

# Bioinformatik

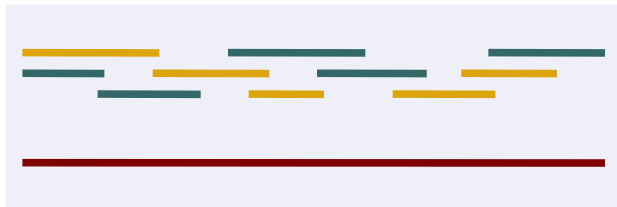
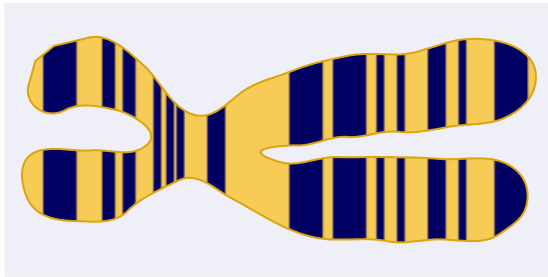
Gene Finding  
Sequenzanalyse mit  
(Hidden) Markov Modellen

Ulf Leser

Wissensmanagement in der  
Bioinformatik

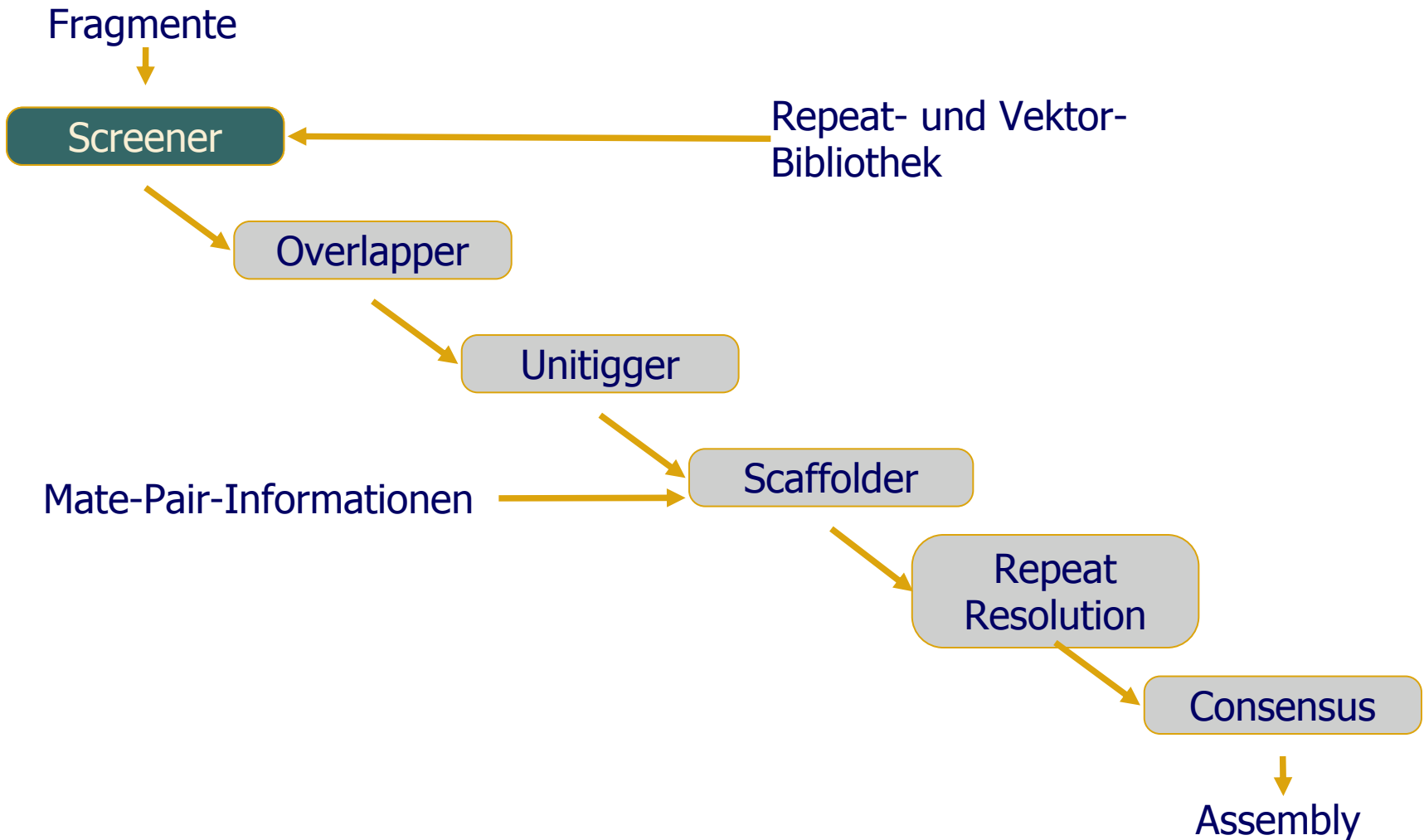


# Whole Genome Shotgun



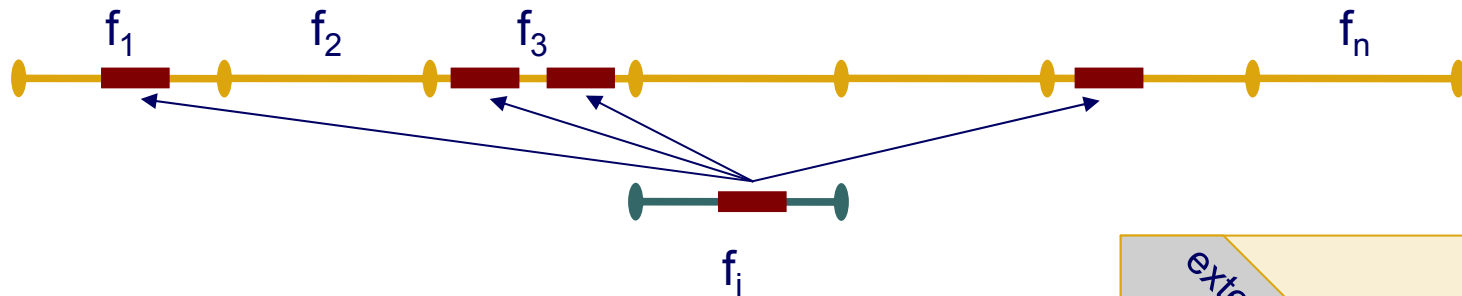
- Zerschneiden eines kompletten Chromosoms in Einzelstücke
- (An)sequenzierung jedes Bruchstücks
- **Assembly** der Einzelsequenzen zur Konsensussequenz
- Konkret
  - Clone der Länge 2kb, 10kb, 50kb und 150kb
  - "Double Barrel" Shotgun
    - **Ansequenzieren** von beiden Seiten

# Assembler - Überblick

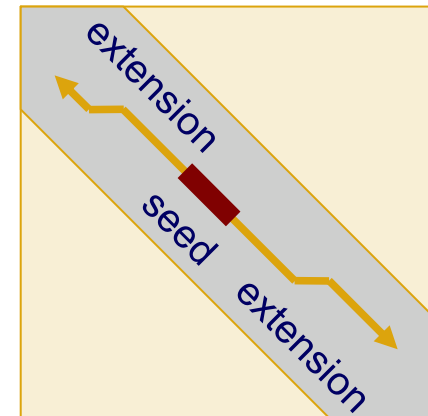


# Seed and Extend

- "Seed and Extend" Ansatz (ähnlich BLAST)
  - Konkatination aller Fragmente
  - Seeds: Suche **exakte Matches der k-mere** von  $f_i$  in der konkatinierten Gesamtsequenz

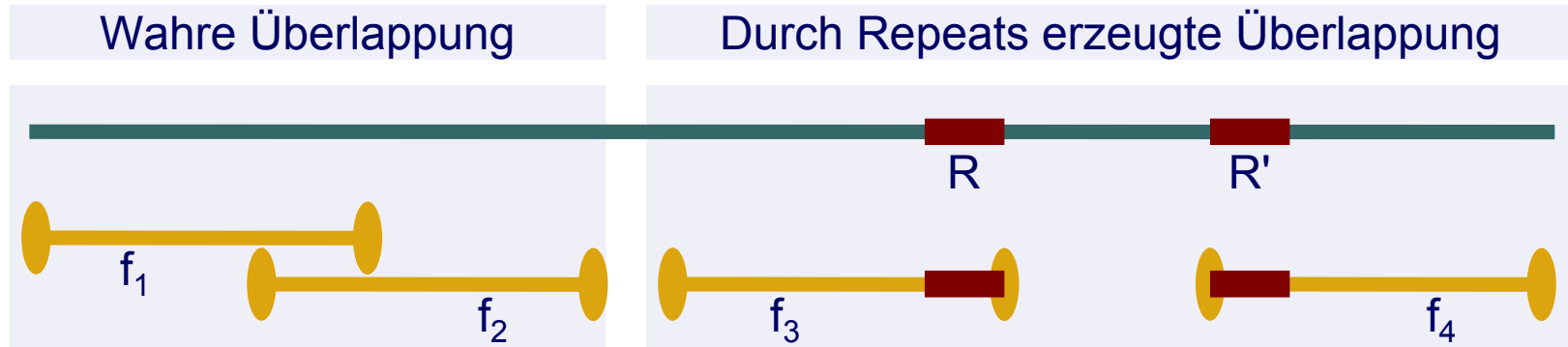


- Extension: Erweiterung der Seeds durch **Banded-Alignment**
  - Bis heuristisches Abbruchkriterium erfüllt



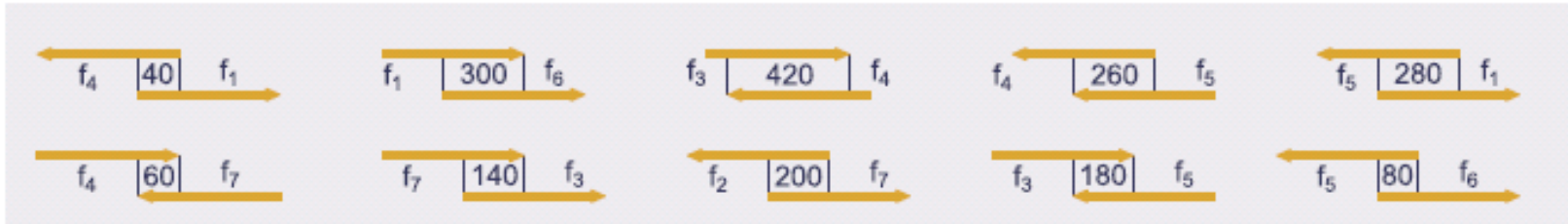
# Repeats: DAS Problem

- Repeat-induced Overlaps

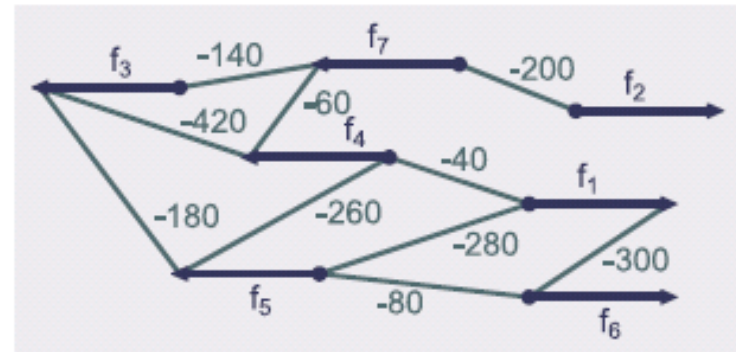


- So gut wie möglich vermeiden durch
  - Screening bekannter Repeats (ALU, SINEs, LINEs, ...)
  - Sehr häufige k-mere ignorieren
- **Repeats bleiben das Hauptproblem des Assemblies**
  - Führen zu falsch-positiven Überlappungen
  - Bilden Verbindungen über das gesamte Chromosom
  - Zerstören/Überlagern die wahre Anordnung

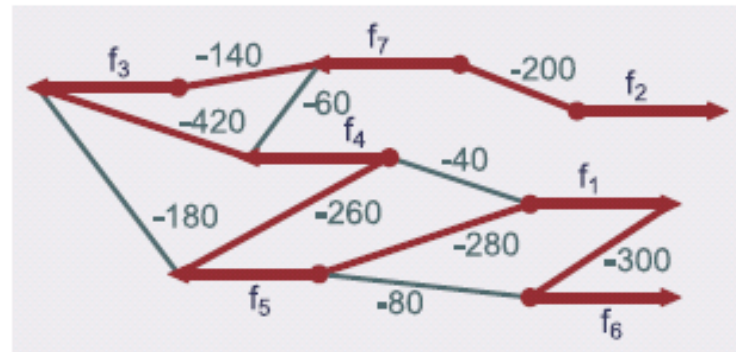
# Beispiel



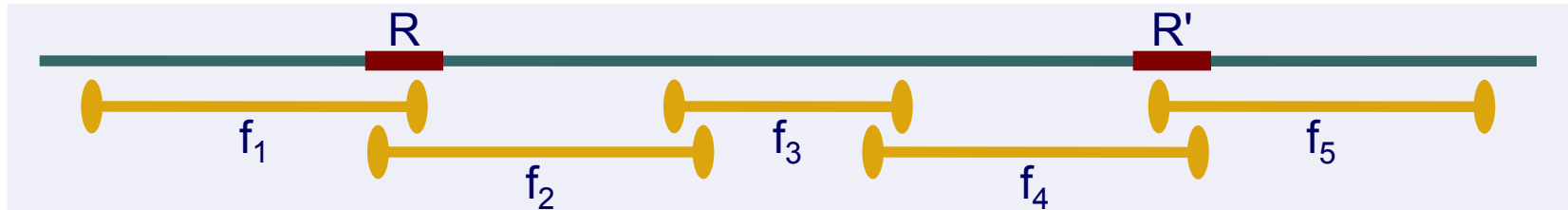
- Overlap Graph bilden
  - Nicht alle Kanten haben Richtung



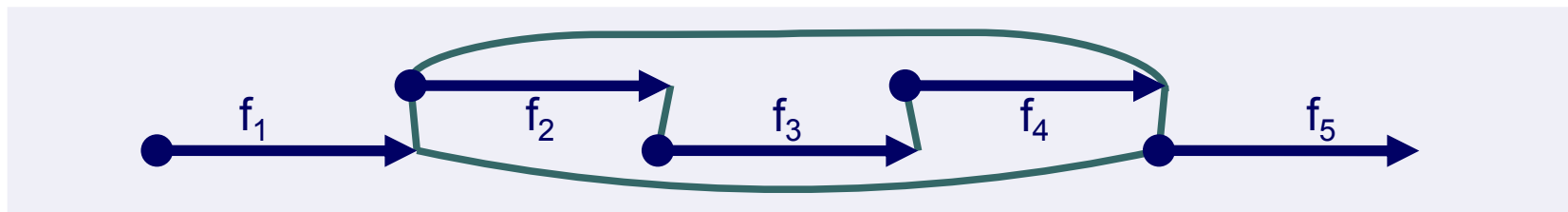
- Kanten sortieren und Greedy markieren
  - -180 würde Kreis schließen
  - Nach -140 Terminierung – danach nur noch Kreise



