Technische Universität Braunschweig

# Theoretische Informatik 2

Arne Schmidt

# Vorstellung

Interessen

- Geometrische Optimierung,
- Programmierbare Materie,
- Komplexitätstheorie

#GernePerDu

Computing MaxMin Edge Length Triangulations

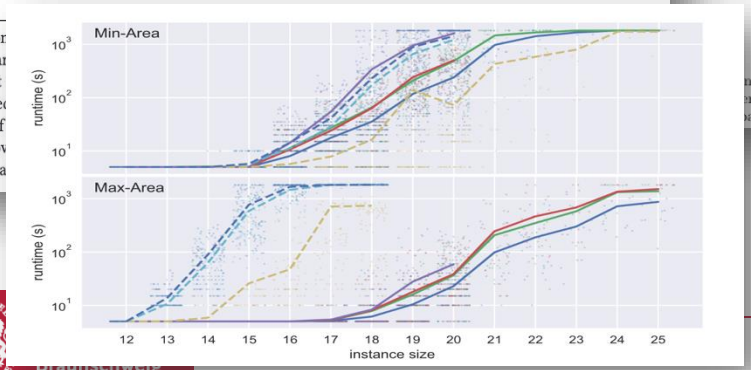
Sándor P. Fekete * Winfried Hellmann* Michael Hemmer* Arne Schmidt*
Julian Troegel*

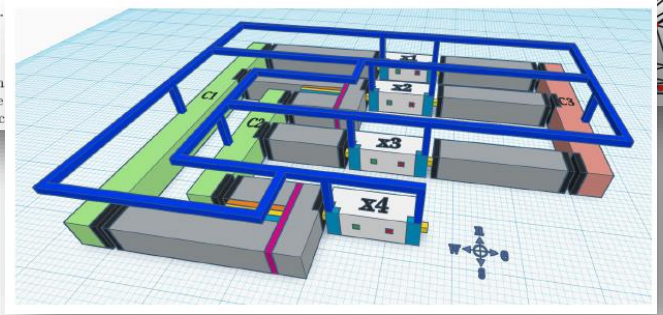
Abstract

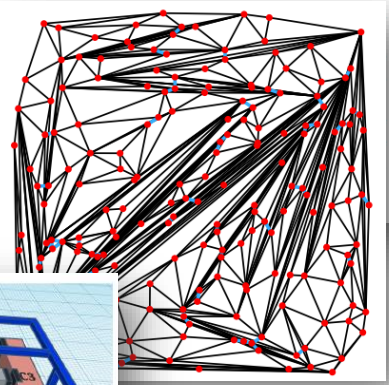
Particle-Based Assembly
Using Precise Global Control*

Computing Area-Optimal Simple Polygonizations

SÁNDOR P. FEKETE, ANDREAS HAAS, PHILLIP KELDENICH, MICHAEL PERK, and ARNE SCHMIDT, Department of Computer Science, TU Braunschweig



# Organisation

Vorlesung + Gr. Übung

- Grundlagen

- Vertiefungen

Arne Schmidt

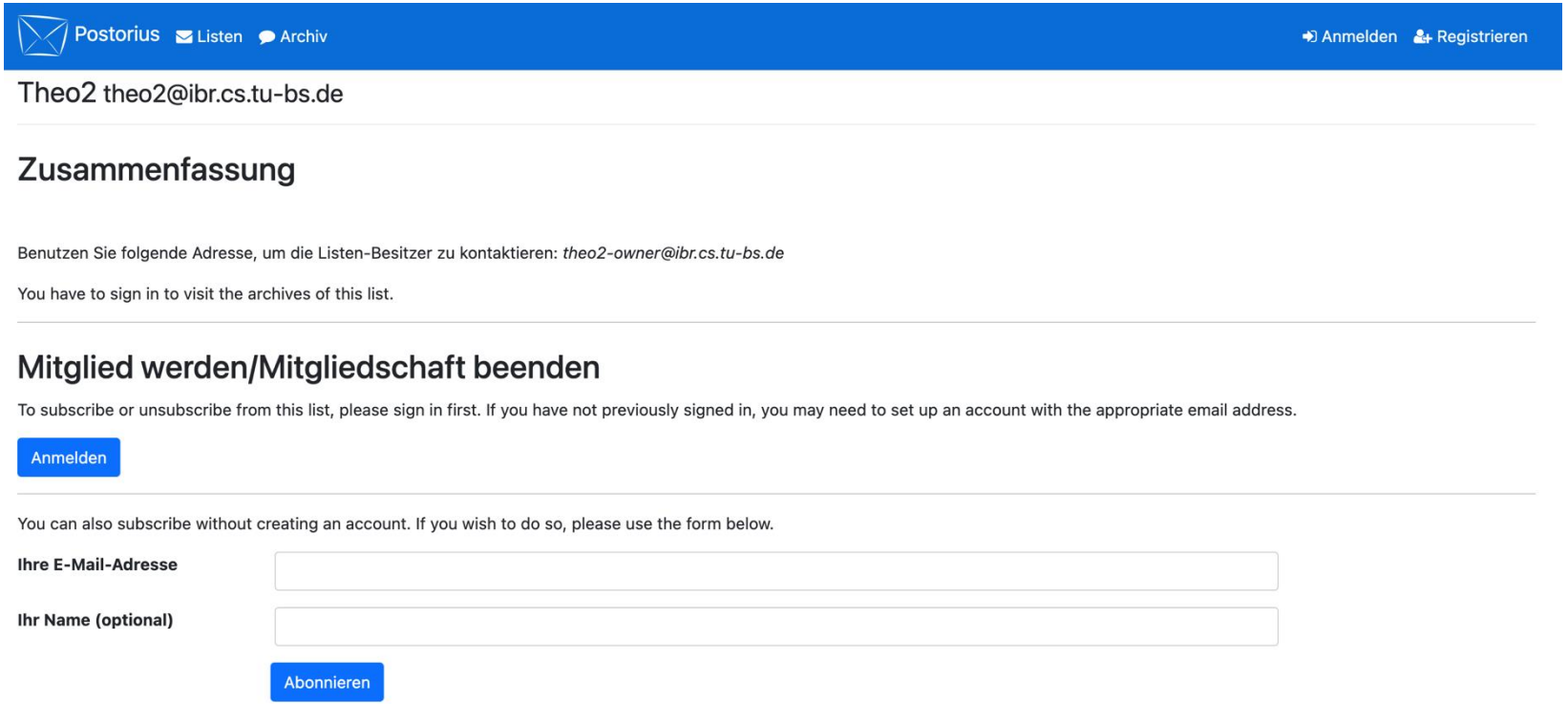
Kl. Übung

Besprechung Hausaufgaben

# Fragen

| | |
|---|---|
| Inhaltlich oder allgemeiner Ablauf | Vorlesung / große Übung |
| Zu Übungsblättern oder Korrektur | Kleine Übungen / Tutoren |
| Individuelle Fragen | **Immer per Mail** an aschmidt@ibr.cs.tu-bs.de (Nicht über StudIP) |
| Sprechstunde | Montags, 09:45 Uhr im Raum IZ 333 (Am besten vorher per Mail ankündigen) |

