Logik für Informatiker
Logic for computer scientists

Till Mossakowski

Wintersemester 2014/15

Überblick

- Motivation
- Über die Vorlesung
- Die Sprache PL1
- Formale Beweise in Fitch
- Die Booleschen Junktoren

Motivation

Warum ist Logik für einen Informatiker wichtig?

- Formale Spezifikation und Verifikation
- Datenbanken, WWW, Künstliche Intelligenz
- Algorithmen und Komplexität
- Metatheorie
- (Semi-)Automatisches Beweisen von Theoremen
- Programmiersprachen

Formale Spezifikation und Verifikation

- Formale Software- und Hardwareentwicklung
- Verifikation von Software und Hardware
- Erzeugung von Testdaten
- Protokoll-Verifikation, Sicherheit (Modale und Temporale Logik)
- Eigenschaften von Telefonsystemen
- Beispiel: Die Arithmetik des Pentium 4 ist vollständig spezifiziert und verifiziert mittels Logik höherer Ordnung
- Beispiel: Die NASA benutzt Logik für ihre Testsoftware

Datenbanken, WWW, Künstliche Intelligenz

- Suche im Internet, Datenbankanfragen (SQL)
- Ontologies und semantic web
- Expertensysetme
- Linguistik
- Beispiel: CYC ist eine maschinenauswertbare Wissensdatenbank des Alltagswissens.
  Sie enthält über 1,5 Millionen Fakten und Sachverhalte des "gesunden Menschenverstandes" – die CYC-Inferenzmaschine benutzt Logik erster Stufe.

Algorithmen und Komplexität

- Für die Frage, ob Eigenschaften von Graphen in Prädikatenlogik zweiter Stufe formuliert werden können, gibt es einen effizienten Algorithmus.
- Komplexitätsklassen können durch Klassen logischer Formeln charakterisiert werden.

Metatheorie

- Die Mengentheorie wurde in Logik erster Stufe formalisiert – das dient als Grundlage für die gesamte Mathematik und theoretische Informatik.
- Gödel's Vollständigkeitssatz der Logik erster Stufe: Semantik kann durch formale Beweise erfasst werden – sogar durch Maschine geschriebene Beweise.
- Gödel's Unvollständigkeitssatz der Logik erster Stufe inklusive der Induktion: Es gibt wesentliche Aussagen der Mathematik und der theoretischen Informatik, die nicht durch formale Systeme erfasst werden können.

(Semi-)Automatisches Beweisen von Theoremen

- Logische Eigenschaften endlicher Automaten können automatisch überprüft werden (model checkers).
- Für komplexere Systeme braucht man semiautomatisches Beweisen.
- Die Verifikation von Beweisen ist relativ einfach und voll automatisch durchführbar.
- Beispiel: Einige Sätze über Boolesche Algebren sind von Computern gefunden worden.
- Beispiel: Etliche mathematische Lehrbücher sind durch semiautomatisches Beweisen verifiziert worden (und kleine aber gravierende Fehler sind gefunden worden).

Programmiersprachen

- Viele Programmiersprachen benutzen die logischen Operatoren “und”, “oder” und “nicht”.
- Prolog: Programmieren in Logik.
- Konzentration auf das Was anstatt auf das Wie.
- Einschließen von nicht-deterministischer Suche.
- Werden benutzt für Anwendungen in der Linguistik und der künstlichen Intelligenz.

Über die Vorlesung

Über die Vorlesung

- Aussagenlogische Folgerung
- Hintikka Spiel
- Aussagenlogische Beweise
- Resolution
- (Semi-)Automatisches Beweisen: SPASS, Isabelle
- Quantoren erster Stufe
- Folgerungen erster Stufe
- Mehrfache Quantifizierung
- Beweise erster Stufe, Resolution
- Induktion, Datenstrukturen
- Modelltheorie
- Korrektheit
- Vollständigkeit
- Anwendungen, Ausblick

Literatur

Jon Barwise und John Etchemendy: *Sprache, Beweis und Logik*. Mentis-Verlag, 2005, ISBN 978-3-89785-440-6.

