



# Informatik II Tutorium

## SS 05

**Vorlesung:**

Prof. Dr. J. Calmet

**Übungsleitung:**

Dipl.-Math. Ralf Eberhardt

**Tutorium:**

12 & 13

**Tutor:**

Christian Maier

Tutorium 9: Mittwoch 22. Juni 2005



## Übersicht heute:

- Erfüllbarkeitsäquivalenz (Wdh.)
- Dynamisches Programmieren
- Textsuche
- Trellis
- Fehlerkorrigierende Codes



## Anmerkung:

Bezüglich der Abgabe der Programmieraufgaben hat Herr Eberhard in der Übung lediglich gesagt, daß "**... eine Abgabe per E-Mail allein nicht mehr ausreichend ist!**" Das heißt NICHT dass die Programme nur noch in der Rechnerübung vorgeführt werden müssen OHNE Abgabe per E-Mail !!!

**Programme die NICHT rechtzeitig per E-Mail an mich gingen, können auch nicht in der Rechnerübung abgenommen werden!!!!** Wer unentschuldig der Rechnerübung ferngeblieben ist und NICHT seine Programme vorgeführt hat, hat **keinen** Anspruch auf „Nachprüfung“!!!



# Tutorium 9



# Wiederholung: Erfüllbarkeitsäquivalenz

- Definition: Eine PL-Formel heißt erfüllbar, wenn mindestens eine Struktur Modell ist
- Satz:  $F$  erfüllbar gdw.  $SKF(F)$  erfüllbar
- Definition: Eine SKNF ist eine bereinigte Pränexform, in der alle Existenzquantoren eliminiert wurden. Die Teilform ohne Allquantoren heißt Matrix. Die Matrix muss in KNF sein (sonst nur Skolemform). SKNF ist erfüllbarkeitsäquivalent zu  $F$ .
- Zwei Formeln heißen erfüllbarkeitsäquivalent, wenn sie beide erfüllbar sind bzw. beide nicht erfüllbar sind. Der Unterschied zur "üblichen" Äquivalenz ist, dass es reicht, wenn es für jede Formel eine Belegung gibt, die sie erfüllt. Das müssen nicht die gleichen Belegungen sein. Bei der "üblichen" Äquivalenz müssen alle Belegungen für beide Formeln den gleichen Wert liefern



## Definition (Erfüllbarkeitsäquivalenz)

Zwei Formeln  $F$  und  $G$  sind genau dann **erfüllbarkeitsäquivalent**, wenn gilt:  
 $F$  ist erfüllbar GDW.  $G$  erfüllbar ist.

- $F$  und  $G$  sind erfüllbarkeitsäquivalent: Es gibt eine erfüllende Struktur  $\mathcal{A}$  zu  $F$  GDW. es eine erfüllende Struktur  $\mathcal{A}'$  zu  $G$  gibt.
- $F$  und  $G$  sind erfüllbarkeitsäquivalent:  $F$  ist unerfüllbar GDW.  $G$  unerfüllbar ist.
- Äquivalenz betrifft gleiche Bewertung durch alle passenden Strukturen.  
Erfüllbarkeitsäquivalenz betrifft die Existenz von erfüllenden Strukturen.  
(Diese können verschieden sein.)
- Erfüllbarkeitsäquivalenz ist schwächer als Äquivalenz,  
d.h. Äquivalenz schliesst Erfüllbarkeitsäquivalenz ein.

## Beobachtung

- Zwei Formeln sind genau dann erfüllbarkeitsäquivalent, wenn beide erfüllbar sind oder beide unerfüllbar sind.



## Problem:

Beim Lösen von Problemen mittels „Teile und Herrsche“ tauchen gleiche Teilprobleme oft mehrfach auf. Rein rekursive Algorithmen verbrauchen hier unnötig viel Zeit durch wiederholte Ausführung identischer Berechnungen.

## Lösung:

Erstelle Tabelle, in der bereits berechnete Teilergebnisse gespeichert bzw. systematisch im Voraus berechnet werden.

Diese Technik heißt **Dynamisches Programmieren**.



## **Problemstellung:**

Finde gegebenes Muster in Text.

## **Einfachste Lösung:**

Muster an jeder Textstelle anlegen und vergleichen.

Aber: Aufwand im Worstcase  $O(m \cdot n)$

( $m$  = Länge Muster,  $n$  = Länge Text)



## Grundgedanke:

Mit Vorwissen über das Muster kann man die Suche beschleunigen.



# Einführung Knuth Morris Pratt Algorithmus

Der **Knuth-Morris-Pratt-Algorithmus**, auch KMP-Algorithmus ist ein Suchalgorithmus. Seine Besonderheit besteht darin, dass durch eine eingehende Überprüfung des Suchbegriffs (auch Muster) die Anzahl der nötigen Textvergleiche reduziert wird.



