*, **22**, 115–116, PMID: 9149528. 3.4.3

Ferbeyre, G., Bourdeau, V., Pageau, M., Miramontes, P. & Cedergren, R. (2000). Distribution of hammerhead and hammerhead-like RNA motifs through the GenBank. *Genome Res.*, **10**, 1011–1019, PMID: 10899150. 4.4

Ferbeyre, G., Smith, J.M. & Cedergren, R. (1998). Schistosome satellite DNA encodes active hammerhead ribozymes. *Mol. Cell. Biol.*, **18**, 3880–3888, PMID: 9632772. 3.5.2, 3.5.2, 4.4

Fichant, G.A. & Burks, C. (1991). Identifying potential tRNA genes in genomic DNA sequences. *J. Mol. Biol.*, **220**, 659–671, PMID: 1870126. 1.3, 4.1

Fire, A., Xu, S., Montgomery, M.K., Kostas, S.A., Driver, S.E. & Mello, C.C. (1998). Potent and specific genetic interference by double-stranded RNA in *Caenorhabditis elegans*. *Nature*, **391**, 806–811, PMID: 9486653. 1.1

Flores, R., Navarro, J.A., de la Peña, M., Navarro, B., Ambrós, S. & Vera, A. (1999). Viroids with hammerhead ribozymes: some unique structural and functional aspects with respect to other members of the group. *Biol. Chem.*, **380**, 849–854, PMID: 10494833. 1.2, 3.5.2

Flores, R., Randles, J.W., Bar-Joseph, M. & Diener, T.O. (1998). A proposed scheme for viroid classification and nomenclature. *Arch. Virol.*, **143**, 623–629, PMID: 9572562. 3.5.2, 4.4

Forster, A.C., Davies, C., Sheldon, C.C., Jeffries, A.C. & Symons, R.H. (1988). Self-cleaving viroid and newt rnas may only be active as dimers. *Nature*, **334**, 265–267, PMID: 2456468. 3.5.2

Forster, A.C. & Symons, R.H. (1987). Self-cleavage of virusoid RNA is performed by the proposed 55-nucleotide active site. *Cell*, **50**, 9–16, PMID: 3594567. 3.5.2

Galtier, N. & Lobry, J.R. (1997). Relationships between genomic G+C content, RNA secondary structures, and optimal growth temperature in prokaryotes. *J. Mol. Evol.*, **44**, 632–636, PMID: 9169555. 1.3

Garrett, T.A., Pabón-Peña, L.M., Gokaldas, N. & Epstein, L.M. (1996). Novel requirements in peripheral structures of the extended satellite 2 hammerhead. *RNA*, **2**, 699–706, PMID: 8756412. 3.5.2, 3.5.2

GCG (2004). Wisconsin Package, Genetics Computer Group, Madison, Wisc., Accelrys Webseite. 1.2

Giegerich, R. & Kurtz, S. (1997). From ukkonen to mccreight and weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, **19**, 331–353, Pdf. 3.3.1

Glockner, G., Eichinger, L., Szafranski, K., Pachebat, J.A., Bankier, A.T., Dear, P.H., Lehmann, R., Baumgart, C., Parra, G., Abril, J.F., Guigo, R., Kumpf, K., Tunggal, B., Cox, E., Quail, M.A., Platzer, M., Rosenthal, A., Noegel, AA & Consortium., Dictyostelium Genome Sequencing (2002). Sequence and analysis of chromosome 2 of dictyostelium discoideum. *Nature*, **418**, 79–85, PMID: 12097910. 3.5.1

Gräf, S., Borisova, B.E., Nellen, W. & Steger, G. Hammann, C. (2004). A database search for double-strand containing RNAs in *Dictyostelium discoideum*. *Biol. Chem.*, **285**, 961–965, PMID: 15551871. 3.5.1, 3.5.1, 4, 5

Gräf, S., Strothmann, D., Kurtz, S. & Steger, G. (2001). HyPaLib: a database of RNAs and RNA structural elements defined by hybrid patterns. *Nucleic Acids Res.*, **29**, 196–198, PMID: 11125089. 3.4.3, 4, 4.3, 5

