TECHNISCHE UNIVERSITÄT DRESDEN

THEORETISCHE INFORMATIK UND LOGIK

1. Vorlesung: Willkommen/Einleitung/Übersicht

Markus Krötzsch

Lehrstuhl Wissensbasierte Systeme

TU Dresden, 9. April 2018

Willkommen zur Vorlesung
Theoretische Informatik und Logik

Raum, Zeit, URL

- **Vorlesungen:**
  Montag, DS2 (9:20–10:50), HSZ/0004 Freitag, DS3 (11:10–12:40), HSZ/0004
- **Keine Vorlesungen:**
  Ausfall: 23.4. (Web Conference, Lyon)
  Pfingstferien: Mo 21.5., Fr 25.5.
- **Vorlesungswebseite**:

  `https://iccl.inf.tu-dresden.de/web/TheoLog2018`

  (Folien, Übungsblätter, Termine, etc.)
- **Quellcode**:

  `https://github.com/mkroetzsch/TheoLog`

  (Fragen, Bug-Reports, Pull-Requests sind willkommen)

Übungen

- **Anmeldung** zu den Übungen über jExam
- **Übungsblätter** jeweils mittwochs für folgende Woche
- **Beginn der Übungen:** 16. April 2018
- **Übungsablauf**, vereinfacht, idealisiert:
  Aufgaben werden zu Hause bearbeitet so gut es geht;
  in der Übung helfen Gruppenleiter/innen bei Fragen und Problemen und zeigen Beispiellösungen

Prüfung und Prüfungsvorbereitung

- schritliche Prüfung (90min) am Ende des Sommersemesters
- prüfungsrelevant:
  kompletter Stoff aus Vorlesung und Übung;
  Wiedergeben (Definieren), Anwenden (Rechnen) und Erklären (Beweisen)
- Modulnote ergibt sich je nach Studiengang
- zur zusätzlichen Vorbereitung gibt es zwei oder drei Repetitorien und eine Probeklausur, jeweils an einem Vorlesungstermin

Motivation

Paris im August 1900

Der 2. Internationale Mathematikerkongress

„Wer von uns würde nicht gern den Schleier lüften, unter dem die Zukunft verborgen liegt, um einen Blick zu werfen auf die bevorstehenden Fortschritte unsrer Wissenschaft und in die Geheimnisse ihrer Entwickelung während der künftigen Jahrhunderte!“

– David Hilbert, Paris, August 1900

Der 2. Internationale Mathematikerkongress

„Wer von uns würde nicht gern den Schleier lüften, unter dem die Zukunft verborgen liegt, um einen Blick zu werfen auf die bevorstehenden Fortschritte unsrer Wissenschaft und in die Geheimnisse ihrer Entwickelung während der künftigen Jahrhunderte!“

– David Hilbert, Paris, August 1900

Hilbert präsentiert eine Liste offener Fragen für die Mathematik des 20. Jahrhunderts:

- 1. Problem: Kontinuumshypothese (und Auswahlaxiom)
- 2. Problem: Widerspruchsfreiheit der Arithmetik
- ...

Hilberts Programm

Aber Hilberts wahres Ziel ist ein neues Verständnis der Mathematik:

> „So unzugänglich diese Probleme uns erscheinen und so ratlos wir zur Zeit ihnen gegenüber stehen – wir haben dennoch die sichere Ueberzeugung, daß ihre Lösung durch eine endliche Anzahl rein logischer Schlüsse gelingen muß.
> [. . .]
> Diese Ueberzeugung von der Lösbarkeit eines jeden mathematischen Problems ist uns ein kräftiger Ansporn während der Arbeit; wir hören in uns den steten Zuruf: Da ist das Problem, suche die Lösung. Du kannst sie durch reines Denken finden; denn in der Mathematik giebt es, kein Ignorabimus*!“
>
> – David Hilbert, Paris, August 1900

*) lat. „wir werden es niemals wissen“

Der Rest ist Geschichte . . .

- 1910–1913: Whitehead und Russel formalisieren in ihrer Principia Mathematica logische Grundlagen der Mathematik

Der Rest ist Geschichte . . .

- 1910–1913: Whitehead und Russel formalisieren in ihrer Principia Mathematica logische Grundlagen der Mathematik
- 1918–1922: Hilbert spezifiziert sein Programm zur widerspruchsfreien Formalisierung der Mathematik

Der Rest ist Geschichte . . .

