TECHNISCHE UNIVERSITÄT DRESDEN

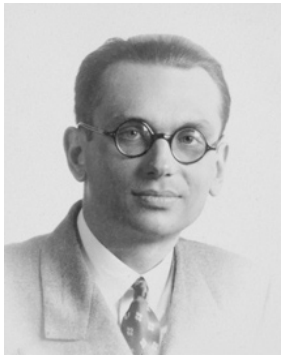
Fakultät Informatik, Institut für Künstliche Intelligenz, Professur Computational Logic

THEORETISCHE INFORMATIK UND LOGIK

24. Vorlesung: Gödel, Turing und der ganze Rest

Hannes Straß

TU Dresden, 11. Juli 2022



Kurt Gödel

Der 1. Gödelsche Unvollständigkeitssatz

Satz (Gödel, 1931): Jedes konsistente formale System, in dem eine gewisse Menge elementarer Arithmetik dargestellt werden kann, ist unvollständig in Bezug auf die Beweisbarkeit von Sätzen der elementaren Arithmetik.

Relevante Begriffe:

- Formales System: Ein implementierbares Verfahren, mit dem man Theoreme endlich beweisen kann.
- Konsistent: Man kann niemals eine Aussage und ihr Gegenteil beweisen.
- Gewisse Menge Arithmetik: Kodierung konkreter natürlicher Zahlen und deren korrekte Addition, Subtraktion, Multiplikation und Vergleich.
- Unvollständig: Es gibt Sätze, die weder bewiesen noch widerlegt werden können.

Beispiele

Beispiel: Natürliche Zahlen und einfache Rechenregeln können mit einer prädikatenlogischen Theorie definiert werden. Mit Resolution kann man daraus korrekte neue Schlüsse ziehen. Laut Gödels erstem Satz kann man auf diese Art aber niemals alle wahren Aussagen der Arithmetik beweisen, außer wenn die Theorie widersprüchlich ist.

Keine prädikatenlogische Theorie kann die elementare Arithmetik vollständig beschreiben.

Beispiel: Die moderne Mathematik basiert auf der Mengenlehre von Zermelo-Fraenkel unter Hinzunahme des Auswahlaxioms. Dieses formale System heißt ZFC. Es ist klar definiert, was ein korrekter mathematischer Beweis in ZFC ist. Laut Gödels erstem Satz gibt es also wahre Aussagen über elementare Arithmetik, die nicht in ZFC bewiesen werden können.

Gödels 2. Unvollständigkeitssatz

Gödel kam direkt zu einer weiteren Schlussfolgerung:

Satz: Jedes konsistente formale System, in dem eine gewisse Menge elementarer Arithmetik dargestellt werden kann, kann nicht seine eigene Konsistenz beweisen.

Auch hier gibt es einige vage Punkte:

- Was ist „eine gewisse Menge elementarer Arithmetik“?
- Was genau bedeutet „seine eigene Konsistenz beweisen“?

Konsistenz beweisen

Konsistenz bedeutet formal:

„Für alle Sätze F gilt: Es gibt keinen Beweis für F oder es gibt keinen Beweis für $\neg F$.“

- Um das ausdrücken zu können, muss das System über die im eigenen System möglichen Beweise reflektieren können.
- Diese Idee ist verwandt mit der Intuition, dass man in Arithmetik universelle Turingmaschinen kodieren kann: Die Beweise eines Systems sind letztlich die akzeptierenden Läufe einer TM.
- Man benötigt dazu etwas mehr Arithmetik als für den Beweis des 1. Unvollständigkeitssatzes. (Details sparen wir uns.)

Beweis des 2. Unvollständigkeitssatzes

Gödels Argument für seinen 1. Unvollständigkeitssatz (vereinfacht):

Sei $\mathcal{S}$ ein korrektes, konsistentes Formales System. Definiere Formel F mittels:

„F ist in System $\mathcal{S}$ nicht beweisbar.“

- Wenn F beweisbar wäre, dann wäre F wahr und daher nicht beweisbar – Widerspruch.
- Also ist F nicht beweisbar, und damit wahr. □

Es stellt sich heraus: Mit einer „gewissen Menge Arithmetik“ kann man diese Argumentation komplett im System $\mathcal{S}$ selbst darstellen.