Green, B., Pabon-Pena, L.M., Graham, T.A., Peach, S.E., Coats, S.R. & Epstein, L.M. (1993). Conserved sequence and functional domains in satellite 2 from three families of salamanders. *Mol. Biol. Evol.*, **10**, 732–750, PMID: 8355598. 3.5.2

Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A. & Eddy, S.R. (2003). Rfam: an RNA family database. *Nucleic Acids Res.*, **31**, 439–441, PMID: 12520045. 4.1

Grillo, G., Licciulli, F., Liuni, S., Sbisa, E. & G., Pesole (2003). PatSearch: A program for the detection of patterns and structural motifs in nucleotide sequences. *Nucleic Acids Res.*, **31**, 3608–3612, PMID: 12824377. 3.3.2, 4.1

Guerrier-Takada, C., Gardiner, K., Marsh, T., Pace, N. & Altman, S. (1983). The RNA moiety of ribonuclease P is the catalytic subunit of the enzyme. *Cell*, **35**, 849–157, PMID: 6197186. 1, 3.5.2

Gusfield, D. (1997). Algorithms on strings, trees and sequences. Cambridge University Press, New York. 3.4, 3.3.1

Hannon, G.J. & Rossi, J.J. (2004). Unlocking the potential of the human genome with RNA interference. *Nature*, **431**, 371–378, PMID: 15372045. 1.1

Haseloff, J. & Gerlach, W.L. (1989). Sequences required for self-catalysed cleavage of the satellite RNA of tobacco ringspot virus. *Gene*, **82**, 43–52, PMID: 2684775. 3.5.2

He, L & Hannon, G.J. (2004). MicroRNAs: small RNAs with a big role in gene regulation. *Nat. Rev. Genet.*, **5**, 522–531, PMID: 15211354. 1

Hentze, M.W. & Kuhn, L.C. (1996). Molecular control of vertebrate iron metabolism: mRNA-based regulatory circuits operated by iron, nitric oxide, and oxidative stress. *Proc. Nat. Acad. Sci. U.S.A.*, **93**, 8175–8182, PMID: 8710843. 1.1

Heus, H.A. & Pardi, A. (1991). Structural features that give rise to the unusual stability of RNA hairpins containing GNRA loops. *Science*, **253**, 191–194, PMID: 1712983. 1.2

Hildebrandt, M. & Nellen, W. (1992). Differential antisense transcription from the Dictyostelium EB4 gene locus: implications on antisense-mediated regulation of mRNA stability. *Cell*, **69**, 197–204, PMID: 1555240. 3.5.1

Hofacker, I.L. (2003). Vienna RNA secondary structure server. *Nucleic Acids Res.*, **31**, 3429–3431, PMID: 12824340. 1.2, 2.5.2

Hofacker, I.L., Fontana, W., Stadler, P.F., Bonhoeffer, S., Tacker, M. & Schuster, P. (1994). Fast folding and comparsion of RNA structures. *Monatsh. Chem.*, **125**, 167–188, Postscript, Vienna RNA Package Webseite. 1.2, 2.5.2, 2.5.3

Human Genome Sequencing Consortium, International (2004). Finishing the euchromatic sequence of the human genome. *Nature*, **431**, 931–945, PMID: 15496913. 1

Hutchins, C.J., Rathjen, P.D., Forster, A.C. & Symons, R.H. (1986). Self-cleavage of plus and minus RNA transcripts of avocado sunblotch viroid. *Nucleic Acids Res.*, **14**, 3627–3640, PMID: 3714492. 3.5.2, 3.5.2

Jeanmougin, F., Thompson, J.D., Gouy, M., Higgins, D.G. & Gibson, T.J. (1998). Multiple sequence alignment with CLUSTAL X. *Trends Biol. Sci.*, **23**, 403–405, PMID: 9810230. 2.5.1, 2.5.3, 3.4.3