# Knuth Morris Pratt: Einleitung

Um die Arbeitsweise des KMP-Algorithmus besser nachvollziehen zu können, ist es sinnvoll, einen Blick auf den **naiven Ansatz** des Suchproblems zu werfen:

Ein *Muster B* kann in einem *Suchtext A* mit folgendem Verfahren gefunden werden (*i* ist dabei das aktuelle Zeichen in *A*):

```
01) Setze  $i = 1$ .
02) Wiederhole folgendes:
03)  Vergleiche das Muster B Zeichen für Zeichen mit dem
      Suchtext A, beginnend mit dem Zeichen  $i$ .
04)   Bei Übereinstimmung:
05)     return  $i$ 
06)   andernfalls:
07)     erhöhe  $i$  um 1
08) Wenn das Textende von A ohne Resultat erreicht wird:
09)   return -1
```

- Dieses Verfahren kann noch verbessert werden, indem man den Vergleich zwischen Muster B und dem Suchtextabschnitt (03) abbricht, sobald sich die Wörter in einem Zeichen unterscheiden.
- Der Nachteil des Verfahrens besteht darin, dass nach einem erfolglosen Vergleich bei  $i + 1$  weitergesucht wird (07), statt die bereits in Schritt (03) verglichenen Zeichen mit in Betracht zu ziehen. Der KMP-Algorithmus untersucht vor der eigentlichen Suche das Muster B und kann daher Rückschlüsse darauf ziehen, inwiefern man in Schritt (07) den Wert  $i$  um mehr als nur 1 erhöhen kann. Diese Erhöhung wird im Folgenden *Verschiebung* genannt.



- Der **erste Teil** des Algorithmus baut eine Tabelle auf, aus der für jede Position im Muster der Verschiebewert abgelesen werden kann.
- Der **zweite Teil** durchläuft den Suchtext und vergleicht ihn mit den Symbolen des Musters. Bei Unstimmigkeiten greift er auf die Tabelle zurück um zu ermitteln um wie weit das Muster nach hinten geschoben werden kann.



# Knuth Morris Pratt: Erstellung des Verschiebefelds

## Vorbereitung

- Über das Suchmuster (im Beispielfall das Wort *gegeben*) schreibt man beginnend beim ersten Buchstaben aufsteigend natürlichen Zahlen inklusive der Null in die Ablesezeile, welche von minus Eins angeführt wird. Dem ersten Buchstaben im Verschiebefeld wird -1 zugewiesen.

```
-1   #           <- Rautezeiger
     0   1   2   3   4   5   6   <- Ablesezeile
     G   E   G   E   B   E   N   <- Suchmuster
           ^           <- Positionszeiger
     -1           <- Verschiebefeld (nach -1 noch leer)
```

## Erklärung:

- Nachdem wir die Ablesezeile und das Suchmuster mit den zwei Zeigern (Rautezeiger und Positionszeiger) versehen haben, bauen wir unser Verschiebefeld auf. Es werden nun die Symbole in der jeweiligen Spalte unserer Zeiger verglichen.
- Die # in der oberen Schiene greift bei einer Unstimmigkeit die darunter stehende Zahl auf, lässt sie in unserem Verschiebefeld bei ^ fallen und sinkt nach links, bis im Verschiebefeld eine -1 oder in der Ablesezeile der selbe Wert steht, wie im Verschiebefeld. Sind die Symbole in den Spalten der Zeiger gleich, wird dem neuen Verschiebefeldwert der Wert unter dem Rautezeiger des bereits erstellten Schiebefeldes zugewiesen.
- Nach jedem Vergleichsschritt (bei Unstimmigkeit vorher sinken lassen) werden beide Zeiger jeweils um einen Buchstaben (Zeiger können verschieden versetzt sein) nach rechts verschoben. Das Verschiebefeld ist fertig, wenn jedem Symbol aus dem Suchmuster ein Zahlenwert im Verschiebefeld zugeordnet ist.



# Knuth Morris Pratt: 1. Schritt

<del>#</del>	<del>#</del>						
-1	0	1	2	3	4	5	6
	G	E	G	E	B	E	N
		^					
	-1	0					

// Erklärung:  $G \neq E \rightarrow$  nächster Wert  
Verschiebefeld 0 (da # bei 0), # sinkt auf  
-1, beide Zeiger jeweils um eins nach  
rechts verschieben



## Knuth Morris Pratt: 2. Schritt

	#							
-1	0	1	2	3	4	5	6	
	G	E	G	E	B	E	N	
			^					
	-1	0	-1					

//Erklärung: G = G -> nächster Wert  
Verschiebefeld -1 (Wert unter # im  
Verschiebefeld), beide Zeiger jeweils um  
eins nach rechts verschieben



