Rechnen mit Molekülen

(DNA-Computing)

Monika Sturm

sturm@tcs.inf.tu-dresden.de
TU Dresden

1. DNA-Computing - ein unkonventionelles Berechenbarkeitsmodell
2. Experimente im molekularbiologischen Labor
3. TT6 (Mehrtubesystem) als Modell für einen DNA-Computer
4. Konstruktion von DNA-Algorithmen mit DNA-HASKELL
5. Simulation von DNA-Algorithmen

Future-Computing

Lösen bestimmter Probleme auf der Basis alternativer Hardware

Ausprägungen

- **Quanten-Computing**
  Ausnutzung der Fähigkeit eines Quantensystems, sich in mehr als einem Quantenzustand befinden zu können (Superposition)
  $\longrightarrow$ Quantencomputer
- **Neural-Computing**
  Aufbau, Training und Einsatz Neuronaler Netze
  $\longrightarrow$ Künstliche Intelligenz, Fuzzy-Logik
- **Molekular-Computing** mit den Spezialformen DNA-Computing, RNA-Computing und Protein-Computing

DNA-Computing

Interdisziplinär geprägtes Forschungsgebiet

- Informatik, Molekularbiologie, Chemie, ...
- Arbeitsgruppe an der TU Dresden „Rechnen mit Molekülen“
  http://wwwtcs.inf.tu-dresden.de/molecules/
- European Molecular Computing Consortium
  (unter Leitung von Prof. Rozenberg, Universität Leiden)
- praxisrelevante Anwendung ausgewählter DNA-Algorithmen (DNA-Chips, DNA-Computer in der Genanalyse, Molecular Programming, DNA-Computer in der Informatik)

DNA-Computing

- **„Rechnen im Reagenzglas - Rechnen mit Molekülen“**
- Betrachtung eines alternativen Rechner-Modells (alternative Hardware)
- Betrachtung von DNA als Datenträger, Realisierung von Rechenoperationen (Abarbeitung von Algorithmen) durch geeignete molekularbiologische und biochemische Prozesse
- Anstoß zur Forschung auf diesem Gebiet durch einen Artikel von Adleman („*Molecular computation of solutions to combinatorial problems*“. Science 266, 1021-1024 (1994))

Vorteile des DNA-Computing

Parallelität und Rechengeschwindigkeit

- jede Operation wirkt parallel auf alle Moleküle eines Reagenzglases, es können bis zu 10^{20} Operationen gleichzeitig stattfinden
- die Rechengeschwindigkeit beträgt bis zu $1,2 \cdot 10^{18}$ ops/s, die Geschwindigkeit heutiger Supercomputer beträgt 10^{12} ops/s

Speicherdichte

- DNA speichert Informationen in einer Dichte von 1 bit/nm^3
- heutige Speichermedien sind auf etwa 1 bit/10^{12} nm^3 begrenzt

Energieverbrauch

- ein DNA-Computer ist 10^{10} mal so effizient wie ein traditioneller Rechner, der Energieverbrauch beträgt $2 \cdot 10^{19}$ ops/J

DNA-Computing

Im Zentrum des DNA-Computing steht damit ein auf DNA-Operationen basierender **biologischer Parallelrechner** mit enormer Speicherkapazität und Operationsgeschwindigkeit.

Daten: Repräsentation durch DNA-Stränge

- Desoxyribonukleinsäure (DNA), Träger der Erbinformation in zelluaren Organismen
- Sequenzen der Bausteine A, C, G und T
- Eingabedaten des Algorithmus werden in DNA-Strängen kodiert (durch Bausteinabfolge und/oder Bausteinanzahl im Strang)
- Herstellung der DNA-Stränge entweder künstlich oder durch Gewinnung aus Organismen

DNA-Computing

Operation: biochemische Reaktion

- wirkt auf **alle** DNA-Stränge im Reagenzglas gleichzeitig
- dadurch massiv datenparallele Abarbeitung

Ausgabe:

- Testreaktion zur Feststellung, ob sich DNA-Stränge im Reagenzglas befinden
- Bestimmung von Stranglängen (Bausteinanzahl) und Strangsequenzen (Bausteinabfolgen) ebenfalls möglich

DNA-Computing

DNA-Algorithmus: Abarbeitungsvorschrift für den Laboranten (oder den Simulationsrechner),

- welche biochemischen Reaktionen
- in welchen Reagenzgläsern
- in welcher Reihenfolge

auszuführen sind.

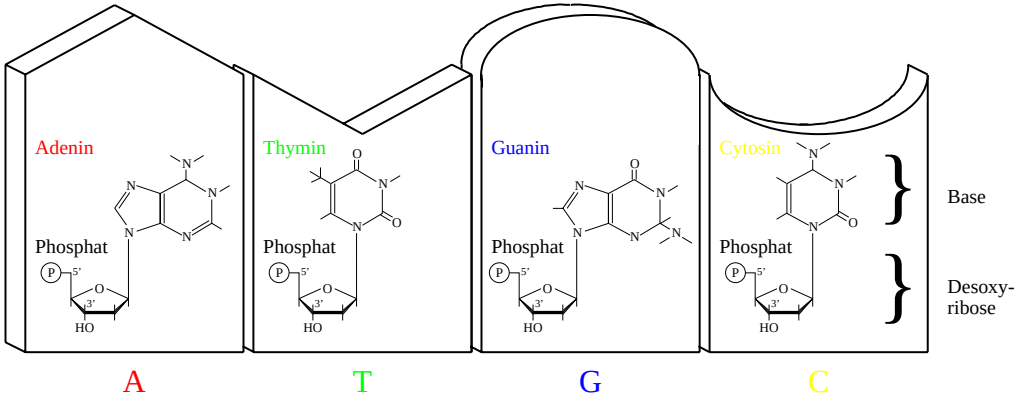
Hardware: molekularbiologisches Labor, in dem die zur Ausführung der Reaktionen benötigten Geräte und Chemikalien vorhanden sind

- Einordnung: SIMD-Modell (Single Instruction Multiple Data)

Aufbau von DNA-Strängen

Baustein (Nukleotid)

- Desoxyribose, an die eine der Basen Adenin, Cytosin, Guanin oder Thymin gebunden ist, entsprechend heißen die Nukleotide A, C, G oder T
- Nukleotide sind die Kettenglieder im DNA-Strang

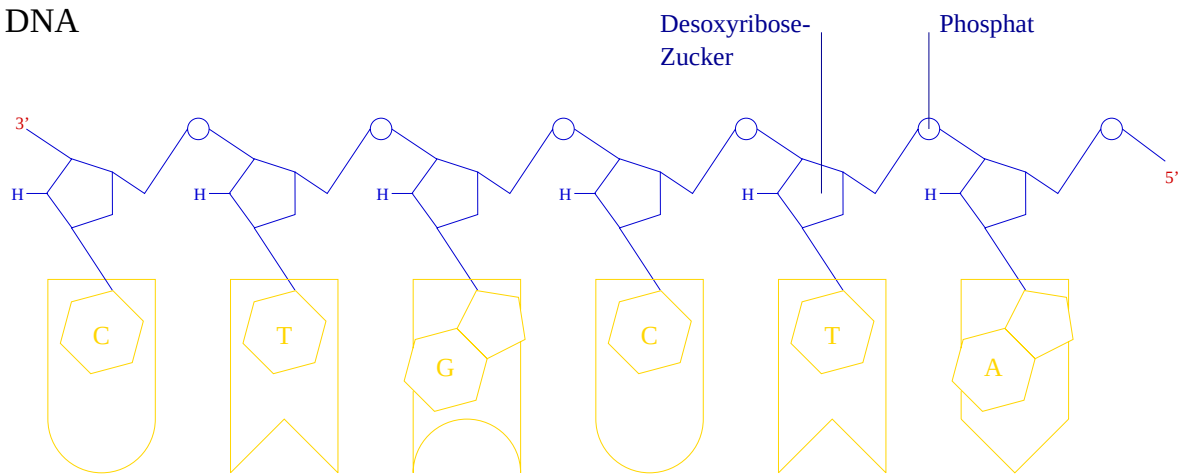


Abbildungsquelle: P.Berg, M.Singer. Die Sprache der Gene. Grundlagen der Molekulargenetik. Spektrum Akademischer Verlag, 1992.

Aufbau von DNA-Strängen

DNA-Einzelstrang

- Bausteinkette, bei der mehrere Desoxyribosen über Phosphatatome miteinander verbunden sind
- besitzt aufgrund der chemischen Struktur eine Richtung



Abbildungsquelle: P.Berg, M.Singer. Die Sprache der Gene. Grundlagen der Molekulargenetik. Spektrum Akademischer Verlag, 1992.

Aufbau von DNA-Strängen

DNA-Doppelstrang

- besteht aus mindestens zwei DNA-Einzelsträngen, die durch Wasserstoffbrückenbindungen zwischen gegenüberliegenden komplementären Bausteinen zusammengehalten werden
- A und T sind komplementär
- C und G sind komplementär
- besitzt Gestalt einer Doppelhelix

Aspekte zum DNA-Computing

Experimenteller Aspekt

- Lösung „ausgewählter“ mathematischer Probleme
- mit dem derzeitigen Stand der Technik nur in nicht vertretbaren Zeitaufwänden lösbare Probleme von hoher praktischer Relevanz
- **Klasse der NP-Probleme**
- **Grundgedanke:** Aus einem initial bereitgestellten umfassenden DNA-Pool, der den gesamten Suchraum DNA-kodiert, werden all jene DNA-Stränge entfernt, die keine Problemlösung darstellen. Ein finaler Test auf Vorhandensein von DNA liefert schließlich das JA-Nein-Ergebnis.

Aspekte zum DNA-Computing

Schwerpunkte

- Nachweis der Komplexität der molekularbiologischen Operationen und daraus zusammengesetzter DNA-Algorithmen
- Erhöhung der Stabilität der Laborimplementation von DNA-Algorithmen (Modellen)
- Reduktion von Seiteneffekten (unerwünschte Wirkungen) bei der Abarbeitung von molekularbiologischen Operationen
- Konstruktion erster Roboter zur Automatisierung von Laborarbeiten, Design erster DNA-Chips
- Verknüpfung von DNA mit RNA (Ribonukleinsäure) in Einheit mit Strangfixierungen an Trägersubstanzen
- Problemgröße aktueller Experimente beträgt ≈ 20

Aspekte zum DNA-Computing

Theoretischer Aspekt

- Modell des DNA-Computing als unkonventionelles Berechnungsmodell
- Erreichen der Berechnungsstärke des Modells der konventionellen Turingmaschine, λ-Kalkül, Klasse der Typ-0-Sprachen, ...
- Modell umfaßt molekularbiologische Operationen, die sich als DNA-Rekombination laborpraktisch realisieren und auch formal beschreiben lassen

Schwerpunkt

- Nachweis der Universalität für ein DNA-Computing-Modell, Simulation des Modells auf dem PC und laborpraktische Implementation (DNA-Computer)

DNA-Operationen

Klassifikation

- Operationen zum Gewinnen von DNA-Strängen,
- zum Knüpfen und Aufbrechen von Wasserstoffbrückenbindungen,
- zum Mischen von DNA-Lösungen
- zum Separieren und Analysieren nach verschiedenen Merkmalen sowie
- enzymatische Reaktionen (**Rekombinationstechnik** im engeren Sinne)

T. Hinze, M. Sturm. *Rechnen mit DNA - Eine Einführung in Theorie und Praxis.* Oldenbourg Wissenschaftsverlag München, Wien, 316 pp., 2004

DNA-Operationen

Synthesis (Generieren von DNA-Einzelsträngen)

Annealing, Melting (Hybridisieren, Denaturieren)

Merging (Mischen, Vereinigen)

Labeling (Markieren von Strangenden)

Ligation (Verketten von DNA-Doppelsträngen)

Cut (Zerschneiden von DNA-Doppelsträngen)

Blunting (Auffüllen/Abbauen von Einzelstrangüberhängen)

Polymerase Chain Reaction (Duplizieren von DNA-Doppelsträngen oder -abschnitten)