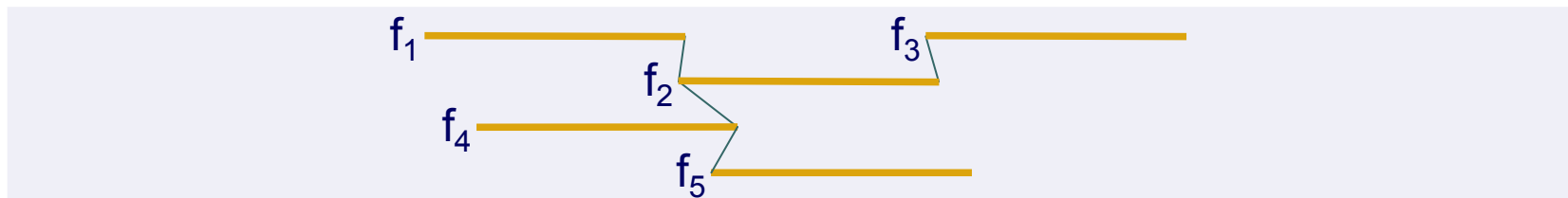
# Repeats und Unitigs



- Erzeugen "falsche" Overlap-Kanten:

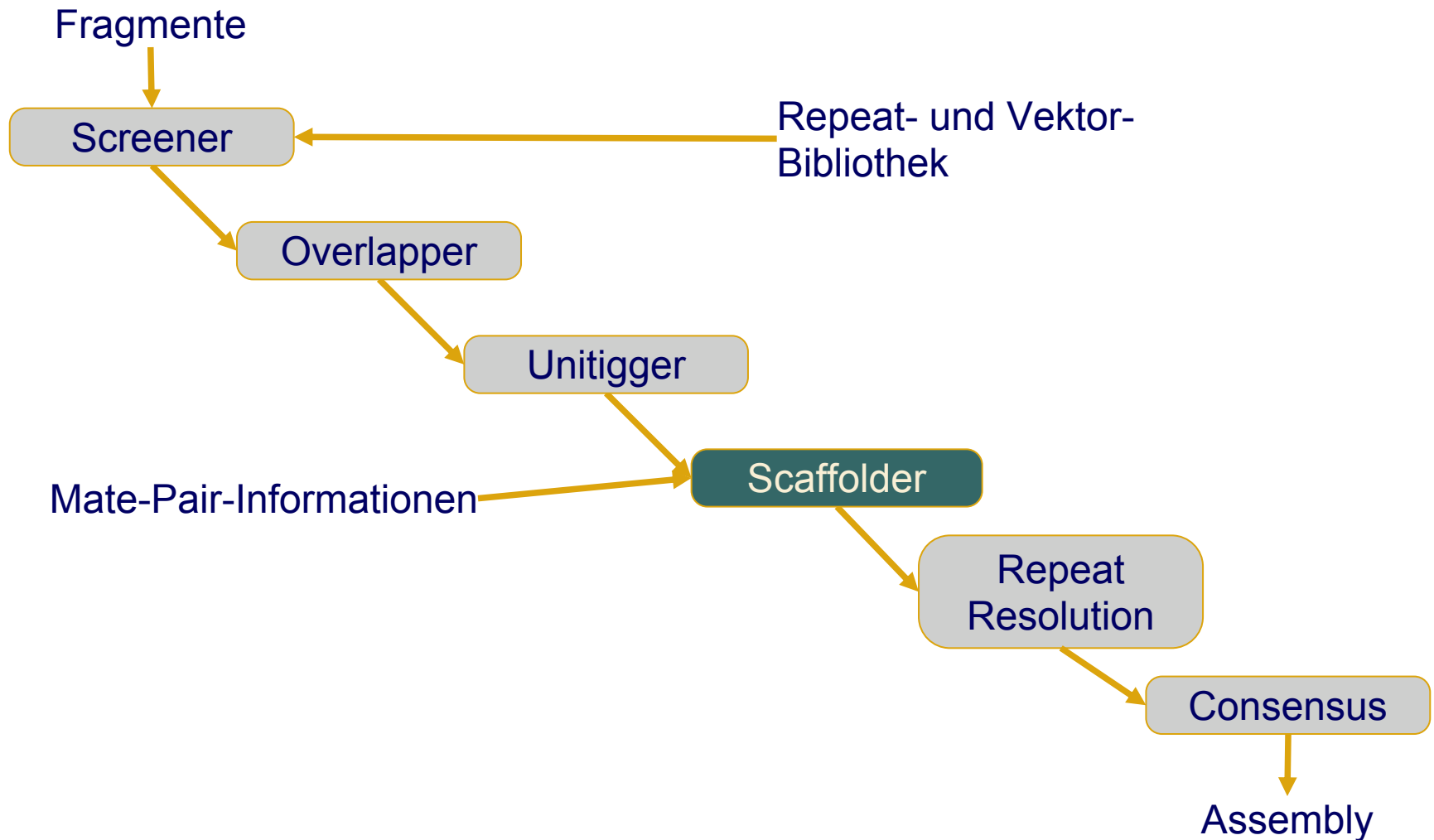


- Erzeugter Contig:



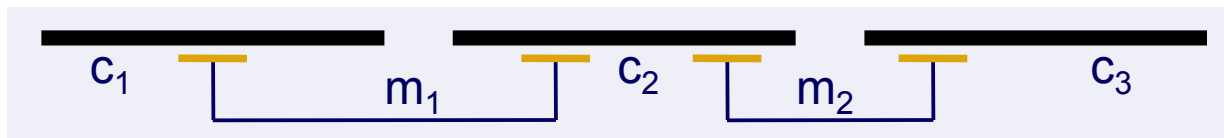
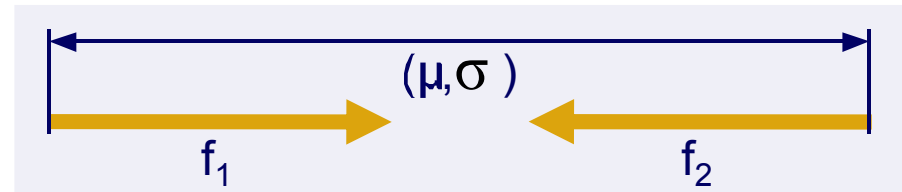
→ Kein konsistentes Layout möglich

# Überblick



# Scaffolder

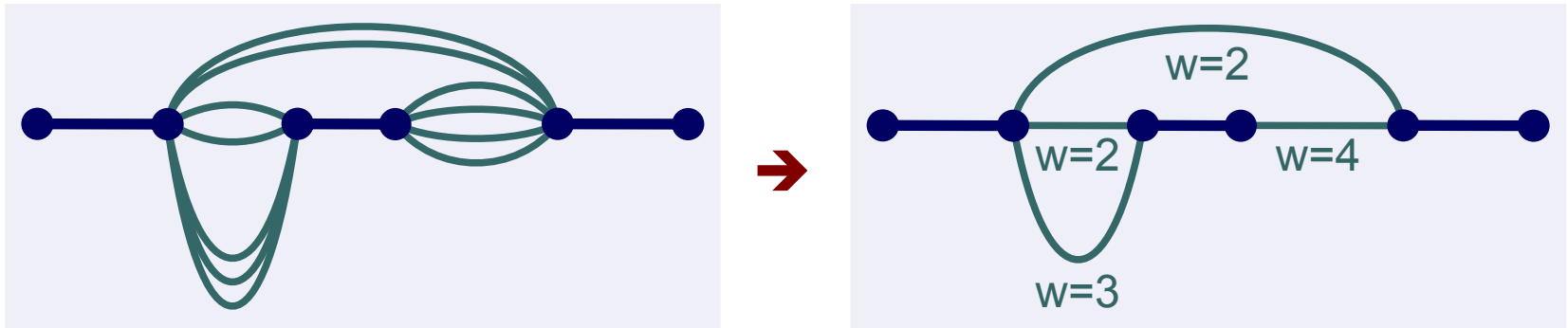
- Verbindet Unitigs zu Scaffolds
  - Gerüst für ein komplettes Chromosom
- Verwendet Unitigs und Mate-Pairs
  - "Double Barrel" Shotgun:  
Paare von Fragmenten mit bekannter Orientierung, Distanz  $\mu$  und Standardabweichung  $\sigma$
- Scaffold
  - Durch Mate-Pairs verbundene Contigs



# Vereinfachung

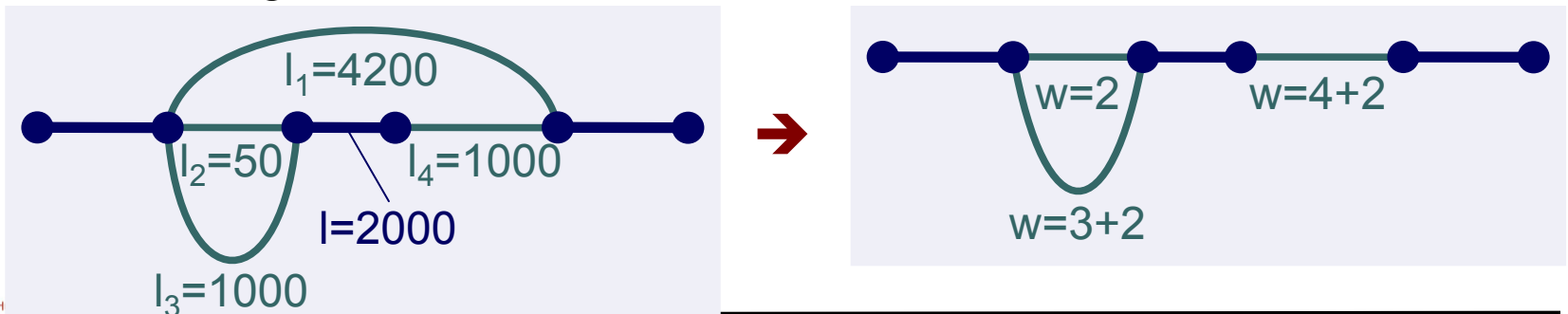
- Edge-Bundling

- Heuristisch sich bestätigende Mate-Kanten verschmelzen



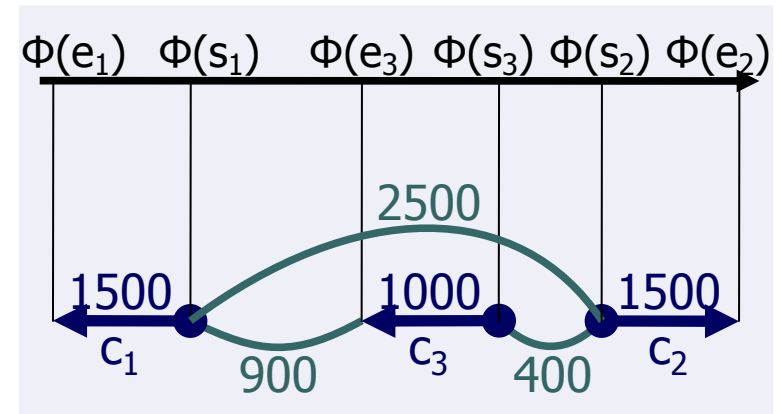
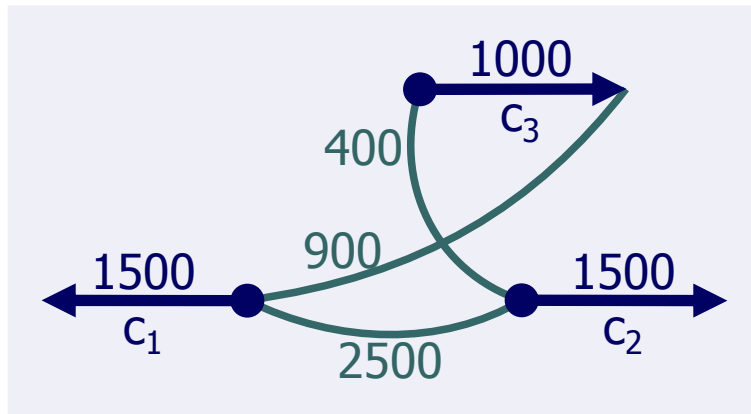
- Transitive **Kantenreduktion**

- Lange Mate-Kanten auf Pfade reduzieren



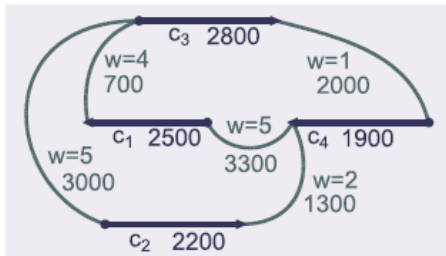
# Happy Mate-Pairs

- Gesucht: **Koordinatenzuordnung** für Knoten
  - Beinhaltet Reihenfolge und Orientierung der Contigs im Graph
  - Ungefähre Erhaltung der Längen und Abstände

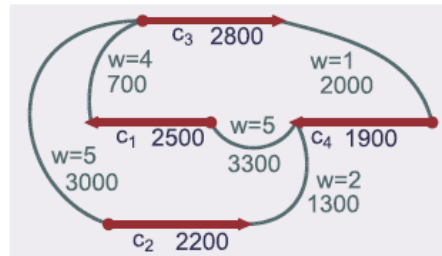


- **Happy Mate-Pairs**
  - Richtige Orientierung, Länge, Abstand bezogen auf Zuordnung  $\Phi$
- Wir suchen Anordnung  $\Phi$  so, dass **möglichst viele „wichtige“ Mate-Pairs happy sind**

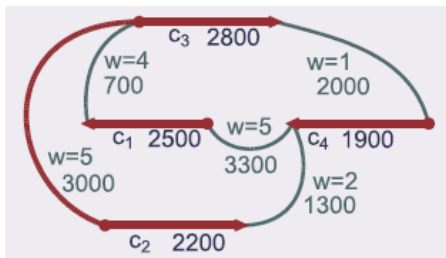
# Folgende Anordnung



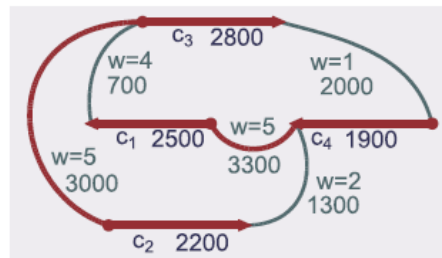
Contig-Mate-Graph nach Edge-Bundling und transitiver Kantenreduktion.



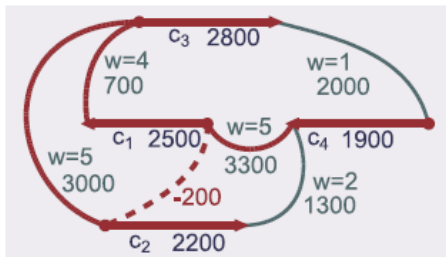
Alle Contig-Kanten werden markiert.



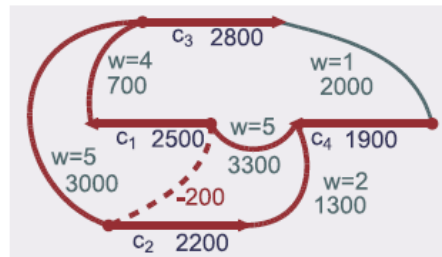
Die Kante mit dem höchsten Gewicht ( $w = 5$ ) verbindet  $c_2$  und  $c_3$ .



