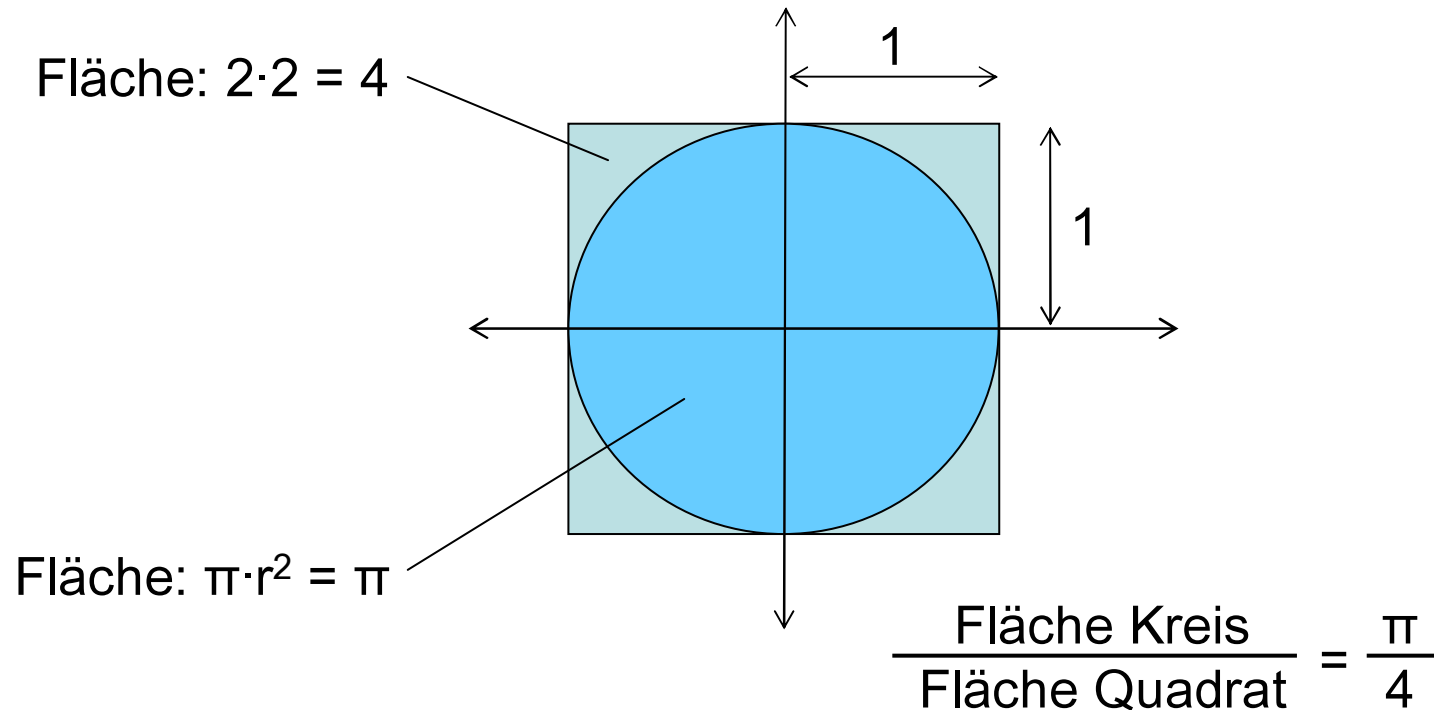


Übungen zu Informatik I Wintersemester 03/04

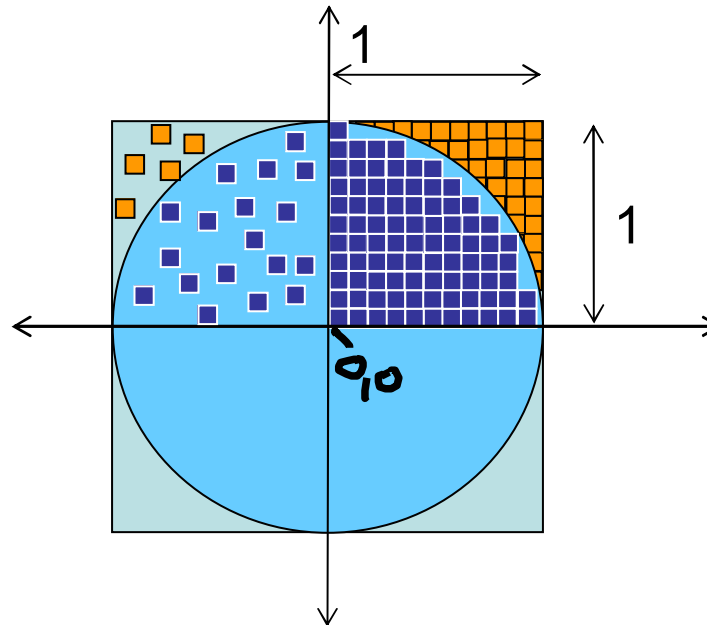
Übungsleiter: Dipl.-Inform. Tom Gelhausen



- Frage: welchen Wert hat π ?
- Berechnung mit probabilistischem Algorithmus:



- Berechnung des Flächenverhältnisses



- Daher: setze zufällig Punkte und zähle die im Einheitskreis
 - Eigenschaft: Abstand vom Nullpunkt ≤ 1
 - Abstandsberechnung: $x^2 + y^2 = d^2$
 - Es gilt: $\text{imKreis} \Leftrightarrow d \leq 1 \Leftrightarrow d^2 \leq 1$
 - also: $\text{imKreis}(x, y) = (x*x + y*y) \leq 1$ (Quadrate \Rightarrow ~~Betrag~~)



1. Wir brauchen Zufallszahlen (Achtung: Qualität entscheidend!)

```
import Random

r :: (Double, Double) -> StdGen -> (Double, StdGen)
r (lo, hi) generator = (lo + ((hi - lo)
                        * (1 + fromInt (fst ng))
                        / fromInt (maxBound)), snd (ng))
  where ng = next generator

rlist range gen = wert : rlist range genneu
  where (wert, genneu) = r range gen
```

2. Aus der Zufallszahlenfolge muss eine Folge von Zufalls-Koordinaten von Punkten (x,y) gemacht werden

```
machKoordinaten [] = []
machKoordinaten [x] = []
machKoordinaten (x:y:xs) = (x,y) : machKoordinaten xs
```



3. Die Folge der genauen Koordinaten interessiert uns nicht, sondern nur, ob der Punkt im Kreis liegt, oder nicht

```
inOutL = map imKreis imEinheitskreis punktMenge
```