Jon Barwise und John Etchemendy: *Sprache, Beweis und Logik II*. Mentis-Verlag, Paderborn, 2006, ISBN 978-3-89785-441-3.

Deutsche Übersetzung des Buches:

Jon Barwise und John Etchemendy: *Language, Proof and Logic*. First Edition, Seven Bridges Press, New York, 1999, ISBN 1-889119-08-3.

Wir arbeiten mit der 1. Auflage. Kaufen Sie nicht die 2. Auflage.

Programme

Boole Erstellen von Wahrheitstabellen

Tarski's world Evaluieren logischer Formeln in einer Welt von Klötzchen

Fitch Konstruieren von Beweisen

Grinder Automatisches Bewerten der Lösungen
$\rightarrow$ erfordert eine ID

Übungsaufgaben

6.36 ➶ Benutzen Sie Tarski's World, um eine Welt zu konstruieren, in welcher die folgenden Sätze alle wahr sind....

6.37 ✎ Reichen Sie einen informellen Beweis dafür ein, dass das folgende Argument logisch gültig ist....

6.38 ➶|✎ Ist das folgende Argument gültig? Wenn ja, benutzen Sie Fitch, um einen formalen Beweis von dessen Gültigkeit zu geben. Wenn nicht, erklären Sie, warum es ungültig ist und reichen Sie Ihre Erklärung bei Ihrem Dozierenden ein.

6.39 ✎* Entwerfen Sie eine PL1-Sprache, welche es Ihnen erlaubt, die folgenden deutschen Sätze auszudrücken....

Organisation

- Vorlesung: Mittwochs 17:15–18:45 Uhr, G44-H6
- Übungen: Einschreibung notwendig, Laptop
- Web:
  `http://theo.cs.uni-magdeburg.de/lehre/lehre14w/logik/`
- Zugang zum internen Bereich: Login: student-logik, Passwort
- Gruppen zu je drei Studenten erhalten eine ID für den Grinder und reichen als Gruppe ihre Lösungen über "Submit" ein.
- Die ID's bitte als Gruppe per E-Mail an `logik@iws.cs.uni-magdeburg.de` anfordern.

Zulassungskriterien für die Klausur

- Übungen: Übungsblätter mit Aufgaben aus dem Buch
- Es müssen zu Beginn jeder Übung Aufgaben votiert werden. $\Rightarrow$ bei Aufforderung vortragen.
  - Lösungsvorschläge werden diskutiert, sie müssen nicht gleich perfekt richtig sein.
- Für die Zulassung zur Klausur sind folgende Leistungen zu erbringen:
  - Mindestens 2/3 der Übungsaufgaben müssen am Ende des Semesters votiert worden sein.
  - Mindestens 50 % der Programmieraufgaben (ab Übung 2) müssen am Ende des Semesters über das Programm "Submit" eingereicht worden sein.
  - Mindestens zweimal pro Semester muss man in den Übungen vorgetragen haben.

Die Sprache PL1

Warum eine künstliche Sprache?

- Studium von Prinzipien des korrekten Schließens.
  - Was ist eine logische Folgerung?
- Erfordert das Eliminieren jedweder Mehrdeutigkeit
- Aufdecken von Mehrdeutigkeiten in natürlichen Sprachen.
- Programmiersprachen sind formale Sprachen.

Den Übergang von einer natürlichen zu einer formalen Sprache nennt man Formalisierung.

Die Sprache PL1: Individuenkonstanten

- Individuenkonstanten sind Symbole, die Personen, Dinge oder Objekte bezeichnen.
- Beispiele:
  - Zahlen: 0, 1, 2, 3, ...
  - Namen: Max, Claire
  - Formale Konstanten: a, b, c, d, e, f, n1, n2
- Jede Individuenkonstante muss ein existierendes Objekt bezeichnen.
- Keine Individuenkonstante kann mehr als ein Objekt bezeichen.
- Ein Objekt kann keinen, einen oder mehrere Namen (Individuenkonstanten) haben.