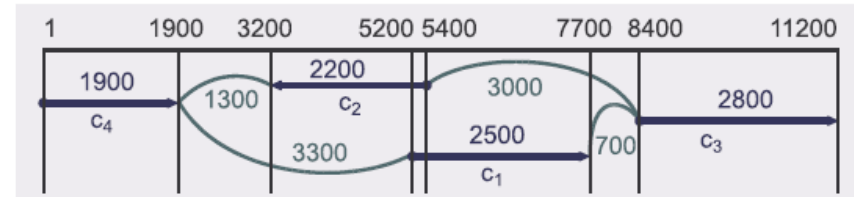
Die nächste Kante mit  $w = 5$  verbindet  $c_1$  und  $c_4$ . Es gibt jetzt zwei Pfade.



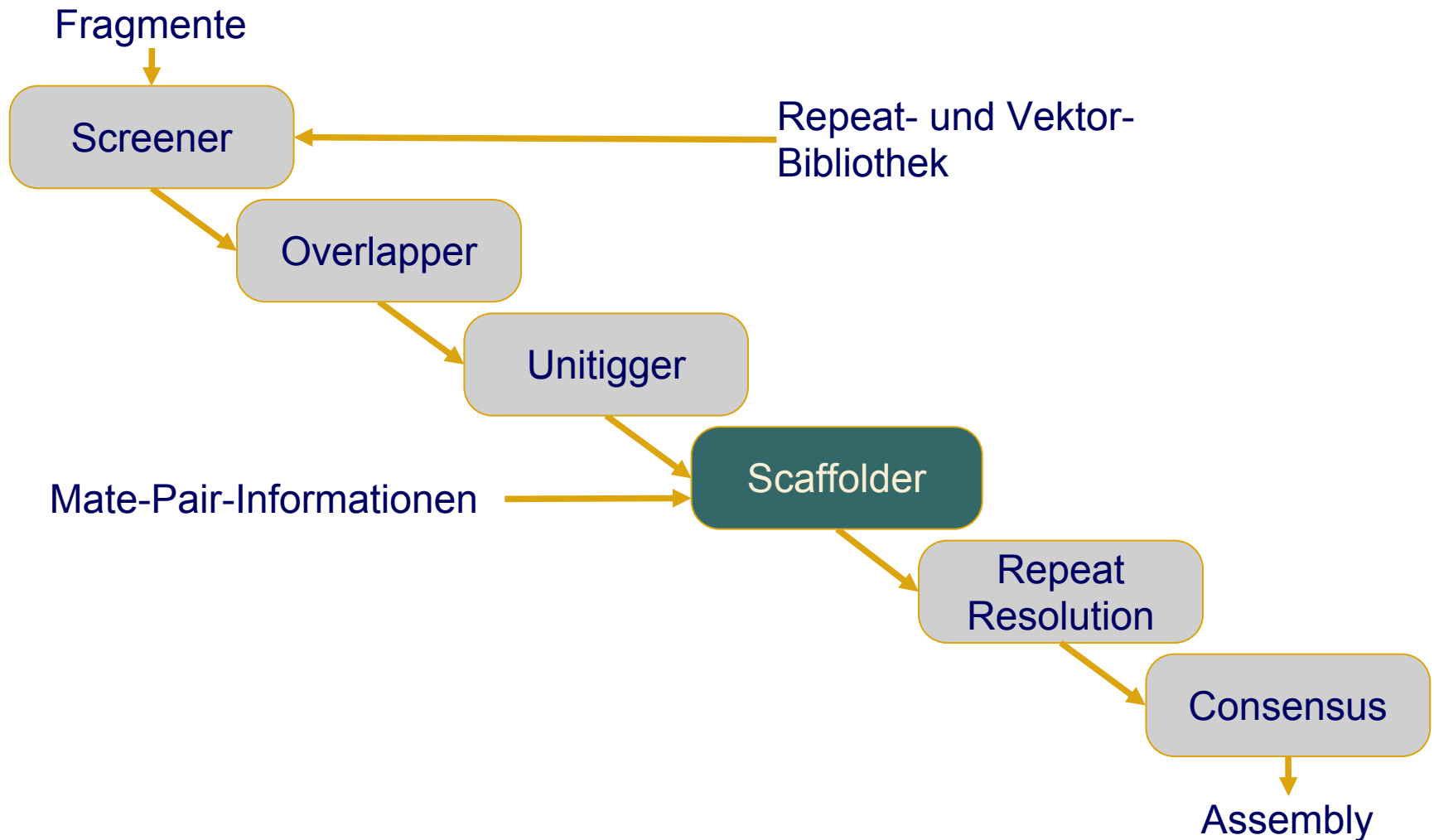
Die Kante mit  $w = 4$  ordnet  $c_1$  zwischen  $c_3$  und  $c_2$  an. Eine neue Kante der Länge -200 wird gefolgt.



Es geht mit der Kante des Gewichtes  $w = 2$  weiter. Sie bestätigt die bisherige Anordnung.



# Überblick



# Menschliches Genom

---

- 27 Millionen Fragmente, Ø-Länge: 550b, 70% mit Mate-Pair Informationen

	CPU-Stunden		Max. Speicher
Screeener	4800	2- 3 Tage auf 10- 20 Computern	2 GB
Overlapper	12000	10 Tage auf 10- 20 Computern	4 GB
Unitigger	120	4- 5 Tage auf 1 Computer	32 GB
Scaffolder	120	4- 5 Tage auf 1 Computer	32 GB
Repeat Res	50	2 Tage auf 1 Computer	32 GB
Consensus	160	1 Tag auf 10- 20 Computern	2 GB
Total	18000		

- Ergebnis: **Assembly (2003) aus 6500 Scaffolds**
  - Umfasst 2,8 Gb Sequenz
  - **150.000 Gaps** (insgesamt 148 Mb)

# Inhalt der Vorlesung

---

- Gene Finding
- CpG Inseln und Markov Modelle
- Hidden Markov Modelle

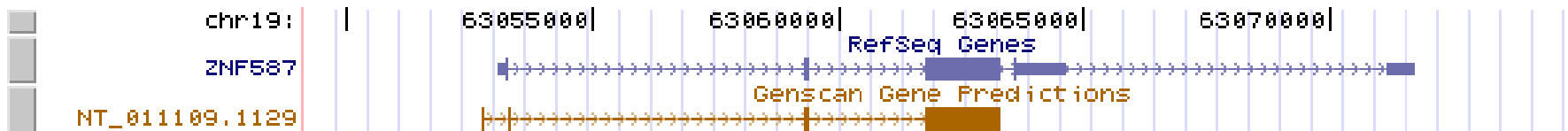
# Gene Finding

---

- Wichtigster Bestandteil eines Genoms sind seine Gene
  - Unsere Definition: Teil eines Chromosoms, der in ein **Protein übersetzt** wird
    - Durch Transkription und Translation
- Wie kann man Gene finden?
  - **Experimentell**: mRNA extrahieren, cDNA erzeugen, im Genom suchen
    - Findet Gene nur teilweise (UniGene kann helfen)
    - Findet nur Gene, die im Experiment „stark“ exprimiert wurden
      - Schwierig: Seltene Gewebe (embryonale, tw. ausdifferenzierte Zellen etc.)
      - Gene, die nur sehr wenig exprimiert werden (low copy genes)
  - **Homologie**: Ähnliche Sequenzen in evolutionär entfernten Spezies
    - Generiert nur eine Hypothese, keinen Beweis (z.B.: Pseudo Genes)
    - Findet auch nicht-kodierende, aber konservierte Bereiche
    - Findet vielleicht gerade die spezies-spezifischsten Gene nicht

# Gene Prediction

- Kann man **Gene vorhersagen**?
  - Ist an der Sequenz eines Gens irgendwas anders als an „Nicht-Gen“ Sequenzen?
  - Wenn ja: Kann man die Unterschiede aus bekannten Gen-/Nichtgen-Sequenzen lernen?
  - Kann man das Gelernte zur Vorhersage eines Gens auf einer neuen Sequenz benutzen?
- Gene Prediction
  - Nach wie vor sehr aktives Forschungsgebiet
  - Aktuelle Verfahren benutzen alle verfügbaren Informationen
    - GRAIL, GeneWise, Gene-ID, GeneScan, ...
  - Vorhergesagte Gene werden typischerweise als „putative“ in die aktuellen **Genomannotationen** übernommen



# Prokaryoten versus Eukaryoten

**(B) PROCARYOTES**

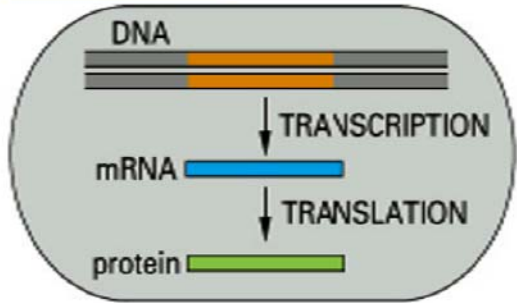


Figure 6-21 part 2 of 2. Molecular Biology of the Cell, 4th Edition.

**(A) EUKARYOTES**

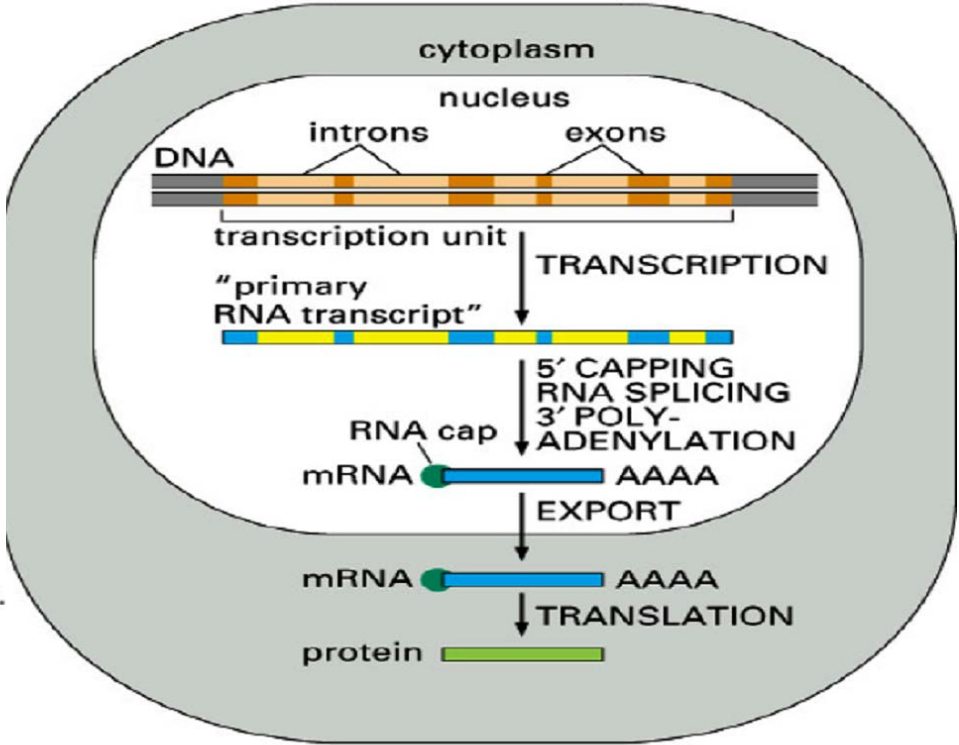
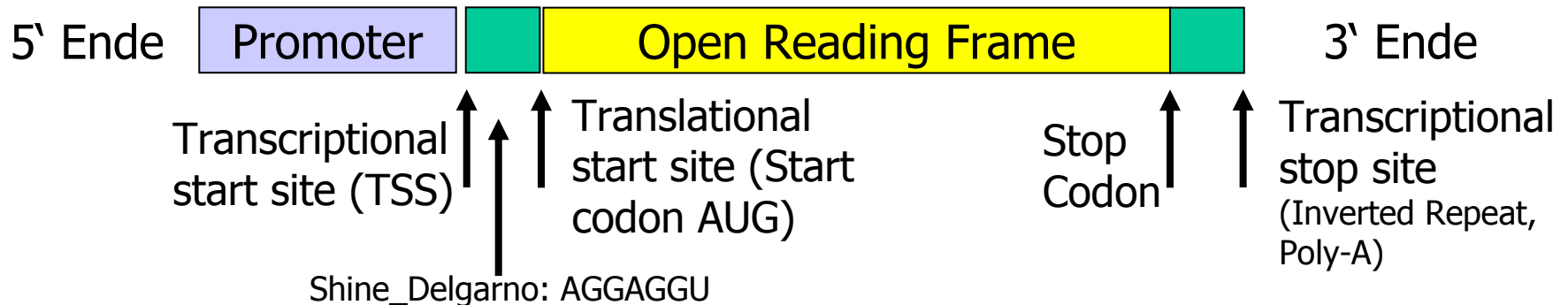


Figure 6-21 part 1 of 2. Molecular Biology of the Cell, 4th Edition.

Quelle: William Stafford Noble

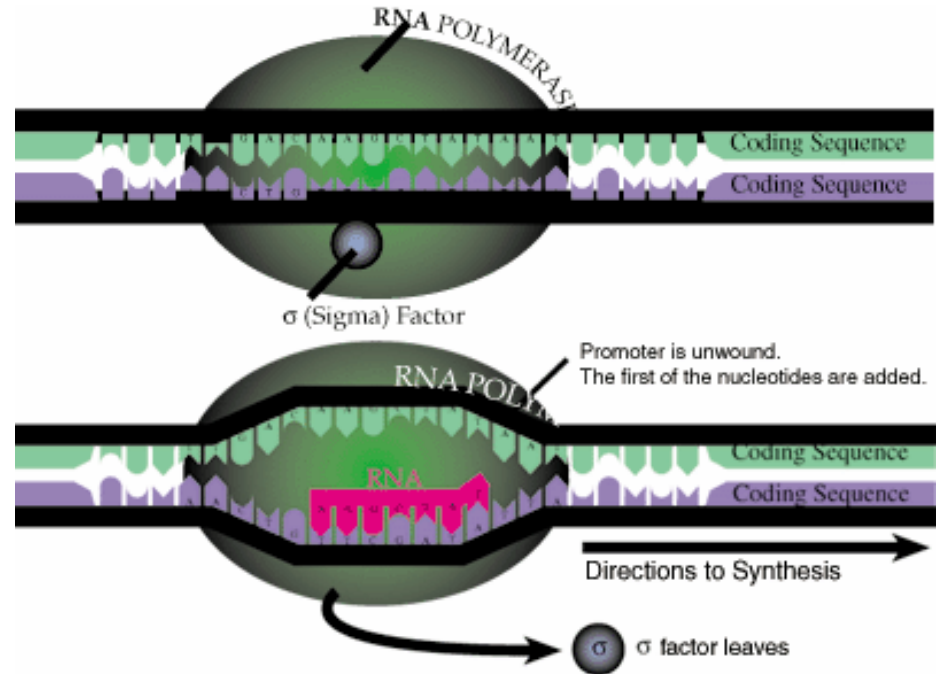
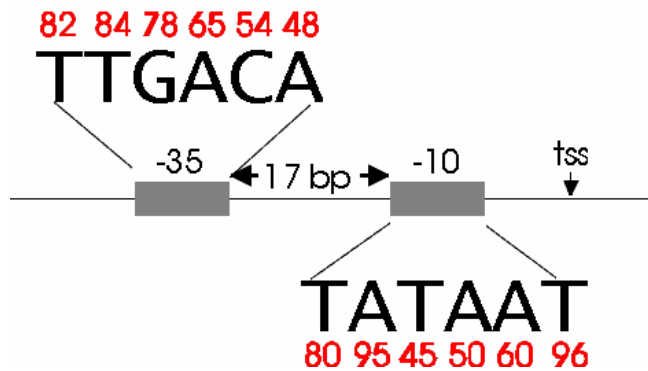
# Gene in Prokaryoten

- Haben eine vergleichsweise einfache Struktur
  - Relativ feste **Start- und Stopcodons**
  - **Open Reading Frame (ORF)**: Sequenz zwischen Start- und Stopcodon von mindestens 100 Basen Länge; Länge durch 3 teilbar
  - Signale für **Anfang und Ende der Transkription**
  - **Promoterregion**: Konservierte Motive im Abstand von -35 bzw. -10 Basen von der Transcriptional Start Site (TSS)



# Promoter Region und RNA Polymerase

Typical Bacterial Promoter



Quelle: Blackwell Pub., 11th hour

- RNA Polymerase: Komplex aus verschiedenen Proteinen
- Sigma-Faktoren erkennen unterschiedliche DNA-Motive
  - Produktion der Sigma-Faktoren hängt von Umwelt ab und regelt z.T. die **Reaktion der Zelle**
- Polymerase bindet erst, wenn Sigma-Faktor gebunden

# Sigma-Faktoren

Faktor	Erkennungssequenz -35	Erkennungssequenz -10	Bedingungen
$\sigma^{70}$	<b>TTGACA</b>	<b>TTGACA</b>	Normal (~70% aller Gene)
$\sigma^{32}$	<b>CTTGAA</b>	<b>CTTGAAA</b>	Hitzestress
$\sigma^{54}$	<b>CTGGCAC</b>	<b>CTGGCAC</b>	Stickstoffmangel
$\sigma^{28}$	<b>TAAA</b>	<b>CTAAA</b>	...
...	...		...