- 1910–1913: Whitehead und Russel formalisieren in ihrer Principia Mathematica logische Grundlagen der Mathematik
- 1918–1922: Hilbert spezifiziert sein Programm zur widerspruchsfreien Formalisierung der Mathematik
- 1928: Hilbert beschreibt das Entscheidungsproblem der Prädikatenlogik

Der Rest ist Geschichte . . .

- 1910–1913: Whitehead und Russel formalisieren in ihrer Principia Mathematica logische Grundlagen der Mathematik
- 1918–1922: Hilbert spezifiziert sein Programm zur widerspruchsfreien Formalisierung der Mathematik
- 1928: Hilbert beschreibt das Entscheidungsproblem der Prädikatenlogik
- 1929: Gödel beweist seinen Vollständigkeitssatz: „es gibt ein Kalkül, das alle Wahrheiten der Prädikatenlogik endlich beweisen kann“

Der Rest ist Geschichte . . .

- 1910–1913: Whitehead und Russel formalisieren in ihrer Principia Mathematica logische Grundlagen der Mathematik
- 1918–1922: Hilbert spezifiziert sein Programm zur widerspruchsfreien Formalisierung der Mathematik
- 1928: Hilbert beschreibt das Entscheidungsproblem der Prädikatenlogik
- 1929: Gödel beweist seinen Vollständigkeitssatz: „es gibt ein Kalkül, das alle Wahrheiten der Prädikatenlogik endlich beweisen kann“
- 1936: Turing definiert ein universelles Rechenmodell: die Turingmaschine

Der Rest ist Geschichte . . .

- 1910–1913: Whitehead und Russel formalisieren in ihrer Principia Mathematica logische Grundlagen der Mathematik
- 1918–1922: Hilbert spezifiziert sein Programm zur widerspruchsfreien Formalisierung der Mathematik
- 1928: Hilbert beschreibt das Entscheidungsproblem der Prädikatenlogik
- 1929: Gödel beweist seinen Vollständigkeitssatz: „es gibt ein Kalkül, das alle Wahrheiten der Prädikatenlogik endlich beweisen kann“
- 1936: Turing definiert ein universelles Rechenmodell: die Turingmaschine
- 1951: Tarski publiziert ein Verfahren, mit dem alle wahren logischen Aussagen über reelle Zahlen, $+$ und $*$ automatisch durch Computer entschieden werden können

Der Rest ist Geschichte . . .

- 1976: Computerbeweis der Vier-Farben-Problems (Appel & Haken)

Der Rest ist Geschichte . . .

- 1976: Computerbeweis der Vier-Farben-Problems (Appel & Haken)
- 1992: IBM's Supercomputer WHILE-S beweist die Fermatsche Vermutung

Der Rest ist Geschichte . . .

- 1976: Computerbeweis der Vier-Farben-Problems (Appel & Haken)
- 1992: IBM's Supercomputer WHILE-S beweist die Fermatsche Vermutung
- ab 1995: erste Programmierumgebungen mit automatischer Verifikation

Der Rest ist Geschichte . . .

- 1976: Computerbeweis der Vier-Farben-Problems (Appel & Haken)
- 1992: IBM's Supercomputer WHILE-S beweist die Fermatsche Vermutung
- ab 1995: erste Programmierumgebungen mit automatischer Verifikation
- 2001: IBM beweist die Goldbachsche Vermutung

Der Rest ist Geschichte . . .

- 1976: Computerbeweis der Vier-Farben-Problems (Appel & Haken)
- 1992: IBM's Supercomputer WHILE-S beweist die Fermatsche Vermutung
- ab 1995: erste Programmierumgebungen mit automatischer Verifikation
- 2001: IBM beweist die Goldbachsche Vermutung
- 2005: Google beweist die Riemannsche Vermutung

Der Rest ist Geschichte . . .

- 1976: Computerbeweis der Vier-Farben-Problems (Appel & Haken)
- 1992: IBM's Supercomputer WHILE-S beweist die Fermatsche Vermutung
- ab 1995: erste Programmierumgebungen mit automatischer Verifikation
- 2001: IBM beweist die Goldbachsche Vermutung
- 2005: Google beweist die Riemannsche Vermutung
- 2007: „American Mathematical Society" benennt sich um in „Association of Mathematical Programmers"

Der Rest ist Geschichte . . .

