UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR INFORMATIONSSYSTEME

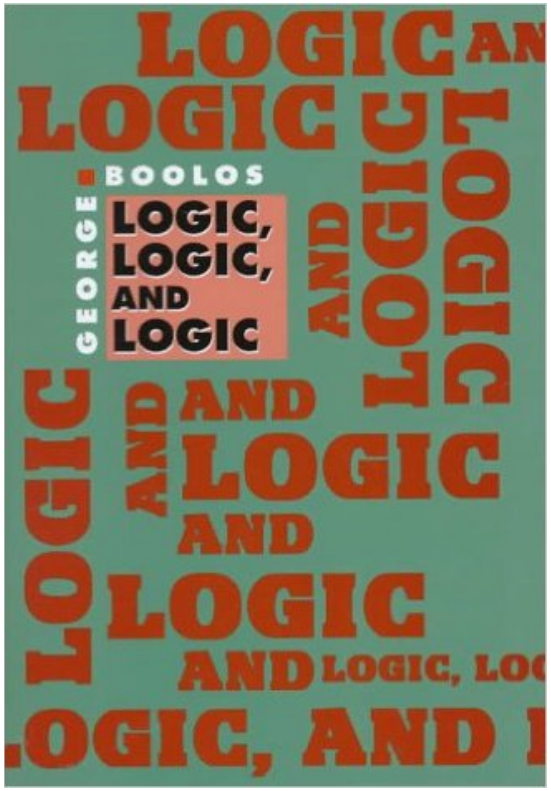
Özgür L. Özçep

Einführung

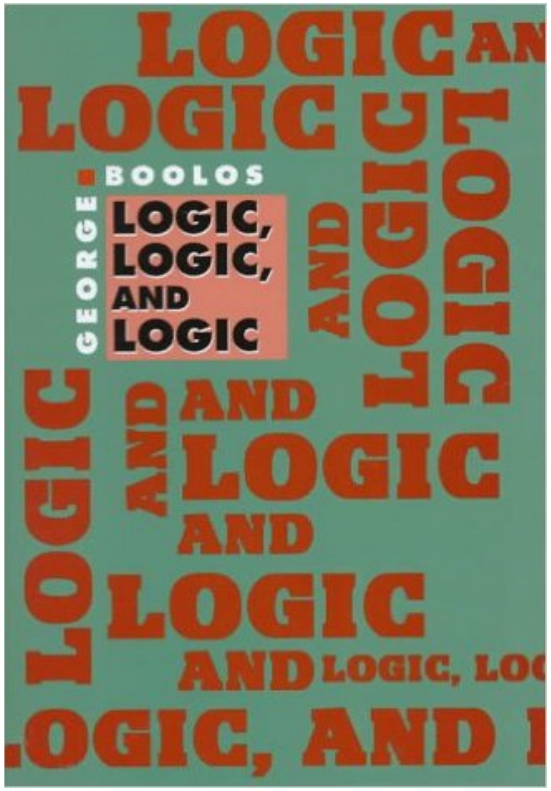
V0: Motivation and Überblick
1 April 2021

Logikprogrammierung CS3055
(Sommersemester 2021)

Logik allenthalben!



Logik allenthalben!



Agenda dieser Vorlesungseinheit

1. Grundidee der Logikprogrammierung
2. Überblick zu Prolog
3. Anwendung: Sprachverarbeitung
4. Schlusswort

Grundidee der Logikprogrammierung

Paradigma der Logikprogrammierung (LP)

Paradigma der Logikprogrammierung (LP)

- Weitere bekannte Paradigmen
  - funktionale Programmierung (Lisp)
  - imperative Programmierung (C)
  - objektorientierte Programmierung (Java)

Paradigma der Logikprogrammierung (LP)

- Weitere bekannte Paradigmen
  - funktionale Programmierung (Lisp)
  - imperative Programmierung (C)
  - objektorientierte Programmierung (Java)

- Problemlösung durch Spezifikation des „Was“ nicht des „Wie“

$\Rightarrow$ Hohe Abstraktion von

  - Steuerstrukturen
  - Speichermodell
  - Verarbeitungsrichtung

Wo ist die Logik in der Logikprogrammierung?