# Mailingliste (https://lists.ibr.cs.tu-bs.de/postorius/lists/theo2.ibr.cs.tu-bs.de/)



# Semesterplan (vorläufig)

| Datum (Woche) | VL Nummer | VL / Ü Inhalt | Hausaufgabe (Ausgabe, Mo) | Hausaufgabe (Abgabe, Mo, 13 Uhr) | Hausaufgabe (Besprechung) |
|---|---|---|---|---|---|
| **07. April** | - + - | | | | |
| **14. April** | 0+1 | Intro, Kontextsensitive Sprachen, Turing Maschinen | | | |
| **21. April** | -+2 | Satz von Kuroda, Determinismus | HA1 | | |
| **28. April** | 3+4 | Satz von Immermann & Szelepcsény, Entscheidbarkeit | | | |
| **05. Mai** | 5+6 | Rekursiv-aufzählbare Sprachen, Unentscheidbarkeit | HA2 | HA1 | |
| **12. Mai** | 7+ O | Reduktionen | | | HA1 |
| **19. Mai** | 8+Ü1 | Satz von Rice | HA3 | HA2 | |
| **26. Mai** | 9+Ü2 | Probleme kontextfreier Grammatik | | | HA2 |
| **02. Juni** | 10+11 | Komplexitätsklassen | HA4 | HA3 | |
| **09. Juni** | Exkursionswoche | | | | |
| **16. Juni** | 12+13 | Komplexitätskarte, L, NL, Reduktionen, PATH | | | HA3 |
| **23. Juni** | 14+Ü3 | 2-SAT | HA5 | HA4 | |
| **30. Juni** | 15+16 | P, NP, Satisfiability | | | HA4 |
| **07. Juli** | 17+Ü4 | Zertifikate | | HA5 | |
| **14. Juli** | 18+19 | QBF, Savitch, Hierarchiesätze | | | HA5 |

O: Online-Fragestunde; Link per Mail

# Hausaufgaben

- 20 Punkte pro Blatt
- 5 Blätter
- 50% aller Punkte (50 Punkte) für die Studienleistung
- Abgaben per Zweierteams
- Auf Papier

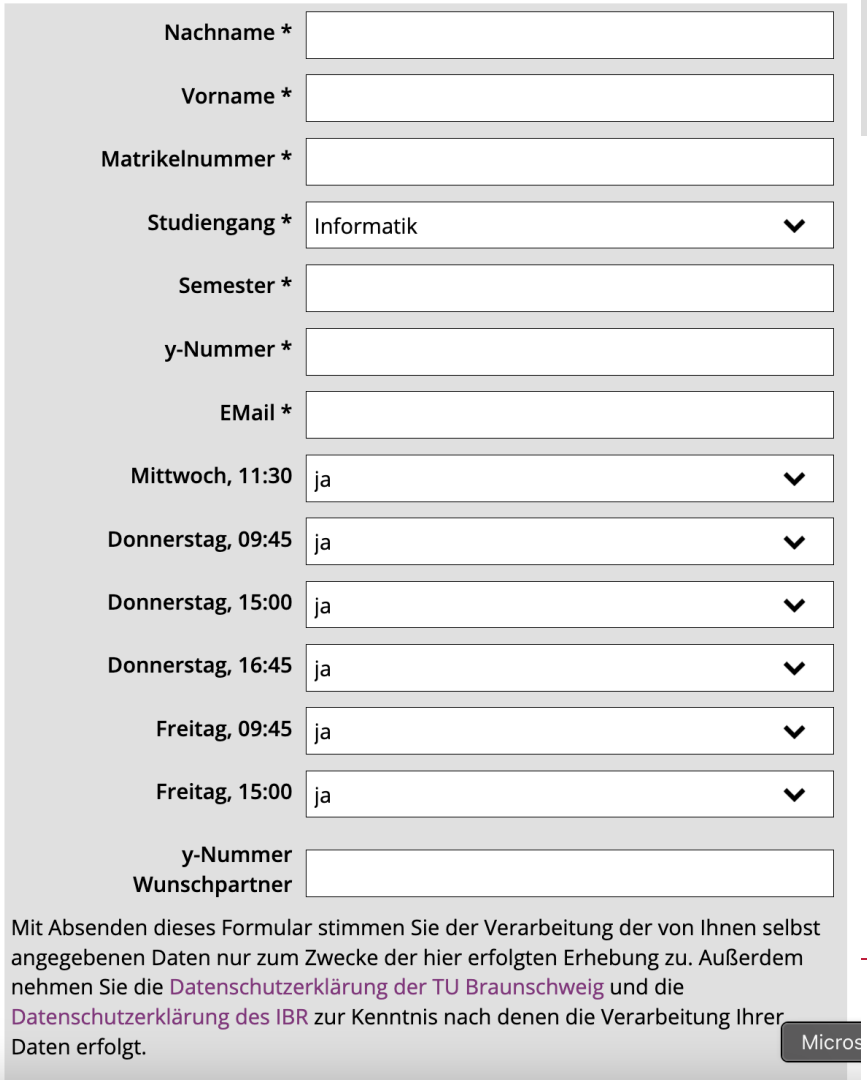
# Anmeldung kleine Übungen

Offen bis einschl. 24.04.

Gruppenaufteilung erfolgt am 25.04. per Mail.

https://www.ibr.cs.tu-bs.de/courses/ss25/theo2/index.xml

Hinweis: Einteilung zu AuD 2 findet früher statt. Falls dadurch ein Termin für Theo 2 nicht mehr funktioniert, einfach erneut für Theo 2 anmelden!