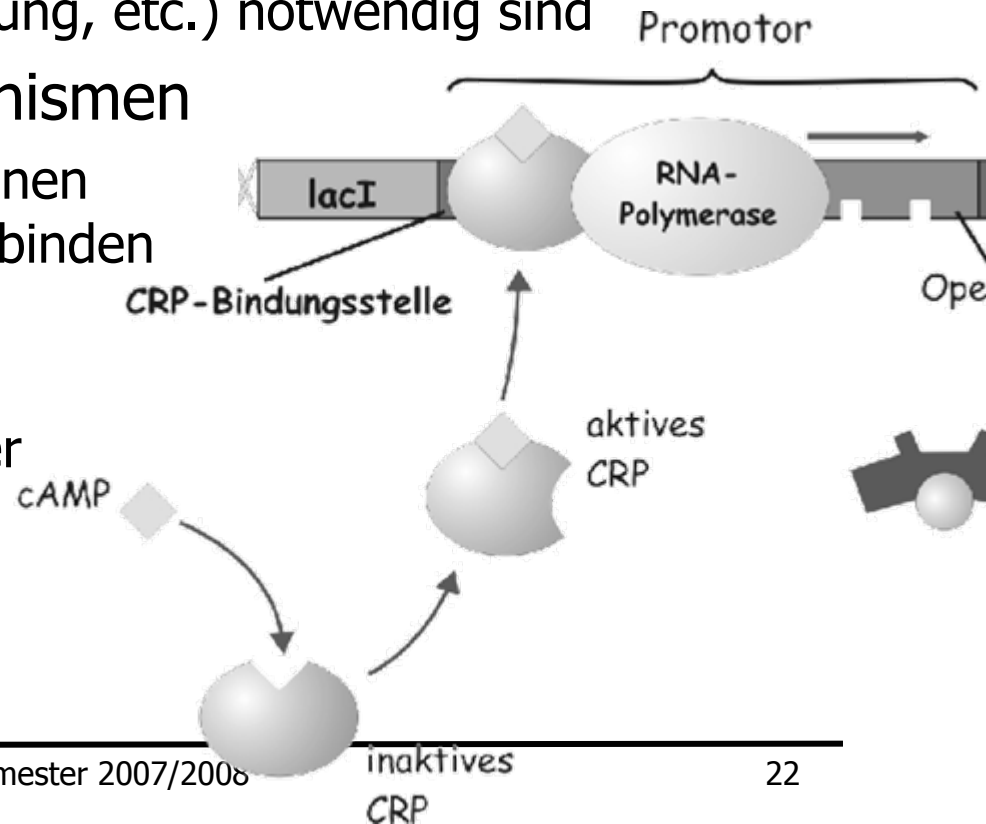
- Verschiedene Sigma-Faktoren binden an verschiedene **Sequenzmotive**
  - E.Coli hat 7 Faktoren; andere haben mehr/weniger
- Motive müssen nicht perfekt erhalten sein
  - Dargestellt sind **Consensus-Sequenzen**
  - Je größer die Abweichung, desto geringer die Expression des regulierten Gens

# Regeln und Abweichungen

- Nicht alle Gene haben eigene Promoterregionen
  - **Operons**: Gruppen von Genen, deren Expression durch einen gemeinsamen Promoter reguliert wird (nur in Prokaryoten)
  - Oftmals Gruppen von Genen, die alle zur Bewältigung einer Aufgabe (Hitzestress, Zellteilung, etc.) notwendig sind

- Weitere Regulationsmechanismen

- **Unterdrückung**: Proteine können zwischen Promotor und TSS binden und Bindung der RNA Polymerase unterdrücken
- **Aktivierung**: Bindung weiterer Proteine in der Nähe des Promoters kann Effizienz der Expression erhöhen



# Open Reading Frames (ORFs)

---

- Prokaryotische Gene haben keine Introns
- Nahezu alle DNA ist kodierend
  - E-.Coli: ca. 80%
- **Open Reading Frame**
  - Bereich auf dem Chromosom, der **kodierend sein könnte**
  - Sollte länger als 60 Codons sein (trifft für ~98% zu)
  - Start-Codon AUG
    - Auch hier sind andere Codons möglich
    - AUG ist auch ein „normaler“ Codon (Methionin) – kein eindeutiges Signal
  - Stop-Codons UAA, UAG, UGA
- ORFs kann man leicht und schnell finden
- ORFs können sich überlappen
  - An beiden Enden?

# Gene Prediction in Prokaryoten

---

- Verfügbare **Evidenzen sammeln**
  - ORFs finden
  - Konservierte Promotor-Sequenzen eines Sigma-Faktors prüfen
  - In einem ORF ist die dritte Base jedes Codons häufiger gleich als statisch erwartet
    - Grund: Spezies favorisieren spezifische Codons für Aminosäuren, bei denen es mehrere Möglichkeiten gibt
  - Weitere Signale: Transcriptional Stop Site, Shine-Delgado-Sequenz, ...
- Wenn man die (fast) alle gefunden hat, hat man mit hoher Wahrscheinlichkeit ein Gen
  - Wahrscheinlichkeit eines **Falsch-Positiven Hits** für ein beliebiges ORF der Länge 60 Codons
    - 60-mal kein Stop-Codon sehen:  $(61/64)^{60} \sim 4\%$

# Eukaryoten – Alles viel schwieriger

## (A) EUCARYOTES

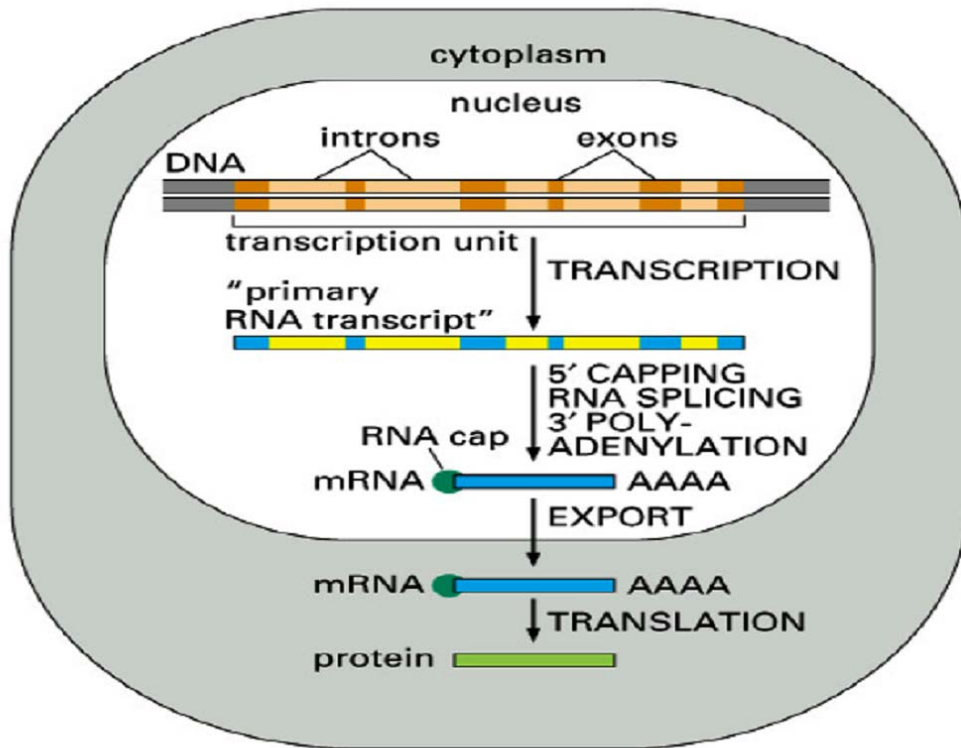
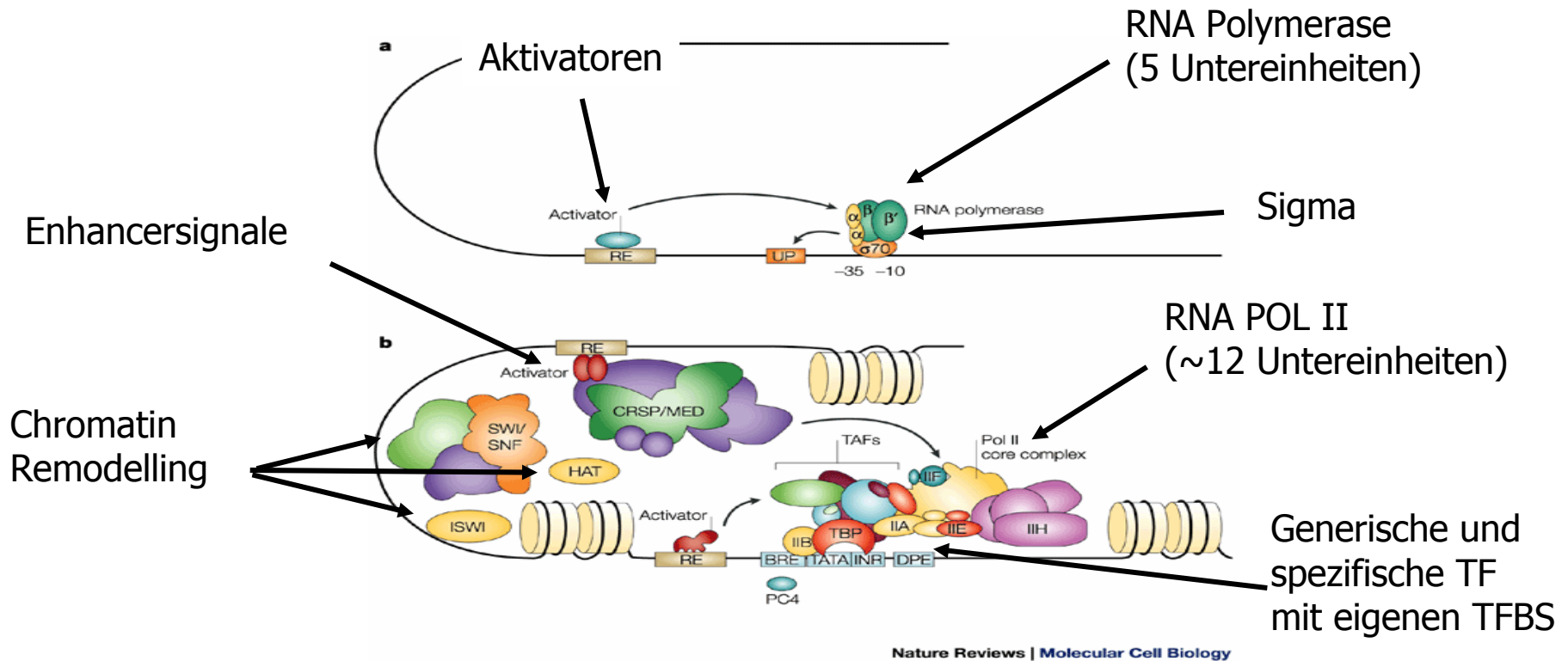


Figure 6–21 part 1 of 2. Molecular Biology of the Cell, 4th Edition.

Quelle: William Stafford Noble

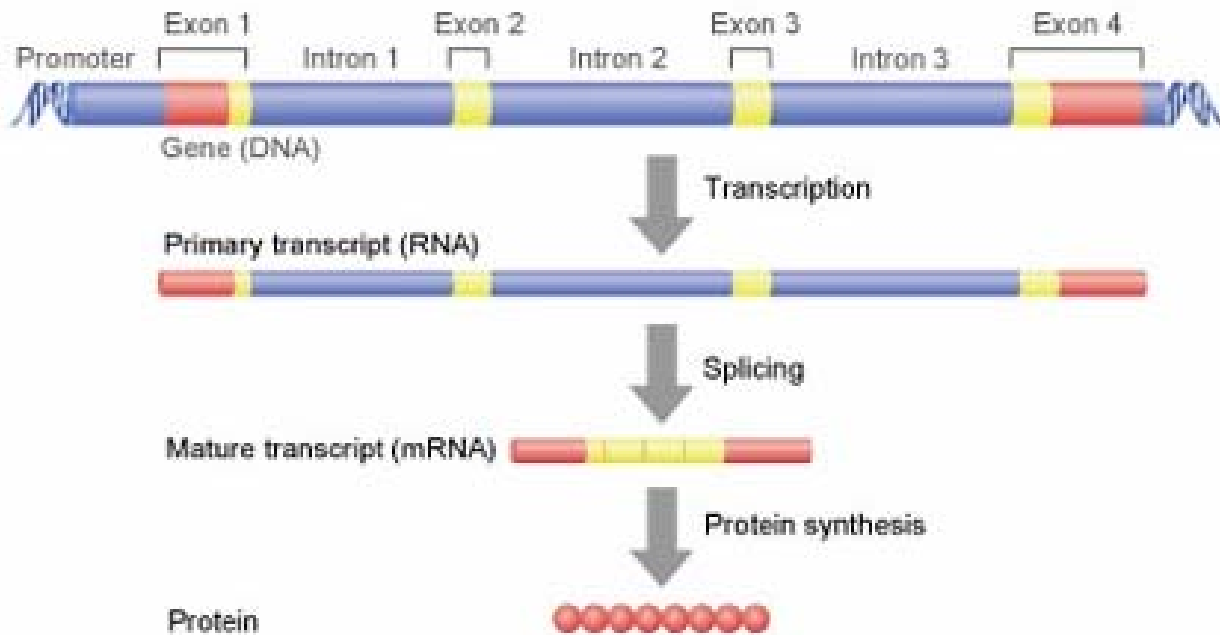
- **Introns** variabler Zahl und Länge
  - können bis zu MB lang sein
- Differentielles Splicing
- 3 RNA Polymerasen
- **Promoterbereiche können mehrere MB** vom Gen entfernt sein
- Polymerase bindet nicht direkt, sondern nur bei Vorhandensein mehrerer **Transcription Factors (TF)**
  - Mensch: Ca. verschiedene 2000 TF
  - Expression benötigt im Schnitt >5 gebundene TFs
- Sehr großer Anteil nicht kodierender DNA
- ...