Wo ist die Logik in der Logikprogrammierung?

Definition (Logik)

Wissenschaft, die sich systematisch mit Strukturen und formalen Sprachen zu deren Beschreibung beschäftigt.

- Syntax (wohlgeformte Formeln)
- Semantik (Wahrheitsbedingungen, Folgerbarkeitsbegriff)
- Kalküle (Inferenzen)

Wo ist die Logik in der Logikprogrammierung?

Definition (Logik)

Wissenschaft, die sich systematisch mit Strukturen und formalen Sprachen zu deren Beschreibung beschäftigt.

- Syntax (wohlgeformte Formeln)
- Semantik (Wahrheitsbedingungen, Folgerbarkeitsbegriff)
- Kalküle (Inferenzen)

- Spezifikation der Domäne in einer Wissensbasis (WB) bestehend aus Formeln mit wahrheitstheoretischer Semantik

Wo ist die Logik in der Logikprogrammierung?

Definition (Logik)

Wissenschaft, die sich systematisch mit Strukturen und formalen Sprachen zu deren Beschreibung beschäftigt.

- Syntax (wohlgeformte Formeln)
- Semantik (Wahrheitsbedingungen, Folgerbarkeitsbegriff)
- Kalküle (Inferenzen)

- Spezifikation der Domäne in einer Wissensbasis (WB) bestehend aus Formeln mit wahrheitstheoretischer Semantik

- Problembeschreibung als Anfrage

Wo ist die Logik in der Logikprogrammierung?

Definition (Logik)

Wissenschaft, die sich systematisch mit Strukturen und formalen Sprachen zu deren Beschreibung beschäftigt.

- Syntax (wohlgeformte Formeln)
- Semantik (Wahrheitsbedingungen, Folgerbarkeitsbegriff)
- Kalküle (Inferenzen)

- Spezifikation der Domäne in einer Wissensbasis (WB) bestehend aus Formeln mit wahrheitstheoretischer Semantik

- Problembeschreibung als Anfrage

- Problemlösung durch bekannte Inferenzverfahren aus der Logik

Prolog

- Prolog: **Pro**grammation en **Log**ique
- Geburtsstunde um 1970, als geforscht wurde zu
  - Theorembeweisern
  - Sprachverarbeitung mit formalen Grammatiken
- Protagonisten
  - R. Kowalski: Theoretischer Beitrag in Form der SL-Resolution
  - A. Colmerauer und P. Roussel: Entwickler

Kowalski

Colmerauer

Überblick zu Prolog

Fakten

- Prolog-Wissensbasis besteht aus sogenannten Klauseln
- Zwei Typen von Klauseln: Fakten und Regeln

- Fakten: repräsentieren Objekte aus der interessierenden Domäne, deren Eigenschaften und deren Beziehungen zu anderen Objekten.

Beispiel für Fakten

```
human(john).        % John ist ein Mensch.
likes(john, mary). % John mag Mary.
```

Boolesche Anfragen

Faktenbasis

```
likes(john, fish).
likes(joe, mary).
likes(mary, book).
likes(john, book).
likes(john, france).
```

Anfrage

```
?- likes(john, fish).
true
?- likes(john, money).
false
```

Boolesche Anfragen

Faktenbasis

```
likes(john, fish).
likes(joe, mary).
likes(mary, book).
likes(john, book).
likes(john, france).
```

Anfrage

```
?- likes(john, fish).
true
?- likes(john, money).
false
```

- Anfragebeantwortung
  - Iteration über Klauseln
  - Abgleichen: Unifikation

- Antwort „true“, wenn Anfragefakt beweisbar.
- Antwort „false“, wenn Anfragefakt nicht beweisbar.