Nachname *

Vorname *

Matrikelnummer *

Studiengang * Informatik

Semester *

y-Nummer *

EMail *

Mittwoch, 11:30 ja

Donnerstag, 09:45 ja

Donnerstag, 15:00 ja

Donnerstag, 16:45 ja

Freitag, 09:45 ja

Freitag, 15:00 ja

y-Nummer Wunschpartner

Mit Absenden dieses Formular stimmen Sie der Verarbeitung der von Ihnen selbst angegebenen Daten nur zum Zwecke der hier erfolgten Erhebung zu. Außerdem nehmen Sie die Datenschutzerklärung der TU Braunschweig und die Datenschutzerklärung des IBR zur Kenntnis nach denen die Verarbeitung Ihrer Daten erfolgt.

Micros

# Klausur – Vorläufig

Datum:
22.07.25

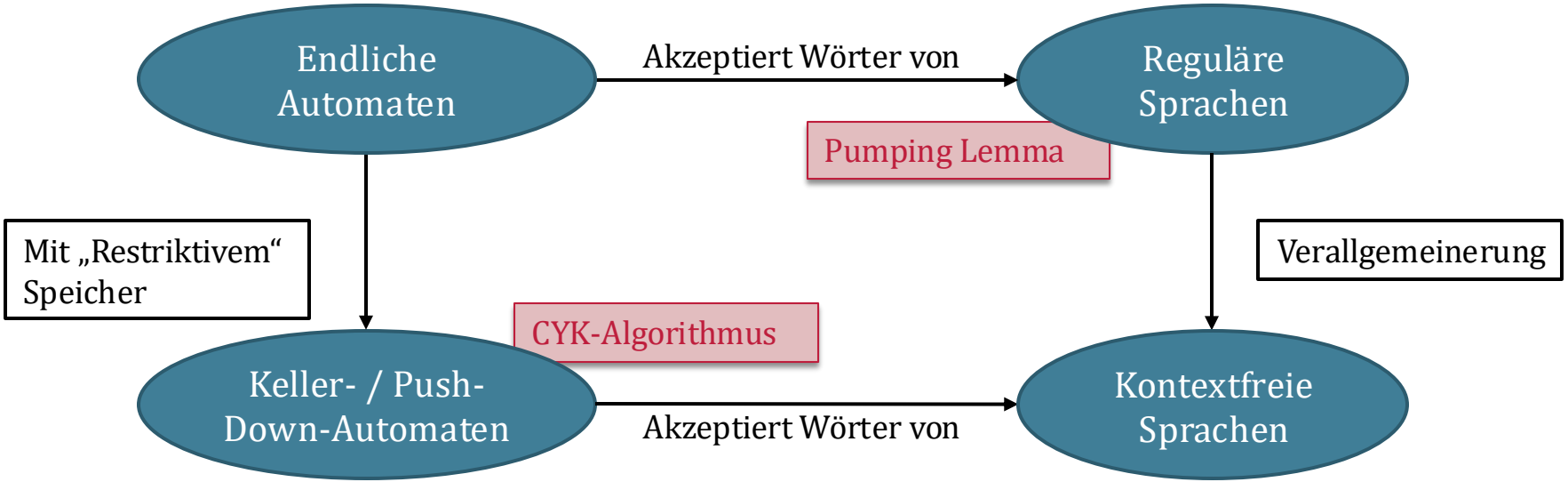
Uhrzeit:
08 Uhr – 10 Uhr

Ort:
TBA

Inhalt:
Hauptsächlich
Vorlesung

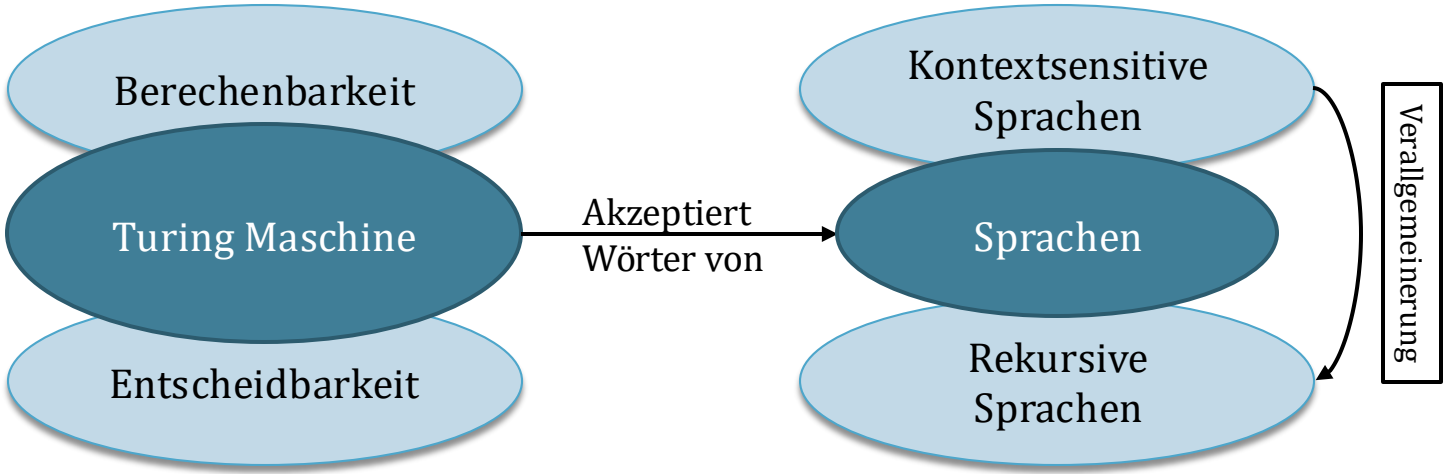
# Überblick

# Theoretische Informatik 1




Generelle Fragen:
Was können die Automaten berechnen? (Berechnungsmächtigkeit)
Besitzen Automaten bestimmte Eigenschaften? (Verifikation)