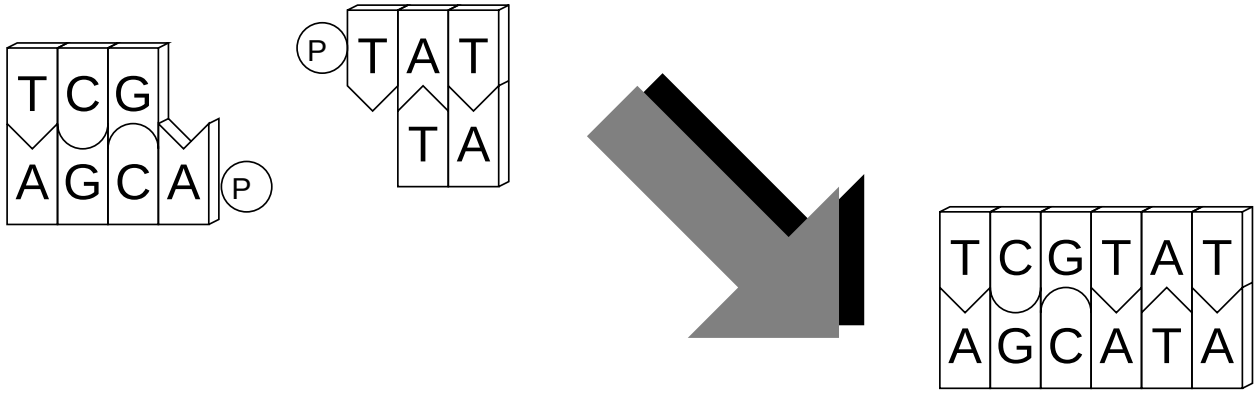
Affinity Purification (Separieren nach Subsequenz)

Electrophoresis (Stranglänge bestimmen, Separieren nach Länge)

Sequencing (Strangsequenz bestimmen) **u. a.**

Ligation

- Synonyme: Concatenate, Verketten



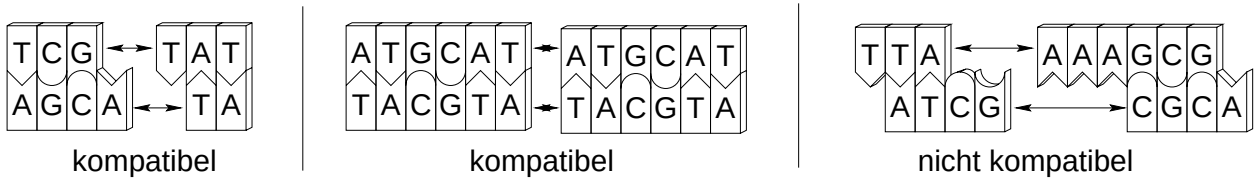
- Verketten von DNA-Doppelsträngen

Ligation

Definitionen

(Kompatibilität von DNA-Doppelstrangenden) Zwei sticky-Enden von DNA-Doppelsträngen sind *kompatibel*, wenn die Einzelstrangüberhänge beider Stränge über ihre gesamte Länge antiparallel komplementär sind. Blunt-Enden von DNA-Doppelsträngen sind generell zueinander *kompatibel*.

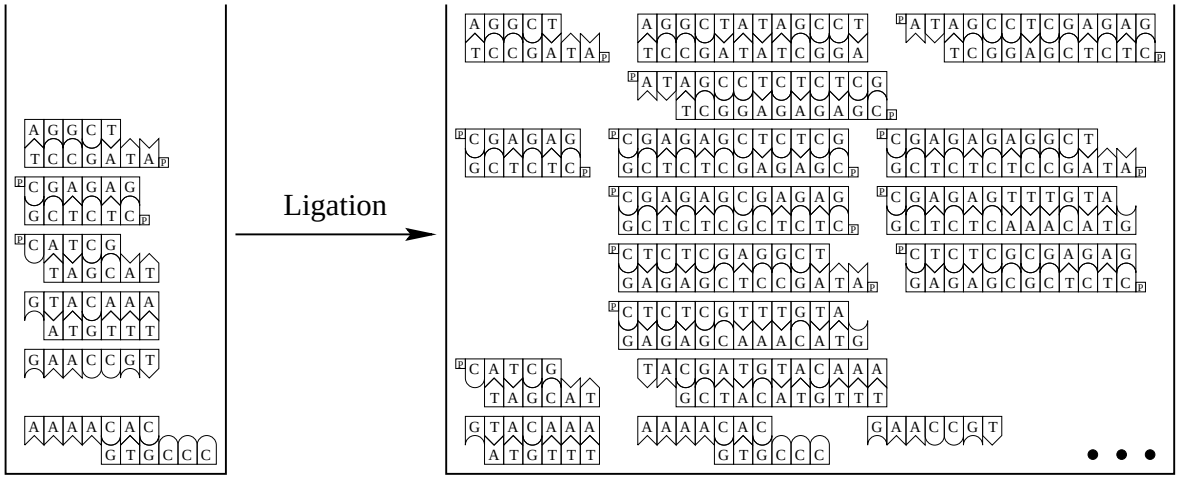
(5'-Phosphorylierung eines DNA-Strangendes) *5'-Phosphorylierung* bedeutet, daß das betreffende 5'-DNA-Strangende mit einer Phosphatgruppe markiert ist.



Ligation

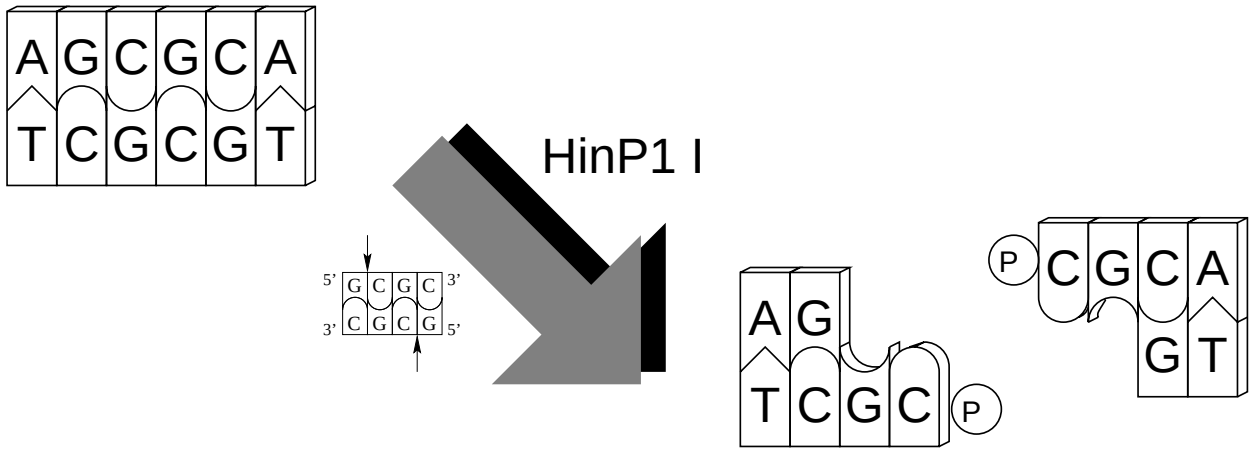
Definition

Unter *Ligation* versteht man eine biochemische Reaktion, bei der DNA-Doppelstränge mit kompatiblen Enden und mindestens einer 5'-Phosphorylierung an diesen kompatiblen Enden fortgesetzt miteinander verkettet werden. Es werden alle möglichen Endenpaare, auch von Zwischenprodukten, in die Reaktion einbezogen. Die Reaktion wird durch eine DNA-Ligase als Enzym katalysiert.



Cut

- Synonyme: Digestion, Cleavage, Verdau, Schnitt

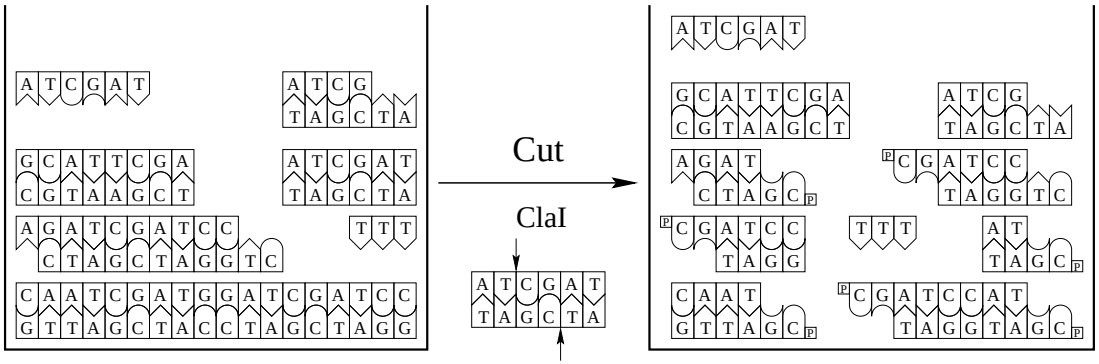


- Zerschneiden von DNA-Doppelsträngen

Cut

Definition

Unter *Cut* versteht man eine biochemische Reaktion, bei der DNA-Doppelstränge an jedem Vorkommen einer durch das Enzym bestimmten Subsequenz (Erkennungssequenz) an ebenfalls durch das Enzym bestimmten Spaltstellen geschnitten werden, wodurch bei jedem ausgeführten Schnitt ein neues 3'-5'-Endenpaar entsteht und auch Einzelstrangüberhänge auftreten können. Die beiden bei jedem ausgeführten Schnitt gebildeten 5'-Enden sind mit einer Phosphatgruppe markiert, die 3'-Enden mit einer Hydroxylgruppe. Die Reaktion wird durch eine auf DNA wirkende Typ-II-Restriktionsendonuclease katalysiert.



Konstruktion von Algorithmen

Klasse NP

Für eine große Klasse (wichtiger) Probleme, eben die sogenannten NP-Probleme, ist die Frage, ob sie alle einen polynomiellen Algorithmus besitzen (können), ungeklärt. Die Klasse NP steht für die Menge der Sprachen (Entscheidungsprobleme), die unter Annahme eines nichtdeterministischen Berechnungsmodells, der sogenannten **n**ichtdeterministischen Turingmaschine, in **p**olynomieller Zeit erkannt werden können.

Klassische NP-Probleme

- Erfüllbarkeitsproblem der Aussagenlogik
- Hamilton-Kreis-Problem
- Rucksackproblem
- ...

Hamilton-Kreis-Problem

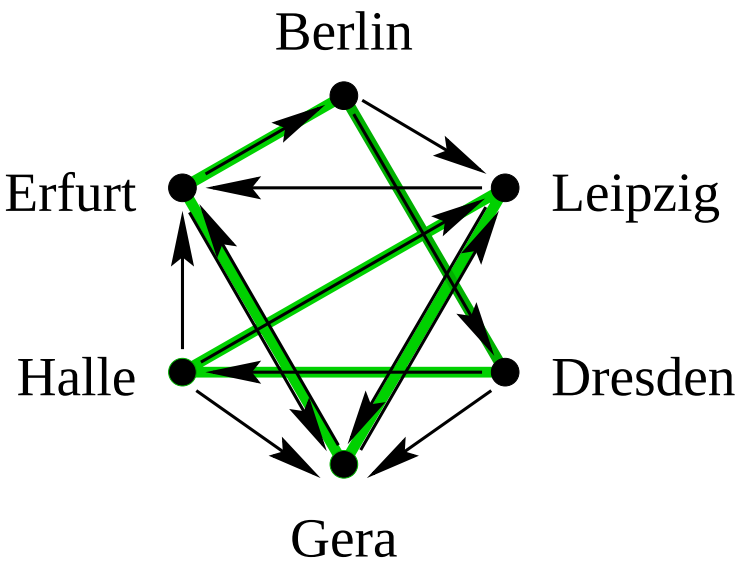
Problemdefinition

gegeben: zusammenhängender endlicher gerichteter Graph $G = (V, E)$ ohne Kantenbewertung mit n Knoten
(V: Knotenmenge, E: Kantenmenge)

gesucht: Gibt es eine Rundreise (Hamilton-Kreis) entlang der Kanten des Graphen, so daß

- **jeder** Knoten **genau einmal** besucht wird und
- die Rundreise wieder am Startknoten endet?