$\Longrightarrow$ Closed world assumption (CWA)
wie in DB-Theorie

Anfragen mit Variablen

Faktenbasis

```
likes(john, fish).
likes(joe, mary).
likes(mary, book).
likes(john, book).
likes(john, france).
```

Anfragen

```
?- likes(john, X).
%  Dinge (X), die John mag
X = fish;
X = book;
X = france;
false
```

- Abarbeitung durch Polog
  - Prolog versucht Anfrage mit einem Fakt zu unifizieren.
  - Prolog markiert Position von Fakten mit erfolgreicher Unifikation. (Nötig für das Backtracking)
  - Weitere Lösungen für `X` via „;“
- Abstrakter Variablenbegriff

Konjunktive Anfrage

Beispiel für eine konjunktive Anfrage

```
likes(mary,X), likes(john,X).
%  Dinge, die sowohl John als auch Mary mag
```

- Abarbeitung durch Polog
  - Teilkonjunkte von links nach rechts abgearbeitet
  - Erfolgreiche Unifikation im ersten Konjunkt führt zu Bindung von X für die gesamte (!) Anfrage.
  - Zweites Konjunkt mit Bindung für X als neue Unteranfrage
  - Nicht erfolgreiche Unifikation in Unteranfragen führt zu Neuversuchen (Backtracking)

Konjunktive Anfrage

Beispiel für eine konjunktive Anfrage

```
likes(mary,X), likes(john,X).
%  Dinge, die sowohl John als auch Mary mag
```

- Abarbeitung durch Polog
  - Teilkonjunkte von links nach rechts abgearbeitet
  - Erfolgreiche Unifikation im ersten Konjunkt führt zu Bindung von X für die gesamte (!) Anfrage.
  - Zweites Konjunkt mit Bindung für X als neue Unteranfrage
  - Nicht erfolgreiche Unifikation in Unteranfragen führt zu Neuversuchen (Backtracking)
- Übrigens: Welchem bekannten SQL-Fragment entsprechen konjunktive Anfragen?

Regeln

- Syntax

```
Kopf :- Körper
```

Regeln

- Syntax

  `Kopf :- Körper`

- Semantik

  `Körper` $\rightarrow$ `Kopf`

Regeln

- Syntax

  `Kopf :- Körper`

- Semantik

  `Körper` $\rightarrow$ `Kopf`

- Intensionales Wissen

Regeln

- Syntax

  `Kopf :- Körper`

- Semantik

  `Körper` $\rightarrow$ `Kopf`

- Intensionales Wissen

- Definition von dynamischen Datenstrukturen (z.B. Listen)

Regel mit einfachem Körper

```
likes(john,X) :- likes(X,wine).
                 % John mag Weinliebhaber
                 % Forall X: likes(X,wine) -> likes(john,X)
```

Regel mit konjunktivem Körper

```
sisterOf(X,Y) :- female(X),
                 parents(X,M,F), parents(Y,M,F).
        % X ist Schwester von Y, wenn X weiblich und
        % X und Y gemeinsame Eltern haben.
```

Anfragebeantwortung bei Regeln

1. Anfrageklausel wird mit Kopf unifiziert
2. Bei erfolgreicher Unifikation entsteht eine Unteranfrage mit den erzielten Variablenbindungen
3. Unteranfrage wird als neue Anfrage behandelt

Rekursion mal anders

TOKYO—Television watching became even more convenient this week with Sony's introduction of a new remote-controlled remote control.

A viewer enjoys the new remote-controlled remote control.

The new device, which can be controlled via remote control through the use of a second remote control unit, will replace older models that needed to be held in the hand to be operable.

"Constantly leaning forward to pick up the remote control from the coffee table is a tiresome, cumbersome chore that will soon be a thing of the past," Sony director of product development Dan Ninomiya said. "These new remotes, should they be left on the coffee table or in some other barely-hard-to-reach place, will not need to be picked up and actually pointed at the screen in order to work."

The new remote control—along with the additional remote it is designed to control—will soon come standard with all Sony televisions, allowing viewers to remain "more immobile, more stationary, and more physically inert than ever before."

"Imagine a remote control capable of switching channels on your television right from its spot on the table, one that requires no clumsy fumbling about with the hands to operate," Ninomiya said. "Well, that bold, inactive future is here."

http://www.theonion.com/article/new-remote-control-can-be-operated-by-remote-1666

Regel mit Rekursion

```
controls(X,Y) :- controlsDirectly(X,Y).
     % X bedient Y direkt.

controls(X,Y) :- controlsDirectly(X,Z), controls(Z,Y).
     % X bedient Y indirekt.
```

Listen

- Listen als 2-stellige Terme mit Kopf und Restliste
- Beispiele: `[]` `[3,a,b]` `[3,X,4,5]` `[2,3 | R]`

Konkatenation via Listenkonsumption

```
%  app(?Liste1, ?Liste2, ?Resultat)
append([],L2,L2).
append([X | Rest], L2, [X | Z]) :- append(Rest,L2,Z).
```

Ist Prolog Logikprogrammierung?