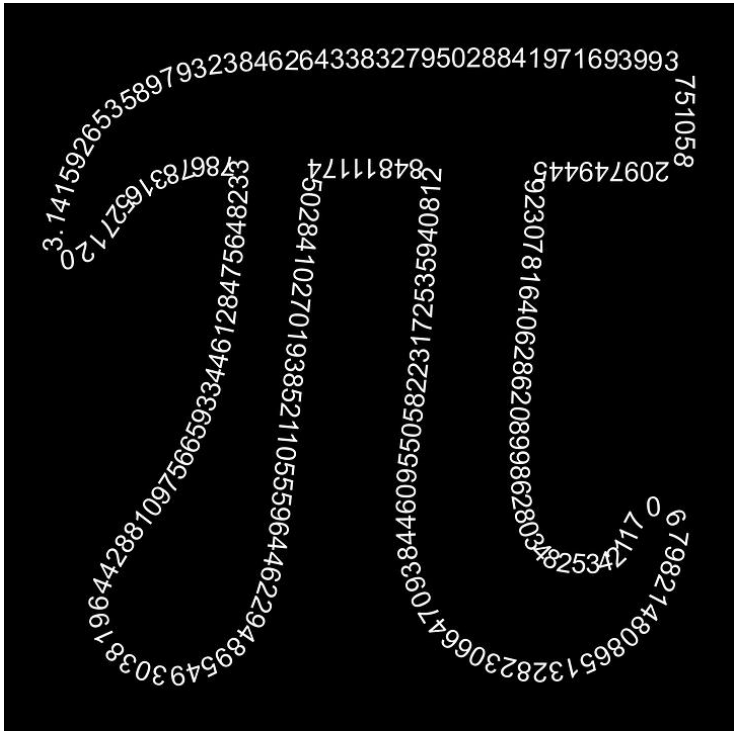
# Theoretische Informatik 2 – Part 1



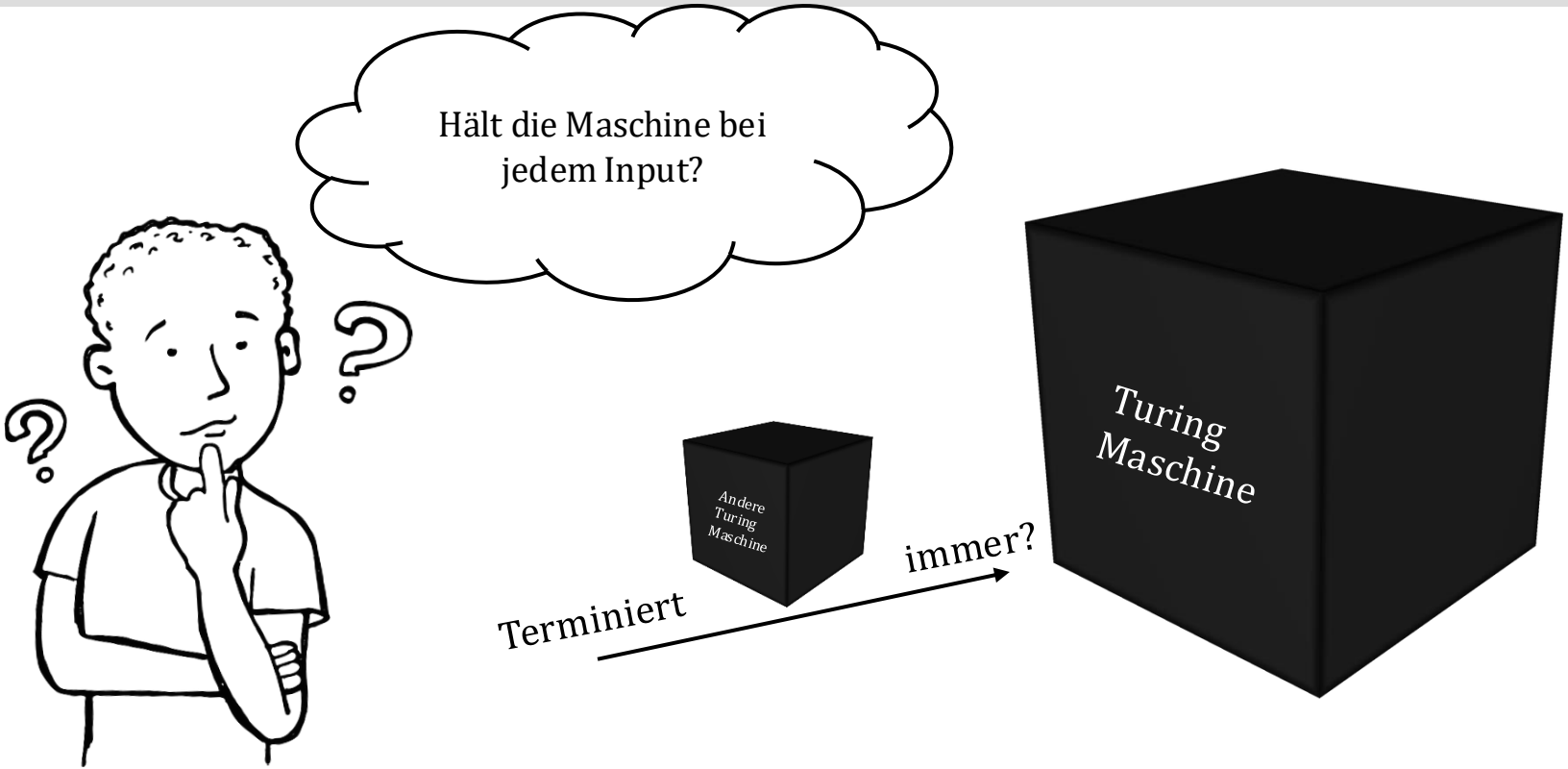
# Berechenbarkeit

Enthält $\pi$ die Zahlenfolge
1597367594816754391
0832418054193875461
3958619854?

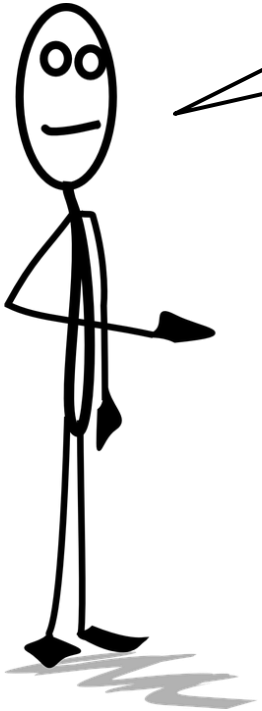
Man kann $\pi$ beliebig approximieren. Aber bis zu welcher Stelle müssen wir das tun? Solange bis wir die Folge gefunden haben? Was passiert, wenn sie nicht existiert?!



# Entscheidbarkeit



# Mächtigkeit



These: Alles, was sich berechnen lässt, kann auch mit einer Turing Maschine berechnet werden.