- 1976: Computerbeweis der Vier-Farben-Problems (Appel & Haken)
- 1992: IBM's Supercomputer WHILE-S beweist die Fermatsche Vermutung
- ab 1995: erste Programmierumgebungen mit automatischer Verifikation
- 2001: IBM beweist die Goldbachsche Vermutung
- 2005: Google beweist die Riemannsche Vermutung
- 2007: „American Mathematical Society“ benennt sich um in „Association of Mathematical Programmers“
- 2010: Zusammenbruch des Banksystems infolge der Veröffentlichung des Computerbeweises zur Entschlüsselung von Public-Key-Kryptographie

Der Rest ist Geschichte . . .

- 1976: Computerbeweis der Vier-Farben-Problems (Appel & Haken)
- 1992: IBM's Supercomputer WHILE-S beweist die Fermatsche Vermutung
- ab 1995: erste Programmierumgebungen mit automatischer Verifikation
- 2001: IBM beweist die Goldbachsche Vermutung
- 2005: Google beweist die Riemannsche Vermutung
- 2007: „American Mathematical Society“ benennt sich um in „Association of Mathematical Programmers“
- 2010: Zusammenbruch des Banksystems infolge der Veröffentlichung des Computerbeweises zur Entschlüsselung von Public-Key-Kryptographie
- 2013: Einstellung des Studiengangs Mathematik an der TU Dresden (Übertragung der Mathematiklehrer-Ausbildung an die Fakultät Informatik)

So ist es nicht gewesen.*

*) alle Ereignisse ab 1992 sind nie passiert

So ist es nicht gewesen.*

Warum nicht?

*) alle Ereignisse ab 1992 sind nie passiert

So ist es nicht gewesen.*

Warum nicht?

(A) Historischer Zufall – es kam einfach anders

*) alle Ereignisse ab 1992 sind nie passiert

So ist es nicht gewesen.*

Warum nicht?

(A) Historischer Zufall – es kam einfach anders

(B) Die Hardware ist noch nicht so weit

*) alle Ereignisse ab 1992 sind nie passiert

So ist es nicht gewesen.*

Warum nicht?

(A) Historischer Zufall – es kam einfach anders

(B) Die Hardware ist noch nicht so weit

(C) Die Software ist noch nicht so weit

*) alle Ereignisse ab 1992 sind nie passiert

So ist es nicht gewesen.*

Warum nicht?

(A) Historischer Zufall – es kam einfach anders

(B) Die Hardware ist noch nicht so weit

(C) Die Software ist noch nicht so weit

(D) Die beschriebene Entwicklung ist in unserem Universum unmöglich

*) alle Ereignisse ab 1992 sind nie passiert

So ist es nicht gewesen.*

Warum nicht?

(D) Die beschriebene Entwicklung ist in unserem Universum unmöglich

*) alle Ereignisse ab 1992 sind nie passiert

Informatik als Naturwissenschaft

Informatik erforscht, was Computer sind
und welche Probleme man mit ihnen lösen kann.

Informatik als Naturwissenschaft

Informatik erforscht, was Computer sind
und welche Probleme man mit ihnen lösen kann.

Computer = ein System das rechnet (CMOS-Schaltkreise, die Turingmaschine, DNA-Moleküle, ein quantenmechanisches System, das Universum, Minecraft, . . .)

Ziel: universelle Erkenntnisse – nicht nur über die Computertechnologie, die wir gerade nutzen, sondern über die Welt, in der wir leben wie Physik

Methode: Spezifikation von einfachen Regeln, aus denen komplexe Systeme entstehen

anders als Physik, die mit dem System anfängt und dessen „Regeln“ sucht

Kernfragen der theoretischen Informatik

- Was heißt „berechnen“?
- Welche Probleme kann man auf reellen Computern lösen?
- Was wäre wenn wir mächtigere Computer hätten?
- Was macht Rechenprobleme „schwer“ oder „einfach“?
- Sind alternative Rechenmodelle denkbar?
- Welche (mathematischen/physikalischen/biologischen) Systeme können sonst noch rechnen?