# Polymerase Initiation Complex



- Warum so komplex? **Unterschiedliche Expressionsmuster**
  - Viele Gewebetypen mit spezifischen Aufgaben
  - Entwicklungsprozess jedes Individuums mit verschiedenen Stadien

# Genstruktur bei Eukaryoten



© Wellcome Trust

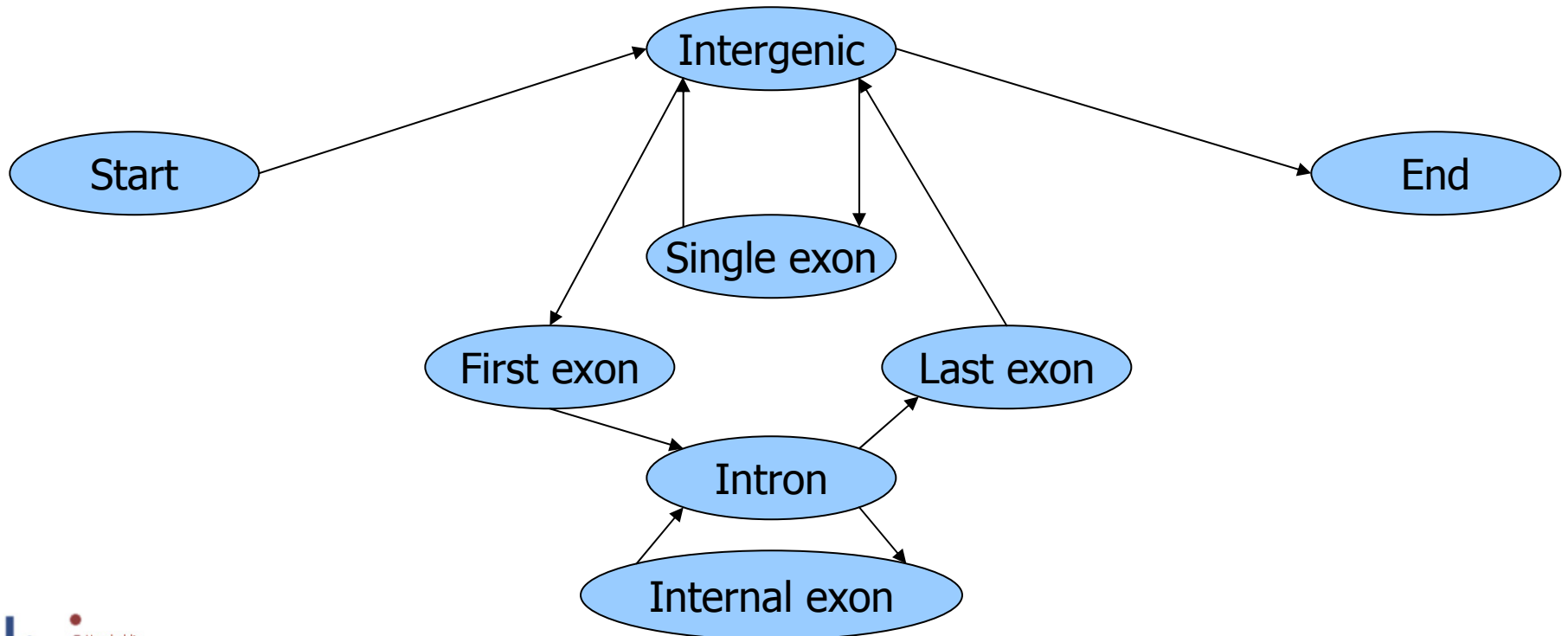
# Module

---

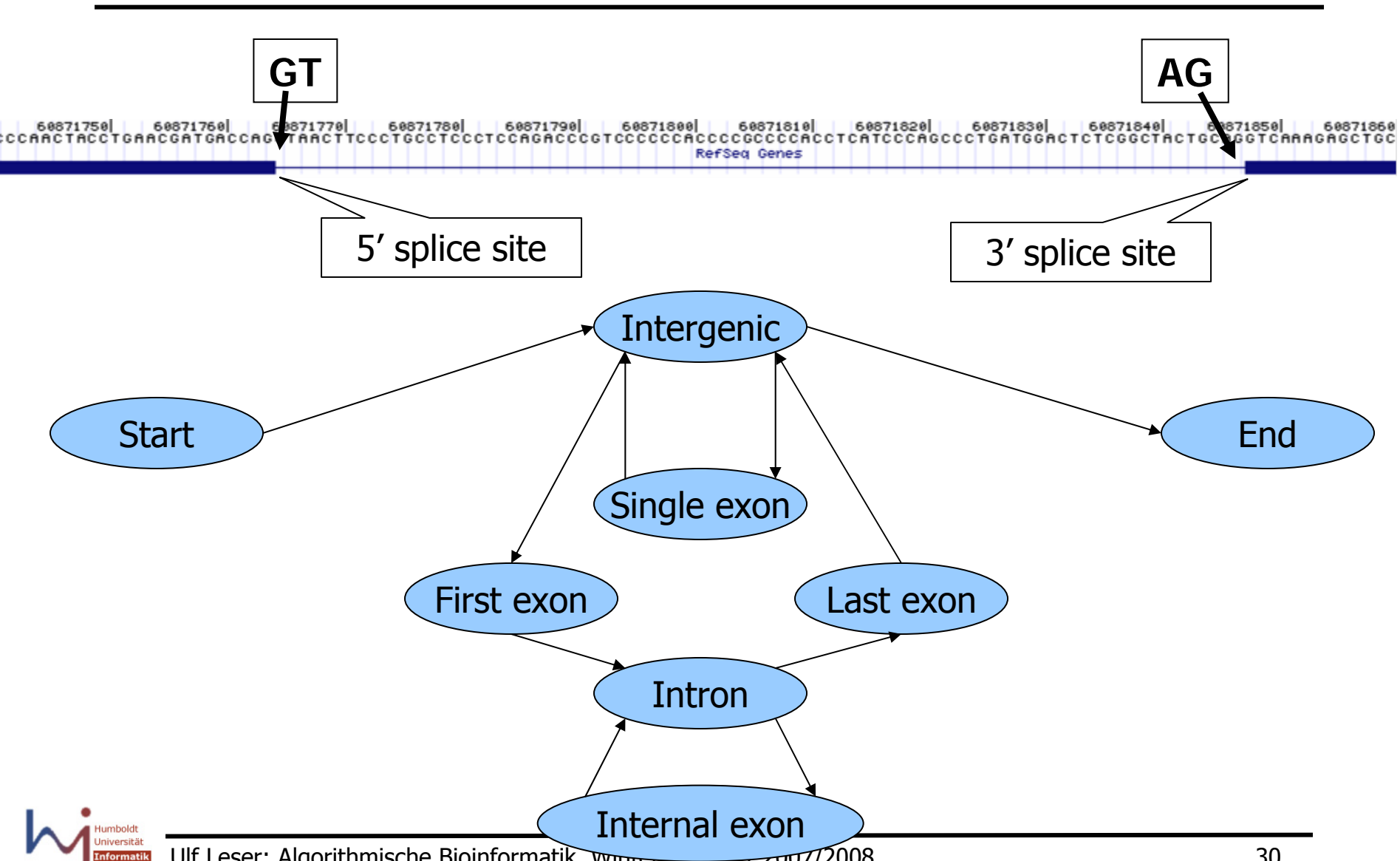
- Exons, Introns, ... nennen wir **Module eines Gens**
- Module sind
  - **Signale**: Feste Länge (kurz) und „relativ“ feste Sequenz
    - Splicestellen, Start- und Stop-Codons, TFBS
  - **Blöcke**: Keine feste Länge, variable Sequenz
    - Exons, Introns, UTRs, Promoterregionen
- Wie kann man ein **Gen samt seiner Modulstruktur** finden?
  - Module haben meistens **keine feste Grenzen**
  - Aber: **Verschiedene Arten von Modulen** haben best. Eigenschaften
    - Länge von Coding Regions durch 3 teilbar
    - Codons haben unterschiedliche Frequenzen
    - Exons sind meistens kürzer als Intros, Intros können seeehr lang sein
    - Start- und Stop-Codons markieren Gengrenzen
    - Splicestellen sind 99% konserviert (GT, AG)
    - Exons und Introns haben unterschiedliche Basenzusammensetzung
    - ...

# Einfaches Zustandsmodell

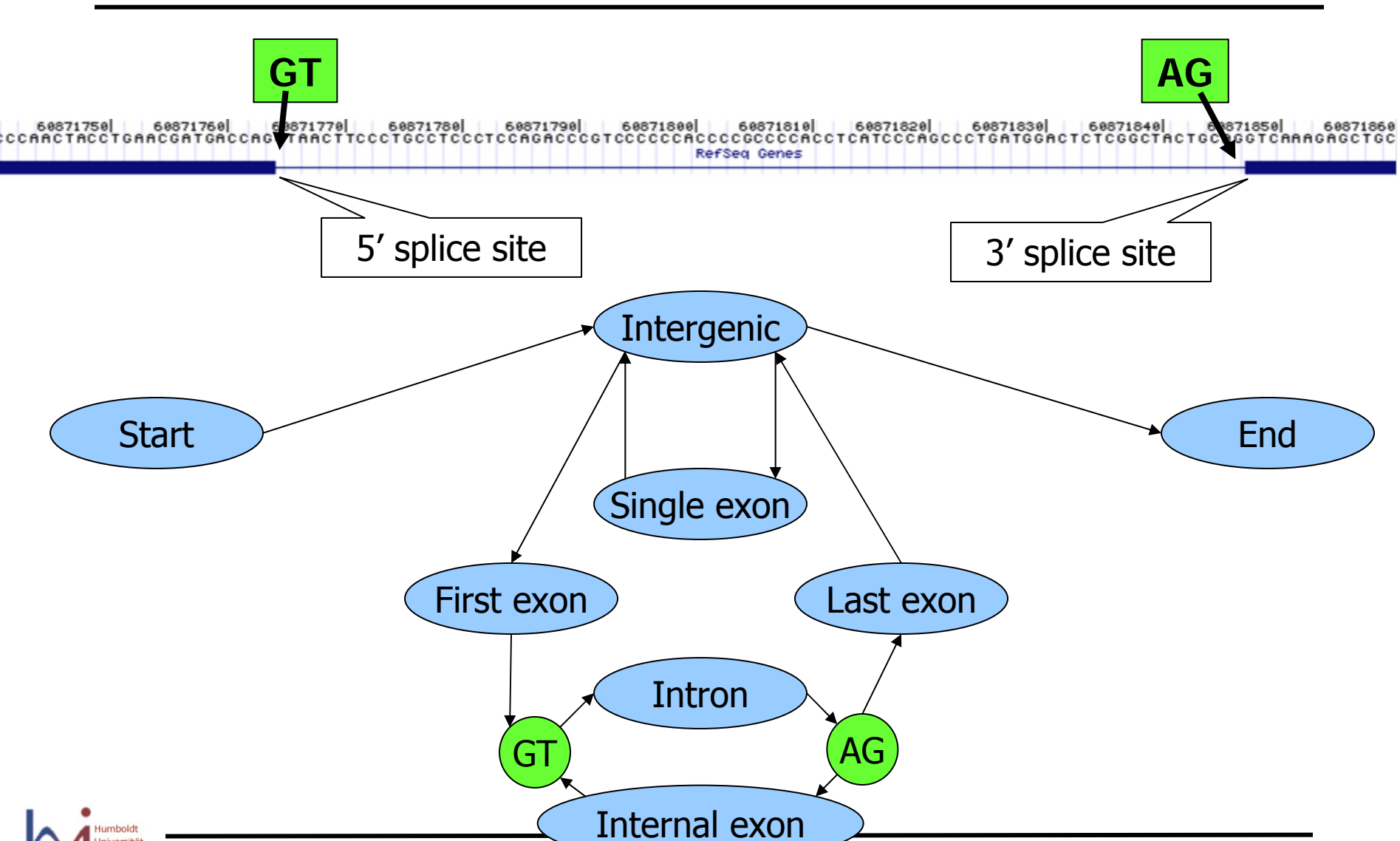
- Stellen wir uns vor, jede Base hat einen Zustand
  - Die **Modulart**, zu der sie gehört
- Folgende **Übergänge** sind erlaubt (einfaches, erstes Modell)
  - Übergänge von Zustand Z zu Zustand Z nicht enthalten