Hamilton-Kreis-Problem



G = (V, E)

V = { Berlin; Dresden; Erfurt;
Gera; Halle; Leipzig }

E = { (Berlin,Dresden); (Berlin,Leipzig);
(Dresden,Gera); (Dresden,Halle);
(Erfurt,Berlin); (Erfurt,Gera);
(Gera,Erfurt); (Gera,Leipzig);
(Halle,Erfurt); (Halle,Gera);
(Halle,Leipzig); (Leipzig,Erfurt);
(Leipzig,Gera) }

Hamiltonkreis:

Berlin -> Dresden -> Halle -> Leipzig -> Gera -> Erfurt -> Berlin

DNA-Algorithmus

Algorithmus nach Adleman

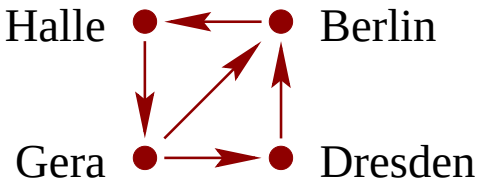
(veröffentlicht *Science*, 266:1021–1024, November 11,1994)

- Erzeuge willkürlich eine Vielzahl von Pfaden durch den Graphen
- Behalte davon nur jene Pfade, die an einem ausgewählten Knoten beginnen und enden
- Behalte davon nur jene Pfade, die genau n Knoten besuchen
- Behalte davon nur jene Pfade, die jeden Knoten mindestens einmal besuchen
- Falls mindestens ein Pfad übriggeblieben ist, lautet die Lösung „ja“, ansonsten „nein“

Beispiel

Vereinfachung gegenüber dem Original-Adleman-Algorithmus
(7 Knoten, 20 Bausteine pro Knoten und Kante)

Graph:



Knoten:

Berlin	T G T A
Dresden	A G C G
Gera	A A C C
Halle	A T G C

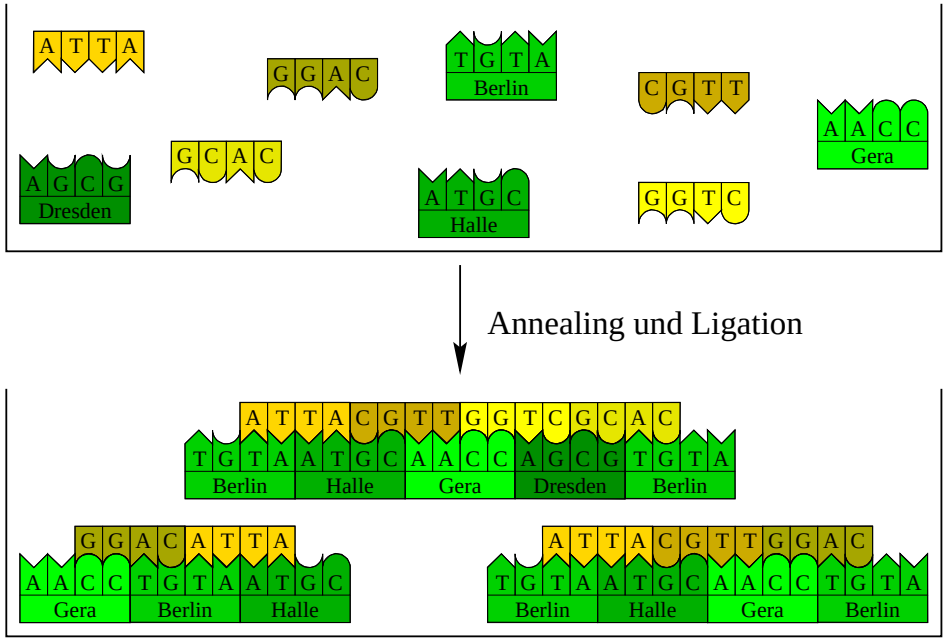
Kanten:

Berlin -> Halle	A T T A
Dresden -> Berlin	G C A C
Gera -> Berlin	G G A C
Gera -> Dresden	G G T C
Halle -> Gera	C G T T

Kodierung des Graphen in DNA

- jeder Knoten → DNA-Einzelstrang mit 4 Bausteinen frei wählbarer, aber halbseitig spezifischer Sequenz (eine Stranghälfte ermöglicht die exakte Knotenzuordnung)
- jede Kante → DNA-Einzelstrang mit 4 Bausteinen:
  - die beiden ersten Bausteine komplementär zur rechten Hälfte der Quellknoten-DNA
  - die beiden letzten Bausteine komplementär zur linken Hälfte der Zielknoten-DNA

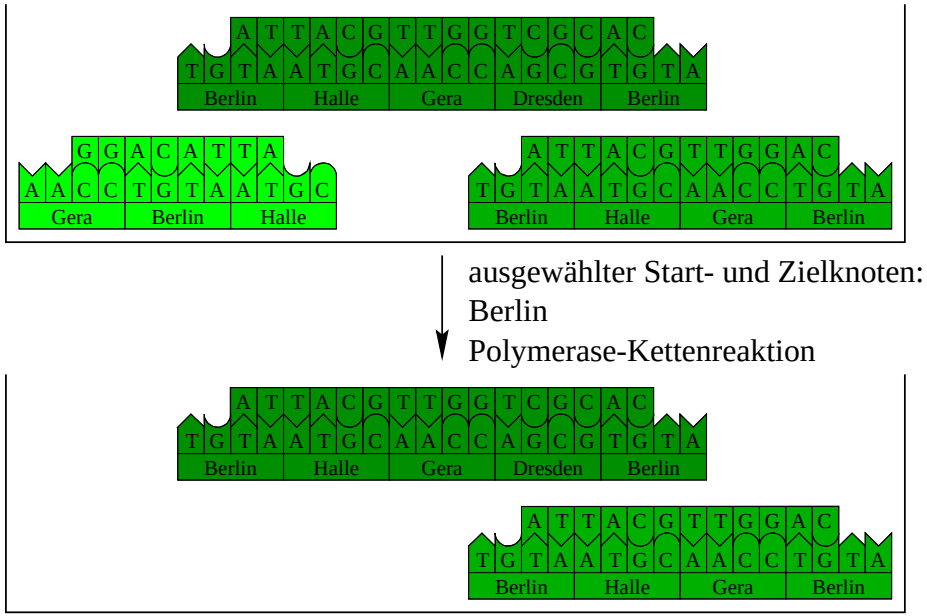
Schritt 1: Erzeuge willkürlich eine Vielzahl von Pfaden durch den Graphen



Annealing: Zusammenlagern komplementärer DNA-Einzelstränge zu Doppelsträngen

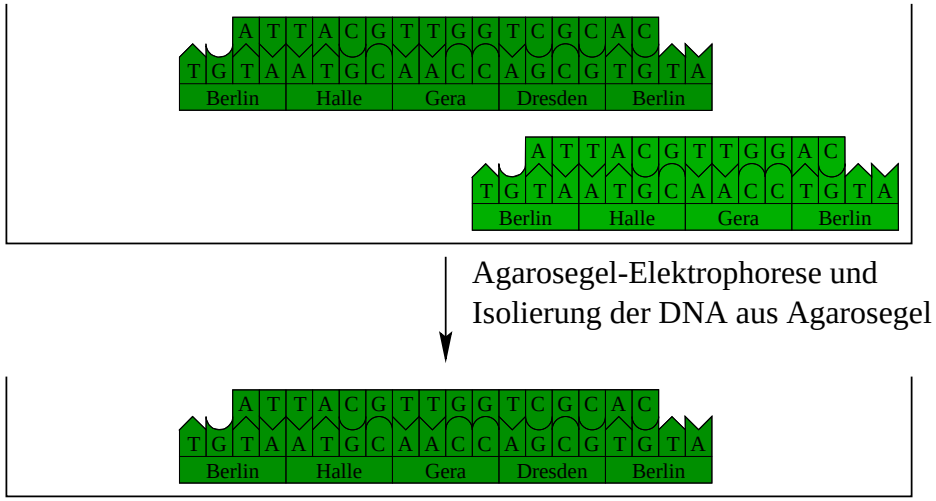
Ligation: Verketten von DNA-Doppelsträngen

Schritt 2: Behalte davon nur jene Pfade, die an einem ausgewählten Knoten beginnen und enden



Polymerase-Kettenreaktion: millionenfaches Duplizieren von DNA-Doppelsträngen mit vorgegebenem Anfangs- und Endstück

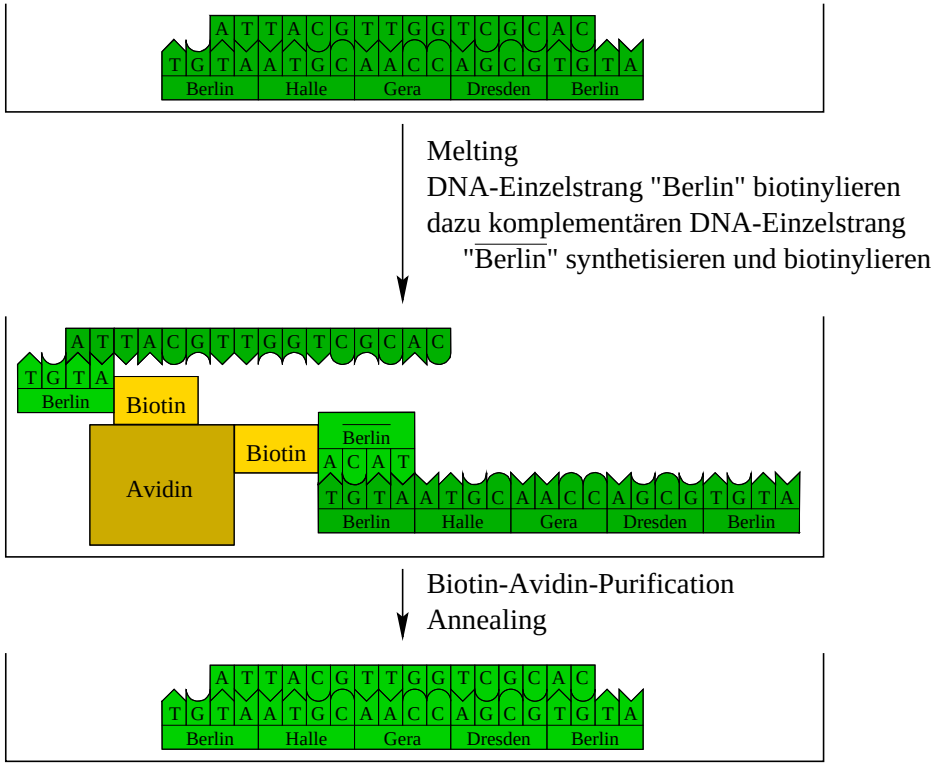
Schritt 3: Behalte davon nur jene Pfade, die genau n Knoten besuchen



Agarosegel-Elektrophorese: Methode zur Längenseparation und Visualisierung von DNA-Doppelsträngen. Die Stränge der Länge $5 \cdot 4 = 20$ Bausteine werden anschließend isoliert.

Schritt 4: Behalte davon nur jene Pfade, die jeden Knoten mindestens einmal besuchen

→ alle Pfade behalten, in denen Berlin vorkommt



Schritt 4:

Melting: Auflösen von DNA-Doppelsträngen zu DNA-Einzelsträngen

Biotinylieren: Anheften von Biotin-Molekülen an Strangenden

Biotin-Avidin-Purification:

- Anlagern komplementärer DNA-Einzelstränge an DNA-Einzelstränge, die mit Biotin und Avidin fixiert sind.
- Alle nicht angelagerten Stränge werden ausgewaschen und verworfen.
- Anschließend trennt man die angelagerten DNA-Einzelstränge wieder ab, wäscht sie ebenfalls aus und sammelt sie in einem neuen Reagenzglas.

