Informatik III

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Christian Schindelhauer

Wintersemester 2006/07

15. Vorlesung

14.12.2006

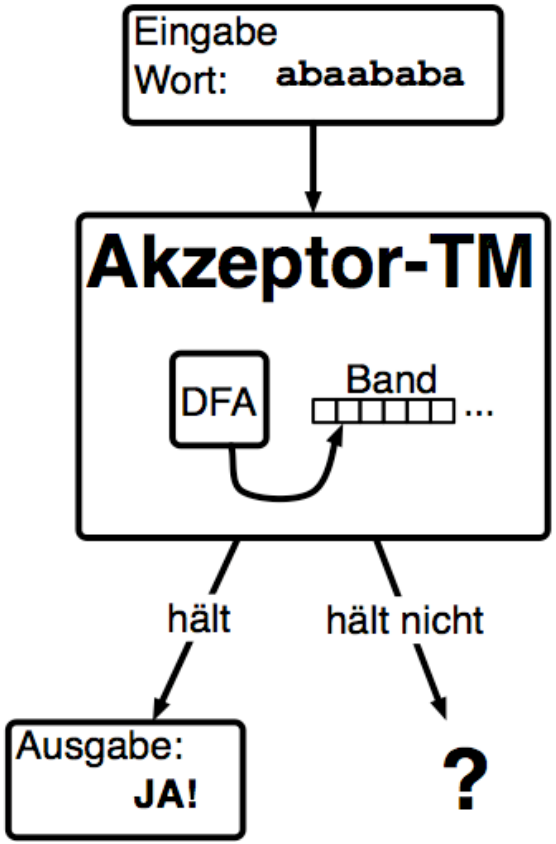
Reduktionen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

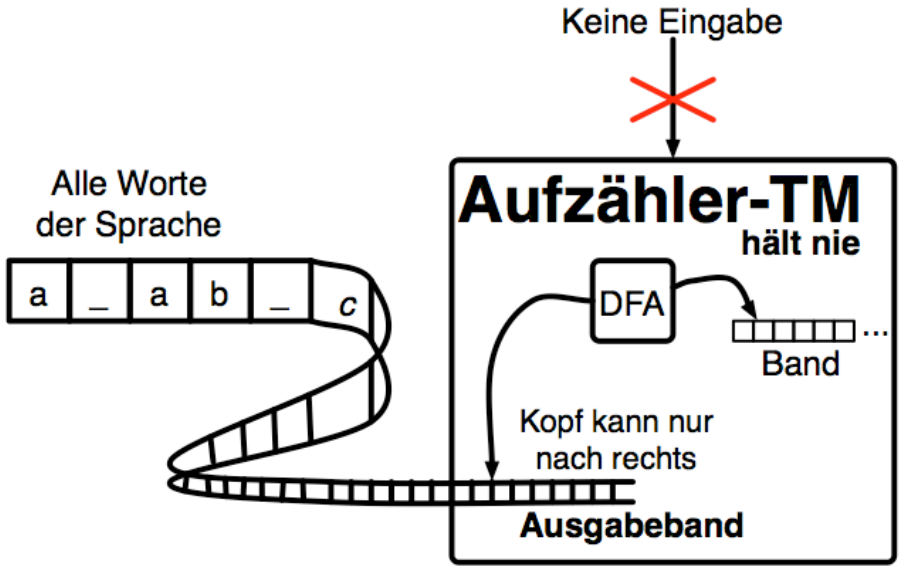
- **Unentscheidbare Probleme**
  - Das Halteproblem
  - Das Leerheitsproblem einer Turingmaschine
- **Ein einfaches nicht berechenbares Problem**
  - Das Postsche Korrespondenzproblem
- **Abbildungsreduktionen**
  - Definition
  - Anwendungen
  - Äquivalenzproblem zweier Turingmaschen
  - Der Satz von Rice
- **Turing-Reduktionen**

Wiederholung: Was ist rekursiv aufzählbar?