- Antwort: „JA!“ für pures Prolog

Ist Prolog Logikprogrammierung?

- Antwort: „JA!“ für pures Prolog
- Antwort: „JEIN!“ für volles Prolog
  - viele gewünschte Operationen nur durch Seitenaffekte lösbar (Bsp.: Input-Output-Verarbeitung)
  - intendierte denotationelle Semantik kann von operationaler Semantik abweichen (Reihenfolge, Cut, Negation)

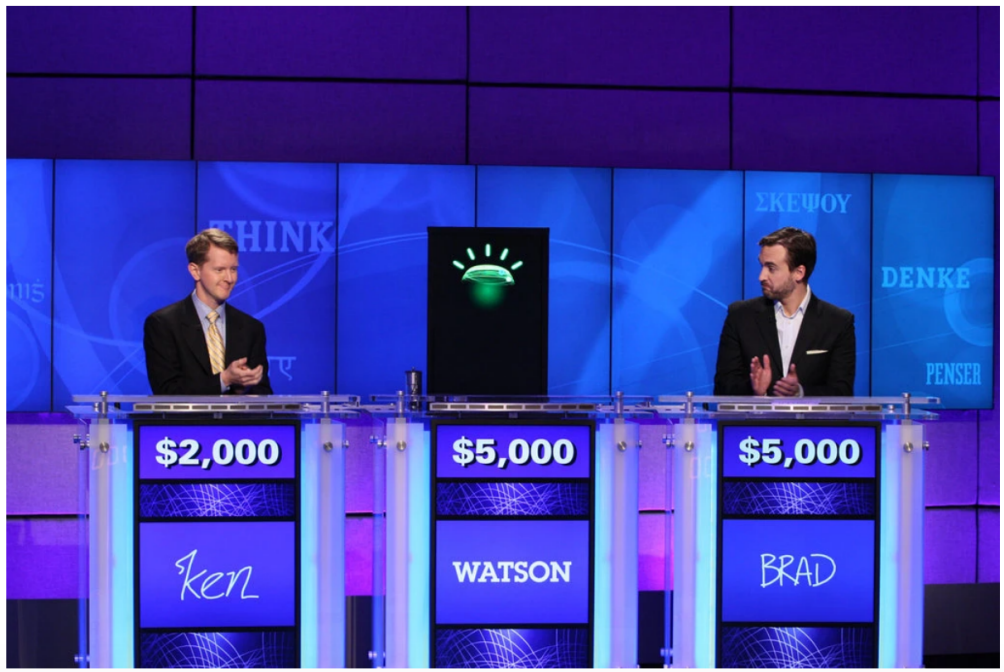
Ist Prolog Logikprogrammierung?

- Antwort: „JA!“ für pures Prolog
- Antwort: „JEIN!“ für volles Prolog
  - viele gewünschte Operationen nur durch Seitenaffekte lösbar (Bsp.: Input-Output-Verarbeitung)
  - intendierte denotationelle Semantik kann von operationaler Semantik abweichen (Reihenfolge, Cut, Negation)
- Alternative Logikprogrammiersprachen entwickelt (z.B. Answer Set Programming (ASP))

Anwendung: Sprachverarbeitung

The New York Times

Computer Wins on 'Jeopardy!': Trivial, It's Not



Example (Category: POETS & POETRY)

Hinweis: "He was a bank clerk in the Yukon before he published 'Songs of a Sourdough' in 1907"

Watson: "Who was Robert W. Service?"

Example (Category: POETS & POETRY)

Hinweis: “He was a bank clerk in the Yukon before he published ‘Songs of a Sourdough’ in 1907”

Watson: “Who was Robert W. Service?”