Annealing: Zusammenlagern komplementärer DNA-Einzelstränge zu Doppelsträngen

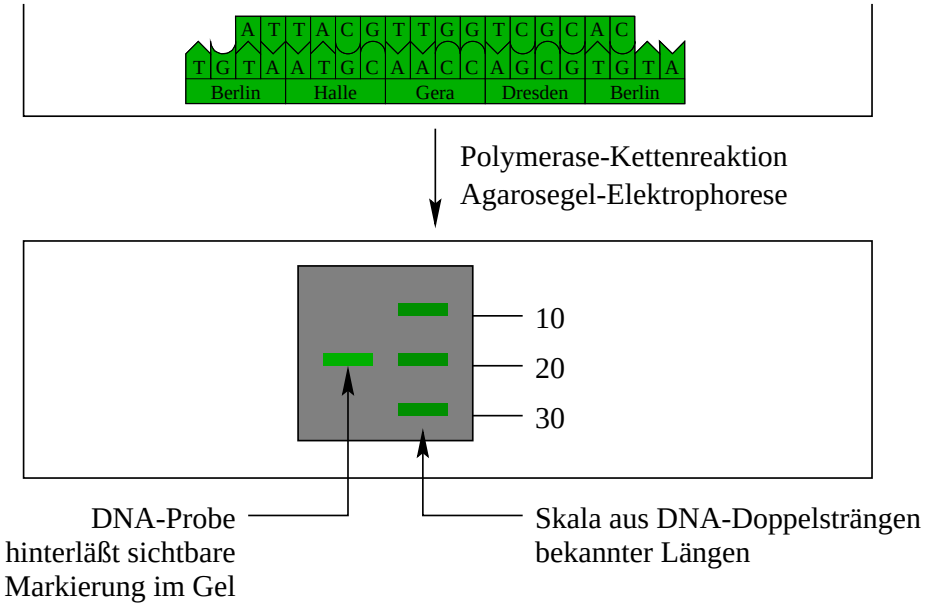
Schritt 4: Behalte davon nur jene Pfade, die jeden Knoten mindestens einmal besuchen

→ alle Pfade behalten, in denen Dresden vorkommt
analoger Ablauf, nur „Dresden“ statt „Berlin“

→ alle Pfade behalten, in denen Gera vorkommt
analoger Ablauf, nur „Gera“ statt „Berlin“

→ alle Pfade behalten, in denen Halle vorkommt
analoger Ablauf, nur „Halle“ statt „Berlin“

Schritt 5: Falls mindestens ein Pfad übriggeblieben ist, lautet die Lösung „ja“, ansonsten „nein“



Lösung des Beispiel-Hamilton-Kreis-Problems: „ja“

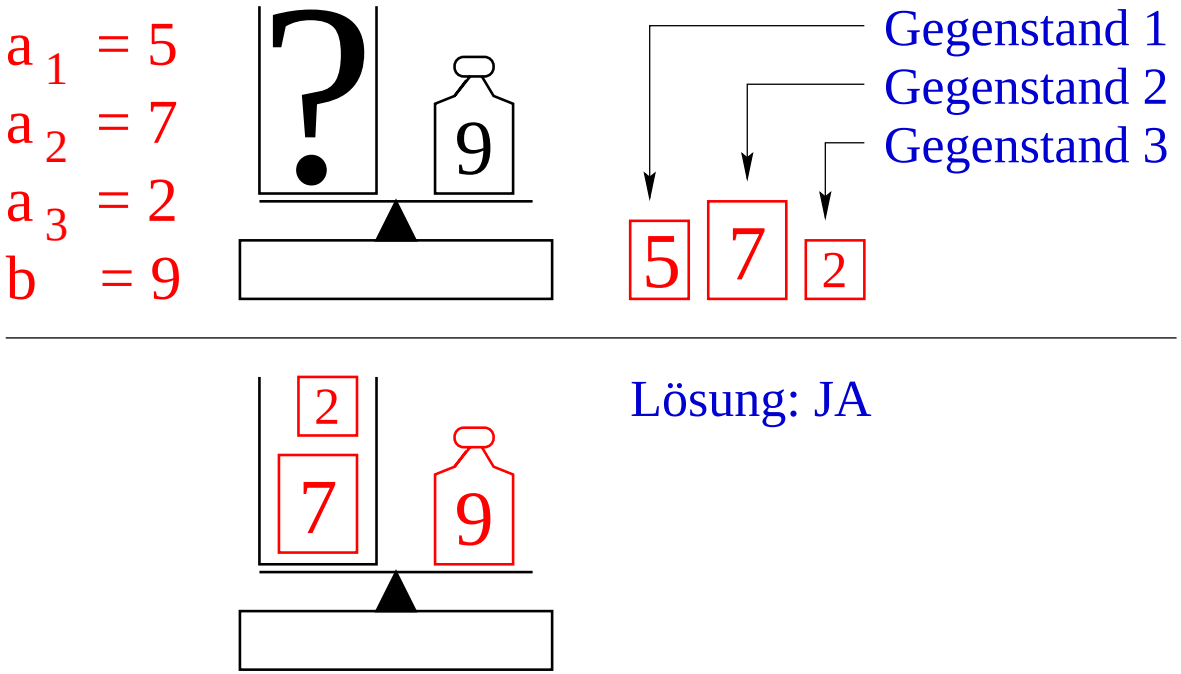
Das Rucksackproblem (knapsack problem)

Problemdefinition

gegeben: n natürliche Zahlen $a_1, \ldots, a_n$ sowie eine natürliche Zahl b

gesucht: Gibt es eine Teilmenge $I \subseteq \{1, 2, \ldots, n\}$ mit $\sum_{i \in I} a_i = b$?

Beispiel eines Rucksackproblems

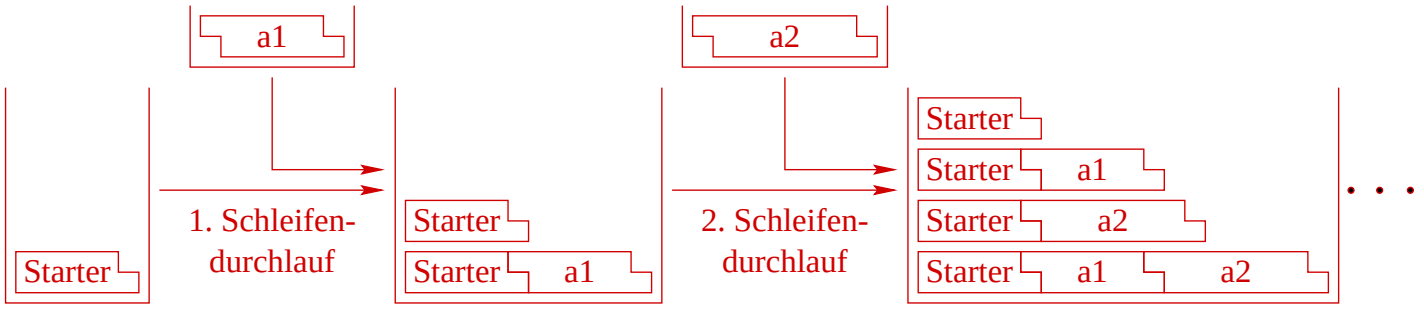


Packmöglichkeit:

{Gegenstand2; Gegenstand3}

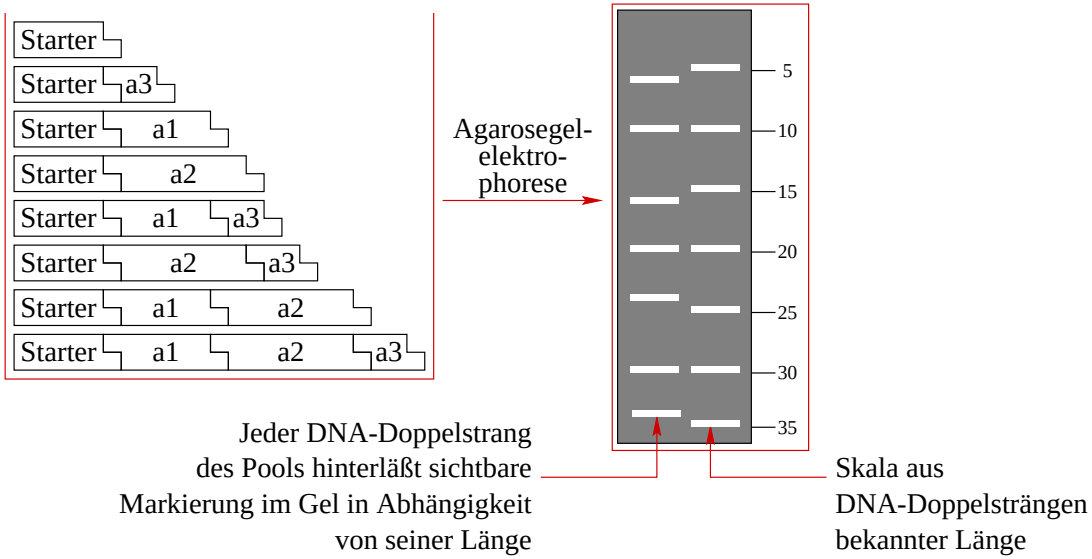
DNA-Algorithmus zur Lösung des Rucksackproblems

- Aufbau aller 2^n Kombinationen (Packmöglichkeiten) durch gezieltes Verketten der DNA-Doppelstränge in n Schleifendurchläufen, danach Längentest
- Jeder Schleifendurchlauf nimmt einen bisher ungenutzten Gegenstand hinzu und verdoppelt die Anzahl der Kombinationen.
- In jedem Schleifendurchlauf dieselbe Abfolge biochemischer Reaktionen

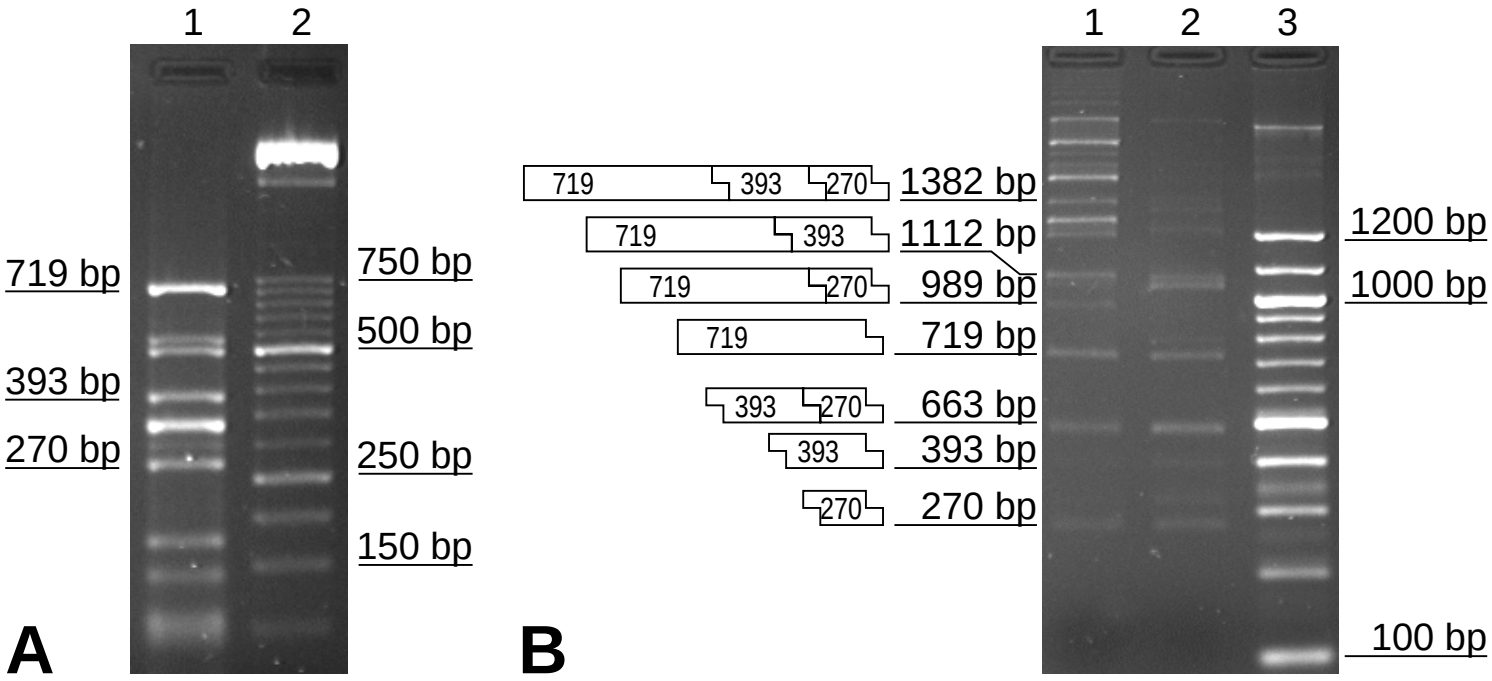


Der Längentest

- Agarosegel-Elektrophorese des finalen DNA-Pools mit 2^n unterschiedlichen DNA-Doppelsträngen, dabei prüfen, ob DNA-Doppelstränge der zu b korrespondierenden Länge existieren
  - Methode zur Längenseparation und Visualisierung von DNA-Doppelsträngen



Auswertung Experiment Rucksackproblem



Bewertung des Algorithmus

- Anzahl Arbeitsschritte linear (und damit polynomiell) in der Anzahl Gegenstände
- prinzipiell skalierbar auf beliebige Problemgrößen (Anzahl Gegenstände), dann jedoch gehäuft Fehler durch Seiteneffekte der biochemischen Reaktionen
- Experiment diente dem prinzipiellen Nachweis, ein NP-Problem im Labor lösen zu können

Modelle des DNA-Computing

Eigenschaften

- Universalität der Modelle, d.h. Modelle berechnen jede berechenbare Funktion und lösen jedes entscheidbare Problem,
- ein universelles Modell ist nicht auf Problemklassen eingeschränkt,
- auf der Basis eines universellen Modells ist es möglich, den programmierbaren Bio-Computer zu konstruieren und zu implementieren.