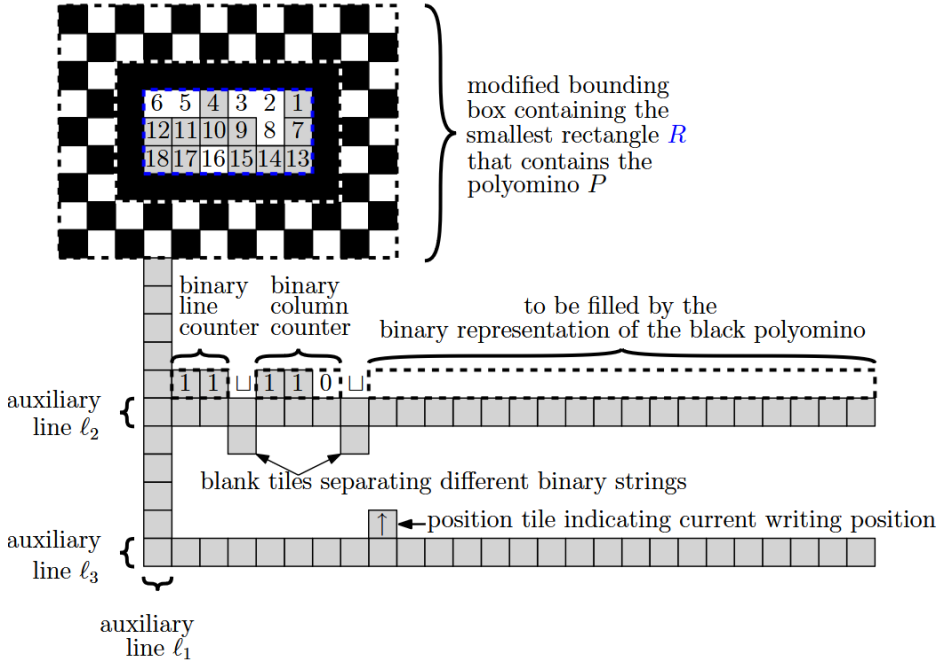
Viele andere Berechnungsmodelle sind genauso mächtig:

- Programmiersprachen: C, Java, Assembler, ...
- Real-RAM-Model
- Lamda-Kalkül
- LaTeX
- PowerPoint, Excel, ...
- Magic the Gathering, Minecraft, Opus Magnum, ...

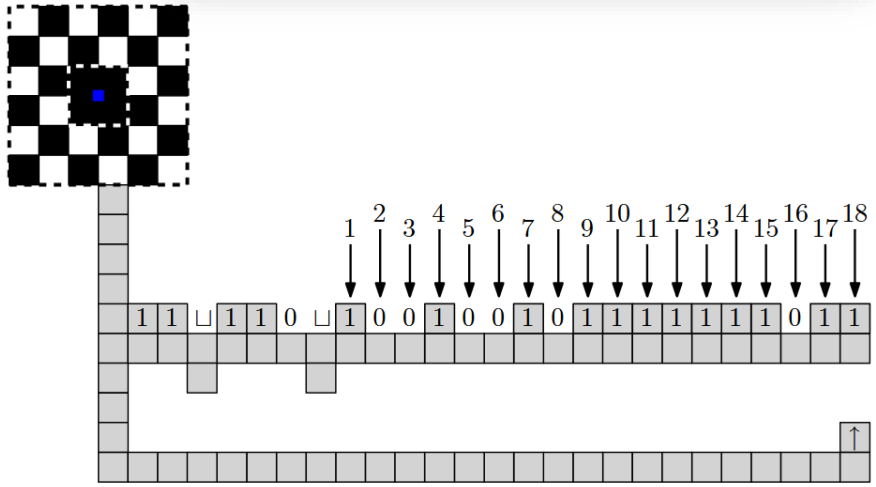
# Turing Maschinen in der Forschung

**CADbots: Algorithmic Aspects of Manipulating Programmable Matter with Finite Automata**

Sándor P. Fekete[1] · Robert Gmyr[2] · Sabrina Hugo[1] · Phillip Keldenich[1] · Christian Scheffer[1] · Arne Schmidt[1]

