Beweisbar unbeweisbar – Formale Systeme und Gödels Unvollständigkeitssätze

Name: Ali Gottschall
Fach: Mathematik
Betreuungsperson: Thomas Foertsch
Jahr der Abgabe: 2020

Kantonsschule Im Lee

Motivation

Der besondere Reiz, den die Mathematik seit jeher auf mich ausübt, gründete nicht zuletzt auch in der Annahme, dass Aussagen innerhalb einer mathematischen Theorie absolut wahr oder falsch seien, und zwar ohne den Interpretationsspielraum, der mir in vielen Bereichen der Geisteswissenschaften immer wieder suspekt war. Als ich dann zum ersten Mal von Gödels Unvollständigkeitssätzen hörte, stürzte ich in eine kleine Existenzkrise, behauptet Gödels erster Unvollständigkeitssatz doch nichts anderes, als dass es auch in hinreichend starken mathematischen Theorien sehr wohl Aussagen gibt, die innerhalb der Theorie weder bewiesen noch widerlegt werden können. Nicht zuletzt um diese Krise zu überwinden, wollte ich im Rahmen meiner Maturitätsarbeit ein tieferes Verständnis für diese Grenzen der Mathematik erlangen.

Vorgehen

In den Sommerferien las ich zunächst das Sachbuch «Gödel, Escher, Bach» von Douglas Hofstadter, fing an zu programmieren und eignete mir die nötige Fachliteratur an. Sie war gewöhnungsbedürftig, da sie nicht an Gymnasiasten gerichtet ist. Die technischen Probleme erforderten Geduld und Gründlichkeit.

Mein Programm

Formale Systeme sind durch Axiome und Folgerungsregeln definiert. Axiome sind grundlegende Annahmen, die nicht weiter bewiesen werden müssen, und Folgerungsregeln sagen uns, wie man aus bekannten Sätzen neue erlangen kann. So sind zum Beispiel mathematische Theorien formale Systeme. Mein Python-Programm soll die abstrakten formalen Systeme greifbar machen. Es ist möglich, sein eigenes System zu erfinden und damit herumzuspielen. Man kann Sätze produzieren lassen oder Sätze beweisen lassen. Ausserdem habe ich Scripts geschrieben, die die Gödel-Nummerierung und Goodstein-Folgen illustrieren.

Beweisskizze

Gödel hat eine Technik gefunden, Sätze mit Zahlen und Zahlen mit Sätzen zu assoziieren. Somit gelang es ihm, «Zahlen über Zahlen» sprechen zu lassen und eine Selbstreferenz in der Zahlenlehre herzustellen. Der Satz «Ich bin ein Lügner» ist widersprüchlich, egal ob ich ein Lügner bin oder nicht. In gleicher Manier konstruierte Gödel Sätze in der Mathematik wie zum Beispiel: «Dieser Satz ist unbeweisbar.» Ist er wahr, dann ist er in der Tat unbeweisbar. Wäre der Satz falsch, müsste es einen Beweis geben. Das stünde aber im Widerspruch mit dem Satz selbst. Folglich ist dieser Satz beweisbar unbeweisbar.

Schlusswort

Die Gödelschen Unvollständigkeitssätze waren bahnbrechende Errungenschaften in der mathematischen Logik. Ihr Einfluss wird verglichen mit dem der Relativitätstheorie auf die Physik. Kurt Gödel legte den Grundstein für weitere Forschungen auf dem Gebiet wie zum Beispiel Turings Halteproblem. Die Arbeit gab mir die Möglichkeit, dieses komplexe Sachgebiet im Selbststudium zu erarbeiten und dabei meine Leidenschaften für Informatik und Mathematik zu vereinen.