Warum ist diese Argumentation dann nicht schon ein Beweis für F?

Weil wir die Annahme verwenden, dass das System $\mathcal{S}$ konsistent ist.

- Wenn wir dies beweisen könnten, so wäre auch F beweisbar.
- Aber laut dem 1. Unvollständigkeitssatz ist F nicht beweisbar.

Also kann die Konsistenz des Systems $\mathcal{S}$ nicht beweisbar sein. □

Beispiele

Beispiel: Es gibt keinen elementaren arithmetischen Beweis für die Widerspruchsfreiheit der Peano-Arithmetik. Man kann deren Konsistenz aber leicht im stärkeren System ZFC beweisen.

Beispiel: Es ist nicht möglich, die Widerspruchsfreiheit der modernen Mathematik (ZFC) aus sich selbst heraus zu beweisen. Auch das kann man allerdings in mächtigeren Systemen erreichen.

Anmerkung: Dadurch wird die Mathematik nicht in eine Sinnkrise gestürzt. Selbst wenn man die Konsistenz von ZFC in ZFC beweisen könnte, wäre dies sicherlich kein starkes Argument für ZFC. Mathematische Systeme erhalten ihre Bedeutung niemals aus sich selbst heraus.

Der universelle elektronische Mathematiker

Oder: „Hilberts Programm als Turingmaschine“

Skizze einer Turingmaschine:

- Bilde systematisch der Reihe nach alle möglichen Beweise der modernen Mathematik (System ZFC).
- Halte, sobald ein Beweis für die Aussage „$0 = 1$“ auftaucht.

Hält diese Turingmaschine?

- Ja, falls ZFC inkonsistent ist.
- Nein, falls ZFC konsistent ist.

$\rightsquigarrow$ Vermutlich hält die TM nicht, aber die moderne Mathematik kann das nicht beweisen.

Anmerkung: Es gibt eine bekannte TM mit 7910 Zuständen, die sich so verhält [Yedidia & Aaronson 2016] und sogar eine mit nur 1919 Zuständen [O'Rear, 2016].

Konsequenzen

Wir haben Turing-Mächtigkeit und Unentscheidbarkeit in vielen Formalismen gezeigt – jedes davon erlaubt die Konstruktion universeller Mathematiker!

Es gibt konkrete Beispiele für

- WHILE-Programme,
- Rust-Programme,
- Typ-0-Grammatiken,
- PCP-Instanzen,
- diophantische Gleichungen,
- prädikatenlogische Theorien,
- ...

deren Halten/Nichtleerheit/Lösbarkeit/Unerfüllbarkeit nicht durch die „übliche Mathematik“ (ZFC) bewiesen oder widerlegt werden kann.

Zumindest, falls die „übliche Mathematik“ konsistent ist.

Hilberts 10. Problem, revisited

Hilbert: „... man soll ein Verfahren angeben, nach welchem sich mittels einer endlichen Anzahl von Operationen entscheiden lässt, ob die Gleichung in ganzen rationalen Zahlen lösbar ist.“

Turing: Es gibt Probleme, die durch kein automatisches Verfahren lösbar sind.

Gödel: Es gibt konstruierbare Beispiele konkreter arithmetischer Sätze, deren Gültigkeit nicht in der modernen Mathematik bewiesen oder widerlegt werden kann.

Matiyasevich/Robinson/Davis/Putnam: Die Lösbarkeit diophantischer Gleichungen ist unentscheidbar.

Alle zusammen: Es gibt also sogar konkrete diophantische Gleichungen, deren Lösbarkeit nicht in der modernen Mathematik bewiesen oder widerlegt werden kann.

Aber: Die Lösbarkeit jeder konkreten diophantischen Gleichung wird durch irgendein Programm entschieden (wir wissen nur oft nicht, welches, bzw. können seine Korrektheit nicht in ZFC mathematisch beweisen).

Ausblick

Komplexitätstheorie

Wesentliche Grundlagen der klassischen Komplexitätslehre haben wir hier schon behandelt.