4. Nun wollen wir wissen, wie viele der Punkte im Kreis liegen, wenn die Stichprobe aus n Zufallskordinaten besteht

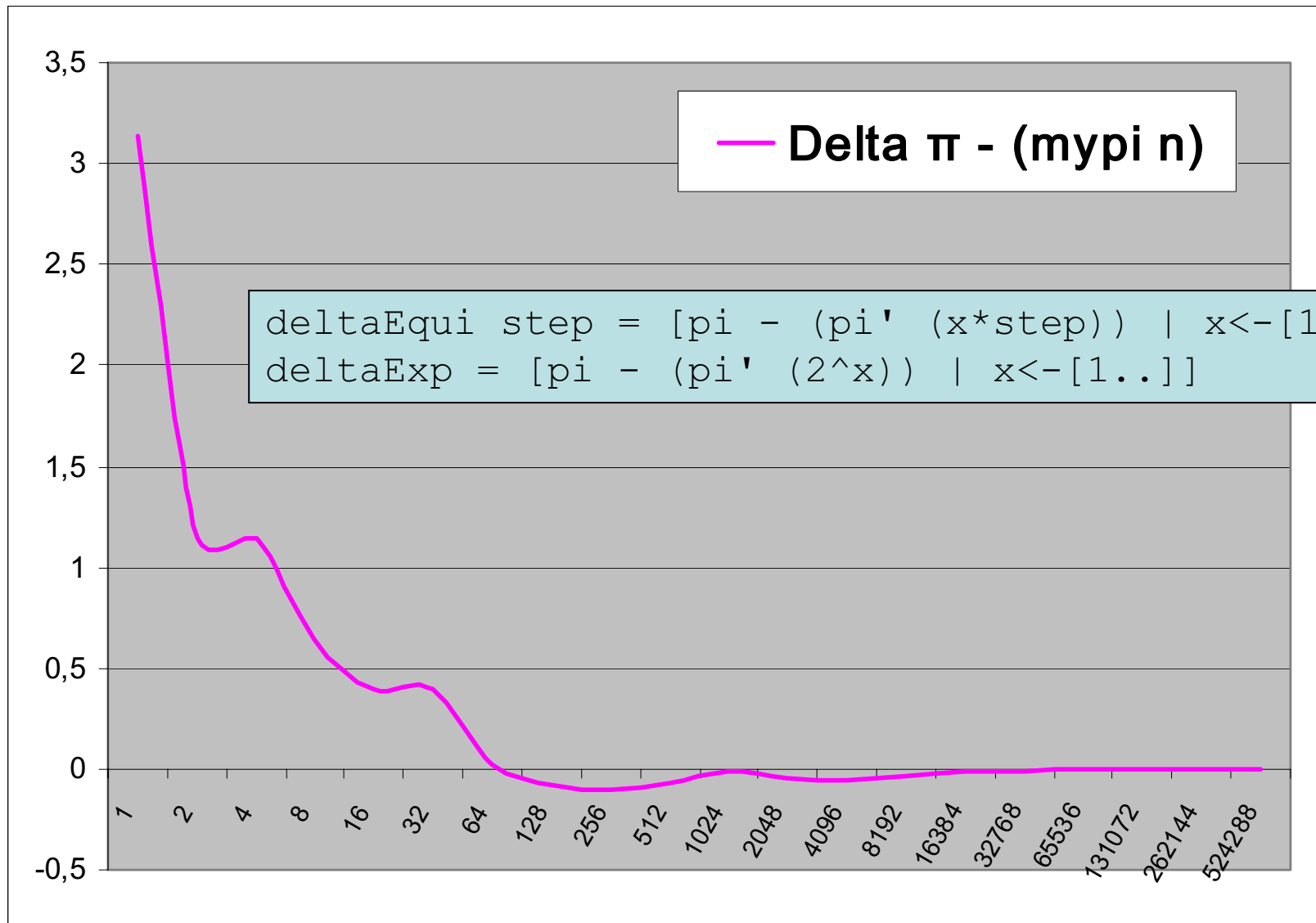
```
anzahlImKreis n = length (filter (==True) (take n inOutL))
```

5. Jetzt brauchen wir nur noch das Verhältnis zu bilden um π mit der Genauigkeit von n Stichproben zu berechnen

```
mypi n = 4.0*(fromInt (anzahlImKreis n))/(fromInt n)
```

$$\frac{\pi}{4}$$





- Wenn ein Algorithmus sich selbst rekursiv aufruft, kann seine Laufzeit oft mit einer Rekurrenz beschrieben werden.
 - Eine Rekurrenz ist eine Gleichung oder Ungleichung, bei der der Funktionswert in Form von Funktionswerten für kleinere Eingabewerte beschrieben wird.
 - Einen allgemeinen Algorithmus zur Lösung einer Rekurrenz gibt es nicht.
 - **Beispiel:**
 - Sortieren durch Mischen:
 - $t(1) = \Theta(1)$
 - $t(n) = 2 \cdot t(n/2) + \Theta(n)$

Liste in zwei
Hälften
aufteilen

Ergebnis
zusammen-
bauen



▪ Lösen von Rekurrenzrelationen: Substitutionsmethode

- Eine Rekurrenz lässt sich lösen, indem eine Lösung „erraten“ wird:
 - Berechne $t(n)$ für einige n .
 - Schätze geschlossene Form für $t(n)$ (oder ggf. nur Schranke).
 - Beweise (Induktion?), dass das geschätzte $t(n)$ korrekt ist.