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



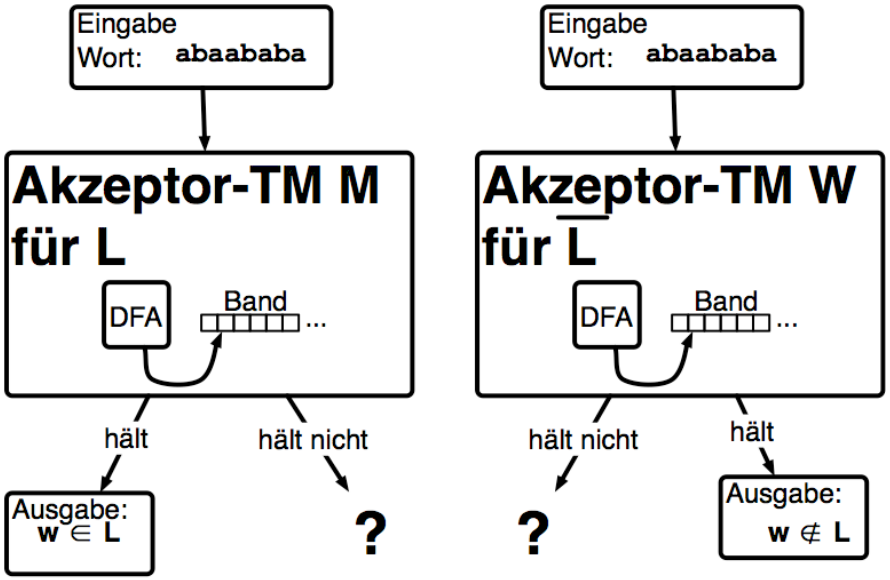
oder

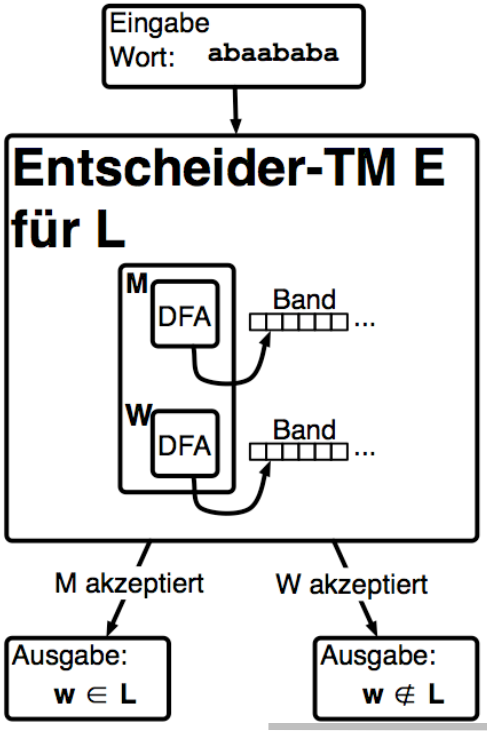


Was ist das Komplement einer rekursiv aufzählbaren Sprache?