Joyce, G.F. (2002). The antiquity of RNA-based evolution. *Nature*, **418**, 214–221, PMID: 12110897. 1.1

Jucker, F.M., Heus, H.A., Yip, P.F., Moors, E.H. & Pardi, A. (1996). A network of heterogeneous hydrogen bonds in GNRA tetraloops. *J. Mol. Biol.*, **264**, 968–980, PMID: 9000624. 1.2

Karlin, S., Campbell, A.M. & Mr'azek, J. (1998). Comparative DNA analysis across diverse genomes. *Annu. Rev. Genet.*, **32**, 185–225, PMID: 9928479. 4.2

Khvorova, A., Lescoute, A., Westhof, E. & Jayasena, S.D. (2003). Sequence elements outside the hammerhead ribozyme catalytic core enable intracellular activity. *Nat. Struct. Biol.*, **10**, 708–712, PMID: 12881719. 1.2, 3.5.2, 3.5.2, 3.5.2, 4.4

Kim, S.H., Sussman, J.L., Suddath, F.L., Quigley, G.J., McPherson, A., Wang, A.H., Seeman, N.C. & Rich, A. (1974). The general structure of transfer RNA molecules. *Proc. Nat. Acad. Sci. U.S.A.*, **71**, 4970–4974, PMID: 4612535. 1.2

Kiss, T. (2001). Small nucleolar RNA-guided post-transcriptional modification of cellular RNAs. *EMBO J.*, **20**, 3617–3622, PMID: 11447102. 1.1

Knuth, D.E., Morris, J.H. & Pratt, V.B. (1977). Fast pattern matching in strings. *SIAM J. Comput.*, **6**, 323–350. 3.3.1

Kruger, K., Grabowski, P.J., Zaug, A.J., Sands, J., Gottschling, D.E. & Cech, T.R. (1982). Self-splicing RNA: autoexcision and autocyclization of the ribosomal RNA intervening sequence of Tetrahymena. *Cell*, **31**, 147–157, PMID: 6297745. 1, 1.1, 3.5.2

Kryukov, G.V., Kryukov, V.M. & Gladyshev, V.N. (1999). New mammalian selenocysteine-containing proteins identified with an algorithm that searches for selenocysteine insertion sequence elements. *J. Biol. Chem.*, **274**, 33888–33897, PMID: 10567350. 1.3, 3.4.3, 3.4.3, 4.1

Kulikova, T., Aldebert, P., Althorpe, N., Baker, W., Bates, K., Browne, P., van den Broek, A., Cochrane, G., Duggan, K., Eberhardt, R., Faruque, N., Garcia-Pastor, M., Harte, N., Kanz, C., Leinonen, R., Lin, Q., Lombard, V., Lopez, R., Mancuso, R., McHale, M., Nardone, F., Silventoinen, V., Stoehr, P.,

Stoesser, G., Tuli, M.A., Tzouvara, K., Vaughan, R., Wu, D., Zhu, W. & Apweiler, R. (2004). The EMBL Nucleotide Sequence Database. *Nucleic Acids Res.*, **32**, 27–30, http://www.ebi.ac.uk/embl, PMID: 14681351. 1.3, 2.1

Lambert, A., Fontaine, J.F., Legendre, M., Leclerc, F., Permal, E., Major, F., Putzer, H., Delfour, O., Michot, B. & Gautheret, D. (2004). The ERPIN server: an interface to profile-based RNA motif identification. *Nucleic Acids Res.*, **32**, 160–165, PMID: 15215371. 4.1

Laslett, D., Canback, B. & Andersson, S. (2002). BRUCE: a program for the detection of transfer-messenger RNA genes in nucleotide sequences. *Nucleic Acids Res.*, **30**, 3449–3453, PMID: 12140330. 4.1

Lathrop, R.H., Webster, T.A., Smith, R.F., Winston, P.H. & Smith, T.F. (1993). Integrating AI with Sequence Analysis. In *Artificial Intelligence and Molecular Biology.* (Hunter, L., Hrsg.). AAAI Press, Distributed by The MIT Press, S. 210–258. http://www.aaai.org/Resources/Classics/Hunter/. 4.2