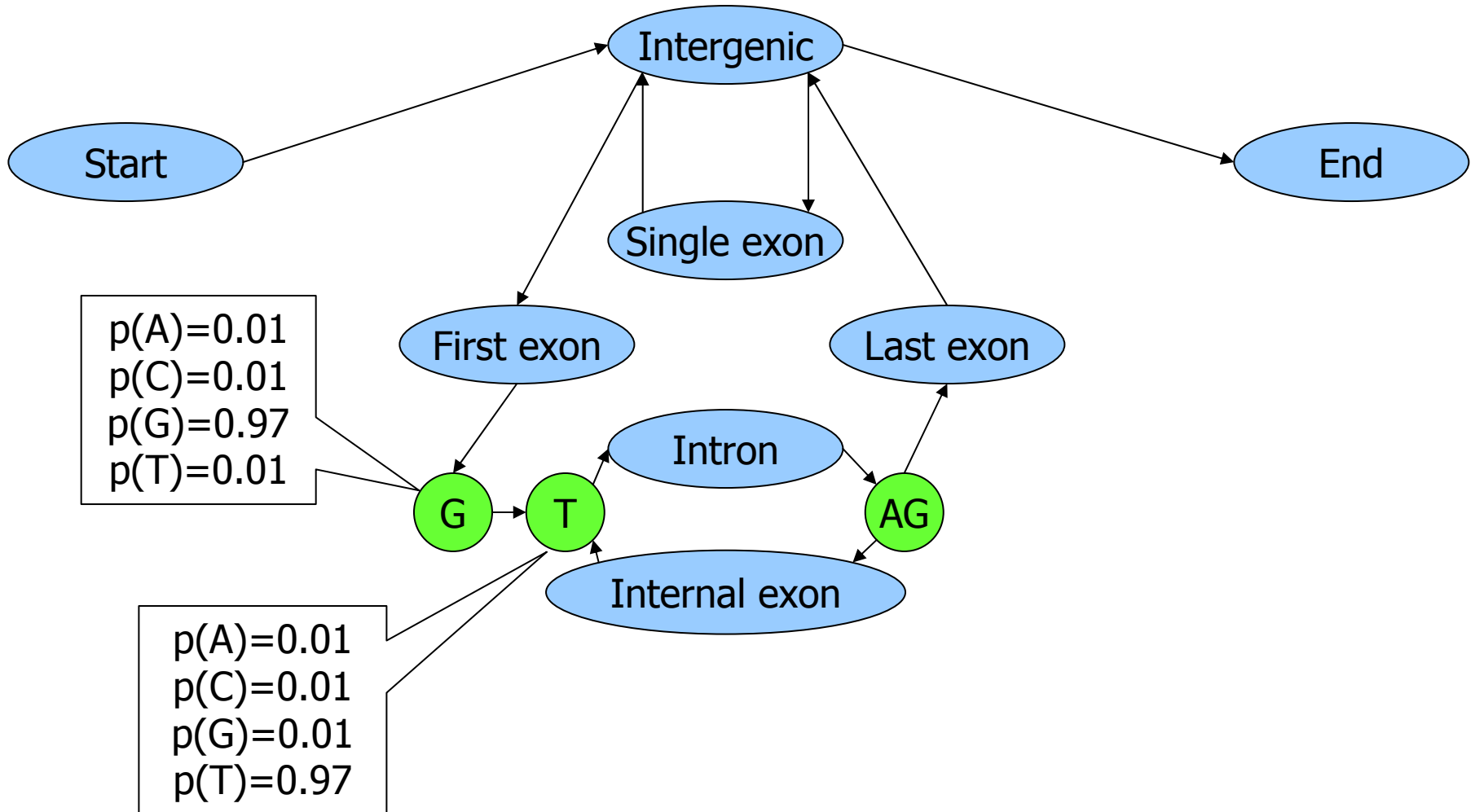
# Exon-Intron-Grenzen



# Signale für Exons/Introns

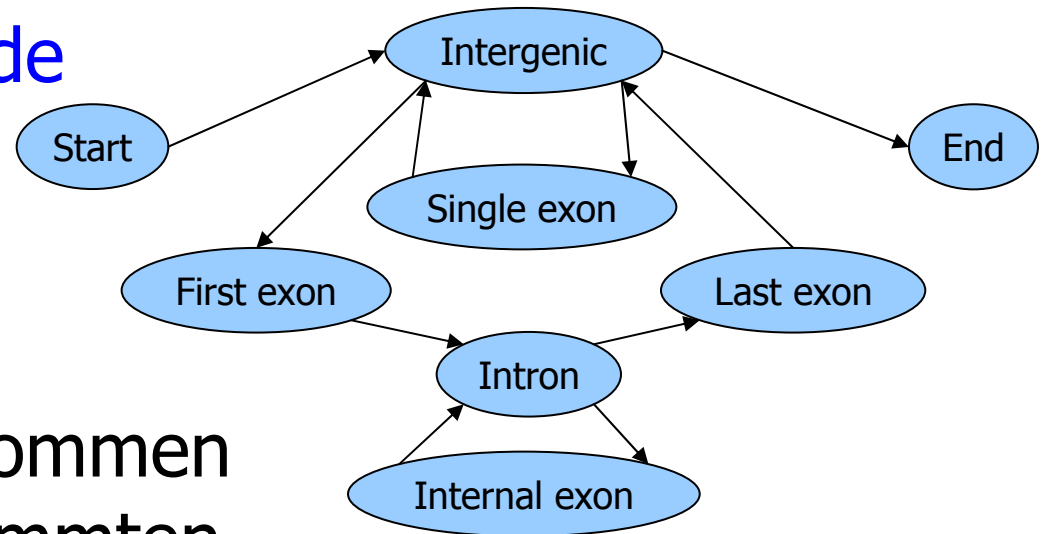


# Emissionswahrscheinlichkeiten

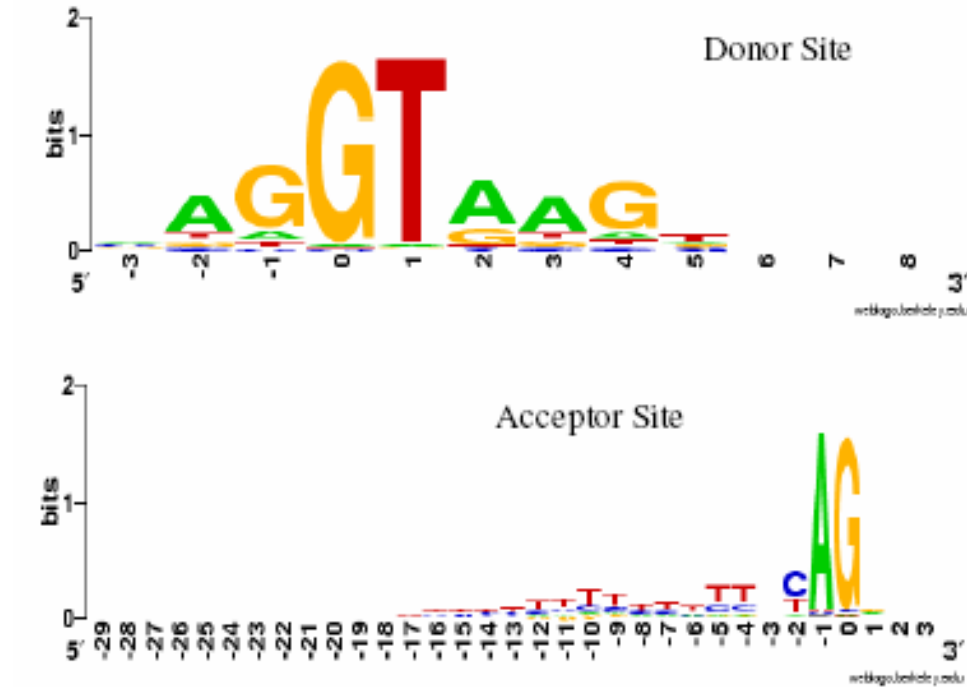


# Module und Zustände

- Module sind **Zustände** eines Modells
- **Zustände** emittieren **Basen**
- In einem Zustand kommen Basen in einer bestimmten Frequenz, Reihenfolge und Anzahl vor
- Pfeile sind Zustandsübergänge
- Übergänge haben eine bestimmte Wsk
- Das ist ein **Hidden Markov Model (HMM)**



# Echte Splicestellen



- Auch Basen links/rechts vom Signal sind "etwas" konserviert
- Kann man als weitere Zustände in das Modell aufnehmen

# Probleme (informell)

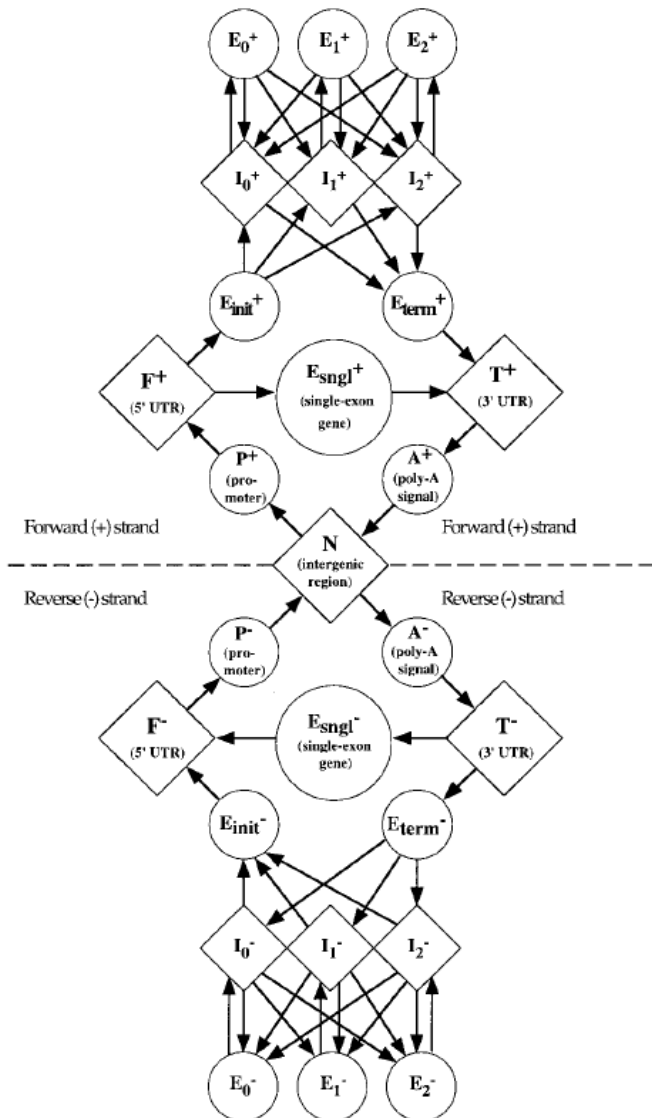
---

- Einer gegebenen Sequenz kann man erst mal nicht ansehen, aus welchen Zuständen in welcher Reihenfolge sie am wahrscheinlichsten generiert wird
  - Alle emittieren A,C,G,T, nur mit (geringfügig) unterschiedlicher Wsk
- Problem 1: Gegeben eine Sequenz und ein Modell: **Finde die Modulgrenzen** (also die Zustandsübergänge)

ACTG	ACT	ACTAAATTGCCGCTCGT	GACGACGATCTACTAG	GGCGCGACCTATGCG
	SSS	EEEEEEEEEEEEEEEEEEEESS	IIIIIIIIIIIIIIIIIIII	SSEEEEEEEEEEEEEEEE...

- Problem 2: Gegeben viele Gene: **Finde die Übergangs- und Emissionswahrscheinlichkeiten** des Modells
  - Und womöglich das Modell selber

# Beispiel: GeneScan



- Burge, C. and Karlin, S. (1997). "Prediction of complete gene structures in human genomic DNA." *J Mol Biol* 268(1): 78-94.
- Modell mit 27 Zuständen
- Erkennungsgenauigkeit (1997)
  - ~90% für Basen (in Gen oder nicht)
  - ~80% für: In Exon oder nicht
  - ~43% für komplette Genstruktur
- Trainingsdaten: ~400 humane Gene

# Inhalt der Vorlesung

---

- Gene Finding
- CpG Inseln und Markov Modelle
- Hidden Markov Modelle

# CpG Inseln

---

- Mit "CpG" bezeichnet man das **Nukleotidpaar CG**
  - CpG: Hintereinander auf einem Strang, nicht die Paarung C-G
  - Das „p“ symbolisiert die Phosphodiesterbrücke zwischen den Basen
- CpG's sind statistisch überraschend relativ selten im humanen (und anderen eukaryotischen) Genom
  - Das C in CpG kann methyliert werden
  - Dadurch erfährt es eine höhere Mutabilität
- Aber: Ab ca. 1500 Basen vor einem Gen ist die **Dichte an CpG „normal“**
  - Erklärung: Methylierung erhöht die Histon-Bindung der DNA
  - Dadurch wird die Expression wesentlich erschwert
  - Zusätzliches **Regulationsprinzip**
  - Wird eng mit gewebespezifischen Expressionsmustern assoziiert

# CpG Inseln

---

- **CpG-Inseln**
  - Sequenzabschnitte, in denen **mehr CpG als erwartet** (bezogen auf absolute Häufigkeit im Genom) vorkommen
  - Die meisten CpG Inseln liegen vor Genen
  - Die meisten Gene liegen hinter einer CpG Insel
- Wie kann man für eine Sequenz entscheiden, ob sie eine CpG Insel ist?
  - Wir wissen, dass bestimmte Dinukleotide häufiger sind als sonst
    - Nach C kommt häufiger ein G als ein A oder T
  - Richtig fest ist aber nichts
  - Erster Versuch: **Markov-Modelle** erster Ordnung

# Markov-Modell (oder Markov-Kette)

---

- Definition

*Gegeben ein Alphabet  $\Sigma$ . Ein **Markov-Modell** erster Ordnung ist ein sequentieller stochastischer Prozess (Zustandsfolge) über  $|\Sigma|$  Zuständen  $s_1, \dots, s_n$  mit*

- *Jeder **Zustand  $s_i$**  emittiert genau ein Zeichen aus  $\Sigma$*
- *Keine zwei Zustände emittieren das selbe Zeichen*
- *Für eine Folge  $z_1, z_2, \dots$  von Zuständen gilt:*

$$p(z_t=s_t | z_{t-1}=s_{t-1}, z_{t-2}=s_{t-2}, \dots, z_1=s_1) = p(z_t=s_t | z_{t-1}=s_{t-1})$$

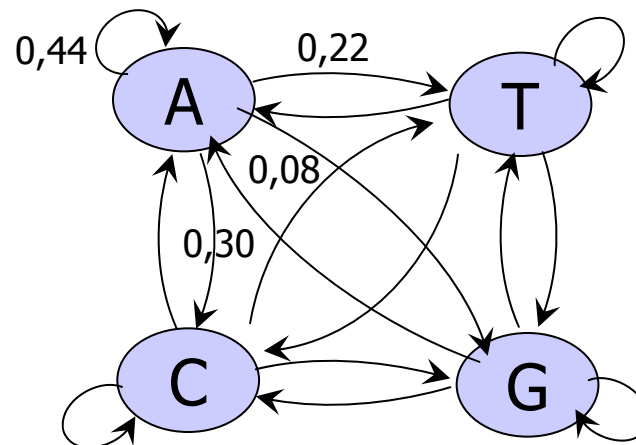
- *Die  $a_{0,i} = p(z_1=s_i)$  heißen **Startwahrscheinlichkeiten***
- *Die  $a_{s_i,s_j} = p(z_t=s_j | z_{t-1}=s_i)$  heißen **Übergangswahrscheinlichkeiten***

- Bemerkung

- Die Wahrscheinlichkeit des Auftretens eines Zustands hängt also **nur vom Vorgängerzustand** ab

# Visualisierung

- Jeder Zustand einer Markov-Kette emittiert genau ein eindeutiges Zeichen des Alphabets
  - Daher können wir **Zustände und Zeichen verschmelzen**
    - Bei HMM geht das nicht, daher trennen wir in der Definition
- Damit können wir ein Markov-Modell als **Zustandsgraph** visualisieren
  - Knoten sind die Zeichen des Alphabets (Zustände)
  - Kanten sind mit Übergangswahrscheinlichkeiten beschriftet



Hier sind alle Zustände mit allen verbunden; das muss nicht so sein ( $a_{ij}=0$ )

# Wahrscheinlichkeit einer Zustandsfolge

---

- Gegeben ein Markov-Modell  $M$  mit Übergangswahrscheinlichkeiten  $a$  und eine Folge  $S$  von Zeichen aus  $\Sigma$  (eine Sequenz)
- Wir lassen den stochastischen Prozess laufen;  $M$  wird eine Sequenz erzeugen
- Wie groß ist die Wahrscheinlichkeit, dass  $M$  genau  $S$  erzeugt? (Oder:  $p(\text{Das } S \text{ von } M \text{ erzeugt wird})$ )

$$p(x_{t-1} = S[i-1]) = \prod_{i=2..n} a_{S[i-1], S[i]} = a_{0, S[1]} * \prod_{i=2..n} a_{S[i-1], S[i]} = a_{0,1} * \prod_{i=2..n} a_{i-1,i}$$

- Bisher ist alles **deterministisch**
- Da Zustände eindeutige Zeichen emittieren, kann jede Zeichenfolge nur durch genau eine Folge von Zuständen erzeugt werden