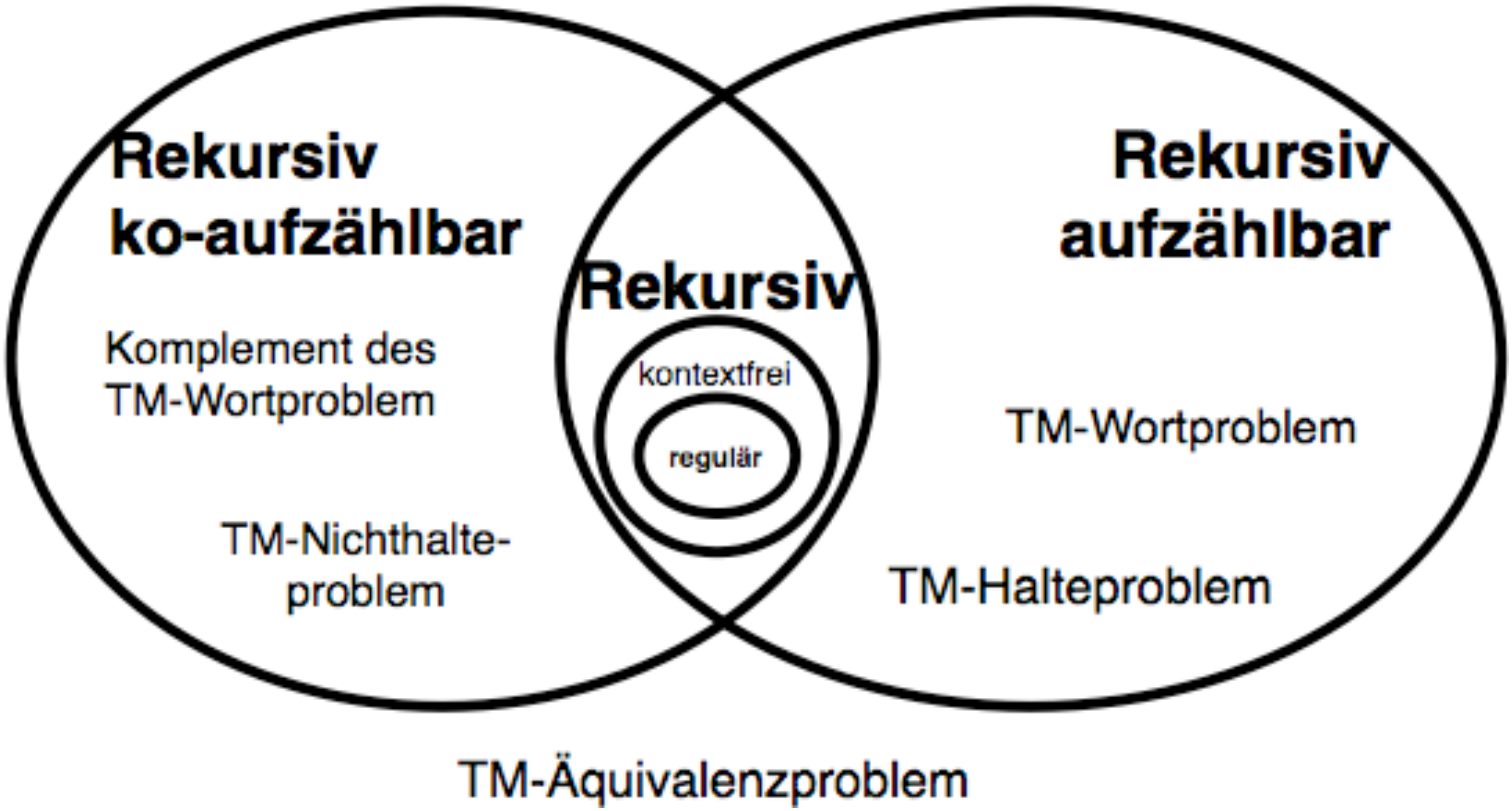
Rekursiv = Aufzählbar + Ko Aufzählbar

- **Definition**
  - Eine Sprache ist **rekursiv ko-aufzählbar**, wenn das Komplement der Menge rekursiv aufzählbar ist.
- **Theorem**
  - Eine Sprache L ist rekursiv, genau dann
    - wenn sie rekursiv aufzählbar
    - und rekursiv ko-aufzählbar ist.
- **Beweis (Rückrichtung)**
  - Betrachte Akzeptor-TM M für L
  - und Akzeptor-TM W für $\Sigma^*\backslash L$
  - Konstruiere TM E für Eingabe x
    - Berechne parallel M(x) und W(x)
  - Falls M(x) zuerst akzeptiert:
    - akzeptiere
  - Falls W(x) zuerst akzeptiert:
    - halte und verwerfe
- **Beweis (Hinrichtung): Übung?**





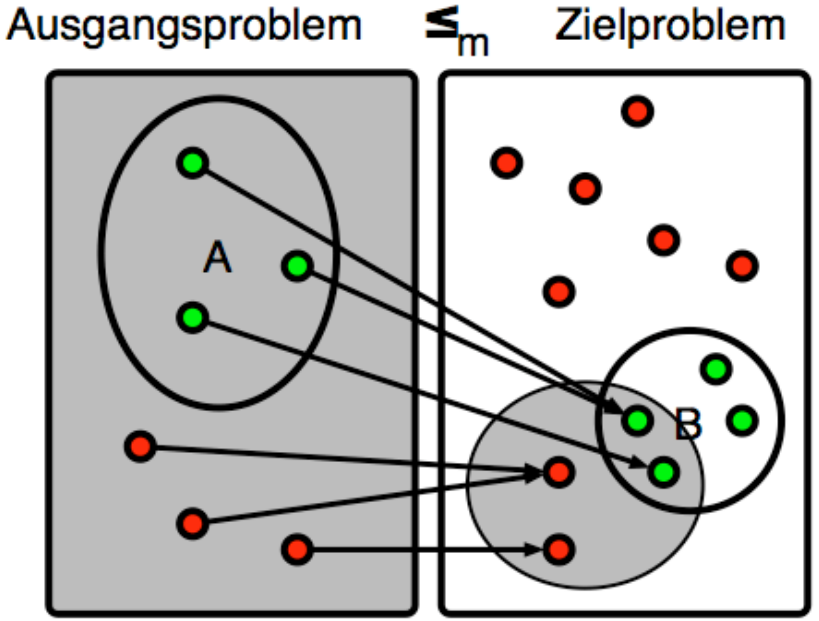
Überblick

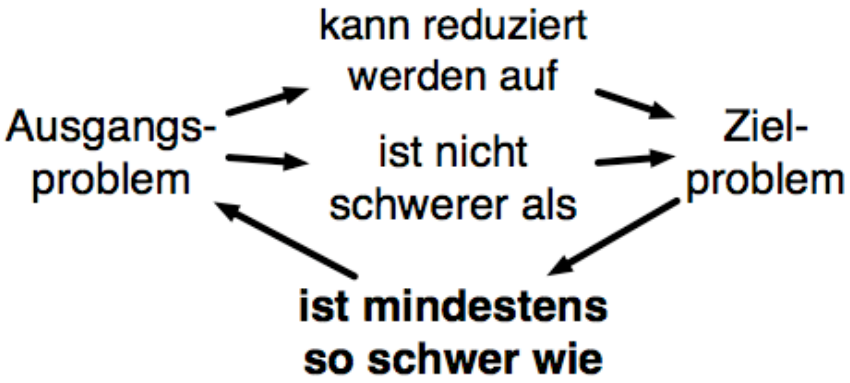

Rekursiv
ko-aufzählbar
Komplement des
TM-Wortproblem
TM-Nichthalte-
problem
Rekursiv
kontextfrei
regulär
Rekursiv
aufzählbar
TM-Wortproblem
TM-Halteproblem
TM-Äquivalenzproblem