Splicing-Systeme

Splicing-Systeme dienen als (mathematisches) Modell des DNA-Computing.

Jede Turingmaschine kann simuliert werden durch ein erweitertes Splicing-System mit Multimenge.

- T. Head. *Formal language theory and DNA: an analysis of the generative capacity of specific recombinant behaviors*. Bulletin of the Mathematical Biology, 49(6):737-759, 1987.
- R. Freund, L. Kari, G. Paun. *DNA computing based on splicing: the existence of universal computers*. Theory of Computing Systems, vol. 32, 69-112,1999.
- M. Sturm, T. Hinze: *Distributed Splicing of $\mathcal{RE}$ with 6 Test Tubes*. Workshop on Multiset Processing, Curtea de Arges (2000).

Splicing-Systeme

Splicing-Systeme basieren auf der **Splicing-Operation**. Diese Operation ist definiert für Wörter und damit für formale Sprachen.

Splicing-Systeme generieren mittels **zweier Sprachen** eine **Ausgabesprache**, von der bekannt ist, zu welcher **Klasse von Sprachen** sie gehört, insbesondere, ob sie zu der Menge der endlichen, regulären oder rekursiv aufzählbaren Sprachen ($\mathcal{FIN}$, $\mathcal{REG}$ oder $\mathcal{RE}$) gehört.

Splicing-Systeme

Interpretation von DNA-Sequenzen als lineare Wörter

$$\begin{array}{cccc} A & C & G & T \\ T & G & C & A \end{array}$$

Darstellung von zwei Wörtern als DNA-Sequenzen

$$\begin{array}{ll} 5'CCCCCTCGACCCCC3' & 5'AAAAAGCGCAAAAA3' \\ 3'GGGGGAGCTGGGGG5' & 3'TTTTTCGCGTTTTT5' \end{array}$$

Verwendung zweier Enzyme *TaqI* **und** *SciNI*

$$\begin{array}{ll} 5'TCGA3' & 5'GCGC3' \\ 3'AGCT5' & 3'CGCG5' \end{array}$$

Splicing-Systeme

Schnitt der beiden Wörter mithilfe der Enzyme

$$5'CCCCCCT \quad CGACCCCCC3' \quad 5'AAAAAG \quad CGCAAAAA3'$$
$$3'GGGGGGAGC \quad TGGGGG5' \quad 3'TTTTTCGC \quad GTTTTT5'$$

Ligation

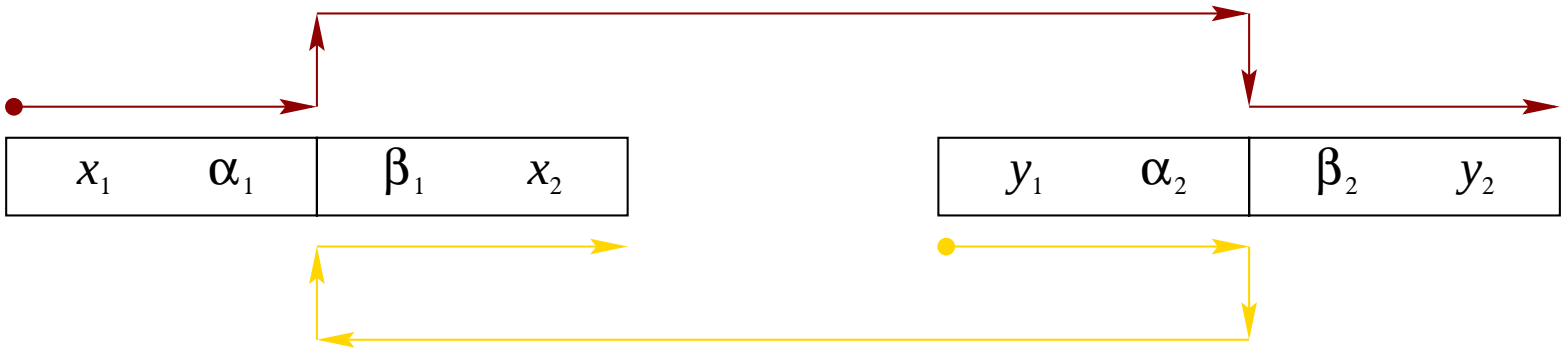
(Ausnutzung der komplementären sticky-Enden der vier Fragmente)

$$5'CCCCCCTCGCAAAAA3' \qquad 5'AAAAAGCGACCCCCC3'$$
$$3'GGGGGGAGCGTTTTT5' \qquad 3'TTTTTCGCTGGGGG5'$$

Splicing-Systeme

Abstraktion vom molekularbiologischen Vorgang

- Nach dem Vorbild der Wirkung der Ligation auf DNA-Stränge mit komplementären sticky-Enden werden aus den Fragmenten geschnittener Wörter neue Wörter generiert.



Ausgabewörter der Splicing-Operation

Splicing-Systeme

Beispiel

Addition zweier positiver natürlicher Zahlen n und m

Gegeben

- das Alphabet $\Sigma = \{a, b, c\}$,
- die Splicing-Regel $r = a\#b\$c\#a$ und
- die Wörter $x = a^n b$, $y = ca^m$.

Splicing-Systeme

$$
\begin{aligned}
(a^n b, ca^m) &= (x, y) \\
&= (\underbrace{a^{n-1}}_{x_1}\underbrace{a}_{\alpha_1}\underbrace{b}_{\beta_1}\underbrace{\lambda}_{x_2}, \underbrace{\lambda}_{y_1}\underbrace{c}_{\alpha_2}\underbrace{a}_{\beta_2}\underbrace{a^{m-1}}_{y_2}) \\
&\to_r (\underbrace{a^{n-1}}_{x_1}\underbrace{a}_{\alpha_1}\underbrace{a}_{\beta_2}\underbrace{a^{m-1}}_{y_2}, \underbrace{c}_{y_1\alpha_2}\underbrace{b}_{\beta_1 x_2}) \\
&= (a^{n+m}, cb) = (z, w)
\end{aligned}
$$

Splicing-Systeme

Definition (Splicing-System)

Ein Splicing-System ist ein Quadrupel $\gamma = (V, \Sigma, A, R)$, in dem V ein Alphabet ist, $\Sigma \subseteq V$, $A \subseteq V^+$ und $R \subseteq V^*\#V^*\$V^*\#V^*$ mit $\#, \$ \notin V$.
V ist das Alphabet des Splicing-Systems γ, Σ das Alphabet der Sprache, A die Menge der Axiome und R die Menge der Splicing-Regeln. Die Elemente von Σ werden als Terminalzeichen und die Elemente von $V - \Sigma$ als Nichtterminalzeichen bezeichnet.

Für $x, y, z, w \in V^+$ und $r = \alpha_1\#\beta_1\$\alpha_2\#\beta_2 \in R$ wird definiert:
$(x, y) \rightarrow_r (z, w)$, wenn
$x = x_1\alpha_1\beta_1x_2$, $y = y_1\alpha_2\beta_2y_2$ und $z = x_1\alpha_1\beta_2y_2$, $w = y_1\alpha_2\beta_1x_2$
für $x_1, x_2, y_1, y_2 \in V^*$ gilt.

Splicing-Systeme

Definition der Sprache (generierbar über ein Splicing-System)

$$L(\gamma) = \sigma^*(A) \cap \Sigma^+$$

$\sigma(A) = \{z \in V^+ \mid (x,y) \vdash_r (z,w) \text{ für } x,y \in A, w \in V^+, r \in R\}$

Transitive und reflexive Hülle für $\sigma(A)$

$$\sigma^*(A) = \bigcup_{i \geq 0} \sigma^i(A)$$

$$\sigma^0(A) = A,\ \sigma^{i+1}(A) = \sigma^i(A) \cup \sigma(\sigma^i(A)),\ i \geq 0$$

Splicing-System
$\gamma = (V, \Sigma, A, R)$ mit $A \in F_1$ und $R \in F_2$ ist vom Typ (F_1, F_2)

$\mathcal{H}(F_1, F_2) = \{L(\gamma) \mid \gamma = (V, \Sigma, A, R), A \in F_1, R \in F_2\}$
(erweitertes Splicing-System: $V \neq \Sigma$)

Splicing-Systeme

Aussage der Tabelle[a] ist, welcher Typ (F_1, F_2) Splicing-System $E\mathcal{H}$ zu welchem Typ Sprache führt.

F_1 / F_2	FIN	REG	CF	CS	RE
FIN	REG	RE	RE	RE	RE
REG	REG	RE	RE	RE	RE
CF	CF	RE	RE	RE	RE
CS	RE	RE	RE	RE	RE
RE	RE	RE	RE	RE	RE

CF - kontextfreie Sprachen (Typ Chomsky-2), CS - kontextsensitive Sprachen (Typ Chomsky-1), REG - reguläre Sprachen (Typ Chomsky-3), FIN - endliche Sprachen, RE - rekursiv aufzählbare Sprachen

[a] R. Freund, L. Kari, G. Paun. DNA computing based on splicing: the existence of universal computers. Theory of Computing Systems, vol. 32, 69-112,1999

Verteilte Splicing-Systeme

Zielstellung

- Konstruktion einer Extension zum klassischen Splicing-System, welches die Klasse $\mathcal{RE}$ generiert
- Extension zum Splicing-System soll laborpraktisch umsetzbar sein
- Extension zum Splicing-System soll mit linearen DNA-Strängen arbeiten
- Extension zum Splicing-System soll mit einer minimalen Anzahl von Testtubes arbeiten (One-Pot)

Klassifikation von Splicing-Systemen

class	splicing system	# tubes	# axioms	# splicing rules
distr.	Multiple	1	$O(\|\mathcal{M}\|)$	$O(\|\mathcal{D}\|^3)$
	TT system	$O(\|\Sigma_G\|)$	$O(\|\mathcal{P}_G\| + \|\Sigma_G\|)$	$O(\|\mathcal{P}_G\| + \|\Sigma_G\|)$
	CDEH system	3	$O(\|\mathcal{P}_G\| + \|\mathcal{V}_G\| + \|\Sigma_G\|)$	$O(\|\mathcal{P}_G\| + \|\mathcal{V}_G\| + \|\Sigma_G\|)$
	TT6 system	6	$O(\|\mathcal{P}_G\| + \|\Sigma_G\|)$	$O(\|\mathcal{P}_G\| + \|\Sigma_G\|)$
inf. set	$EH(\mathcal{FIN}, \mathcal{REG})$	1	$O(\|\mathcal{P}_G\| + \|\mathcal{V}_G\| + \|\Sigma_G\|)$	∞
nonlin.	Circular	1	$O(\|\mathcal{P}\| + \|\mathcal{A}\|)$	$O(\|\mathcal{P}\|)$
multiset	$EH(m\mathcal{FIN}, \mathcal{FIN})$	1	$O(\|\mathcal{P}_G\| + \|\mathcal{V}_G\| + \|\Sigma_G\|)$	$O(\|\mathcal{P}_G\| \cdot (\|\mathcal{V}_G\| + \|\Sigma_G\|)^5)$
		1	$O(\|\mathcal{V}\|^2\|\mathcal{Q}\| + \|\mathcal{Q}\|^2\|\mathcal{V}\|)$	$O(\|\mathcal{V}\|^4\|\mathcal{Q}\|)$
		1	$O(\|\Sigma\| \cdot \|\mathcal{P}\|)$	$O(\|\Sigma\| \cdot \|\mathcal{P}\|)$