Lavorgna, G., Dahary, D., Lehner, B., Sorek, R., Sanderson, C.M. & Casari, G. (2004). In search of antisense. *Trends Biol. Sci.*, **29**, 88–94, PMID: 15102435. 1.1, 3.5.1, 3.5.1

Leclerc, F., Cedergren, R. & Ellington, A.D. (1994). A three-dimensional model of the Rev-binding element of HIV-1 derived from analyses of aptamers. *Nat. Struct. Biol.*, **1**, 293–300, PMID: 7664035. 3.4.3

Lehner, B., Williams, G., Campbell, R.D. & Sanderson, C.M. (2002). Antisense transcripts in the human genome. *Trends Genet.*, **18**, 63–65, PMID: 11818131. 3.5.1, 3.5.1

Leontis, N.B. & Westhof, E. (2001). Geometric nomenclature and classification of RNA base pairs. *RNA*, **7**, 499–512, PMID: 11345429. 1.2

Lippman, Z. & Martienssen, R. (2004). The role of RNA interference in heterochromatic silencing. *Nature*, **431**, 364–370, PMID: 15372044. 1.1

Lowe, T.M. & Eddy, S.R. (1997). tRNAscan-SE: a program for improved detection of transfer RNA genes in genomic sequence. *Nucleic Acids Res.*, **25**, 955–964, PMID: 9023104. 1.3, 4.1

Lück, R., Gräf, S. & Steger, G. (1999). ConStruct: A tool for thermodynamic controlled prediction of conserved secondary structure. *Nucleic Acids Res.*, **27**, 4208–4217, PMID: 10518612, ConStruct Webseite. 1.2, 1.3, 2.5.3, 3.3.2, 3.4.3, 3.5.2, 3.33, 4.2

Lück, R., Steger, G. & Riesner, D. (1996). Thermodynamic Prediction of Conserved Secondary Structure: Application to the RRE Element of HIV, the tRNA-like Element of CMV and the mRNA of Prion Protein. *J. Mol. Biol.*, **258**, 813–826, PMID: 8637012. 2.5.3, 3.4.3

Luzi, E., Eckstein, F. & Barsacchi, G. (1997). The newt ribozyme is part of a riboprotein complex. *Proc. Nat. Acad. Sci. U.S.A.*, **94**, 9711–9716, PMID: 9275189. 3.5.2, 3.5.2

Macke, T.J., Ecker, D.J., Gutell, R.R., Gautheret, D., Case, D.A. & Sampath, R. (2001). RNAMotif, an RNA secondary structure definition and search algorithm. *Nucleic Acids Res.*, **29**, 4724–4735, PMID: 11713323. 1.3, 4.1

Manber, U. & Myers, E.W. (1993). Suffix arrays: a new method for on-line string searches. *SIAM J. Comput.*, **22**, 935–948. 3.3.1

Mandal, M. & Breaker, R.R. (2004). Gene regulation by riboswitches. *Nat. RevṀolĊellḂiol.*, **5**, 451–463, PMID: 15173824. 4

Martens, H., Novotny, J., Oberstrass, J., T.L., Steck, Postlethwait, P. & W., Nellen (2002). RNAi in Dictyostelium: the role of RNA-directed RNA polymerases and double-stranded RNase. *Mol. Biol. Cell.*, **13**, 445–453, PMID: 11854403. 3.5.1

Matousek, J., Orctova, L., Steger, G., Skopek, J., Moors, M., Dedic, P. & Riesner, D. (2004). Analysis of thermal stress-mediated PSTVd variation and biolistic inoculation of progeny of viroid "thermomutants" to tomato and Brassica species. *Virology*, **323**, 9–23, PMID: 15165815. 4.4

Mattick, J.S. & Gagen, M.J. (2001). The evolution of controlled multitasked gene networks: the role of introns and other noncoding RNAs in the development of complex organisms. *Mol. Biol. Evol.*, **18**, 1611–1630, PMID: 11504843. 1