Weiterführende Themen (Beispiele):

- Hierarchietheoreme: Kann man mit mehr Zeit/Speicher wirklich mehr berechnen?
- Relative Komplexität: Orakel
- Komplexitäten unterhalb von P: Schaltkreise als Rechenmodell
- Rechnen mit Zufall: Randomisierte Komplexität
- Einführung in Quantencomputer
- ...

⇝ siehe Vorlesung Complexity Theory (Wintersemester)

Datenbanktheorie

Die Theorie der Datenbanken ist ein wichtiges Anwendungsgebiet für viele Themen aus theoretischer Informatik und Logik.

Weiterführende Themen (Beispiele):

- Anfragesprachen vergleichen bzgl. Komplexität und Ausdrucksstärke
- Relationales Kalkül (=Prädikatenlogik)
- Datalog: rekursive Anfragesprache, Fragment der Logik zweiter Stufe
- Graph-Anfragen: Erreichbarkeit und Co. berechnen
- Anfragen unter Berücksichtigung von Constraints
- ...

⇝ siehe Vorlesung Database Theory (Sommersemester)

Verifikation

Programm- und Hardwareverifikation ist eine wichtige Anwendung logischer Methoden.

Weiterführende Themen (Beispiele):

- Modellierung reaktiver Systeme: Transaktionssysteme
- Automatenmodelle zur Darstellung verifizierbarer Eigenschaften
- Logiken mit linearer und verzweigender Zeit (LTL, CTL, CTL*)
- Probabilistische und zeitgesteuerte Automaten
- ...

⇝ siehe Vorlesung Model Checking

Symbolische Wissensrepräsentation

Anwendungsgebiet der formalen Logik, bei dem menschliches Wissen logisch kodiert und automatisch ausgewertet wird.

Weiterführende Themen (Beispiele):

- Entwicklung logischer Sprachen, für die Schlussfolgerung (effizient) entscheidbar ist
- Meist durch Einschränkung auf Teilmengen der Prädikatenlogik, z.B. bei Beschreibungslogiken
- Ontologien: logische Wissensmodelle
- Entwicklung effizienter Ableitungsalgorithmen und Implementierungen
- ...

⇝ verschiedene Vorlesungen, z.B. Foundations of Knowledge Representation (Wintersemester), Description Logics

Semantic Web & Datenaustausch

Anwendungsgebiet zwischen Datenbanken, Wissensrepräsentation und Webtechnologie.

Weiterführende Themen (Beispiele):

- Standards zur Kodierung von Fakten und Schemainformationen: RDF, OWL, ...
- Informationsintegration in (Web)Graphdatenbanken
- Anfragesprachen: SPARQL und ontologiebasierte Anfragen
- Anwendungen (z.B. Wikidata)
- ...

⇝ siehe Vorlesungen Foundations of Semantic Web Technologies (Wintersemester), Knowledge Graphs (Wintersemester)

Forschung (und studentische Arbeiten)

Einige Themen der Professur Computational Logic

Wissensrepräsentation: Wie kann menschliches Wissen digital dargestellt werden?

- Herausforderung: Flexibilität vs. Nutzbarkeit
- Herausforderung: Ausdrucksstärke vs. Verarbeitungskomplexität
- Schwerpunkt: logische Ontologie- und Anfragesprachen (Regeln, Beschreibungslogiken, Semantic Web, Fragmente der Prädikatenlogik, ...)

Nichtmonotones Schließen: Wie schließt man aus unvollständigem Wissen?

- Herausforderung: Rücknahme nachträglich invalidierter Folgerungen
- Herausforderung: Entwicklung von Beweistheorien, Algorithmen, Implementierungen
- Schwerpunkt: Antwortmengenprogrammierung, Regeln, An- und Ausnahmen

Verstehen natürlicher Sprache:
Wie kann die Bedeutung natürlichsprachlicher Ausdrücke repräsentiert werden?

- Herausforderung: Formalisierung impliziter Annahmen (Hintergrundwissen)
- Herausforderung: Umgang mit Mehrdeutigkeit, Unsicherheit, Ungenauigkeit
- Schwerpunkt: Logische Formalismen zur Darstellung natürlicher Sprache

Beispiele für BSc/MSc-Themen

Wissensrepräsentation: Wie kann menschliches Wissen digital dargestellt werden?