Elementare Formale Systeme (EFS): $EFS = (\mathcal{D}, \Sigma, \mathcal{M})$, Post-Normal-Systeme (PNS): $G = (\mathcal{V}, \Sigma, \mathcal{P}, \mathcal{A})$, Chomsky-Typ-0-Grammatiken: $G = (\mathcal{V}_G, \Sigma_G, \mathcal{P}_G, S_G)$, Deterministische Turingmaschinen: $TM = (\mathcal{Q}, \mathcal{V}, \Sigma, \{L, R\}, \mathcal{F}, q_0, \delta, \square)$

Verteilte Splicing-Systeme

$G_{KNF} = (\mathcal{V}_G, \Sigma_G, \mathcal{P}_G, S_G)$ ist eine Chomsky-Typ-0-Grammatik in Kuroda-Normalform mit den Komponenten

$\mathcal{V}_G$: Menge der Nichtterminalzeichen $V_G = \{v_1, \ldots, v_n\}$

Σ_G: Menge der Terminalzeichen $\Sigma_G = \{\sigma_1, \ldots, \sigma_n\},\ \mathcal{V}_G \cap \Sigma_G = \emptyset$

$\mathcal{P}_G$: Menge der Produktionsregeln $\mathcal{P}_G \subseteq (\mathcal{V}_G \times (\mathcal{V}_G \otimes \mathcal{V}_G)) \cup (\mathcal{V}_G \times \Sigma_G) \cup (\mathcal{V}_G \times \{\varepsilon\}) \cup ((\mathcal{V}_G \otimes \mathcal{V}_G) \times (\mathcal{V}_G \otimes \mathcal{V}_G))$

S_G: Satzsymbol $S_G \in \mathcal{V}_G$

Verteilte Splicing-Systeme

$\Gamma = (\mathcal{V}, T_1, T_2, T_3, T_4, T_5, T_6)$ ist ein auf G_{KNF} basierendes TT6-EH-System mit den Komponenten

$\mathcal{V}$: Menge der Alphabetzeichen $\quad \mathcal{V} = \mathcal{V}_G \cup \Sigma_G \cup \{B, \alpha, \beta, X, X', Y, Y', Y'_\alpha, Y'_\beta, Z, Z', Z''\}$

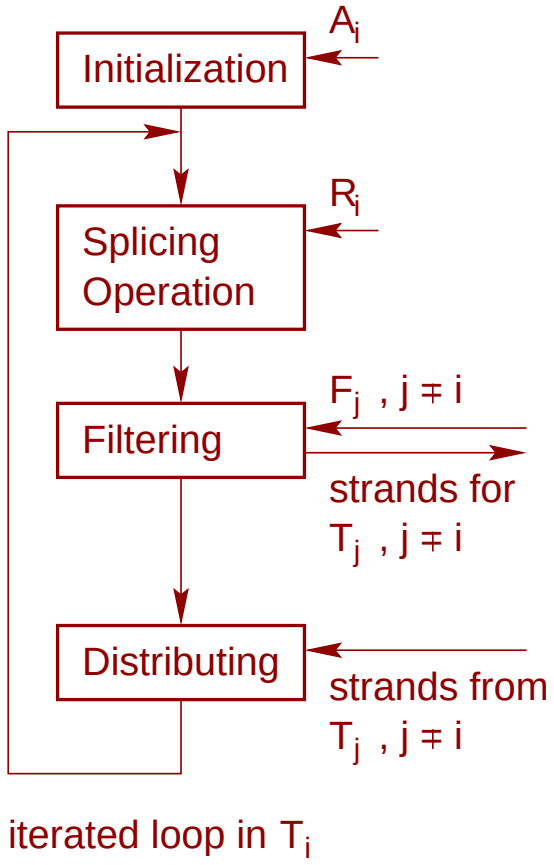
T_i: Testtube i, $i = 1, \ldots, 6$ mit $\quad T_i = (\mathcal{A}_i, \mathcal{R}_i, \mathcal{F}_i)$

- $\mathcal{A}_i$: Menge der Axiome $\quad \mathcal{A}_i \subset \mathcal{V}^*$
- $\mathcal{R}_i$: Menge der Splicingregeln $\quad \mathcal{R}_i \subset \mathcal{V}^* \otimes \{\#\} \otimes \mathcal{V}^* \otimes \{\$\} \otimes \mathcal{V}^* \otimes \{\#\} \otimes \mathcal{V}^*$
- $\mathcal{F}_i$: Menge der Filtermuster $\quad \mathcal{F}_i \subset \mathcal{V}^* \otimes \mathcal{V}^*$

Verteiltes Splicing mit TT6

Schleife

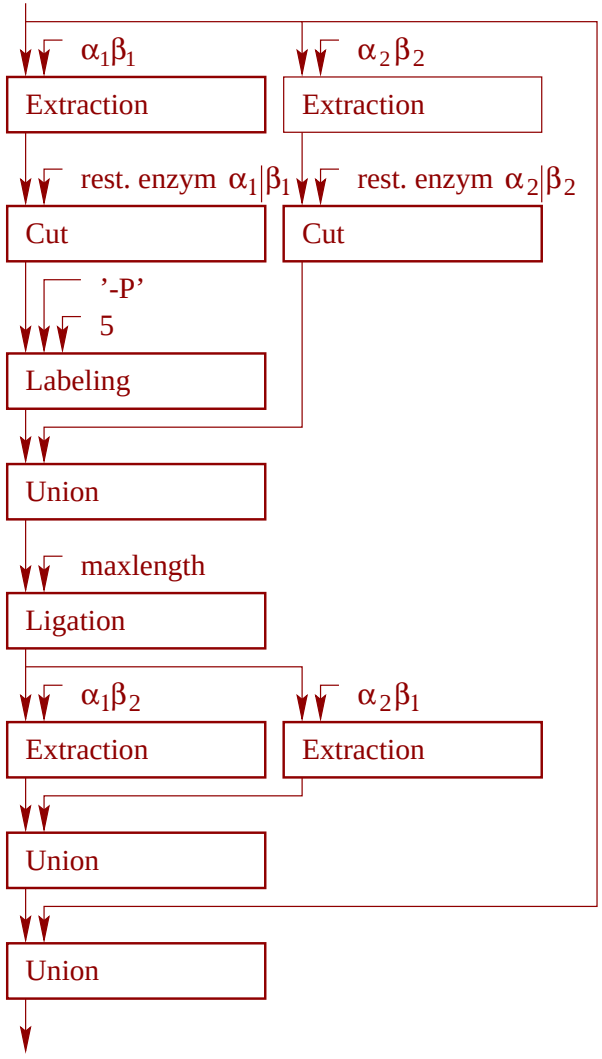
- die Inhalte der Testtubes T_1 bis T_5 durchlaufen iterativ eine Schleife
- eine Initialisierung erfolgt durch die Menge der Axiome $\mathcal{A}_i$ (kodiert als DNA-Stränge)
- die Schleife besteht aus den DNA-Operationen Splicing, Filtern und Verteilung
- T_6 erhält durch Filtern nur Stränge, die Wörter der Sprache repräsentieren ($\in \mathcal{L}(G)$)



Splicing-Operation des TT6-Systems

Splicing-Operation

- Selektion und Anwendung einer passenden Splicing-Regel aus $\mathcal{R}_i$
- für die Komponenten α_1, β_1, α_2, β_2 werden Restriktionsenzyme und DNA-Stränge für eine Extraktion eingesetzt
- mihilfe von Digestion und Ligation wird die Splicing-Operation ausgeführt, weitere Operationen (Extraktion und Labeling) dienen dem Ausschluß nicht gewünschter DNA-Stränge



Verteiltes Splicing mit TT6

$$(\mathcal{L}_1, \mathcal{L}_2, \ldots, \mathcal{L}_6) \xrightarrow{1}_{TT6} (\mathcal{L}'_1, \mathcal{L}'_2, \ldots, \mathcal{L}'_6)$$

mit

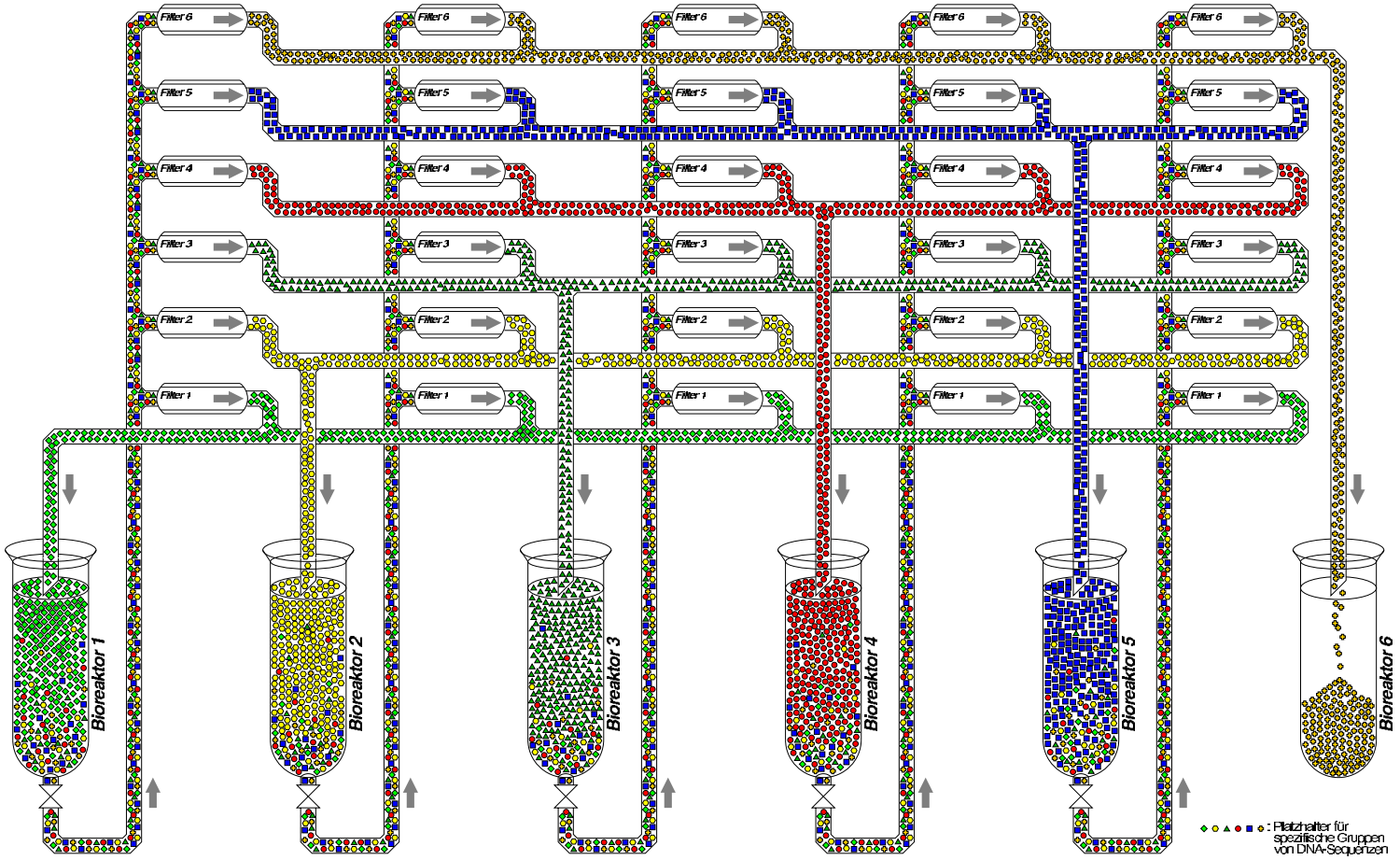
$$\mathcal{L}'_i = \left(\left(\bigcup_{\substack{j=1 \wedge \\ j \neq i}}^{6} \sigma_j^1(\mathcal{L}_j) \right) \cap \mathcal{S}_i \right) \cup \sigma_i^1(\mathcal{L}_i) \quad \forall i \in \{1, \ldots, 6\}$$

$$S_k = \{w \in V^* \mid \exists (p_a, p_e) \in F_k\}$$

$$\mathcal{L}(\Gamma) = \left\{ w \in \mathcal{L}_6 \cap \Sigma^* \mid (\mathcal{A}_1, \ldots, \mathcal{A}_6) \xrightarrow{*}_{TT6} (\mathcal{L}_1, \ldots, \mathcal{L}_6) \right\}$$