## Knuth Morris Pratt: 3. Schritt

		#						
-1	0	1	2	3	4	5	6	
	G	E	G	E	B	E	N	
				^				
	-1	0	-1	0				

//Erklärung: E = E -> nächster Wert  
Verschiebefeld 0 (Wert unter # im  
Verschiebefeld), beide Zeiger jeweils um  
eins nach rechts verschieben



## Knuth Morris Pratt: 4. Schritt


			#					
-1	0	1	2	3	4	5	6	
	G	E	G	E	B	E	N	
					^			
	-1	0	-1	0	2			

//Erklärung:  $G \neq B \rightarrow$  nächster Wert

Verschiebefeld 2 (da # bei 2), # sinkt bis  
-1, beide Zeiger jeweils um eins nach  
rechts verschieben




## Knuth Morris Pratt: 5. Schritt

	#							
-1	0	1	2	3	4	5	6	
	G	E	G	E	B	E	N	
								
	-1	0	-1	0	2	0		

//Erklärung:  $G \neq E \rightarrow$  nächster Wert  
Verschiebefeld 0 (da # bei 0), # sinken  
lassen bis -1, beide Zeiger jeweils um  
eins nach rechts verschieben



## Knuth Morris Pratt: 6. Schritt

	#						
-1	0	1	2	3	4	5	6
	G	E	G	E	B	E	N
							
	-1	0	-1	0	2	0	0

//Erklärung:  $G \neq N \rightarrow$  nächster Wert  
Verschiebefeld 0 (da # bei 0), #  
rücksetzen

Nun ist das Verschiebefeld aufgebaut und kann auf den Suchtext angewendet werden. Die Anzahl der Schritte ist von der Länge des Suchworts abhängig.



# Knuth Morris Pratt: Anwendung des Verschiebefelds



- Muster: „gegeben“
- Belegung der Reihung  $f$  (Verschiebefeld): „-1 0 -1 0 2 0 0“
- Gesucht: Vorgehen Suche Muster im Text:

g	e	g	e	b	a	e	g	e	g	e	b	e	n	a	g
g	e	g	e	b	e										
		.			g										
						g									
							g	e	g	e	b	e	s		
		2	1	0	0										
		g	e	g	e										



## Aufgabe 2.2: Textsuche nach KMP (Info2/SS 04)

Suchen Sie mit Hilfe des Knuth-Morris-Pratt-Algorithmus im Text "etpeterestetepetete" das Muster "etepetete". Nutzen Sie hierbei das Vorwissen über das gesuchte Wort optimal aus (shift-Vorausberechnung!).

*Shift- Vorausberechnung:*

<b>j</b>	0	1	2	3	4	5	6	7	8
<b>s[j]</b>	e	t	e	p	e	t	e	t	e
<b>shift(j)</b>	-	0	0	1	0	1	2	3	2

*KMP- Algorithmus:*

e	t	p	e	t	e	r	e	s	t	e	t	e	p	e	t	e	t	e	k	j	shift(j)
e	t	e	p	e	t	e	t	e											0	2	0
		e	t	e	p	e	t	e	t	e									2	0	0
			e	t	e	p	e	t	e	t	e								3	3	1
				e	t	e	p	e	t	e	t	e							5	1	0
					e	t	e	p	e	t	e	t	e						6	0	0
						e	t	e	p	e	t	e	t	e					7	1	0
							e	t	e	p	e	t	e	t	e				8	0	0
								e	t	e	p	e	t	e	t	e			9	0	0
									e	t	e	p	e	t	e	t	e		10		



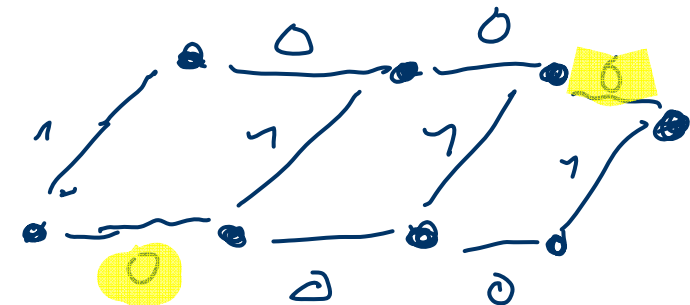
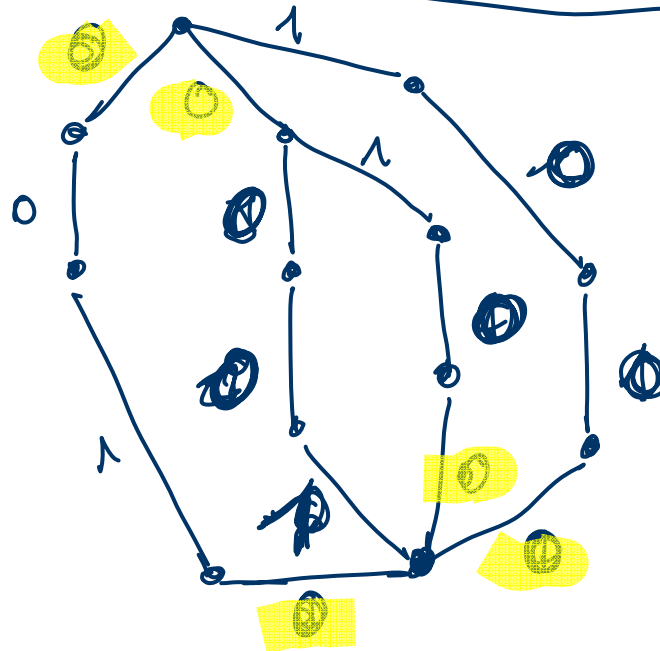
## Mehr Infos zu KMP:

- <http://www-igm.univ-mlv.fr/~lecroq/string/node32.html#SECTION00320>  
(inkl. Java Demo)
- <http://www.itf.fh-flensburg.de/lang/algorithmen/pattern/kmp.htm>
- [http://de.wikipedia.org/wiki/Algorithmus\\_von\\_Knuth-Morris-Pratt](http://de.wikipedia.org/wiki/Algorithmus_von_Knuth-Morris-Pratt)
- Weiterer Textsuchalgorithmus: Boyer-Moore



was sind Trellis?  
(engl. Gitter, Rastergitter)  
sind tree-like-structures

Überführen Sie das  
folgende Trelli in  
minimaler Form





## Begriffe:

- *Informationswort*  
 $(i_0, \dots, i_k) \in \mathbb{F}_2^k$
- *Darstellung als Polynom*  
 $i(x) = \sum_{j=0}^k i_k x^j$
- *Generatorpolynom*  
 $g(x) = \sum_{j=0}^{n-k} g_j x^j$
- *Codewort als Polynom*  
 $c(x) = i(x)g(x) := \sum_{j=0}^n c_j x^j$   
 $c = (c_1, \dots, c_n) \in \mathbb{F}_2^n$

## Kanalmodell:

Codewort  $c \in \mathbb{F}_2^n \rightarrow$  moduliertes Codewort  $m \in \mathbb{R}^n \rightarrow$  Empfangswort  $y \in \mathbb{R}^n$  mit  $y = e + m$ , Fehler  $e \in \mathbb{R}^n$

**Fragen:** Wie komme ich zurück zu 0, 1? Welches Codewort passt am besten zu  $y$ ?

$c \rightarrow m, 0 \mapsto 1 \in \mathbb{R}, 1 \mapsto -1 \in \mathbb{R}$

Wie suche ich ein passendes Codewort?



# Fehlerkorrigierende Codes: Beispiel

$$i = (0, 1), i(x) = x, g(x) = x + 1, c(x) = g(x)i(x) = x^2 + x \Rightarrow c = (0, 1, 1), m = (1, -1, -1)$$

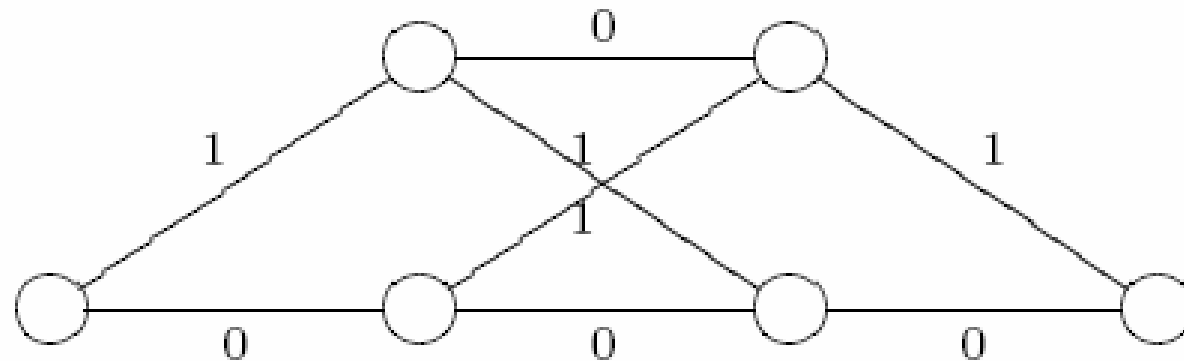
$$00 \mapsto (0, 0, 0)$$

$$01 \mapsto (0, 1, 1)$$

$$10 \mapsto (1, 1, 0)$$

$$11 \mapsto (1, 0, 1)$$

Trellis:





## Zu Aufgabe 5

b) bitte folgende „Muster“-Tabelle verwenden:

Ebene	Schritt 1	Schritt 2	Schritt 3	Schritt 4	Schritt 5
Knoten 0					



## Ein bisschen Sarkasmus?