Ich habe ein Programm geschrieben, das neue Sätze aus eingegebenen Axiomen und Regeln bildet.

```
    @classmethod
    def from_file(cls, path):
        parsed = toml.load(path)
        return cls(parsed["axiom"], parsed["rule"])


    def apply_rules(self, sequence, as_theorems=True) -> set:
        """ Wendet alle Regeln möglichst viel mal an
        wenn as_theorems wird eine Menge von Theorem Objekten returnt, sonst
        strings"""
        news = set()
        for i, rule in enumerate(self.rules):
            if rule.regex.search(sequence):
                for p in range(len(sequence)+1):
                    res = rule.regex.sub(rule.sub, sequence, pos=p-1, count=1)
                    if res not sequence:
                        if as_theorems:
                            news.add(Theorem(res, None, i))
                        else:
                            news.add(res)
        return news


def _step_production(self, verbose=True):
    """ Generator, der pro Schritt wenn möglich ein neues Axiom produziert
    und alle Regeln auf alle aktiven Sätze anwendet"""
    nexts = list()
    for gen in self._generators:
        new_axiom = next(gen)
        if verbose: print("New axiom added")
        self._active.append(new_axiom)
        yield new_axiom
    if self._active:
        for a in self._active:
            seq = a.theorem
            news = self.apply_rules(seq)
            for new_theorem in news:
                new_theorem.parent = a
                if new_theorem.theorem not in self.theorems:
                    nexts.append(new_theorem)
                    yield new_theorem
        self._active = nexts
```

Ausschnitte der Resultate eines Testlaufs:

```
AAAALLLLLALLLLLLI <- Rule 2 from AAAALLLLLALLLLLI
AAAAALLLLLALLLLLI <- Rule 0 from AAAALLLLLALLLLI
AAALLLLLLLALLLLLLI <- Rule 2 from AAALLLLLLALLLLLLI
AALLLLLLLLLLLLLLLLI <- Rule 2 from AALLLLLLLLLLLLLLLI
ALLLLLLLLLLALLLLLLLLI <- Rule 2 from ALLLLLLLLLALLLLLLLLI
LALALALALLLLLALLLLI <- Rule 2 from ALALALALLLLLALLLLI
ALALLALALLLLLALLLLI <- Rule 2 from ALALALALLLLLALLLLI
AAALALALLLLLALLLLI <- Rule 0 from ALALALALLLLLALLLLI
AALLALALLLLLALLLLI <- Rule 2 from ALALALALLLLLALLLLI
ALLLALLLLALLLLALLLLI <- Rule 2 from ALLALLALLLLALLLLI
LALLLALLALLLLALLLLI <- Rule 2 from ALLALLALLLLALLLLI
ALLALALLLLALLLLI <- Rule 1 from ALLALLALLLLALLLLI
LLALLLALALLLLALLLLI <- Rule 2 from LALLALALLLLALLLLI
LAALALLLLALLLLI <- Rule 1 from LALLALALLLLALLLLI
LLLLALLLLALLLLALLLLI <- Rule 2 from LLLALLLALLLLALLLLI
LLLALALLLLALLLLI <- Rule 1 from LLLALLLALLLLALLLLI
ALALAALLLLALLLLI <- Rule 2 from AAAALLLLALLLLI
AAAAAALLLLALLLLI <- Rule 0 from AAAALLLLALLLLI
AALALAALLLLALLLLI <- Rule 2 from AAAALLLLALLLLI
LALAAAALLLLALLLLI <- Rule 2 from AAAALLLLALLLLI
LLALLAAALLLLALLLLI <- Rule 2 from LALAALLLLALLLLI
LALLALAALLLLALLLLI <- Rule 2 from LALAALLLLALLLLI
ALLALLAALLLLALLLLI <- Rule 2 from ALALALLLLALLLLI
LLLALLLAALLLLALLLLI <- Rule 2 from LLALLALLLLALLLLI
LLAAALLLLALLLLI <- Rule 1 from LLALLAALLLLALLLLI
>>> f.produce(3)
LALLLALLALALLLI <- Rule 2 from LALLALALALLLI
LLALLLALALALLLI <- Rule 2 from LALLALALALLLI
LAALALALLLI <- Rule 1 from LALLALALALLLI
>>> f.
f.apply_rules(      f.from_file(        f.proof(            f.theorems
f.axioms            f.is_proof(         f.rules             f.tree_produce(
f.derive(           f.produce(          f.step_production(
>>> len(f.theorems)
2129
>>> f.proof("ALLALLALLI")
Starting to derive from
ALLALLALLI
ALALALLI <- Rule 2 from AAALLI 2
AAALLI <- Rule 0 from AALLI 2
AALLI <- Rule 8 from ALLI 0
ALLI <- Rule 1 from LLALLI 8
LLALLI <- Rule 2 from LALI 1
```