(p_a ist Präfix von w und p_e ist Suffix von w,
$\xrightarrow{*}_{TT6}$ ist die reflexive und transitive Hülle von $\xrightarrow{1}_{TT6}$)

Prinzipskizze zum Splicing-System TT6

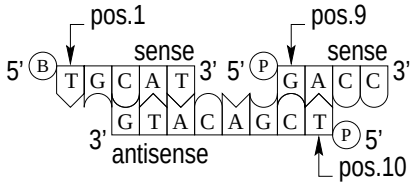
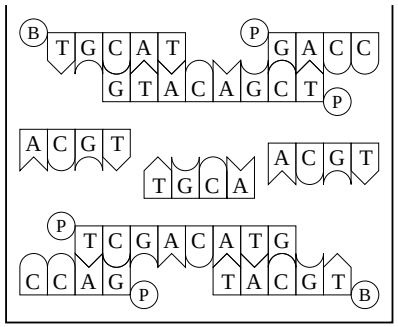


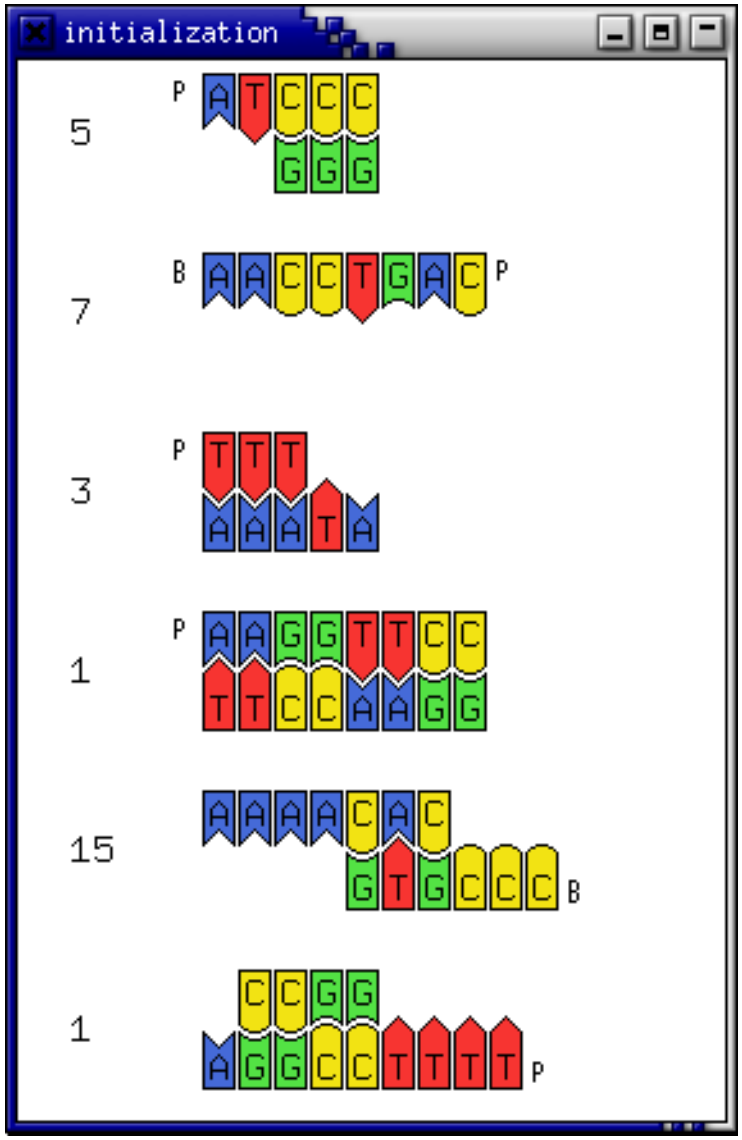
DNA-Haskell

- Entwickler TU Dresden
- "Functional Workbench for Design of DNA Algorithms with Experimental Study"
- Basis: funktionale Programmiersprache HASKELL
- Ziel: Konstruktion von DNA-Algorithmen
- Spezifikation von molekularbiologischen Operationen
- Benutzerdefinierte rekursive Datentypen (DNA-Einzel-und Doppelstränge einschließlich Einzelstrangüberhänge, 5'- und 3'-Label)
- Implementation von Multimengen
- Nachweis von Programmeigenschaften und Programmtransformationen

DNA-Haskell

Beispiel für die Repräsentation von DNA-Strängen und Testtubes in Haskell

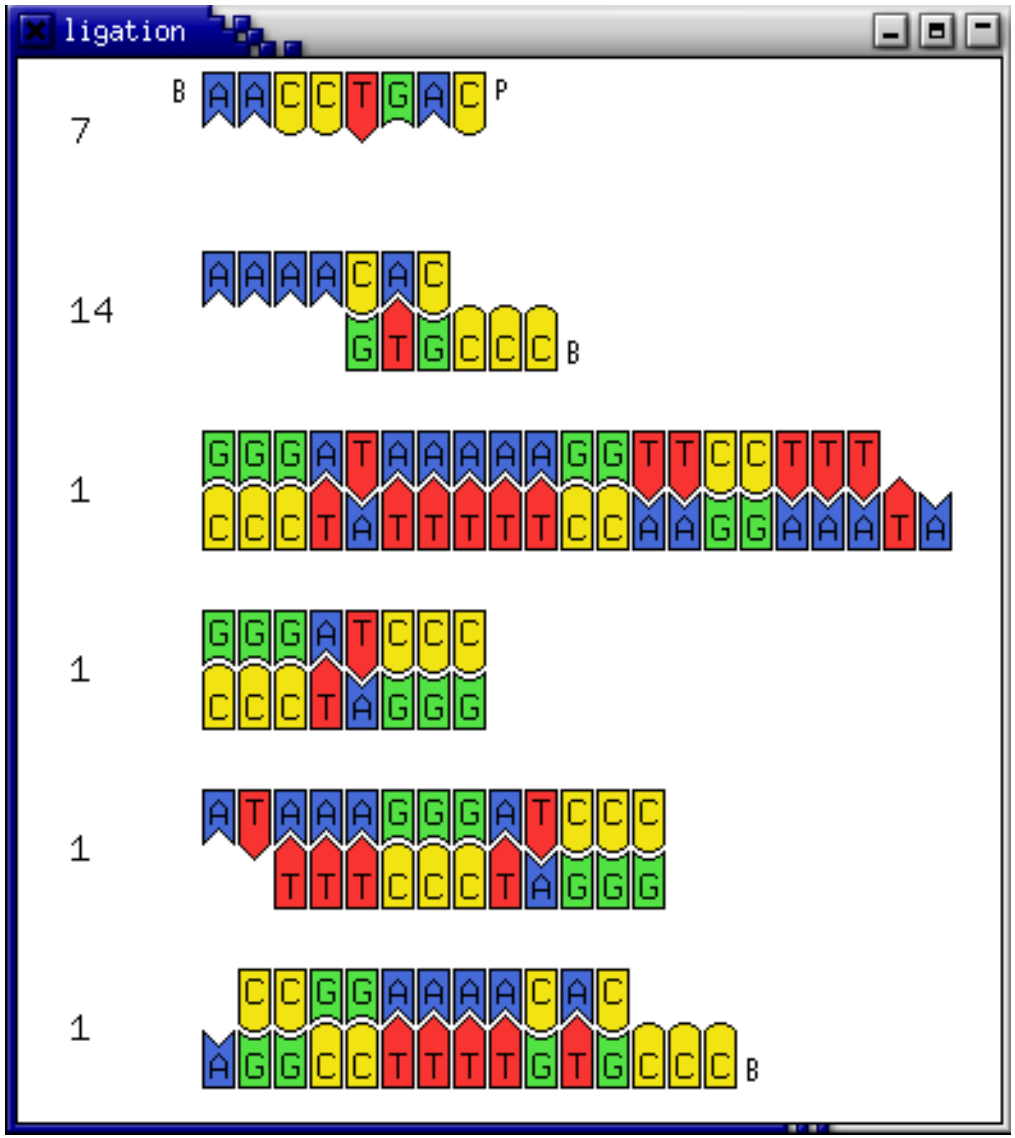
type	example of a value from type Strand			example of a value from type Tube	
	biological representation		encoding in Haskell	biological representation	encoding in Haskell
Strand composed of			[(strand1, 1), (strand2, 9), (strand3, -10)]		[([((H, [A, C, G, T], H), 1)], 3), ([((B, [T, G, C, A, T], H), 1), ((P, [G, A, C, C], H), 9), ((P, [T, C, G, A, C, A, T, G], H), -10)], 2)]
BasicStrand		strand1	(B, [T, G, C, A, T], H)		
BasicStrand		strand2	(P, [G, A, C, C], H)		
BasicStrand		strand3	(P, [T, C, G, A, C, A, T, G], H)		



DNA-Haskell

Im Testtube befinden sich insgesamt 32 DNA-Stränge (Einzelstränge, Doppelstränge mit blunt- und/oder sticky-Enden). Dabei treten 6 voneinander unterschiedliche Strangsequenzen auf. Für die Ausführung der Operation Ligation kommt es zur Auswahl eines dieser 32 DNA-Stränge. Dieser wird untersucht, inwieweit eine Verbindung mit allen restlichen Strängen möglich ist.