McCaskill, J.S. (1990). The equilibrium partition function and base pair binding probabilities for RNA secondary structure. *Biopolymers*, **29**, 1105–1119, PMID: 1695107. 1.2

McCreight, E.M. (1976). A space-economical suffix tree construction algorithm. *Journal of the ACM.*, **23**, 262–272. 3.3.1

Mehldau, G. & Myers, G. (1993). A system for pattern matching applications on biosequences. *Comp. Appl. Biosci.*, **9**, 299–314, PMID: 8324630. 1.3, 3.4.3, 4.1

Meister, G. & Tuschl, T. (2004). Mechanisms of gene silencing by double-stranded RNA. *Nature*, **431**, 343–349, PMID: 15372041. 1.1

Mello, C.C. & Conte, D. Jr. (2004). Revealing the world of RNA interference. *Nature*, **431**, 338–342, PMID: 15372040. 1.1

Miller, W.A., Hercus, T., Waterhouse, P.M. & Gerlach, W.L. (1991). A satellite RNA of barley yellow dwarf virus contains a novel hammerhead structure in the self-cleavage domain. *Virol.*, **183**, 711–720, PMID: 1713001. 3.5.2, 3.5.2

Molinaro, M. & Tinoco, I. Jr. (1995). Use of ultra stable UNCG tetraloop hairpins to fold RNA structures: thermodynamic and spectroscopic applications. *Nucleic Acids Res.*, **23**, 3056–3063, PMID: 7544890. 1.2

Moore, P.B. & Steitz, T.A. (2002). The involvement of RNA in ribosome function. *Nature*, **41**, 229–235, PMID: 12110899. 1.1

Moszer, I., Jones, L.M., Moreira, S., Fabry, C. & Danchin, A. (2002). Subtilist: the reference database for the bacillus subtilis genome. *Nucleic Acids Res.*, **30**, 62–65, PMID: 11752255, `http://genolist.pasteur.fr/SubtiList/`. 2

Nissen, P., Hansen, J., Ban, N., Moore, P.B. & Steitz, T.A. (2000). The structural basis of ribosome activity in peptide bond synthesis. *Science*, **289**, 920–930, PMID: 10937990. 1.1

Novotny, J., Diegel, S., Schirmacher, H., Mohrle, A., Hildebrandt, M., Oberstrass, J. & W., Nellen (2001). Dictyostelium double-stranded ribonuclease. *Methods Enzymol.*, **342**, 193–212, PMID: 11586892. 3.5.1

Nussinov, R., Pieczenik, G., Griggs, J.R. & Kleitman, D.J. (1978). Algorithms for loop matchings. *SIAM J. Appl. Math.*, **35**, 68–82. 1.2, 3.3.2

Ogle, J.M., Brodersen, D.E., Clemons, W.M. Jr, Tarry, M.J., Carter, A.P. & Ramakrishnan, V. (2001). Recognition of cognate transfer RNA by the 30S ribosomal subunit. *Science*, **292**, 897–902, PMID: 11340196. 1.1

Ohno, M., Segref, A., Bachi, A., Wilm, M. & Mattaj, I.W. (2000). PHAX, a mediator of U snRNA nuclear export whose activity is regulated by phosphorylation. *Cell*, **101**, 187–198, PMID: 10786834. 3.5.2

Olsen, M.V. (1995). A time to sequence. *Science*, **270**, 394–396. (document)

O'Rear, J.L., Wang, S., Feig, A.L., Beigelman, L., Uhlenbeck, O.C. & Herschlag, D. (2001). Comparison of the hammerhead cleavage reactions stimulated by monovalent and divalent cations. *RNA*, **7**, 537–545, PMID: 11345432. 1.2

Pabón-Peña, L.M., Zhang, Y. & Epstein, L.M. (1991). Newt satellite 2 transcripts self-cleave by using an extended hammerhead structure. *Mol. Cell. Biol.*, **11**, 6109–6115, PMID: 1944278. 3.5.2, 3.5.2