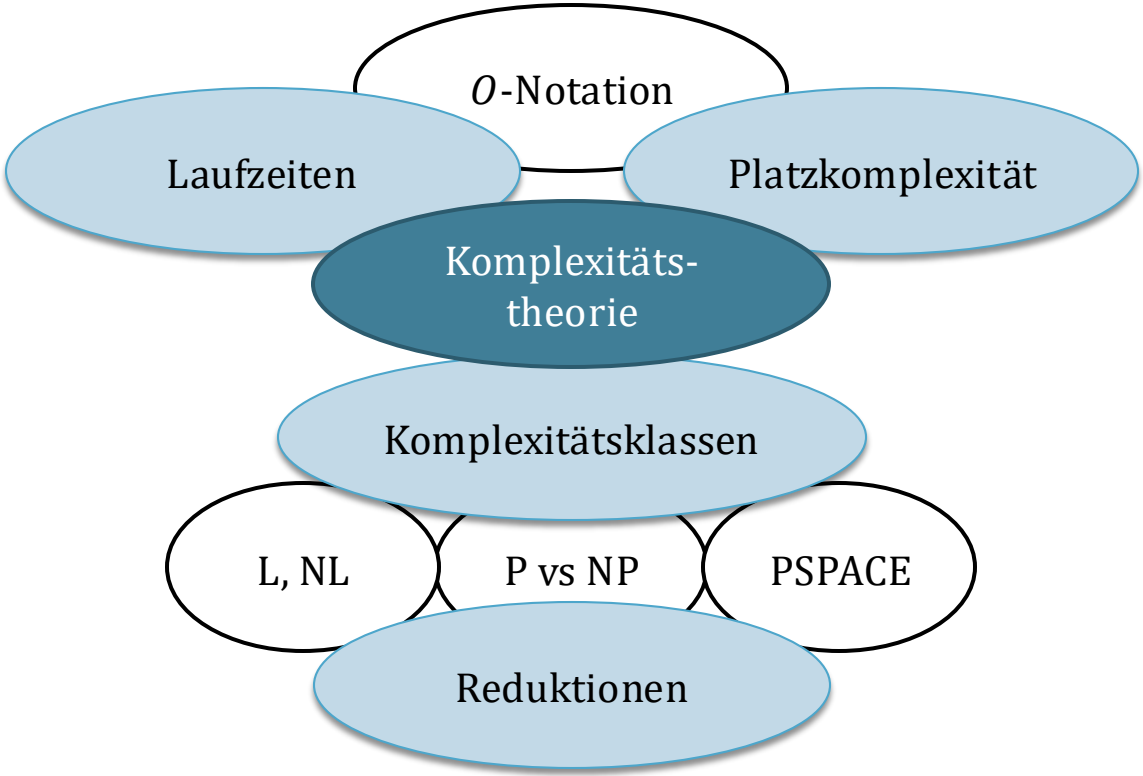



**(a)** State after counting numbers of lines and columns.



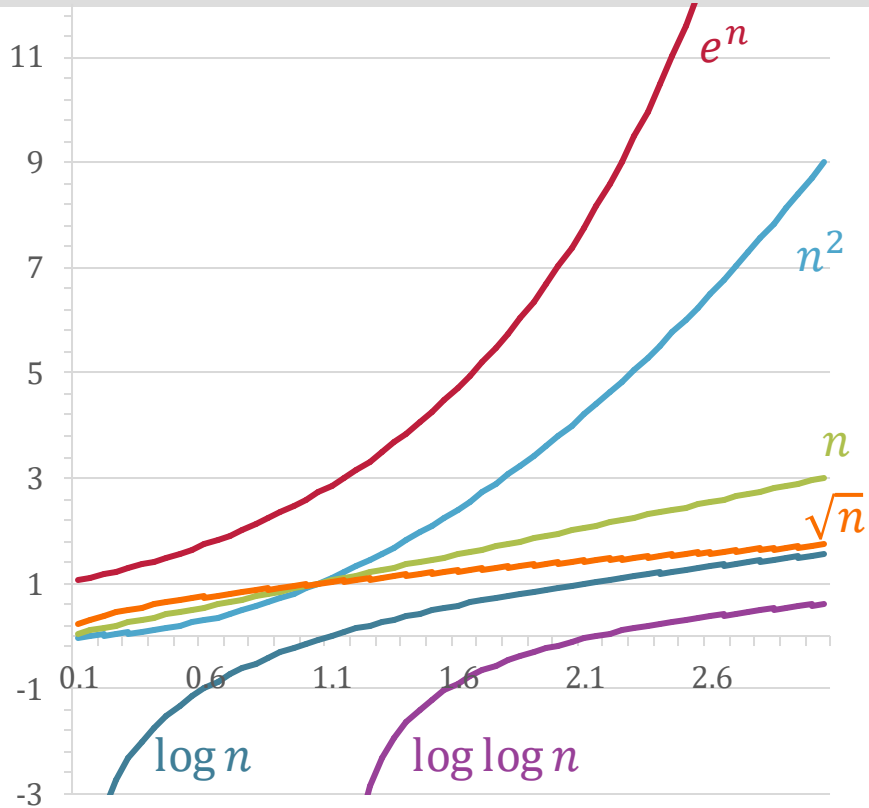

**(c)** Final state of Phase (2) after writing the binary representation of the input polyomino.

# Theoretische Informatik 2 – Part 2

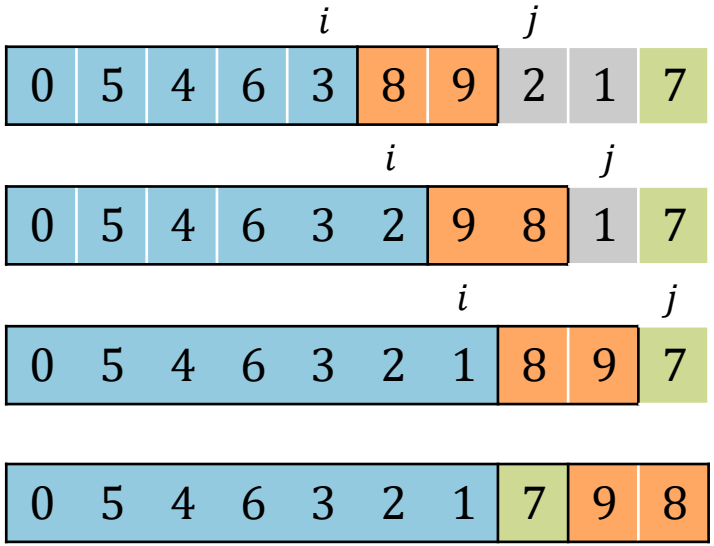


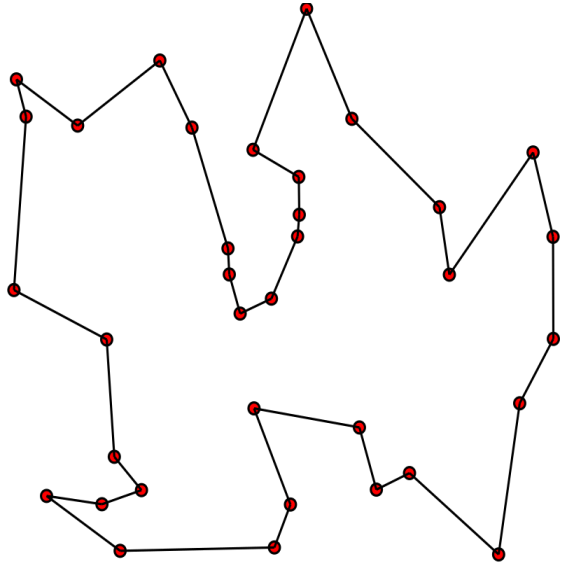
# Laufzeiten


Wie lange benötigt eine Maschine?
Wie viel Platz benötigt eine Maschine?
Wie „schwer“ sind Probleme?


11
9
7
5
3
1
-1
-3
0.1
0.6
1.1
1.6
2.1
2.6
$e^n$
$n^2$
$n$
$\sqrt{n}$
$\log n$
$\log\log n$

