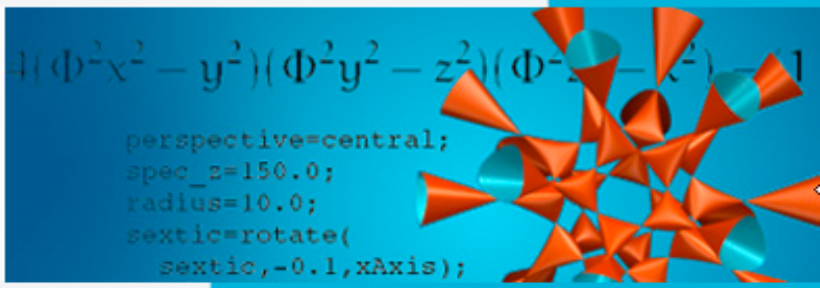
Philipps Universität Marburg

Deklarative Programmierung

Sommersemester 2018

Prof. Christoph Bockisch
(Programmiersprachen und –werkzeuge)
Steffen Dick, Alexander Bille, Johannes Frankenau,
Patrick Frömel, Niclas Schmidt, Jonas Stettin,
Robert Tran, Julian Velten



[The art of Prolog: Einleitung – 1.7]

Logikprogrammierung

- Ausgangspunkt:
  - Wie will ich Probleme/Lösungen beschreiben?
  - Nicht: Welche Beschreibungen versteht die Maschine?
  - Ähnlich wir funktionale Programmierung

- Logik Programm
  - Wissen explizit ausdrücken: **logische Axiome**
  - Problem als logische Aussage beschreiben (“**Zielaussage**“)
  - Programm: Menge von Axiomen
  - Programmausführung: konstruktiver Beweis der Zielaussage über dem Programm

Logikprogrammierung

- Zielaussage typischerweise existential quantifiziert:
  “Es gibt eine Liste X, sodass das Sortieren der Liste [3, 1, 2] die Liste X ergibt.“

- Ergebnis des Programms
  - Folgt die Zielaussage aus den Annahmen? - Ja/Nein
  - Konstruktiver Beweis: Angabe von Werten für die Unbekannten für die die Aussage wahr wird.

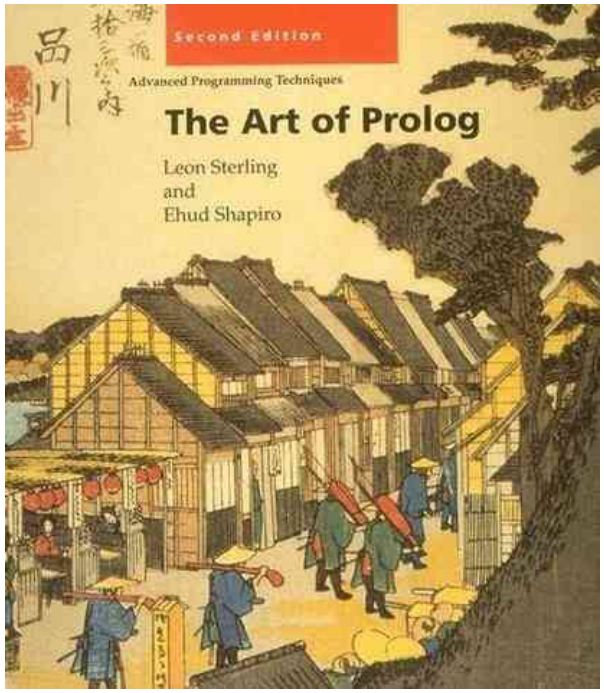
- Beispiel, angenommen, ausreichende Axiome über “Sortieren“ wurden definiert:
  - X = [1, 2, 3]

Geschichte von Prolog

- Entwickelt 1970er (Kowalski, Colmerauer)
- PROrammieren in LOGik
- Beeinflusste viele Entwicklungen:
  - 5th Generation Project
  - Deductive Databases
  - Constraint Logic Programming
- Prolog-Implementierung:
  - TuProlog
  - http://apice.unibo.it/xwiki/bin/view/Tuprolog/
  - https://bitbucket.org/tuprologteam/tuprolog/downloads/2p-3.3.0.zip

Literatur

- Leon Sterling, Ehud Shapiro. The Art of Prolog
- http://www.learnprolognow.org



Programmieren in Prolog

- Programmieren in Prolog bedeutet
  - Tatsachen (Fakten) über Objekte und deren Beziehungen zu deklarieren
  - Regeln über Objekte und deren Beziehungen zu definieren
  - Fragen zu den Objekten und Beziehungen zu stellen

Sokrates ist ein Mensch.