Pesole, G., Liuni, S. & M., D'Souza (2000). PatSearch: a pattern matcher software that finds functional elements in nucleotide and protein sequences and assesses their statistical significance. *Bioinformatics*, **16**, 439–450, PMID: 10871266. 1.3, 3.3.2

Poland, D. (1974). Recursion relation generation of probability profiles for specific-sequence macromolecules with long-range correlations. *Biopolymers*, **13**, 1859–1871, PMID: 4415504. 3.5.2

Przybilski, R., Gräf, S., Lescoute-Phillips, A., Nellen, W., Westhof, E., Steger, G. & Hammann, C. (2005). Functional Hammerhead Ribozymes Naturally Encoded in the Genome of *Arabidopsis thaliana*. *eingereicht*. 3.5.2, 4, 4.4, 5

Reese, M.G., Kulp, D., Tammana, H. & Haussler, D. (2000). Genie – gene finding in drosophila melanogaster. *Genome Res.*, **10**, 529–538, PMID: 10779493. 1

Rhee, S.Y., Beavis, W., Berardini, T.Z., Chen, G., Dixon, D., Doyle, A., Garcia-Hernandez, M., Huala, E., Lander, G., Montoya, M., Miller, N., Mueller, L.A., Mundodi, S., Reiser, L., Tacklind, J., Weems, D.C., Wu, Y., Xu, I., Yoo, D., Yoon, J. & Zhang, P. (2003). The arabidopsis information resource (tair): a model organism database providing a centralized, curated gateway to arabidopsis biology, research materials and community. *Nucleic Acids Res.*, **31**, 224–228, PMID: 12519987, `http://arabidopsis.org`. 1, 3.29, 4.2

Rice, P., Longden, I. & Bleasby, A. (2000). EMBOSS: The european molecular biology open software suite. *Trends Genet.*, **16**, 276–277, PMID: 10827456. 2.5.4, 3.5.1

Riks, J. (2001). Vorhersage konservierter strukturelemente in einzelsträngiger ribonukleinsäure. Diplomarbeit, Heinrich-Heine-Universität Düsseldorf. 3.3.2

Rivas, E. & Eddy, S.R. (2001). Noncoding RNA gene detection using comparative sequence analysis. *BMC Bioinformatics*, **2**, 8, PMID: 11801179. 1.3, 4.1

Rojas, A.A., Vazquez-Tello, A., Ferbeyre, G., Venanzetti, F., Bachmann, L., Paquin, B., Sbordoni, V. & Cedergren, R. (2000). Hammerhead-mediated processing of satellite pdo500 family transcripts from dolichopoda cave crickets. *Nucleic Acids Res.*, **28**, 4037–4043, PMID: 11024185. 3.5.2, 4.4

Rosenblad, M.A., Gorodkin, J., Knudsen, B., Zwieb, C. & Samuelsson, T. (2003). SRPDB: Signal Recognition Particle Database. *Nucleic Acids Res.*, **31**, 363–364, PMID: 12520023. 3.4.3

Schattner, P. (2002). Searching for RNA genes using base-composition statistics. *Nucleic Acids Res.*, **30**, 2076–2082, PMID: 11972348. 4.2

Scott, W.G., Murray, J.B., Arnold, J.R., Stoddard, B.L. & Klug, A. (1996). Capturing the structure of a catalytic RNA intermediate: the hammerhead ribozyme. *Science*, **274**, 2065–2069, PMID: 8953035, PDB-Id: 299d. 3.24, 3.5.2, 3.5.2, 3.5.2

Searls, D.B. (1993). The Computational Linguistics of Biological Sequences. In *Artificial Intelligence and Molecular Biology.* (Hunter, L., Hrsg.). AAAI Press, Distributed by The MIT Press, S. 47–120. http://www.aaai.org/Resources/Classics/Hunter/. 3.4.3

Stabenau, A., McVicker, G., Melsopp, C., Proctor, G., Clamp, M. & Birney, E. (2004). The Ensembl core software libraries. *Genome Res.*, **14**, 929–933, PMID: 15123588. 4.3