Zusammenfassung: Abbildungsreduktionen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Eine Sprache A ist kann durch Abbildung auf eine Sprache B reduziert werden:** $A \leq_m B$**,**
  - falls es eine berechenbare Funktion $f: \Sigma^* \to \Sigma^*$ gibt,
  - so dass für alle w: $w \in A \Leftrightarrow f(w) \in B$
  - Die Funktion f heißt die **Reduktion** von A auf B.
- **Theorem**
  - Falls $A \leq_m B$ und B ist entscheidbar, dann ist A entscheidbar.
- **Korollar**
  - Falls $A \leq_m B$ und A ist nicht entscheidbar, dann ist B auch nicht entscheidbar.
- **Theorem**
  - Falls $A \leq_m B$ und B ist rekursiv aufzählbar, dann ist A rekursiv aufzählbar.
- **Korollar**
  - Falls $A \leq_m B$ und A ist nicht rekursiv aufzählbar, dann ist B nicht rekursiv aufzählbar.





Turing-Reduktionen: Vorüberlegung

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Die Abbildungsreduktion ist ein "schwacher" Reduktionsbegriff**

- **Gedankenexperiment:**
  - Sei B eine entscheidbare Sprache
  - Dann gibt es ein haltendes Programm M, das B entscheidet.
  - Wir benutzen M nun als Unterprogramm
  - Wenn ein Programm M' jetzt ein anderes Problem A entscheiden will,
    - kann es M beliebig häufig als Unterprogramm verwenden
      - weil M immer hält
  - Wenn das Programm M' auf jeder Ausgabe von M immer hält und eine Entscheidung trifft,
    - dann löst es A mit Hilfe von B.
  - A ist entscheidbar,
    - weil man M und M' zu einem Programm verschmelzen kann.

- **Diese Art Reduktion ist nicht durch den Begriff der Abbildungsreduktion abgedeckt.**

Orakel-Turing-Maschinen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition**
  - Ein **Orakel** für eine Sprache B
    - ist eine externe Einheit, welche für ein gegebenes Wort w entscheidet, ob w ein Element von B ist.
  - Eine **Orakel-Turing-Maschine (OTM)**, ist eine modifizierte Turing-Maschine,
    - welche beliebig häufig ein Orakel befragen kann.
      - Hierzu schreibt die **OTM** die Anfrage auf ein separates Orakelband
      - und findet nach dem Schreiben des Endsymbols
      - sofort die Antwort auf dem selben Band.
- **Beobachtung**:
  - Orakel müssen nicht notwendigerweise berechenbar sein.
  - Z.B. mit dem Halteproblem als Orakel lässt sich das TM-Wortproblem lösen:
    1. Frage Halteproblem-Orakel, ob gegebene TM M auf gegebener Eingabe w hält
    2. Falls nein, gib nein aus.
    3. Falls ja, führe M auf Eingabe w aus
    4. Gib Ergebnis von M(w) aus

Der Begriff der Turing-Reduktion

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Definition**
  - Eine Sprache A ist entscheidbar bezüglich Orakel B,
    - falls eine OTM M existiert mit Orakel B,
    - die auf allen Ausgaben von Orakel B hält und
    - die Sprache A entscheidet.

- **Definition**
  - Eine Sprache A ist **Turing-reduzierbar** auf eine Sprache B,