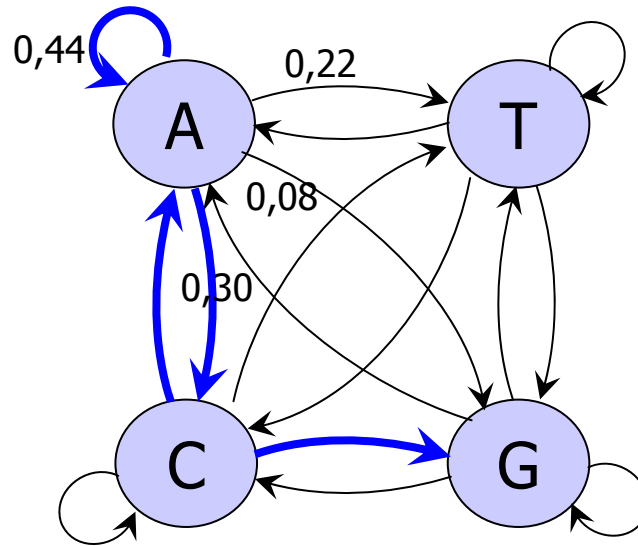
# Vereinfachung

---

- Startzustände machen die Formeln hässlich
- Vereinfachung
  - Einführung eines **expliziten neuen Startzustands**  $s_0$
  - Jede Zustandsfolge beginnt mit  $z_0 = s_0$
  - Seine Wahrscheinlichkeit ist fix 1 und er emittiert kein Zeichen des Alphabets
  - Damit schreiben wir

$$p(S | M) = a_{0,1} * \prod_{i=2..n} a_{i-1,i} = \prod_{i=1..n} a_{i-1,i}$$

# Beispiel



- $$P(\text{CAACG}) = p(z_1=C|z_0) * p(z_2=A|z_1=C) * p(z_3=A|z_2=A) * p(z_4=C|z_3=A) * p(z_5=G|z_4=C)$$
$$= a_{0C} * a_{CA} * a_{AA} * a_{AC} * a_{CG}$$

# Geschichte

---

- Andrej Andrejewitsch Markov (1856-1922)
  - Russischer Mathematiker
  - Entwickelte Markov-Ketten-Modelle für **Anwendungen in der Sprache**
  - Statistische Analyse der Buchstabenfolgen in Novellen
  - Markov, A. A. (1913). "Beispiel statistischer Untersuchungen des Textes ‚Eugen Onegin‘, das den Zusammenhang von Ereignissen in einer Kette veranschaulicht (Original in Russisch)." *Bulletin de l'Academie Imperiale des Sciences de St.-Petersbourg*. 153-162.

# CpG Inseln revisited

---

- Wie unterscheiden sich CpG Inseln von anderen Sequenzen?
- Durch Ihre **Übergangswahrscheinlichkeiten**

M+	A	C	G	T
A	.180	.274	.426	.120
C	.171	.368	.274	.188
G	.161	.339	.375	.125
T	.079	.355	.384	.182

M-	A	C	G	T
A	.300	.205	.285	.210
C	.233	.298	.078	.302
G	.248	.246	.298	.208
T	.177	.239	.292	.292

# CpG Inseln erkennen

---

- Erster Versuch: Wir bilden **zwei Markov-Modelle**
  - Modell M+ für die Übergangshäufigkeiten in CpG Inseln
  - Modell M- für die Übergangshäufigkeiten in normaler Sequenz
  - Berechnung des Log-Odds-Score

$$s = \log\left(\frac{p(S | M+)}{p(S | M-)}\right) = \sum_{i=1}^n \frac{a_{i-1,i}^+}{a_{i-1,i}^-}$$

- $s > 0$ : Die Sequenz ist **wahrscheinlich eine CpG Insel**
  - Je größer  $s$ , desto wahrscheinlicher
- $s < 0$ : Die Sequenz ist wahrscheinlich keine CpG Insel

# Wo kommen die Tabellen her?

---

- Am einfachsten: **Zählen**
- Man benötigt „hinreichend“ viele CpG Inseln und „hinreichend“ viele normale Sequenzen
- Zählen der Häufigkeiten aller Zeichen  $x_i$  und aller Substrings  $x_i x_j$  der Länge 2. Dann:

$$a_{i,j} = \frac{h(x_i x_j)}{h(x_i)}$$

- Und die Startwahrscheinlichkeiten?
  - Am einfachsten  $a_{0,i} = h(x_i)$

# CpG Inseln finden

---

- Die Frage: „Ist Sequenz S eine CpG Insel?“ ist nicht wirklich relevant
- Wichtiger: „**Wo in S sind CpG Inseln?**“
- Problem: Die Markov-Kette kann überall in S beginnen
- Lösung 1: **Sliding Window** (sei  $|S|=n$ )
  - Wir schieben ein Fenster der Größe  $w$  über S
  - Für jede Position bestimmen wir den Score  $s$  mit  $M+$  und  $M-$
  - Laufzeit:  $O(n)$  – Wie?
  - Problem: **Welches  $w$ ?**
    - CpG Inseln haben keine fixen Längen
    - Jede Wahl ist falsch
    - Besser wäre ein **längenunabhängiger Mechanismus**

# Inhalt der Vorlesung

---

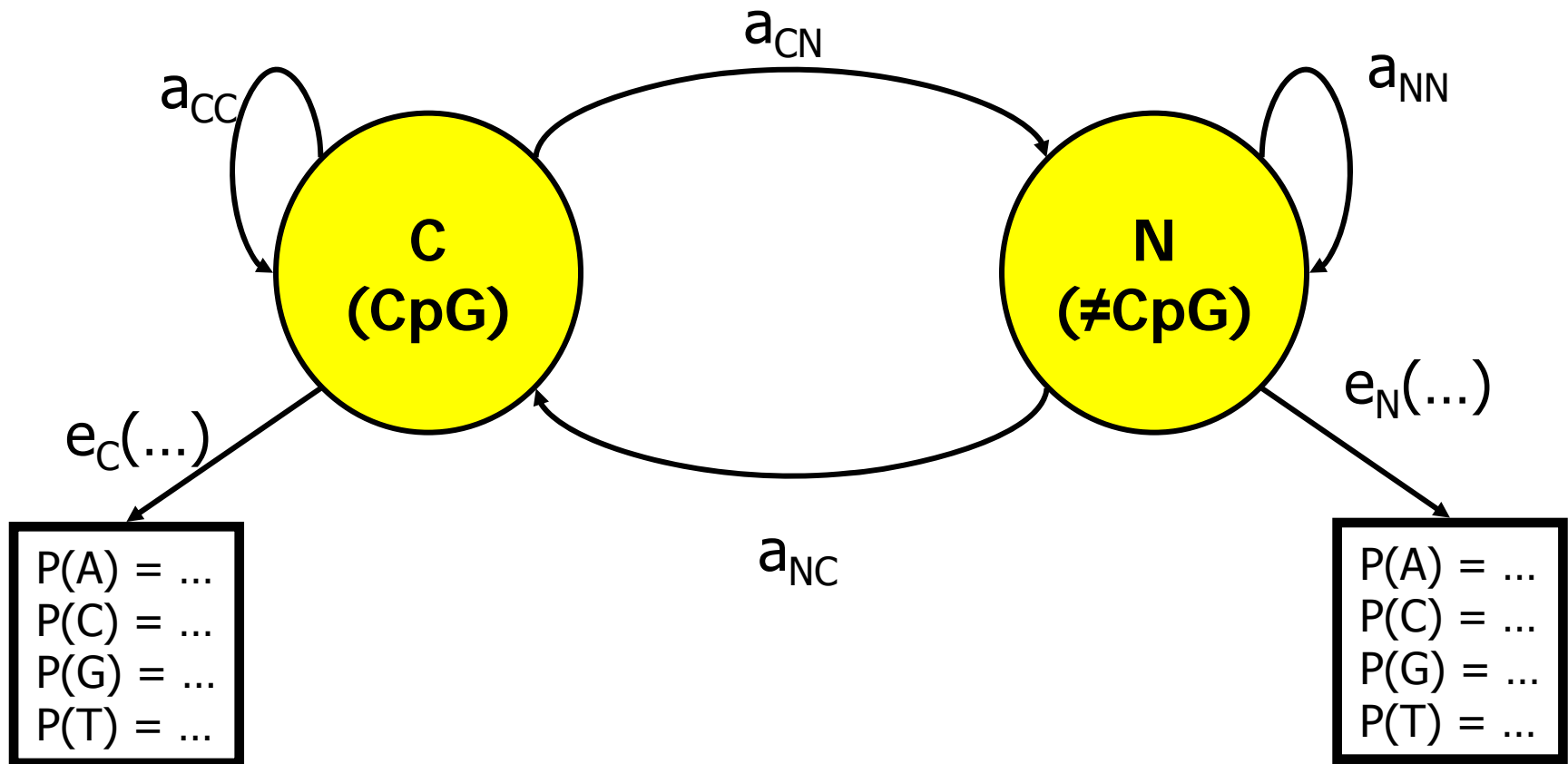
- Gene Finding
- CpG Inseln und Markov Modelle
- Hidden Markov Modelle

# Lösung 2: Hidden Markov-Modelle (HMM)

---

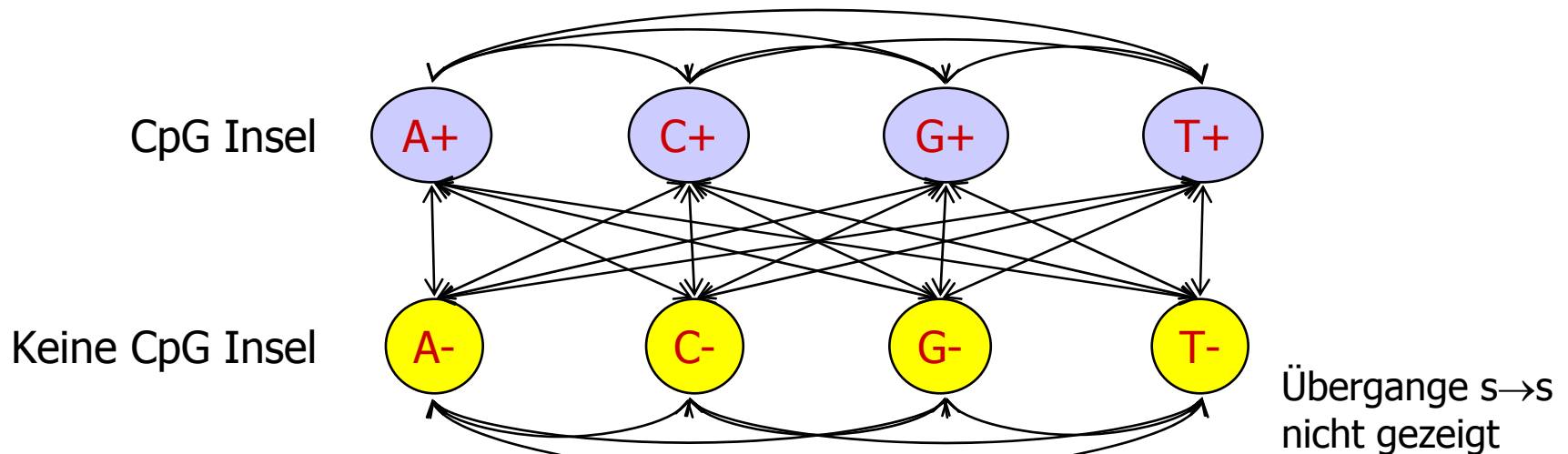
- Baum, L. E. and Petrie, T. (1966). "Statistical Inference for Probabilistic Functions of Finite State Markov Chains." *Annals of Mathematical Statistics* 37: 1554-1563.
- Folgende Idee
  - Wir denken uns einen Automaten mit zwei „Meta-Zuständen“: Einer für CpG, einer für nicht-CpG
  - Jeder Zustand kann alle Zeichen des Alphabets mit einer bestimmten Emissions-Wsk emittieren
    - Das ist neu: Zustände  $\neq$  Zeichen
  - Damit kann man die Zustandsfolge nicht mehr einfach aus der Zeichenfolge ablesen
    - Das ist neu: Zustände sind verborgen (hidden)
  - Die Emissions-Wsk und Übergangs-Wsk in den beiden Metazuständen sind unterschiedlich und modellieren die Unterschiede zwischen normaler DNA und CpG-Inseln

# Hidden Markov Modell 1: CpG



# Hidden Markov Modell 2: CpG

- Wir **verdoppeln die Anzahl der Zustände**:  $A \rightarrow A+, A- \dots$
- Die Zustände  $s+$  und  $s-$  emittieren jeweils **dasselbe Zeichen**
- Die Übergangswahrscheinlichkeiten zwischen allen  $s+$  Zuständen entnehmen wir dem  $M+$  Modell
- Die Übergangswahrscheinlichkeiten zwischen allen  $s-$  Zuständen entnehmen wir dem  $M-$  Modell
- Übergang von jedem  $s+$  Zustand zu jedem  $s-$  Zustand und umgekehrt ist mit einer bestimmten (kleinen) Wahrscheinlichkeit erlaubt



# Formale Definition von HMMs

---

- Definition

*Gegeben  $\Sigma$ . Ein Hidden Markov Modell ist ein sequentieller stochastischer Prozess über  $k$  Zuständen  $s_1, \dots, s_k$  mit*

- *Zustand  $s$  emittiert Zeichen  $x \in \Sigma$  mit Wahrscheinlichkeit  $p(x|s)$*
- *Die Folge der Zustände ist eine Markov-Kette, d.h.:*