Diese finden sich wieder in zahlreichen Teilgebieten (Berechenbarkeit, Automaten, Komplexität, Quantencomputing, logisches Schließen, künstliche Intelligenz, . . .)

Übersicht „TheoLog“

Teil 1: Berechenbarkeit

- Turingmaschinen und andere Modelle
- Entscheidbarkeit und unentscheidbare Probleme

Teil 2: Komplexität

- Zeit und Raum
- P, NP und PSpace

Teil 3: Prädikatenlogik

- Syntax und Semantik
- Modell-Checking (Anwendung: Datenbanken)
- Logisches Schließen mit Resolution

Teil 4: Logik + Berechnung

- Logik und formale Sprachen
- Gödel
- Weitere Themen . . .

Literatur: Lehrbücher

Der Vorlesungsstoff gehört zu fast jeder Informatikausbildung. Es gibt viele Lehrmaterialien und eine weitgehend einheitliche Notation.

- Uwe Schöning: **Theoretische Informatik – kurz gefasst.** Spektrum Akademischer Verlag
  klassischer deutschsprachiger Text
- Michael Sipser: **Introduction to the Theory of Computation.** Cengage Learning
  Standardtext zu Sprachen und Berechnungskomplexität
- Christopher Moore, Stephan Meterns: **The Nature of Computation.** Oxford University Press
  moderner Text zu Komplexität und Berechnung, weniger formell

Zu speziellen Themen der Logik werden wir bei Gelegenheit noch Literaturangaben machen.

Literatur: Mehr Bücher

Teile des Stoffs der Vorlesung werden auch in vielen, zum Teil recht kurzweiligen Büchern weniger formal behandelt, z.B.:

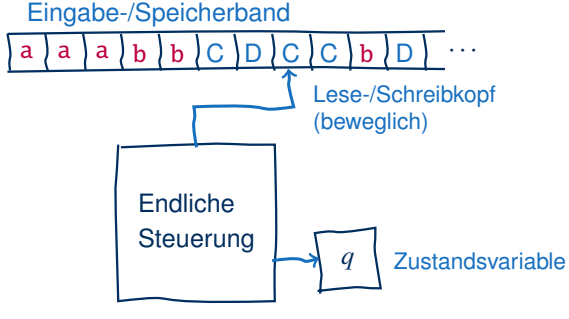
- Apostolos Doxiadis, Christos Papadimitriou:
  Logicomix: An Epic Search for Truth. Bloomsbury
  Graphic Novel, inspiriert von Russels Leben und der Geschichte der Logik
- Scott Aaronson: **Quantum Computing Since Democritus.** Cambridge
  eher informeller Text mit interessanten Denkanstößen
- Douglas Hofstadter:
  Gödel, Escher, Bach: An Eternal Golden Braid. Basic Books
  der Klassiker

Wiederholung: Turingmaschinen

Alan Turing (5 Jahre alt)

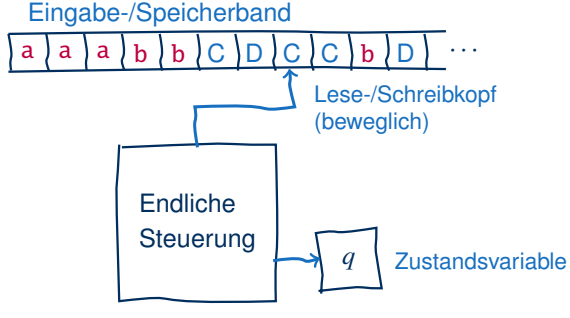
Turingmaschinen – informell

Schematische Darstellung:



Turingmaschinen – informell

Schematische Darstellung:



Übergangsfunktion:

- Eingabe: aktueller Zustand, gelesenes Zeichen
- Ausgabe: neuer Zustand, geschriebenes Zeichen, Änderung Lese-/Schreibadresse ($\hat{=}$ Bewegung Lese-/Schreibkopf)

Definition DTM

Eine deterministische Turingmaschine (DTM) ist ein Tupel $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$ mit den folgenden Bestandteilen:

- Q: endliche Menge von Zuständen
- Σ: Eingabealphabet
- Γ: Arbeitsalphabet mit $\Gamma \supseteq \Sigma \cup \{\sqcup\}$
- δ: Übergangsfunktion, eine partielle Funktion

$$Q \times \Gamma \to Q \times \Gamma \times \{L, R, N\}$$

- q_0: Startzustand $q_0 \in Q$
- F: Menge von akzeptierenden Endzuständen $F \subseteq Q$