• **Beispiel:**

- $t(n) = 2 \cdot t(n/2) + 4$ und $t(1) = 1$
- $t(1) = 1 = 5 \cdot 1 - 4$
- $t(2) = 4 + 2 \cdot 1 = 6 = 5 \cdot 2 - 4$
- $t(4) = 4 + 2 \cdot 6 = 16 = 5 \cdot 4 - 4$
- $t(8) = 4 + 2 \cdot 16 = 36 = 5 \cdot 8 - 4$
- $t(16) = 4 + 2 \cdot 36 = 76 = 5 \cdot 16 - 4$
- $t(n) = \dots = 5 \cdot n - 4$

$\Rightarrow t(n) \in O(n)$



▪ Lösen von Rekurrenzrelationen: Hauptsatz über Rekurrenzen

- Wenn $a, b, p > 0, n = p^m, m > 0$ ganzzahlig, wobei
 - a = Anzahl der Teile, in die das Problem zerlegt wird
 - b = Kosten für Teilen und Zusammenfügen (Verw.-aufwand) in jedem Schritt (\rightarrow „ $\cdot n$ “)
 - p = Faktor, um den das Problem beim Rekursionsabstieg kleiner
 - daher Ansatz nur geeignet, wenn
 - sich der Problemumfang bei jedem Aufruf um einen konstanten Faktor reduziert
 - nicht bei „-1“

$$t(n-1)$$

- Dann hat die Rekurrenz $t(n) = a \cdot t(n/p) + b \cdot n \quad (n > 1)$

die Lösung

$t(n) =$	$\Theta(n)$	$a < p$
	$\Theta(n \cdot \log n)$	$a = p$
	$\Theta(n^{\log_p a})$	$a > p$



$$f(0) = 1$$

$$f(n) = (f(x)) + (f(x)) + (f(x)) + (f(x))$$

$$\text{where } x = n/4$$

- a = Anzahl der Teile, in die das Problem zerlegt wird
- b = Kosten für Teilen und Zusammenfügen (Verw.-aufwand) in jedem Schritt (\rightarrow „ n “)
- p = Faktor, um den das Problem beim Rekursionsabstieg kleiner
- $t(n) = a \cdot t(n/p) + b \cdot n$

- $a = 4, b = 1, p = 4$

- $t(n) = 4 \cdot t(n/4) + 1 \cdot n$

- $t(0) = 1$
- $t(n) \in \Theta(n \cdot \log n)$, da $a=p$



- $t(n) = 7 \cdot t(n/2) + 18 \cdot (n/2)$

$$t(1) = 1$$

$$f [] = 1$$

$$f l = (f x) + (f y) * (f x) / (f y) + (f x) - (f y) - (f x)$$

where

```
x = take len (take len (take len ( ... l)))
```

```
y = drop len (take len (take len ( ... l)))
```

```
len = (length l) div 2
```

- $a = 7$

- $b = 9$

- $p = 2$

- $t(0) = 1$

- $t(n) = a \cdot t(n/p) + b \cdot n \in \Theta(n^{\log_p a})$, da $a > p$



- $t(1) = 1$
- $t(n) = n + t(n-1)$

- Ungeeignet, da das Problem nicht um einen festen Faktor kleiner wird

- Idee: $t(n) = n + (n-1) + \dots + 2 + 1 = n(n+1)/2, \quad n > 0$

- Beweis mit vollständiger Induktion:
 - I.A.: $t(1) = 1$
 - I.V.: Gelte $t(n) = n(n+1)/2$ für ein festes n
 - I.S.:
$$\begin{aligned} t(n+1) &= (n+1) + t(n) \\ &= (n+1) + n(n+1)/2 \\ &= (2n+2 + n^2+n)/2 \\ &= (n+1)(n+2)/2 \end{aligned}$$



Zu den Fragen...



- Regel bei Markov/Semi-Thue: $\mapsto, \mapsto.$
- Ableitungen bei Markov/Semi-Thue: $\Rightarrow, \Rightarrow^*, \Rightarrow^+$
- Produktionen bei Grammatiken: \mapsto
- Konversionen beim Lambda-Kalkül: $=^R$ mit $R \in \{\alpha, \beta, \eta\}$
- Implikation: \Rightarrow (auch \rightarrow)
- Ableitung (siehe 4.1): $\vdash_{\text{Kalkül}}$

$\mathcal{F} \vdash F$	}	Prämisse
$\mathcal{F} \vdash F \rightarrow G$		
$\mathcal{F} \vdash G$		
Konklusion		
- Folgerung (siehe 4.1): \vDash
also $\{p, p \rightarrow q\} \vDash q$