Die Sprache PL1: Prädikatsymbole

- Prädikatsymbole bezeichnen eine Eigenschaft eines Objektes oder Beziehungen zwischen Objekten
- Jedes Prädikatsymbol hat eine Stelligkeit, die die Anzahl der Objekte angibt, die in Beziehung stehen.
- Beispiele:
  - Stelligkeit 0: Gate0_is_low, A, B, ...
  - Stelligkeit 1:
    Cube (Würfel), Tet (Tetraeder), Dodec (Dodekaeder),
    Small (Klein), Medium (Mittelgroß), Large (Groß)
  - Stelligkeit 2:
    Smaller (Kleiner), Larger (Größer),
    LeftOf (Links Von), BackOf (Hinter),
    Adjoins (Benachbart), SameSize (Gleichgroß), ...
  - Stelligkeit 3: Between (Zwischen)

Die Interpretation von Prädikatsymbole

- In Tarski's world, besitzt jedes Prädikatsymbol eine feste Bedeutung, die immer mit der Interpretation in einer natürliche Sprache übereinstimmt
- In anderen PL1-Sprachen kann die Interpretatiuon von Prädikatsymbolen variieren. Zum Beispiel kann $\leq$ eine Ordnung von Zahlen, von Zeichenketten, Bäumen, etc. bedeuten.
- Wir vereinbaren, dass das Symbol "$=$" immer die Gleichheit, Identität bedeutet.

Atomare Sätze

- In der Aussagenlogik (Boole):
  - Die aussagenlogischen Symbole: A, B, C, ...
- In PL1 (Tarski's world):
  - Anwendung von Prädikatsymbolen auf Konstanten: Larger(a,b)
  - Reihenfolge der Argumente ist wichtig: Larger(a,b) vs. Larger(b,a)
  - Atomare Sätze haben genau einen Wahrheitswert: true (wahr, 1) oder false (falsch, 0)

Terme und Funktionssymbole

- Individuenkonstanten sind Terme.
- Mit Funktionssymbolen können komplexere Terme gebildet werden. Beispiele für Funktionssymbole sind
  - father, mother
  - +, -, *, /
- Damit ergeben sich z. B. die neuen Terme
  - father(max), mother(father(max))
  - 3*(4+2)
- Und daraus lassen sich z. B. neue atomare Sätze bilden:
  - Larger(father(max),max)
  - 2<3*(4+2)

Tautologie, Erfüllbarkeit

Ein Satz A heißt Tautologie, wenn er unter allen Umständen wahr ist.

Ein Satz A heißt erfüllbar, wenn er unter mindestens einem Umstand wahr ist.

Ein Umstand ist

- in der Aussagenlogik: eine Belegung der atomaren Formeln mit den Wahrheitswerten aus der Menge $\{\text{true}, \text{false}\}$.
- in Tarski's world: Eine Klötzchenwelt.

Folgerungen ...

Logische Folgerungen

Ein Satz B ist eine logische Folgerung von $A_1, \ldots, A_n$, wenn unter allen Umständen, die $A_1, \ldots, A_n$ wahr machen, auch B wahr ist. In Zeichen: $A_1, \ldots, A_n \models B$.

Die Sätze $A_1, \ldots, A_n$ heißen Prämissen, B heißt Konklusion.

In diesem Fall ist es ein gültiges Argument, B von $A_1, \ldots, A_n$ abzuleiten.

Ein gültiges Argument mit wahren Prämissen heißt korrekt (oder auch beweiskräftig, schlüssig).

Logische Folgerungen – Beispiele

- Alle Menschen sind sterblich. Sokrates ist ein Mensch. Also ist Sokrates sterblich. (gültig, korrekt)
- Alle reichen Schauspieler sind gute Schauspieler. Brad Pitt ist ein reicher Schauspieler. Also ist er ein guter Schauspieler. (gültig, aber nicht korrekt)
- Alle reichen Schauspieler sind gute Schauspieler. Brad Pitt ist ein guter Schauspieler. Also ist er ein reicher Schauspieler. (nicht gültig)