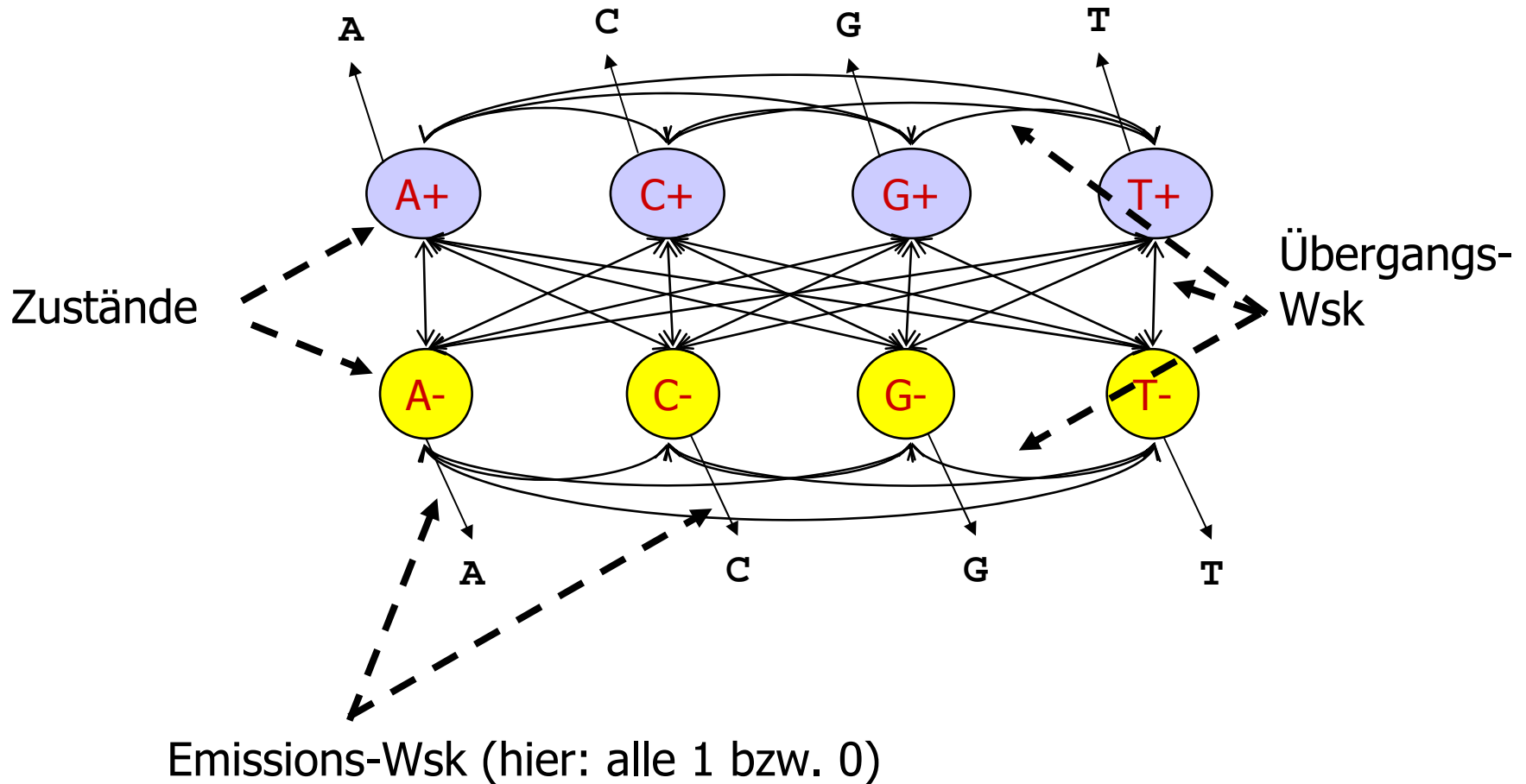
$$p(z_t=s_t|z_{t-1}=s_{t-1}, z_{t-2}=s_{t-2}, \dots, z_0=s_0) = p(z_t=s_t|z_{t-1}=s_{t-1}) = a_{t-1,t}$$

- *Die  $a_{0,1}$  heißen Startwahrscheinlichkeiten*
- *Die  $a_{t-1,t}$  heißen Übergangswahrscheinlichkeiten*
- *Die  $e_s(x) = p(x|s)$  heißen Emissionswahrscheinlichkeiten*

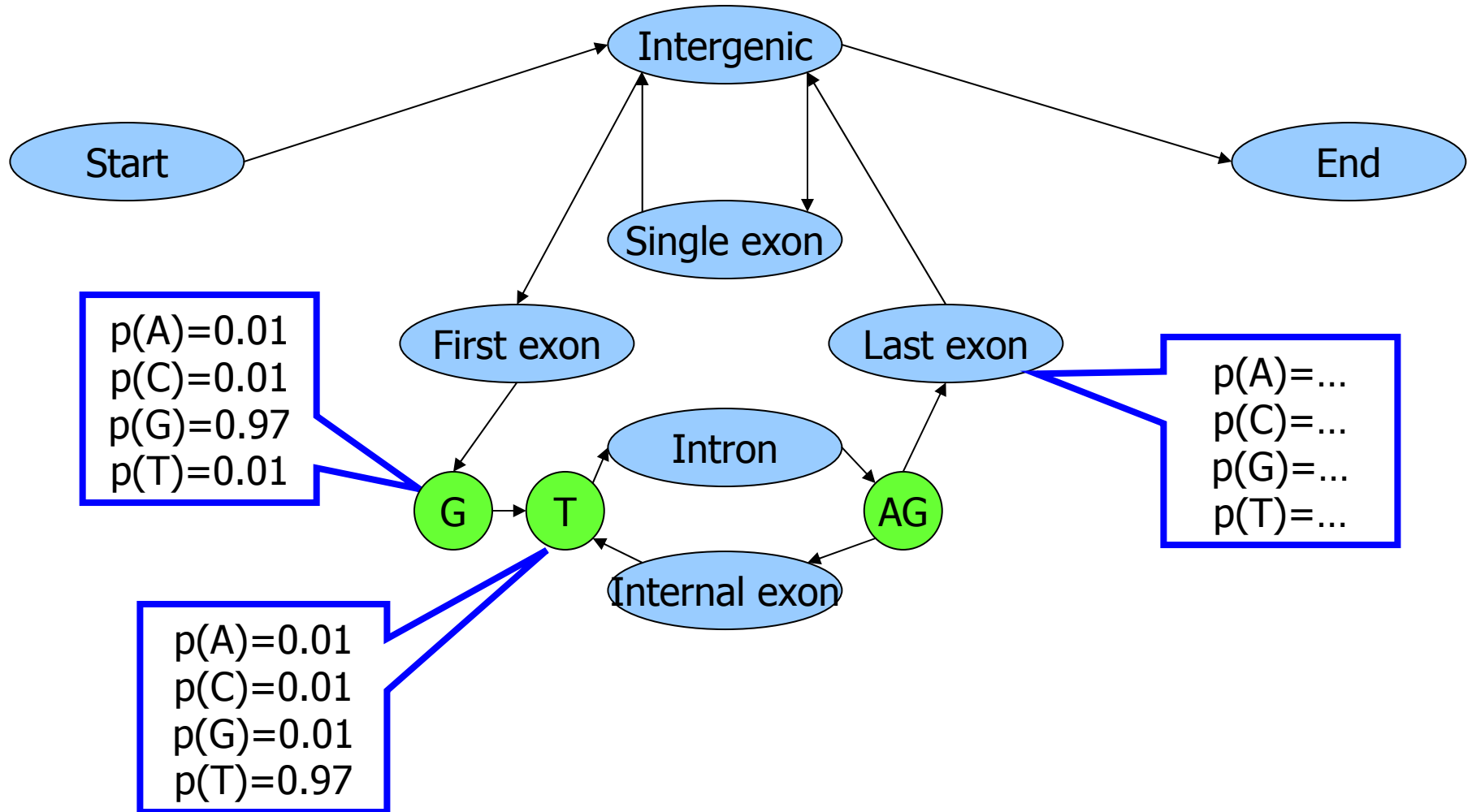
- Bemerkung

- Jetzt ist die Unterscheidung zwischen Zustand und Zeichen wichtig
- Die existierenden Zustände in  $M$  sind  $s_1, \dots, s_k$
- Eine konkrete Zustandsfolge ist  $z_0, z_1, \dots, z_n$
- Eine **Sequenz kann durch viele Zustandssequenzen** emittiert werden

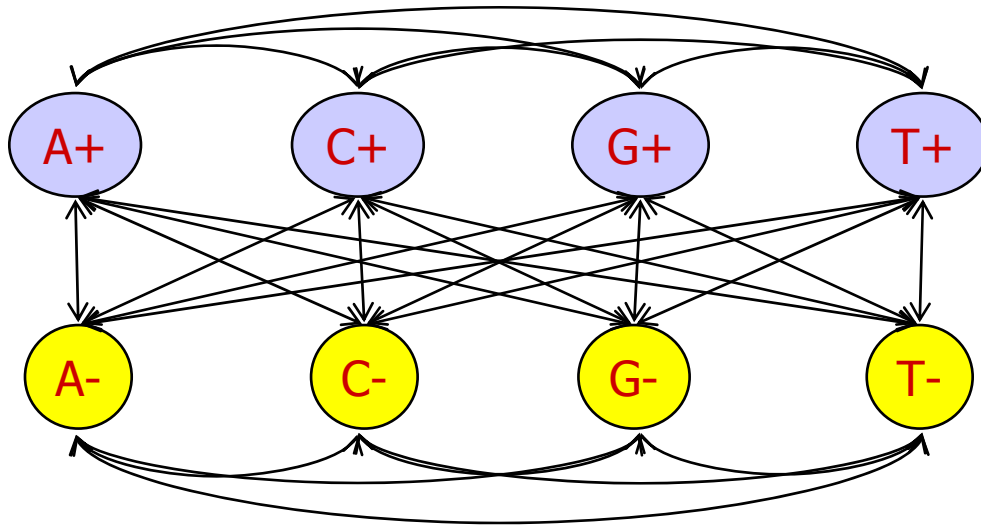
# Beispiel



# Emissionswahrscheinlichkeiten



# Sequenzen und Zustandsfolgen



M+	A	C	G	T
A	.180	.274	.426	.120
C	.171	.368	.274	.188
G	.161	.339	.375	.125
T	.079	.355	.384	.182

M-	A	C	G	T
A	.300	.205	.285	.210
C	.233	.298	.078	.302
G	.248	.246	.298	.208
T	.177	.239	.292	.292

A C T G A C  
 A+C+T-G-A-C-  
  
 A C T G A C  
 A+C+T+G+A+C+  
  
 A C T G A C  
 A-C+T+G+A-C+

- Alle möglich
- Aber **nicht alle gleich wahrscheinlich**
  - Start-Wsk alle 1,  $p(M+ \leftrightarrow M-)$ : 0.01
  - A+C+T-G-A-C-: 0.0000406...
  - A+C+T+G+A+C+: 0.0008726...
  - A-C+T+G+A-C+: 0.0000000...

# Problemfelder

---

- Die drei klassischen HMM Probleme
  - **Dekodierung/Parse**: Gegeben eine Sequenz  $S$  und ein HMM  $M$ ; durch welche Zustandsfolge wurde  $S$  wahrscheinlich erzeugt?
    - Lösung: Viterbi Algorithmus
  - **Evaluation**: Gegeben eine Sequenz  $S$  und HMM  $M$ ; mit welcher Wahrscheinlichkeit wurde  $S$  durch  $M$  erzeugt?
    - Muss alle Zustandssequenzen berücksichtigen, die  $S$  erzeugen
    - Lösung: Forward/Backward Algorithmus
  - **Lernen/Trainieren**: Gegeben eine Sequenz  $S$ ; welche HMM  $M$  (mit gegebener Zustandsmenge) erzeugt  $S$  mit der größten Wahrscheinlichkeit?
    - Lernen der Übergangs- und Emmissionswahrscheinlichkeiten aus  $S$
    - Lösung: Baum-Welch Algorithmus



# Example: The Dishonest Casino

A casino has two dice:

- Fair die

$$P(1) = P(2) = P(3) = P(5) = P(6) = 1/6$$

- Loaded die

$$P(1) = P(2) = P(3) = P(5) = 1/10$$

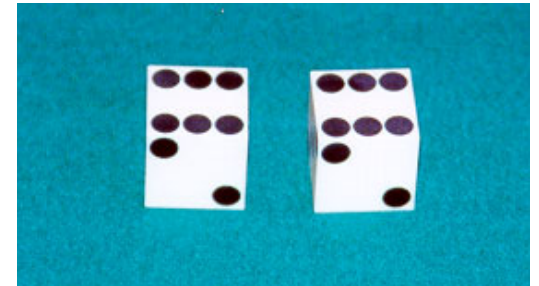
$$P(6) = 1/2$$

Casino occasionally switches between dice

(and you want to know when)

## Game:

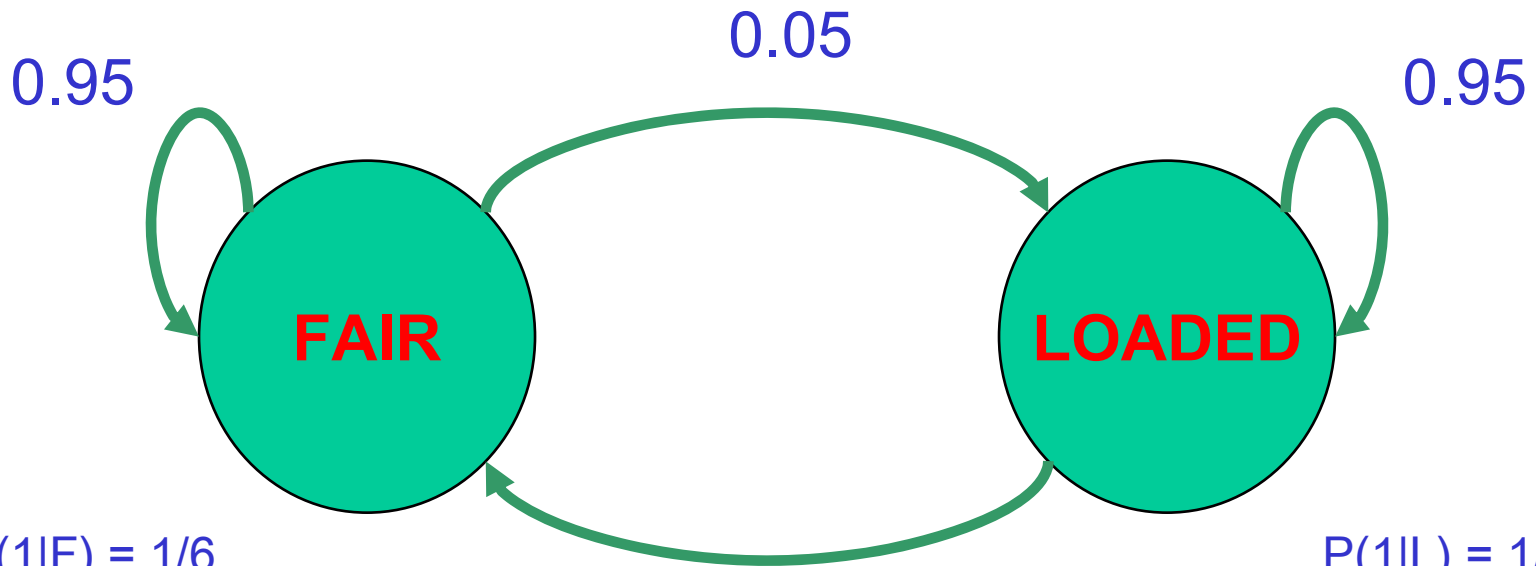
1. You bet \$1
2. You roll (always with a fair die)
3. You may bet more or surrender
4. Casino player rolls (with some die...)
5. Highest number wins



Quelle: Batzoglou, Stanford

# The dishonest casino model

---



$$\begin{aligned} P(1|F) &= 1/6 \\ P(2|F) &= 1/6 \\ P(3|F) &= 1/6 \\ P(4|F) &= 1/6 \\ P(5|F) &= 1/6 \\ P(6|F) &= 1/6 \end{aligned}$$

$$\begin{aligned} P(1|L) &= 1/10 \\ P(2|L) &= 1/10 \\ P(3|L) &= 1/10 \\ P(4|L) &= 1/10 \\ P(5|L) &= 1/10 \\ P(6|L) &= 1/2 \end{aligned}$$

# Question # 1 – Decoding

---

**GIVEN** A sequence of rolls by the casino player

62146146136136661664661636616366163616515615115146123562344

**QUESTION** What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question in HMMs

# Question # 2 – Evaluation

---

**GIVEN** A sequence of rolls by the casino player

62146146136136661664661636616366163616515615115146123562344

**QUESTION** How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem in HMMs

# Question # 3 – Learning

---

**GIVEN** A sequence of rolls by the casino player  
6146136136661664661636616366163616515615115146123562344

## QUESTION

How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

This is the **LEARNING** question in HMMs

[Note: We need to know how many dice there are!]