$(\mathcal{F} \vdash_K F$ bezeichnet die syntaktische Herleitung einer Folgerung $\mathcal{F} \vDash F)$



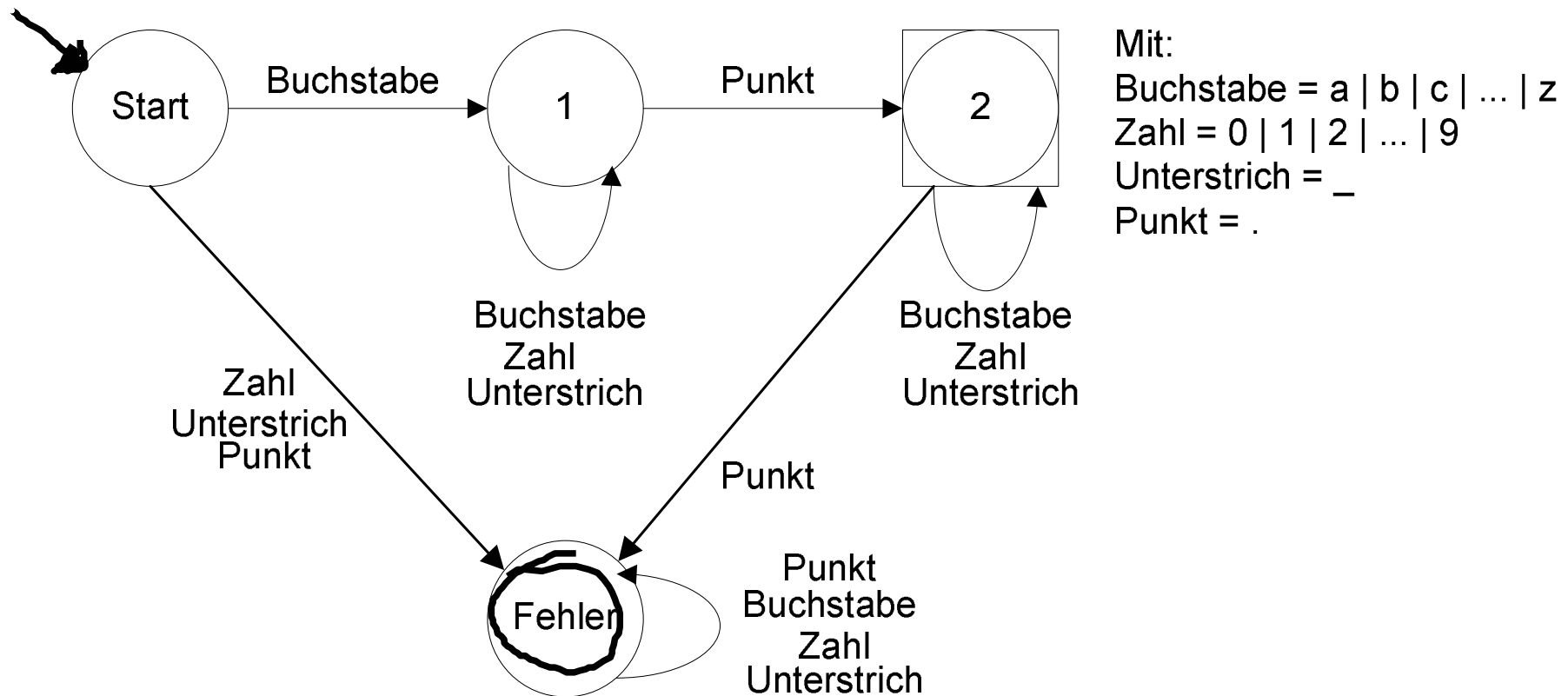
- „Bereinigte“ Form: DNF oder KNF
- Pränexe Form: alle Quantoren vorne
- Skolem-Form: in pränexer Form, jedoch alle auftretenden Existenzquantoren durch Skolem-Funktionen ersetzt

- Es wird nicht gefordert
 - dass die pränexe Form bereinigt ist, insbesondere nicht,
 - dass eine DNF oder eine KNF vorliegt
 - dass die Skolem-Form bereinigt ist, insbesondere nicht,
 - dass eine DNF oder eine KNF vorliegt

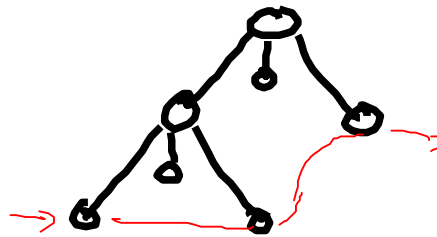
- Aber es wird **dringend geraten**, zuerst die bereinigte Form herzustellen!