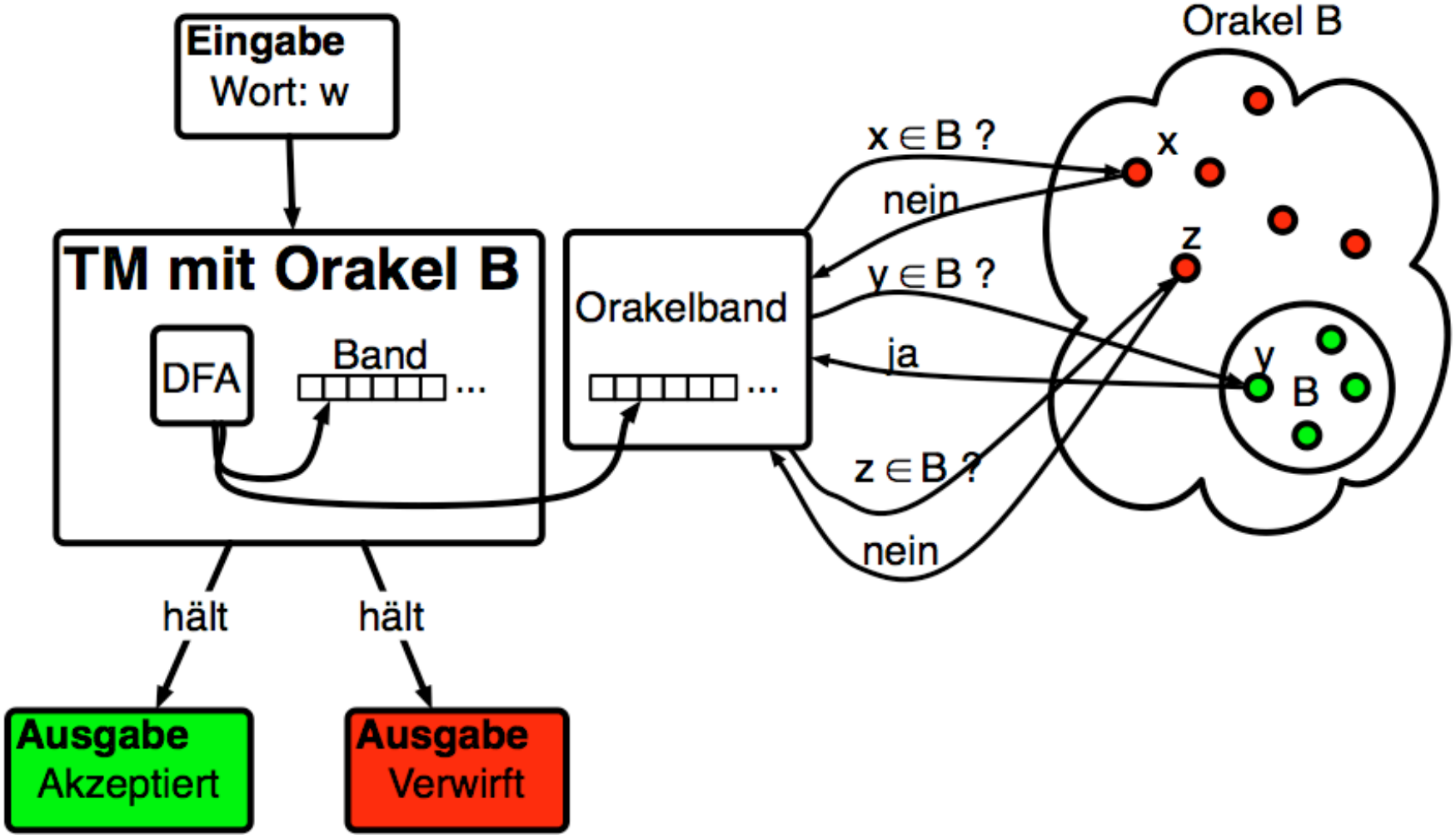
$$\mathbf{A \leq_T B}$$

  - falls A entscheidbar ist bezüglich des Orakels B.

- **Korollar**
  - Aus $A \leq_m B$ folgt $A \leq_T B$.

Die Orakel-Turing-Maschine und ihr Orakel

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



Der Nutzen der Turing-Reduktion

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

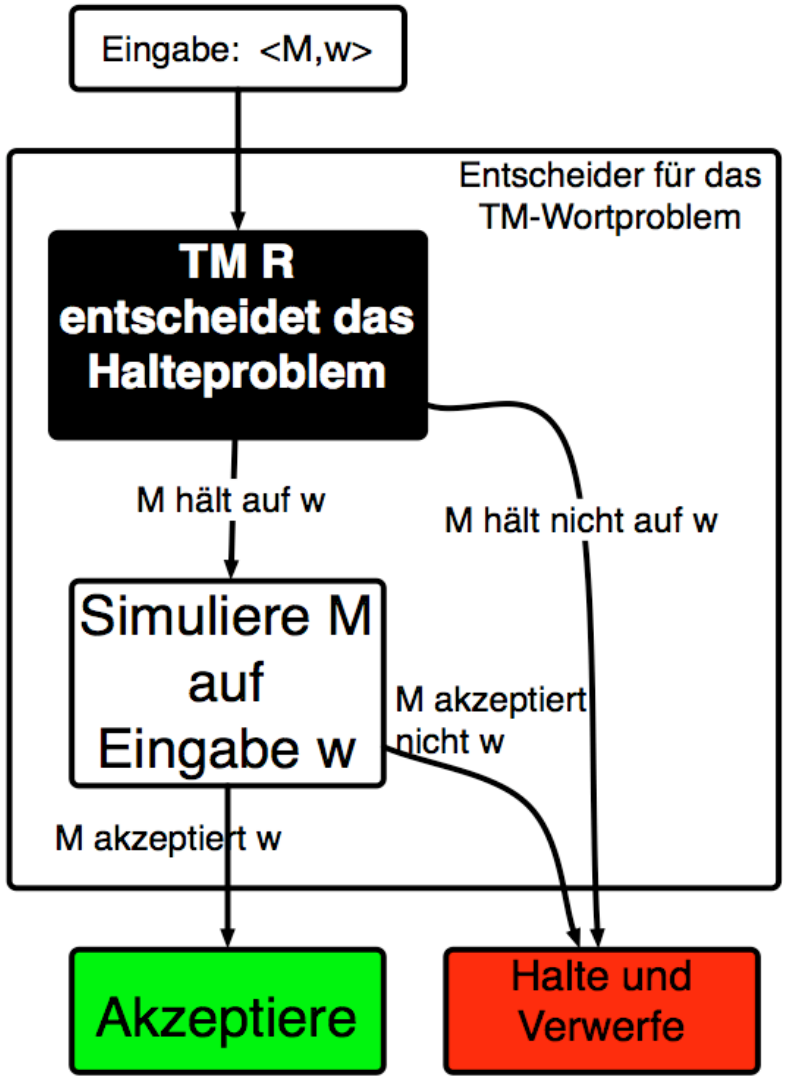
- **Definition**
  - Eine Sprache A ist **Turing-reduzierbar** auf eine Sprache B,
    $$\mathbf{A \leq_T B}$$
  - falls A entscheidbar ist bezüglich des Orakels B.
- **Theorem**
  - Falls $A \leq_T B$ und B ist entscheidbar, dann ist A entscheidbar.
- **Korollar**
  - Falls $A \leq_T B$ und A ist nicht entscheidbar, dann ist B nicht entscheidbar.