- Weiterentwicklung aktueller Sprachen, Komplexitätsresultate, Schließverfahren, ...
- Analyse und Vergleich mit anderen Ansätzen, Konzeption von Übersetzungen
- Implementierung und Optimierung von Schlussverfahren, experimentelle Analyse

Nichtmonotones Schließen: Wie schließt man aus unvollständigem Wissen?

- Ansätze von Aussagenlogik auf Prädikatenlogik verallgemeinern
- Modellierung der Behandlung von Ausnahmen bei Schlussregeln

Verstehen natürlicher Sprache:
Wie kann die Bedeutung natürlichsprachlicher Ausdrücke repräsentiert werden?

- Weiterentwicklung logischer Formalismen zum Umgang mit
  - Unsicherheit
  - Ungenauigkeit
  - Mehrdeutigkeit
  - implizitem Hintergrundwissen

Zusammenfassung

Übersicht

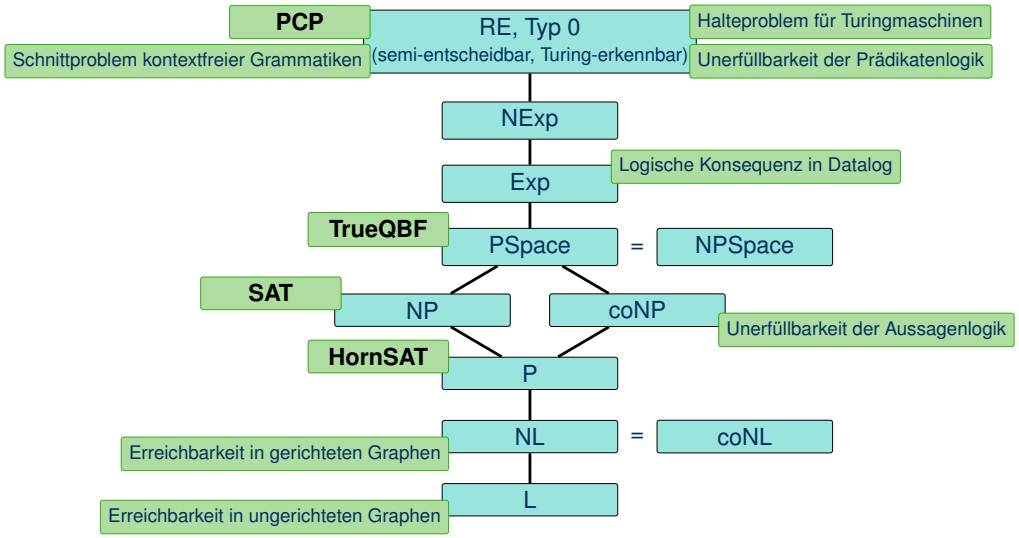
Berechenbarkeit: Turingmaschinen, LOOP/WHILE, Entscheidbarkeit, Beispielprobleme (Halten, PCP)

Komplexität: NP, PSpace, Many-One-Reduktionen, Übersicht weiterer wichtiger Klassen

Logik: Aussagenlogik (SAT), QBF, Prädikatenlogik, Resolution (mit vielen Teilschritten), Herbrand-Interpretationen, Datalog

Bonusmaterial: Gödel, Metamathematik, SQL, Geschichte und Geschichten

Komplexität: Typische Probleme



Querschnittsthemen

Informatik ist überall dort, wo gerechnet wird.

Rechnen = Schlussfolgern

Beweisen – Nachvollziehen – Verstehen

The End

Fragen?

Was erwartet uns als nächstes?

- Probeklausur
- Nachbesprechung der Probeklausur
- Repetitorium vor der Klausur?
- Klausur

Literatur und Bildrechte

Literatur

- A. Yedidia, S. Aaronson: **A Relatively Small Turing Machine Whose Behavior Is Independent of Set Theory.** Complex Systems 25(4), 2016.
- S. O'Rear: **Metamath Turing Machines.** `https://github.com/sorear/metamath-turing-machines`

Bildrechte

Folie 2: Fotografie um 1926, gemeinfrei