- Akzeptor für „...stets mit einem Buchstaben beginnen auf den noch beliebig viele Zahlen, Buchstaben oder Unterstriche folgen können. Dann folgt ein Punkt auf den eventuell noch beliebig viele Unterstriche, Zahlen und Buchstaben folgen können“



- Lineare Datenstrukturen
 - Keller (Zugriff: LIFO) (engl.: stack)
 - Schlange (Zugriff: FIFO) (engl.: queue)
 - Prioritätenschlange
 - Reihung (Zugriff: beliebig) (engl.: array)
 - Sequenz/Datei (Zugriff: aktuelle Position, vorwärts, rückwärts) (engl.: file)
- Baumstrukturen
 - Binärbaum
 - Tiefensuche
 - Breitensuche
 - Verallgemeinerungen: B-Baum und B*-Baum
- Mengen
 - Einfachmengen (mathematische Mengen)
 - Vielfachmengen (mit Duplikaten, ohne Reihenfolge)



▪ Haskell

- Keine Bibliotheken außer Prelude.hs
- Version aus
 - "Haskell 98 Language and Libraries - The Revised Report,, von Simon Peyton Jones
- Nein, Sie brauchen das Ding nicht auswendig zu können.
- Nein, wirklich nicht!
(Verwenden Sie die Lernzeit lieber für etwas anderes...)

▪ Prolog

- keine Bibliotheken
- nur die Sprachkonstrukte, die in Vorlesung oder Übung behandelt wurden



- Wir unterscheiden CH-X und Chomsky-X um den Unterschied zwischen Sprachklasse und Grammatikklasse deutlich zu machen
 - Verwende **CH-X**, wenn es um die Klasse der Grammatik geht.
 - Verwende **Chomsky-X**, wenn es um die Klasse der Sprache geht.

- Wenn nach dem „höchsten“ oder dem „speziellsten“ Typ einer Sprache oder Grammatik gefragt ist,
 - gib CH/Chomsky - Typ 3 an, falls das aber nicht geht
 - gib CH/Chomsky - Typ 2 an, und falls das auch nicht geht,
 - gib CH/Chomsky - Typ 1 an und falls das nicht geht,
 - gib CH/Chomsky - Typ 0 an.



- Ja, Folgefehler werden i.A. berücksichtigt
 - aber z.B. dann nicht, wenn die Lösung des Restes der Aufgabe durch den Fehler trivial wird

- **Beispiel:**

- Gegeben seien die Produktionen einer Grammatik:

$$P1 = \left\{ \begin{array}{l} S \rightarrow aBb, \\ B \rightarrow aBb \mid c \end{array} \right\}$$

- Wie lautet die Sprache?

$$\cancel{a^n b c^n} \quad a^n c b^n \quad n \geq 1$$

$$d^m \quad m \geq 27$$

- Was ist der Typ der Sprache?

~~Chomsky-1~~

Chomsky-2

Chomsky-3



- Ankreuzaufgaben – falsche Freunde!
- „Vorschreiben“ – keine Zeit
- Leserlich schreiben – macht dem Korrektor gute Laune ;-)
- Nur das lösen, was gelöst werden soll – nicht mehr!
- Zeit beachten – Time's running out...



- Das Übungsteam
 - verabschiedet sich von Ihnen
 - bedankt sich für Ihre Aufmerksamkeit
 - und wünscht Ihnen viel Erfolg bei der Klausur!

- Newsgroup <news://news.rz.uni-karlsruhe.de> Gruppe uka.info1

- Zum Übungsteam gehörten
 - 25 Tutoren
 - 2 Hiwis
 - Dominique Lerch
 - Dennis Schieferdecker
 - Tom Gelhausen
 - ...und natürlich Kara!

- Quelle <http://studium.dauweb.de/kara/>