- **Achtung:**
  - Solche Sätze gelten nicht für die rekursive Aufzählbarkeit!

Das Halteproblem ist immer noch nicht entscheidbar

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - $A_{TM} \leq_T HALT_{TM}$.
- **Beweis**
  - Konstruiere nun OTM S:
  - S = “Auf Eingabe <M,w>, Kodierung einer TM und Zeichenkette w
    - Frage Halteproblem-Orakel auf Eingabe <M,w>
    - Falls Orakel verwirft, verwirft S
    - Falls Orakel akzeptiert:
      - Simuliere M auf Eingabe w bis M hält
      - Falls M akzeptiert, dann akzeptiert S
      - Falls M verwirft, dann verwirft S”
  - **S Turing-reduziert A_{TM} auf $HALT_{TM}$**
- **Theorem**
  - Aus $A_{TM} \leq_T HALT_{TM}$ und der Nichtentscheidbarkeit von A_{TM} folgt, dass $HALT_{TM}$ nicht entscheidbar ist.



Berechenbarkeitstheorie für Fortgeschrittene

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Das Rekursionstheorem**
  - Selbstreferenz
  - Das Theorem
  - Terminologie
  - Anwendungen
- **Kolmogorov-Komplexität**
  - Optimale Kompression
  - Zufall und Komprimierbarkeit

Thesen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

1. **Lebewesen sind Maschinen.**
2. **Lebewesen können sich reproduzieren.**
3. **Maschinen können sich nicht reproduzieren.**

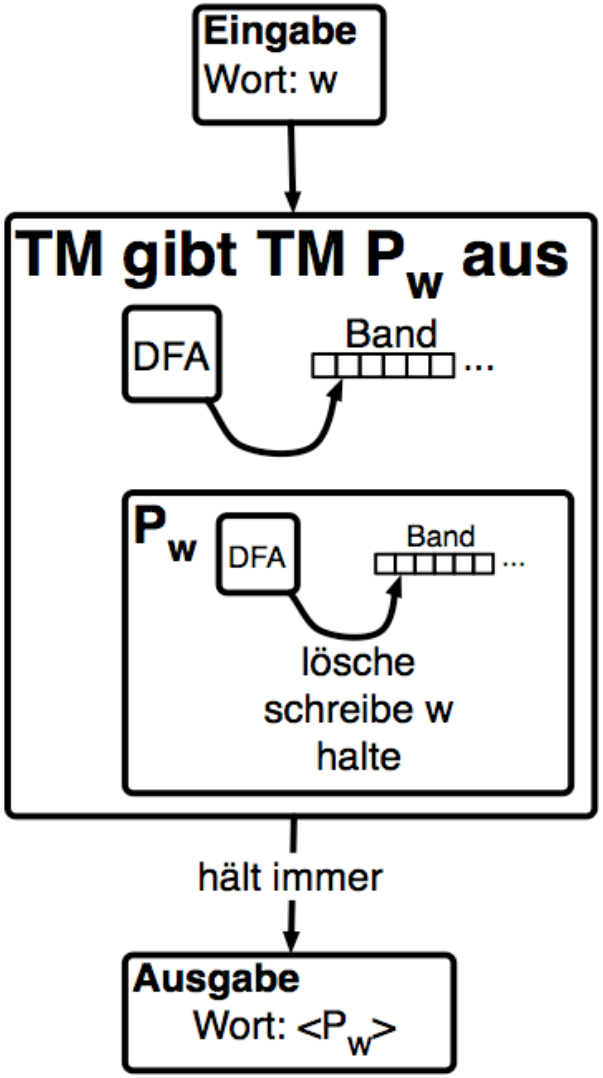
- **Wer werden zeigen: These 3 ist falsch!**