Dabei bedeutet $\delta(q, a) = \langle p, b, D \rangle$:

„Liest die TM in Zustand q unter dem Lese-/Schreibkopf ein a, dann wechselt sie zu Zustand p, überschreibt das a mit b und verschiebt den Lese-/Schreibkopf gemäß $D \in \{L, R, N\}$ (nach links, nach rechts, gar nicht).“

Definition NTM

Eine nichtdeterministische Turingmaschine (NTM) ist ein Tupel $\mathcal{M} = \langle Q, \Sigma, \Gamma, \delta, q_0, F \rangle$ mit den folgenden Bestandteilen:

- Q: endliche Menge von Zuständen
- Σ: Eingabealphabet
- Γ: Arbeitsalphabet mit $\Gamma \supseteq \Sigma \cup \{\sqcup\}$
- δ: Übergangsfunktion, eine totale Funktion

  $$Q \times \Gamma \to 2^{Q \times \Gamma \times \{L,R,N\}}$$

  wobei $2^{Q \times \Gamma \times \{L,R,N\}}$ die Potenzmenge von $Q \times \Gamma \times \{L, R, N\}$ ist
- q_0: Startzustand $q_0 \in Q$
- F: Menge von akzeptierenden Endzuständen $F \subseteq Q$

Dabei bedeutet $\langle p, b, D \rangle \in \delta(q, a)$:

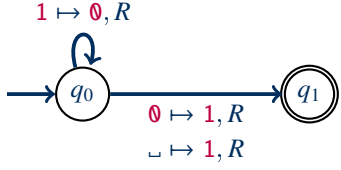
„Liest die TM in Zustand q unter dem Lese-/Schreibkopf ein a, dann kann sie zu Zustand p wechseln, a mit b überschreiben und den Lese-/Schreibkopf gemäß $D \in \{L, R, N\}$ verschieben."

Beispiel

Wir können TMs in Diagrammen darstellen:

Ein Pfeil $s_1 \mapsto s_2, D$ von q_1 nach q_2 bedeutet
$\delta(q_1, s_1) = \langle q_2, s_2, D \rangle$ (DTM) bzw. $\langle q_2, s_2, D \rangle \in \delta(q_1, s_1)$ (NTM)

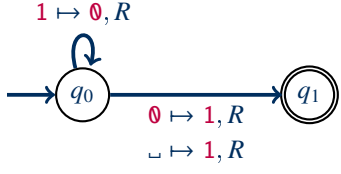
Beispiel:



NTM oder DTM? Was tut diese Turingmaschine?

TM: Beispiel Abarbeitung

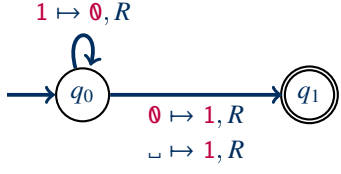
TMs gehen schrittweise von einer Konfiguration in die nächste über:



Eingabe: 1101

TM: Beispiel Abarbeitung

TMs gehen schrittweise von einer Konfiguration in die nächste über:

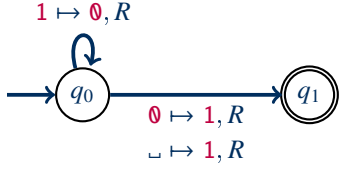


Eingabe: 1101

$q_0 1101$

TM: Beispiel Abarbeitung

TMs gehen schrittweise von einer Konfiguration in die nächste über:

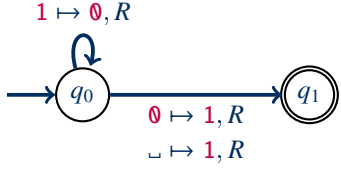


Eingabe: 1101

$q_0 1101 \vdash 0 q_0 101$

TM: Beispiel Abarbeitung

TMs gehen schrittweise von einer Konfiguration in die nächste über:

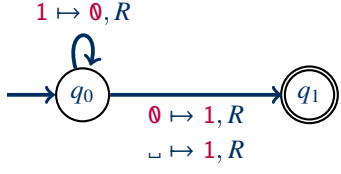


Eingabe: 1101

$q_0 1101 \vdash 0 q_0 101 \vdash 00 q_0 01$

TM: Beispiel Abarbeitung

TMs gehen schrittweise von einer Konfiguration in die nächste über:

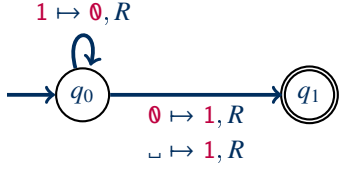


Eingabe: 1101

$q_0 1101 \vdash 0 q_0 101 \vdash 00 q_0 01 \vdash 001 q_1 1$

TM: Beispiel Abarbeitung

TMs gehen schrittweise von einer Konfiguration in die nächste über:



Eingabe: 1101

$q_0 1101 \vdash 0q_0 101 \vdash 00q_0 01 \vdash 001q_1 1$

Ausgabe: 0011

Wiederholung Grundbegriffe

Konfiguration: der „Gesamtzustand“ einer TM, bestehend aus Zustand, Bandinhalt und Position des Lese-/Schreibkopfs;

geschrieben als Wort (Bandinhalt), in dem der Zustand vor der Position des Kopfes eingefügt ist

Übergangsrelation: Beziehung zwischen zwei Konfigurationen wenn die TM von der ersten in die zweite übergehen kann (deterministisch oder nichtdeterministisch); geschrieben als $\vdash$

Lauf: mögliche Abfolge von Konfigurationen einer TM, beginnend mit der Startkonfiguration; kann endlich oder unendlich sein

Halten: Ende der Abarbeitung, wenn die TM in einer Konfiguration keinen Übergang mehr zur Verfügung hat

Was ist das Ergebnis einer TM-Berechnung?

Es gibt zwei wesentliche Arten, um DTMs zu benutzen:

(1) Transducer: Ausgabe der TM ist der Inhalt des Bandes, wenn sie hält, oder undefiniert, wenn sie nicht hält (partielle Funktion); Endzustände spielen keine Rolle und können weggelassen werden

(2) Entscheider: Ausgabe der TM hat nur zwei Werte: die TM „akzeptiert“, wenn sie in einem Endzustand hält und sie „verwirft“ wenn sie in einem Nicht-Endzustand oder gar nicht hält; Bandinhalt beim Halten spielt keine Rolle und kann ignoriert werden

Wir werden NTMs nur als Entscheider verwenden: in diesem Fall reicht es, wenn mindestens ein möglicher Lauf akzeptiert.

NTM, DTM und die Church-Turing-These

Wir wissen: (aus „Formale Systeme“, Winter 2017/2018, 19. Vorlesung)

Satz: Jede NTM kann durch eine DTM simuliert werden. Insbesondere akzeptieren deterministische und nichtdeterministische Turingmaschinen die selben Sprachen.

NTM, DTM und die Church-Turing-These

Wir wissen: (aus „Formale Systeme“, Winter 2017/2018, 19. Vorlesung)

Satz: Jede NTM kann durch eine DTM simuliert werden. Insbesondere akzeptieren deterministische und nichtdeterministische Turingmaschinen die selben Sprachen.

Allgemeiner gilt: kein bekanntes Berechnungsmodell kann mehr berechnen als TMs

Church-Turing-These: Jede (partielle) Funktion, die intuitiv als berechenbar angesehen werden kann, ist auch mit einer Turingmaschine berechenbar.

- **Lesart 1:** Vorschlag einer mathematischen Definition der intuitiven Idee von Berechenbarkeit
- **Lesart 2:** „Naturgesetz“ über die Möglichkeiten und Grenzen des Rechnens an sich

Zusammenfassung und Ausblick

Die Theorie der Informatik untersucht Systeme im Hinblick auf ihre Fähigkeit zur Informationsverarbeitung (Berechnung)

Grundbegriffe: Turingmaschine (det./nichtdet.), Konfiguration, Lauf, Akzeptanz

Church-Turing-These: „Alle Computer sind gleich“

Was erwartet uns als nächstes?

- Probleme
- Paradoxien
- Phenomenal große Zahlen

Bildrechte

Folie 7: Fotografie von 1900, gemeinfrei
Folie 8: Fotografie von 1912, gemeinfrei
Folie 18: Fotografie von 1917, gemeinfrei