Stajich, J.E., Block, D., Boulez, K., Brenner, S.E., Chervitz, S.A., Dagdigian, C., Fuellen, G., Gilbert, J.G., Korf, I., Lapp, H., Lehvaslaiho, H., Matsalla, C., Mungall, C.J., Osborne, B.I., Pocock, M.R., Schattner, P., Senger, M., Stein, L.D., Stupka, E., Wilkinson, M.D. & Birney, E. (2002). The BioPerl toolkit: Perl modules for the life sciences. *Genome Res.*, **12**, 1611–1618, http://www.bioperl.org, PMID: 12368254. 2.3.1

Steger, G. (1994). Thermal denaturation of double-stranded nucleic acids: prediction of temperatures critical for gradient gel electrophoresis and polymerase chain reaction. *Nucleic Acids Res.*, **22**, 2760–2768, PMID: 8052531. 3.5.2

Steger, G. (2003). Bioinformatik: Methoden zur Vorhersage von RNA- und Proteinstruktur. Birkhäuser Verlag, Basel. 1.1, 1.2

Stein, L.D., Mungall, C., Shu, S., Caudy, M., Mangone, M., Day, A., Nickerson, E., Stajich, J.E., Harris, T.W., Arva, A. & Lewis, S. (2002). The generic genome browser: a building block for a model organism system database. *Genome Res.*, **12**, 1599–1610, PMID: 12368253. 2.3.1

Strothmann, D. (2005). A system for the declarative description and efficient search of hybrid patterns. PhD thesis, Universität Bielefeld. 3.3.3, 3.3.3, 3.3.3, 4

Sucgang, R., Chen, G., Liu, W., Lindsay, R., Lu, J., Muzny, D., Shaulsky, G., Loomis, W., Gibbs, R. & Kuspa, A. (2003). Sequence and structure of the extrachromosomal palindrome encoding the ribosomal RNA genes in Dictyostelium. *RNA*, **31**, 2361–2368, PMID: 22598211. 3.5.1

Szymanski, M., Barciszewska, M.Z., Erdmann, V.A. & Barciszewski, J. (2002). 5 S Ribosomal RNA Database. *Nucleic Acids Res.*, **30**, 176–178, PMID: 11752286. 3.4.3

Takamatsu, N., Watanabe, Y., Meshi, T. & Okada, Y. (1990). Mutational analysis of the pseudoknot region in the 3' noncoding region of tobacco mosaic virus RNA. *J. Virol.*, **64**, 3686–3693, PMID: 2370679. 3.4.3

Teune, J.-H. (2004). Analyse genomischer Sequenzdaten mit strukturbasierten RNA-Mustern. Diplomarbeit, Heinrich-Heine-Universität Düsseldorf. 3.4, 4.3

Tomari, Y., Du, T., Haley, B., Schwarz, D.S., Bennett, R., HA, Cook, BS, Koppetsch, Theurkauf, W.E. & Zamore, P.D. (2004). RISC assembly defects in the Drosophila RNAi mutant armitage. *Cell*, **116**, 831–841, PMID: 15035985. 3.5.1

Tuerk, C., Gauss, P., Thermes, C., Groebe, D.R., Gayle, M., Guild, N., Stormo, G., d'Aubenton Carafa, Y., Uhlenbeck, O.C., Tinoco, I. Jr & *et al.* (1988). CUUCGG hairpins: extraordinarily stable RNA secondary structures associated with various biochemical processes. *Proc. Nat. Acad. Sci. U.S.A.*, **85**, 1364–1368, PMID: 2449689. 1.2

Ukkonen, E. (1995). On-line construction of suffix-trees. *Algorithmica*, **14**, 249–260. 3.3.1

Wassarman, K.M., Repoila, F., Rosenow, C., Storz, G. & S., Gottesman (2001). Identification of novel small RNAs using comparative genomics and microarrays. *Genes Dev.*, **15**, 1637–1651, PMID: 11445539. 1.3