- **Wir zeigen**:
  - **Maschinen können sich vollständig reproduzieren**
  - **Maschinen können ihre eigene Beschreibung verwenden**
  - **Es gibt Maschinen, wenn man ihren Code quadriert, erhält man eine funktionsgleiche Maschine.**
  - **Es gibt Maschinen, wenn man ihre Beschreibung mit 217 multipliziert, erhält man eine funktionsgleiche Maschen.**
  - **Es gibt Maschinen, die in Spiegelschrift notiert die gleiche Funktionalität besitzen.**

Selbstreferenz - Vorbereitung

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Lemma**
  - Es eine berechenbare Funktion $q: \Sigma^* \rightarrow \Sigma^*$,
    - wobei für jedes Wort w
    - q(w) eine Turing-Maschine P_w beschreibt,
    - die w ausgibt und hält.
- **Beweis:**
  - Konstruiere TM Q:
  - Q = “Auf Eingabe w:
    1. Konstruiere P_w= “Für jede Eingabe:
       1. Lösche die Eingabe
       2. Schreibe w auf das Band
       3. Halte.”
    2. Gib <P_w> aus.”



Die SELBST-Maschine

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Die SELBST-Maschine gibt eine Beschreibung von sich selbst aus.**

- **Wir konstruieren zwei TM-Teile A und B**
- **A = $P_{<B>}$,**
- **B = "Auf Eingabe <M>, wobei M ein Anfang eines TM-Programms ist**
  1. Berechne q(<M>)
  2. Kombiniere das Ergebnis mit <M>, um die TM zu vervollständigen
  3. Berechne die Beschreibung der TM und halte"
- **Kombiniere Teile A und B**

- **Resultierende Maschine AB:**

- **A:**
  - Produziert Maschinenbeschreibung von <B> auf das Band
- **B:**
  - Findet Eingabe <B>
  1. B berechnet q(<B>) = $P_{<B>}$ = A
  2. Aus A wird AB
  3. Aus AB wird <AB>

- **Also ist das Ergebnis:**
  - <AB>

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **C:**

```
char*f="char*f=%c%s%c;main(){printf(f,34,f,34,10);}%c";
  main(){printf(f,34,f,34,10);}
```

- **Java:**
  - ```
    class S{public static void main(String[]a){String s="class S{public
    static void main(String[]a){String s=;char
    c=34;System.out.println(s.substring(0,52)+c+s+c+s.substring(52));}}";char
    c=34;System.out.println(s.substring(0,52)+c+s+c+s.substring(52));}}
    ```
- **Pascal:**
  - ```
    const a='const a=';b='begin write(a,#39,a,#39#59#98#61#39,b,#39#59#10,b)
    end.';
    begin write(a,#39,a,#39#59#98#61#39,b,#39#59#10,b) end.
    ```
- **Lisp:**
  - ```
    ((lambda (x)
        (list x (list (quote quote) x)))
    (quote
        (lambda (x)
                (list x (list (quote quote) x)))))
    ```
- **Brainf**k (besser bekannt als Brainfuck):**
  - ```
    >>+++++++>>++>>++++>>++++++++>>+>>++++>>+>>+++>>+>>+++++>>+>>++>>+>>++++++>>++>>++++>>++++++++>>+>>+++++
    >>++>>+>>+>>+++++>>+++++++>>+>>++++++>>+>>+>>+>>++++>>++++++++>>+>>++++++>>+++++++++++++++>>+>>+>>++++++>>+++
    ++++>>+>>++++++>>++>>+>>+>>++++>>+++++++>>+>>++++++>>++++++++++++++++++++++++++++>>+>>+>>+++++>>++++++++>
    >+>>++++++>>++>>+>>+>>+++++>>+>>+++++++>>+>>++>>+>>+++++++>>+>>++>>+>>++++++>>+>>++>>+>>++++++++>>+>>++>>+>
    >++++++>>+>>++>>+>>+++++++>>+>>++>>+>>++++++++>>++>>++++>>++++++++>>+>>+++++>>++++++++>>+>>++++++++>>+>>+>>+>>
    ++++>>+>>++>>+>>+++++++>>+>>+++++>>++++++++>>+>>++++>>+>>+>>++>>+++++>>+>>+++>>+>>++++>>+>>+>>+>>+++++++
    >>+>>++++++>>++++++++++++++++++>>++>>++>>+++>>++>>+>>++>>++++>>++++++>>++>>+++++>>++++++++++++>>+>>++>>
    ++++>>+>>++>>+>>+++++++>>++++++>>+>>+>>++++++>>+>>+++++++>>++>>++++>>++++++++>>++>>++++>>+>>+++++++[<<]>>[
    >++++++[-<<++++++++++>>]<<++..------------------>[-<.>>+<]>[-<+>]>]<<[-[-[-[-[-[-
    [>++>]<+++++++++++++++++++++++++++>]<++>]<++++++++++++++>]<+>]<++>]<<[->.<]<<]
    ```