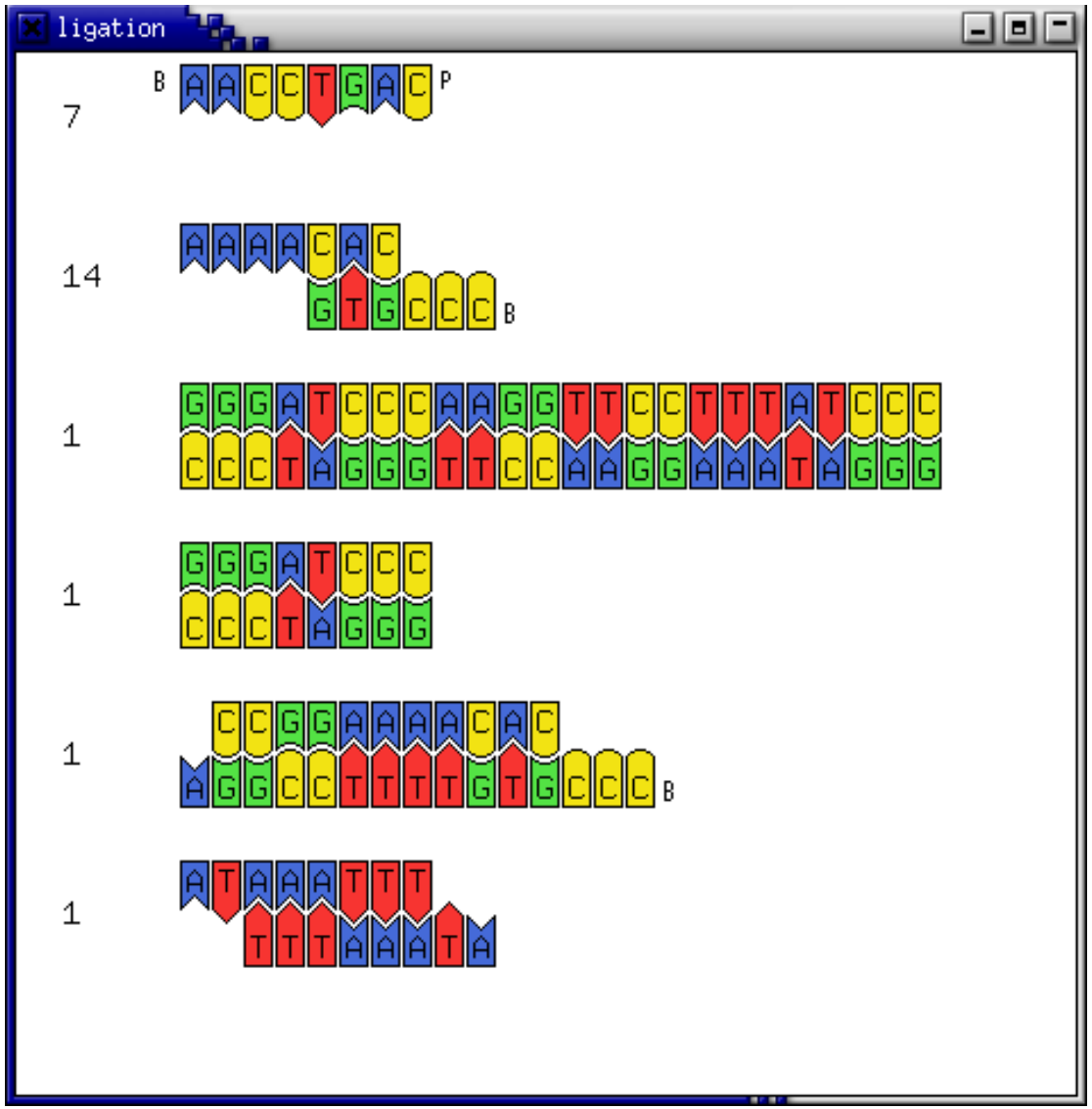
Initialisierung im Testtube



DNA-Haskell

Die Ligation ist eine biochemische Reaktion, die durch das Enzym DNA-Ligase katalysiert wird und das fortgesetzte Verketten von DNA-Doppelsträngen an ihren Enden bewirkt. Die Enden müssen hierzu kompatibel, also paßgenau sein und eine Phosphatgruppe besitzen.

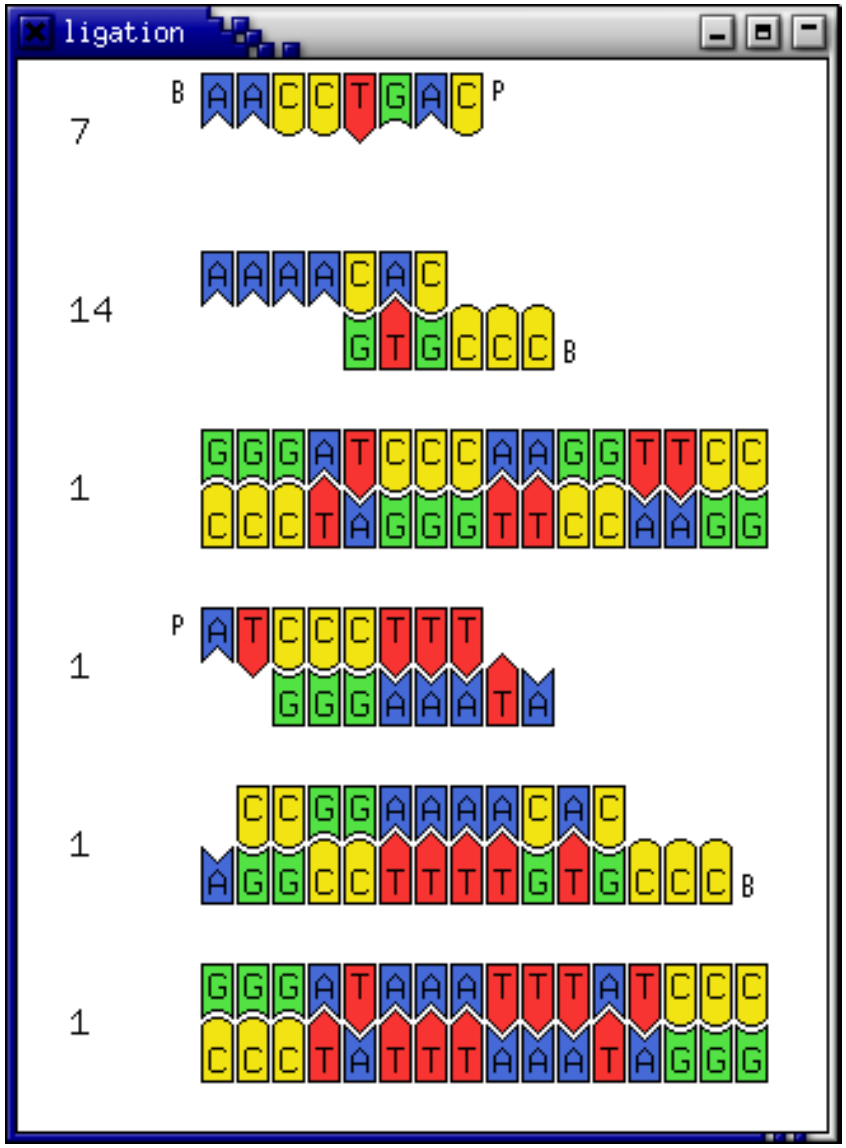
Ergebnis Ligation 1



DNA-Haskell

Bei der Ligation werden aus Bakterien gewonnene und kommerziell angebotene DNA-Ligasen verwendet (z. B. T4-DNA-Ligase). Eine in-vitro-Ligation führt man gewöhnlich bei einer Temperatur zwischen 16°C und 20°C durch, wobei die Inkubationszeit i.a. bis zu 16 h beträgt.

Ergebnis Ligation 2



DNA-Haskell

Die Ligation ist mit mehreren Seiteneffekten behaftet. Infolge intramolekularer Ligation können sich DNA-Ringe bilden. Desweiteren ist es möglich, daß die DNA-Konzentrationen der durch die Ligation enstandenen DNA-Fragmentketten-Kombinationen stark voneinander abweichen.

Ergebnis Ligation 3

Simulation molekularbiologischer Prozesse

- Entwickler TU Dresden
- Simulationstool zur Überwindung der Diskrepanz zwischen formalen Modellen des DNA-Computing und ihrer laborpraktischen Implementation
- Modellierung von linearen DNA-Einzel- und Doppelsträngen auf der Basis von Nukleotiden und Strangendenmarkierungen
- Spezifikation von DNA-Operationen, die auf eine Multimenge von DNA-Strängen spezifisch und gesteuert über Parameter einwirken
- Parameter unterteilen sich in Operations- und Seiteneffektparameter
- parametergesteuerte Ausführung jeder Operation auf der Basis eines probabilistischen Ansatzes

Simulation molekularbiologischer Prozesse

- Anzahl von Strangkopien bildet ein Maß für ihre Konzentration im Tube und beeinflußt Prozessabläufe bei Operationsausführung
- Implementation des Simulationstools nutzt die Programmiersprache Java und realisiert eine objektorientierte Simulation

Ziel

- Einbeziehung nichtlinearer DNA-Strukturen und ihre Integration in die Spezifikation der DNA-Operationen
- Aufnahme bisher nicht erfaßter Seiteneffekte, ihre geeignete Parametrisierung und Berücksichtigung ihrer Wirkung auf die betrachteten DNA-Operationen

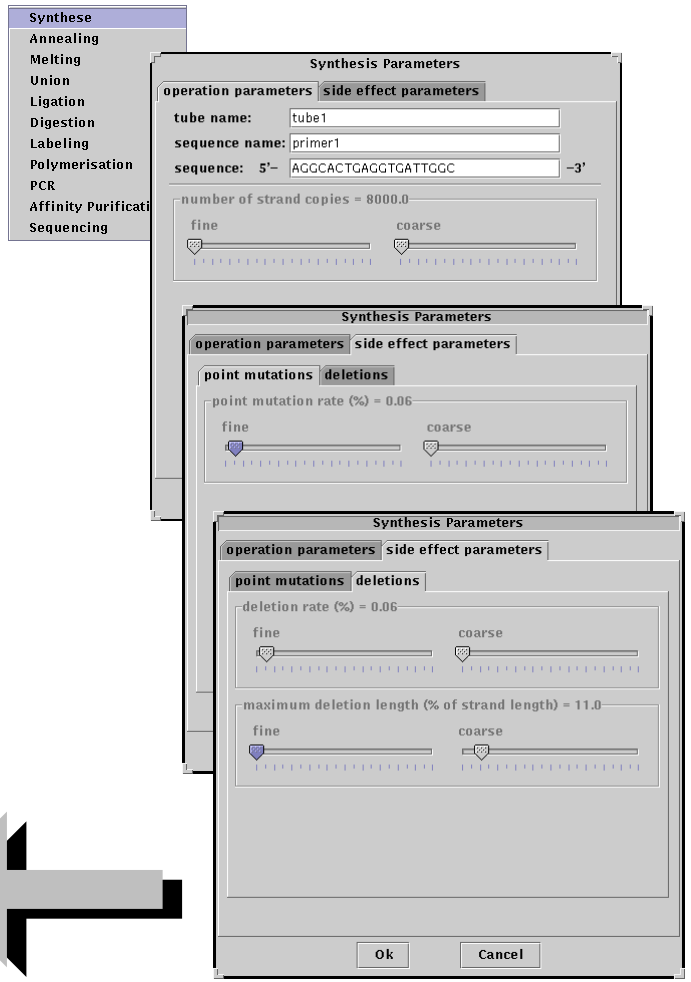
Beispiel - Synthese

operation parameters:

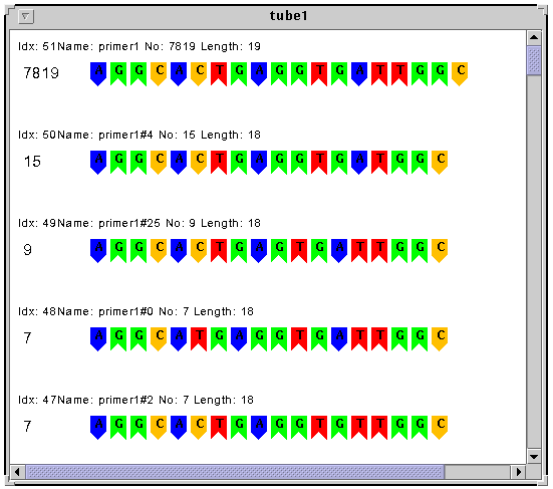
tube name:	tube1
nucleotide sequence (5'-3'):	AGGCACTGAGGTGATTGGC
number of strand copies:	8 000

side effect parameters:

point mutation rate:	0.06%
deletion rate	0.06%
maximum deletion length:	11% of strand length



output of test tube contents



Ausblick

Zielstellungen

- Implementation des modellierten Mehrtubesystems für Beispielgrammatiken vom Typ 0
- Bereitstellung von Modulen (spezifische Definition der Komponenten von T_1 bis T_5 entsprechend einer anwendungsorientierten Problemstellung)
- Konstruktion spezieller Algorithmen mithilfe des Tools „DNA-Haskell“
- Erweiterung des Simulationstools, Fertigstellung des Tools „DNA-Haskell“
- Verfeinerung des funktionalen Modells zur formalen Beschreibung molekularbiologischer Abläufe einschließlich der Erfassung von Seiteneffekten
- Präzisierung in der Vorhersage von Laborergebnissen

Literatur

- T. Hinze, M. Sturm. *Rechnen mit DNA - Eine Einführung in Theorie und Praxis*. ISBN 3-486-27530-5, Oldenbourg Wissenschaftsverlag München, 316 S., 2004
- T. Hinze, M. Sturm. *Eine DNA-Arithmetik auf der Basis von Chomsky-Grammatiken*. In S. Bensch, O. Boldt, H. Bordihn, H. Jürgensen (Hrsg.), Tagungsband 14. Theorietag „Automaten und Formale Sprachen“, Caputh, ISSN 0946-7580, Universität Potsdam, 2004
- M. Sturm, T. Hinze. *Distributed Splicing of RE with 6 Test Tubes*. Romanian Journal of Information Science and Technology, ISSN 1453-8245, Editura Academiei Romane 4(1-2):211-234, 2001 und in C.S. Calude, M.J. Dinnen, G. Paun, editors, Proceedings Workshop on Multiset Processing, Curtea de Arges, CDMTCS Research Report No. 140, pp. 236-248, 2000