# Probleme

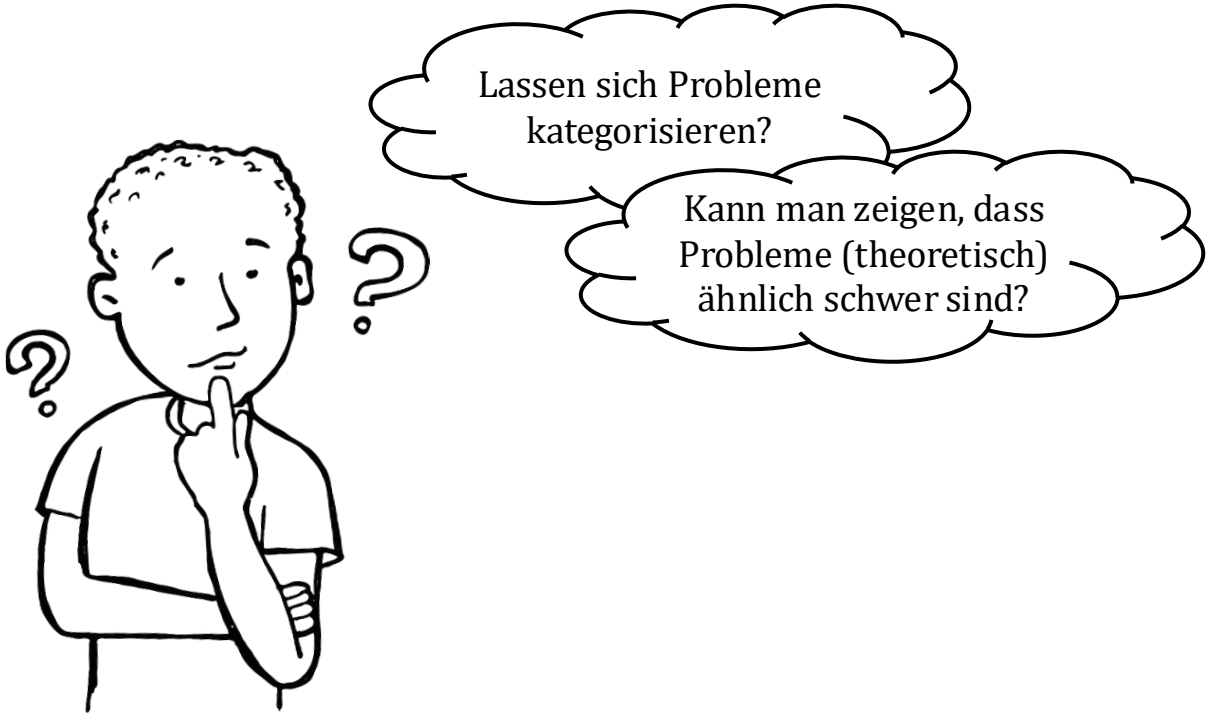


| Sortieren | Best-Case | Average-Case | Worst-Case |
|---|---|---|---|
| Quicksort | $\Theta(n \log n)$ | $\Theta(n \log n)$ | $\Theta(n^2)$ |

**Traveling Salesman Problem**
Vermutlich existiert kein „effizienter“ Algorithmus

# Kategorien


Lassen sich Probleme kategorisieren?
Kann man zeigen, dass Probleme (theoretisch) ähnlich schwer sind?

# Part I – Berechenbarkeit & Entscheidbarkeit

# Kapitel 1 – Kontextsensitive Sprachen

# Grammatik und Sprachen

**Definition 1.1**
Eine **Typ-0-Grammatik** ist eine Grammatik $G = (N, \Sigma, P, S)$, wobei die Produktionen die Form $P \subseteq (N \cup \Sigma)^* \times (N \cup \Sigma)^*$ besitzen.

Eine Grammatik ist **kontextsensitiv** (oder Typ-1), falls für alle Produktionen $\alpha_1 \to \alpha_2$ gilt, dass $|\alpha_1| \leq |\alpha_2|$. Die Produktion $S \to \varepsilon$ ist erlaubt, $S$ darf aber nicht auf der rechten Seite einer Produktion auftreten.

Eine Sprache $\mathcal{L} \subseteq \Sigma^*$ ist **kontextsensitiv**, falls es eine kontextsensitive Grammatik $G$ mit $\mathcal{L}(G) = \mathcal{L}$ gibt. Kontextsensitive Sprachen werden mit CSL abgekürzt.

Eine Sprache $\mathcal{L} \subseteq \Sigma^*$ ist **rekursiv aufzählbar**, falls eine Typ-0-Grammatik $G$ mit $\mathcal{L}(G) = \mathcal{L}$ existiert.

# Beispiel 1.2

Betrachte folgende Sprache

$\mathcal{L} = \{w \in \{a, b, c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w)\}$

Zeige: $\mathcal{L}$ ist kontextsensitiv.

Wir erstellen eine Typ-1-Grammatik $G = (\{S, R, A, B, C\}, \{a, b, c\}, P, S)$, die $\mathcal{L}$ erzeugt:

$S \to \varepsilon \mid R$

$R \to ABC \mid ABCR$

$A \to a$

$B \to b$

$C \to c$

Erstelle Wörter mit gleich vielen a‘s, b‘s und c‘s

$AB \to BA$

$AC \to CA$

$BA \to AB$

$BC \to CB$

$CA \to AC$

$CB \to BC$

Buchstaben dürfen in beliebiger Reihenfolge stehen.

# Unterschied zu kontextfrei

# Wortproblem

**Theorem 1.3**
Für jede kontextsensitive Grammatik $G = (N, \Sigma, P, S)$ kann das Wortproblem für $\mathcal{L}(G)$ („ist $w \in \mathcal{L}(G)$?“) in exponentieller Zeit gelöst werden.

Da durch Ableiten die Satzform nur länger werden kann, können wir einfach alle Satzformen der Länge $\leq |w|$ enumerieren und testen, ob $w$ darunter ist.

Das sind maximal $(|N| + |\Sigma| + 1)^{|w|}$ viele Möglichkeiten.
Für eine fixe Grammatik, kann $|N| + |\Sigma| + 1$ als konstant angesehen werden.

Damit wächst die Laufzeit in etwa wie $2^{|w|}$.

# Beispiel Wortproblem

Betrachte folgende Grammatik $G = (\{S, S', B, C\}, \{a, b, c\}, P, S)$ mit folgenden Produktionsregeln.

1. $S \rightarrow S'$
2. $S' \rightarrow aS'BC \mid aBC$
3. $CB \rightarrow BC$
4. $aB \rightarrow ab$
5. $bB \rightarrow bb$
6. $bC \rightarrow bc$
7. $cC \rightarrow cc$

Zeige oder widerlege folgende Aussagen:

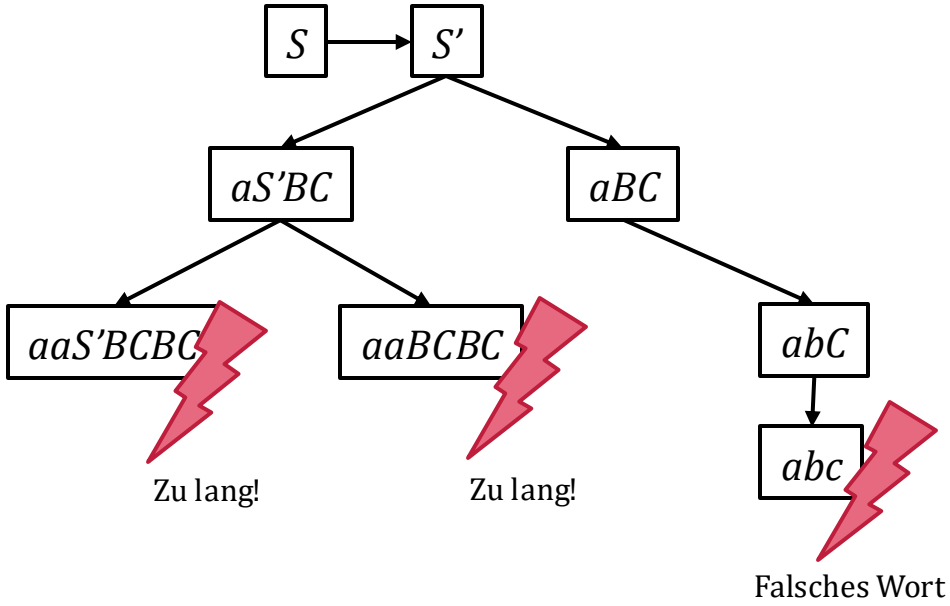
- $aabcc \in \mathcal{L}(G)$
- $aaabbbccc \in \mathcal{L}(G)$