Watson, J.D. & Crick, F.H. (1953). Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, **171**, 737–738, PMID: 13054692. 1.2

Weiner, P. (1973). Linear pattern matching algorithms. In *Proceedings of the 14th IEEE Annual Symposium on Switching and Automata Theory*. The University of Iowa., Iowa City., S. 1–11. 3.3.1

Wheeler, D.L., Church, D.M., Edgar, R., Federhen, S., Helmberg, W., Madden, T.L., Pontius, J.U., Schuler, G.D., Schriml, L.M., Sequeira, E., Suzek, T.O., Tatusova, T.A. & Wagner, L. (2004). Database resources of the National Center for Biotechnology Information: update. *Nucleic Acids Res.*, **32**, 35–40, PMID: 14681353. 3.4.3

Whitfield, M.L., Kaygun, H., Erkmann, J.A., Townley-Tilson, W.H., Dominski, Z. & Marzluff, W.F. (2004). SLBP is associated with histone mRNA on polyribosomes as a component of the histone mRNP. *Nucleic Acids Res.*, **32**, 4833–4248, PMID: 15358832. 1.1

Williams, A.S. & Marzluff, W.F. (1995). The sequence of the stem and flanking sequences at the 3' end of histone mRNA are critical determinants for the binding of the stem-loop binding protein. *Nucleic Acids Res.*, **23**, 654–662, PMID: 7899087. 1.1

Woese, C.R., Winker, S. & Gutell, R.R. (1990). Architecture of ribosomal RNA: constraints on the sequence of "tetra-loops". *Proc. Nat. Acad. Sci. U.S.A.*, **87**, 8467–8471, PMID: 2236056. 1.2

Yeramian, E. (2000). The physics of dna and the annotation of the plasmodium falciparum genome. *Gene*, **255**, 151–168, PMID: 11024276. 3.5.2

Yusupov, M.M., Yusupova, G.Z., A, Baucom, K, Lieberman, Earnest, T.N., Cate, J.H. & Noller, H.F. (2001). Crystal structure of the ribosome at 5.5 a resolution. *Science*, **292**, 883–896, PMID: 11283358. 1.1

Yusupova, G.Z., Yusupov, M.M., Cate, J.H. & Noller, H.F. (2001). The path of messenger RNA through the ribosome. *Cell*, **106**, 233–241, PMID: 11511350. 1.1

Zamore, P.D., Tuschl, T., Sharp, P.A. & Bartel, D.P. (2000). RNAi: double-stranded RNA directs the ATP-dependent cleavage of mRNA at 21 to 23 nucleotide intervals. *Cell*, **101**, 25–33, PMID: 10778853. 3.5.1

Zanger, S. (2005). Vergleich von Suchprogrammen für RNA-Motive. Diplomarbeit, Heinrich-Heine-Universität Düsseldorf. 3.3.3, 4.3

Zdobnov, E.M. & Apweiler, R. (2001). Interproscan–an integration platform for the signature-recognition methods in interpro. *Bioinformatics*, **17**, 847–848, PMID: 11590104. 3.5.1

Zhang, M.Q. (2002). Computational prediction of eukaryotic protein-coding genes. *Nat. RevĠenet.*, **3**, 698–709, PMID: 12209144. 1.3, 4.4

Zhang, Y. & Epstein, LM. (1996). Cloning and characterization of extended hammerheads from a diverse set of caudate amphibians. *Gene*, **172**, 183–190, PMID: 8682301. 3.5.2, 3.5.2, 4.4

Zuker, M. (2003). Mfold web server for nucleic acid folding and hybridization prediction. *Nucleic Acids Res.*, **31**, 3406–3415, PMID: 12824337. 1.2, 2.5.2

Zuker, M. & Stiegler, P. (1981). Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic Acids Res.*, **9**, 133–148, PMID: 6163133. 1.2, 2.5.2

Zwieb, C. (1997). The uRNA database. *Nucleic Acids Res.*, **25**, 102–103, PMID: 9016512. 3.4.3