Parsing

```
lemma(1, ‘‘he’’).
partOfSpeech(1,pronoun).
lemma(2, ‘‘publish’’).
partOfSpeech(2,verb).
lemma(3, ‘‘Songs of a Sourdough’’).
partOfSpeech(3,noun).
subject(2,1).
object(2,3).
```

Example (Category: POETS & POETRY)

Hinweis: “He was a bank clerk in the Yukon before he published ‘Songs of a Sourdough’ in 1907”

Watson: “Who was Robert W. Service?”

Bestimmung der relationalen Struktur

```
authorOf(Author,Composition) :-
    createVerb(Verb),
    subject(Verb,Author),
    author(Author),
    object(Verb,Composition),
    composition(Composition).

createVerb(Verb) :-
    partOfSpeech(Verb,verb),
    lemma(Verb,VerbLemma),
    member(VerbLemma, ["write", "publish",...]).
```

Schlusswort

Warum Logikprogrammierung?

- „Bir lisan bir insan. İki lisan iki insan“

Warum Logikprogrammierung?

- „Bir lisan bir insan. İki lisan iki insan“
- Schnelle Prototypentwicklung

Warum Logikprogrammierung?

- „Bir lisan bir insan. İki lisan iki insan“
- Schnelle Prototypentwicklung
- Logikprogrammierung lebt!
  - Renaissance von Datalog
  - Probabilistische Logikprogrammierung
    https://dtai.cs.kuleuven.be/problog/

Literatur

Prolog Referenzen

W.F. Clocksin, C.S. Mellish: Programming in Prolog.

(Der Klassiker, kompakt und Programmiersprachen-orientiert)

L. Sterling, E. Shapiro: The Art of Prolog—Advanced Programming Techniques.

(Umfangreiches Buch zur fortgeschritteneren Prolog-Programmierung mit vielen klassischen KI-Themen und Konzepten)

I. Bratko: Prolog: Programming for Artificial Intelligence.

(Umfangreiches Buch, das viele klassische KI-Themen abdeckt)

P. Blackburn, J. Bos, K. Striegnitz: Learn Prolog Now!.

(Kompaktes und leicht verständliches Buch neueren Datums mit Online-Tutorial http://www.learnprolognow.org/)

M.A. Covington: Natural Language Processing for Prolog Programmers.

(Schwerpunkt NLP mit Prolog mit Betonung der Prologtechniken. Online frei als PDF erhältlich. http://www.covingtoninnovations.com/books/NLPPP.pdf)

Prolog Referenzen

F.C.N. Pereira, S.M. Shieber: Prolog and Natural-Language Analysis.

(Schwerpunkt NLP mit Prolog. Online frei als PDF erhältlich.

http://www.mtome.com/Publications/PNLA/pnla.html)

P.M. Nugues: Language Processing with Perl and Prolog: Theories, Implementation, and Application.

Aktuelles Buch zu NLP in (Perl und) Prolog

U. Nilsson, J. Maluszynski: Logic, Programming and Prolog

(Fokus auf Logik und Prolog als Logikprogrammiersprache. Online frei als PDF erhältlich

http://www.ida.liu.se/~ulfni/lpp)

A.Lally, P. Fodor: Natural Language Processing With Prolog in the IBM Watson System. The Association for Logic Programming.

(http://www.cs.nmsu.edu/ALP/2011/03/

natural-language-processing-with-prolog-in-the-ibm-watson-system/)

Prolog Referenzen

- SWI-Prolog (http://www.swi-prolog.org/)

  (Robuster und freier Prolog-Interpreter, der Grundlage für diesen Kurs ist)

- Pronto (http://ai1.ai.uga.edu/mc/pronto/)

  (NLP-Tools von Studenten der Georgia Universität (Covington))

- NLP techniques in Prolog

  (http://cs.union.edu/~striegnk/courses/nlp-with-prolog/html/)

- Adventure in Prolog (http://www.amzi.com/AdventureInProlog/index.php)

- The Prolog 1000 database of real Prolog applications (http://www.cs.cmu.edu/afs/cs/project/ai-repository/ai/lang/prolog/doc/pl_1000/pl1000v1.gz)

  (etwas veraltete doch aufschlussreiche Liste)