Die Programmiersprache Brainfuck

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Erfunden von Urban Müller 1993 als Erweiterung von P''**
  - P'' von Corrado Böhm 1964 ist Brainfuck ohne Ein-/Ausgabe-Operationen "," und ".".
- **Daten:**
  - 1 Zeiger
  - 30.000 Zellen mit je einem Byte
- **Programm besteht aus den Befehlen: > < + - . , [ ]**
- **8 Befehle:**
  - > : Erhöhe Zeigerwert um 1. Der Zeiger zeigt im nächsten Schritt auf die nächste rechte Zelle
  - < : Verringere Zeigerwert um 1. Der Zeiger zeigt im nächsten Schritt auf die nächste linke Zelle.
  - + : Erhöhe das Byte an der Zeigerstelle um 1.
  - - : Verringere das Byte an der Zeigerstelle um 1.
  - . : Ausgabe des Byte-Werts an der Zeigerstelle.
  - , : Einlesen eines Byte-Werts mit Speicherung an der Zeigerstelle.
  - [: Überspringe das Programmstück bis direkt nach der passenden Klammer], falls das Byte an der Stelle 0 ist.
  -] : Springe zurück zur passenden Klammerstelle, falls das Byte an der Zeigerstelle nicht 0 ist.
- **Brainfuck ist Turing-vollständig,**
  - wenn man beliebig viele Zellen verwendet (statt 30.000)
- **weil jede TM sich mit Brainfuck darstellen lässt**

Selbst-Reproduktion in DEUTSCH

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Hier nochmal die Konstruktion in der Programmiersprache "DEUTSCH"**

- **P_w entspricht:**
  - "Schreiben Sie "w"".
- **AB entspricht:**
  - Schreiben Sie den folgenden Satz zweimal: Das erste Mal so; das zweite Mal in Anführungszeichen: "Schreiben Sie den folgenden Satz zweimal: Das erste Mal so; das zweite Mal in Anführungszeichen:"
- **Also A =**
  - Schreiben Sie den folgenden Satz zweimal: Das erste Mal so; das zweite Mal in Anführungszeichen:
- **Und B =**
  - "Schreiben Sie den folgenden Satz zweimal: Das erste Mal so; das zweite Mal in Anführungszeichen:"

Das Rekursionstheorem

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Rekursionstheorem**
  - Sei T eine TM, welche die Funktion t: $\Sigma^* \rightarrow \Sigma^*$ berechnet.
  - Dann gibt es eine TM R,
    - welche die Funktion r: $\Sigma^* \rightarrow \Sigma^*$ für alle w berechnet:
    - r(w) = t(<R>,w).
- **Interpretation:**
  - Für die Berechnung einer Funktion kann man eine TM finden, welche ihre eigene Kodierung mitverwenden kann.

- **Achtung:**
  - T “weiß” nicht (unbedingt), wie seine eigene Kodierung aussieht
  - R wird dahingehend konstruiert, dass die Kodierung zur Verfügung steht.

Beweis des Rekursionstheorems

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Rekursionstheorem**
  - Sei T eine TM, welche die Funktion t: $\Sigma^* \rightarrow \Sigma^*$ berechnet.
  - Dann gibt es eine TM R,
    - welche die Funktion r: $\Sigma^* \rightarrow \Sigma^*$ für alle w berechnet:
    - r(w) = t(<R>,w).
- **Beweis**
  - Konstruiere eine TM R aus drei Programmteilen A,B und T (gegeben)
  - A = $P_{<BT>}$ mit der Funktionalität von q',
    - wobei q' eine TM baut,
    - welche die Ausgabe <BT> an eine existierende Bandanschrift w anhängt
  - B = "Lies das Band und wende q' auf den mit einem Sondersymbol getrennten rechten Bandinhalt an
    - Als Ergebnis wird $<P_{<BT>}>$ = <A> rechts angehängt
    - Kombiniere A, B, und T zu einer einzigen TM: <ABT> = <R>
    - Kodiere diese Beschreibung zusammen mit w zu <R,w> auf das Band
    - Übergebe Kontrolle an T"

Anwendungen des Rekursionstheorems

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Computer-Viren**
  - sind reproduzierende Programme
  - welche sich ohne Einfluss eines Benutzers selbst kopieren und ausbreiten.

- **Computer-Viren "leben" das Rekursionstheorem**
  - sie kennen ihre eigene Beschreibung
  - sie sind in der Lage Kopien von sich selbst zu schreiben

Ende der 15. Vorlesung

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Christian Schindelhauer

Wintersemester 2006/07

15. Vorlesung

14.12.2006