# Beispiel Wortproblem

1. $S \rightarrow S'$
2. $S' \rightarrow aS'BC \mid aBC$
3. $CB \rightarrow BC$
4. $aB \rightarrow ab$
5. $bB \rightarrow bb$
6. $bC \rightarrow bc$
7. $cC \rightarrow cc$

Zeige oder widerlege folgende Aussagen:

- $aabcc \in \mathcal{L}(G)$

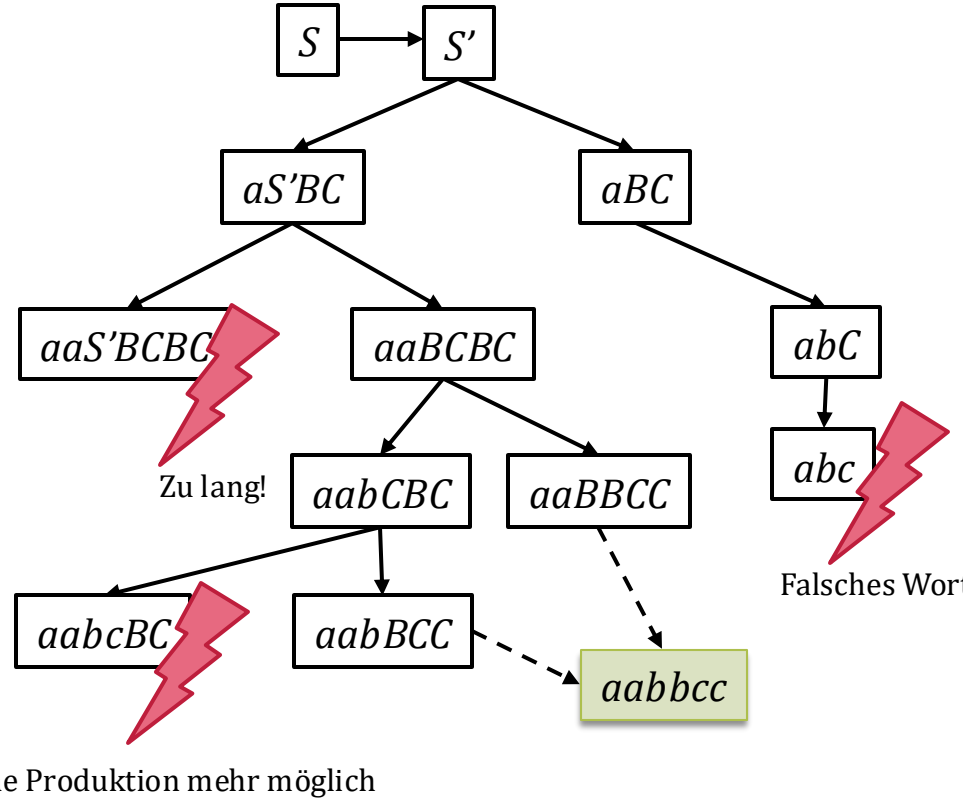


# Beispiel Wortproblem

1. $S \rightarrow S'$
2. $S' \rightarrow aS'BC \mid aBC$
3. $CB \rightarrow BC$
4. $aB \rightarrow ab$
5. $bB \rightarrow bb$
6. $bC \rightarrow bc$
7. $cC \rightarrow cc$

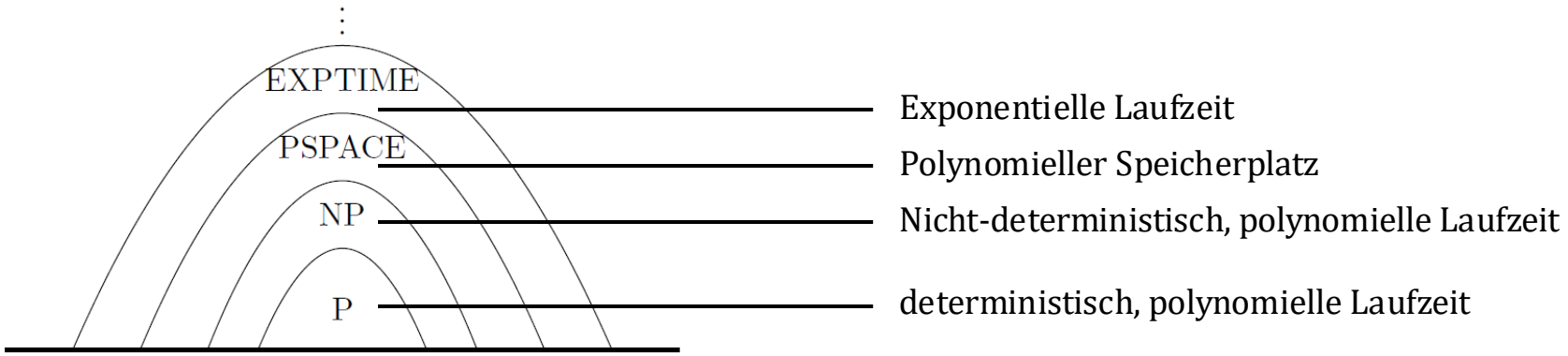
Zeige oder widerlege folgende Aussagen:

- $aabbcc \in \mathcal{L}(G)$



# Ausblick und Bemerkung 1.4

Das Wortproblem lässt sich mit polynomiell viel Speicherplatz (in $|w|$) lösen.




PSPACE ist eine wichtige Komplexitätsklasse für Verifikationsprobleme:

- Erreichbarkeit in Boolschen While-Programmen
- Erreichbarkeit in Multithreaded-Programmen
- Rekonfigurationsprobleme in der (Schwarm-)Robotik

# Nächstes Mal

Endliche und Keller-Automaten können nicht benutzt werden, um solche Wörter zu erkennen.

Wir benötigen also einen mächtigeren Automaten.