Alle Menschen sind sterblich.

Ist Sokrates sterblich?

- Nur **eine Datenstruktur: logischer Term**

Fakten

- Aussage: eine Relation (Prädikat) gilt zwischen Objekten
- Objekte werden als atomarer Bezeichner dargestellt
- Beispiel
  - father(abraham,isaac).

Die Relation “father“ gilt zwischen abraham und isaac

Relation

Atome

- Konvention: Relationen und Atome beginnen mit kleinem Buchstaben

Fakten

- (Arithmetische) Operation lassen sich als Fakten darstellen
  - Hier unvollständig

- Beispiel
  - plus(0,0,0). plus(0,1,1). plus(0,2,2). plus(0,3,3).
    plus(1,0,1). plus(1,1,2). plus(1,2,3). plus(1,3,4).

- Eine endliche Menge an Fakten definiert ein Programm

Fakten

- father(terach,abraham).
  father(terach,nachor).
  father(terach,haran).
  father(abraham,isaac).
  father(haran,lot).
  father(haran,milcah).
  father(haran,yiscah).
- mother(sarah,isaac).
- male(terach).
  male(abraham).
  male(nachor).
  male(haran).
  male(isaac).
  male(lot).
- female(sarah).
  female(milcah).
  female(yiscah).

Abfragen

- Abfragen von Information aus einem Logikprogramm
- Abfragen, ob eine Relation zwischen Objekten erfüllt ist
- Beispiel
  - ?- father(abraham, isaac).
  - *yes*

Gilt die Relation father zwischen abraham und isaac?

Abfragen

- Abfragen und Fakten haben dieselbe Syntax
  - Aus dem Kontext ergibt sich, ob es eine Abfrage oder ein Fakt ist

- **Fakt**
  P.
  - Aussage, dass das Ziel P wahr ist

- **Abfrage**
  ?- P.
  - Frage, ob P wahr ist

- Eine einfache Abfrage besteht aus einem einzigen Ziel

Abfragen

- Beantworten einer Abfrage bezüglich eines Programms
  - Ist die Abfrage eine logische Konsequenz des Programms?
- Logische Konsequenz wird bestimmt durch Anwenden von **Deduktionsregeln**
- Deduktionsregel ***Identität***: Aus P folgt P.
  - Suche einen Fakt, der die Abfrage impliziert
  - Wird so ein Fakt gefunden, ist die Abfrage erfüllt (Antwort Ja)
  - Die Antwort ist Nein, wenn kein solcher Fakt gefunden wird

Abfragen

- Antwort Nein bedeutet
  - Die Aussage konnte mit aus dem Programm heraus nicht bewiesen werden
- Antwort Nein bedeutet nicht,
  - Dass die Aussage tatsächlich falsch ist
- Beispiel
  - plus(0,0,0). plus(0,1,1). plus(0,2,2). plus(0,3,3).
  - plus(1,0,1). plus(1,1,2). plus(1,2,3). plus(1,3,4).
  - ?- plus(1, 4, 5).
    - Ergibt *Nein*, weil die Fakten über die Addition unvollständig sind

Logische Variablen

- **Variable**: unspezifiziertes Objekt
- **Abstraktion**, die für mehrere Abfragen steht
- Beispiel: Von wem ist abraham der Vater?
  - Alle Objekte durchgehen, bis eine Abfrage Ja ergibt
    - ?- father(abraham, lot).
    - ?- father(abraham, milcah).
    - ...
    - ?- father(abraham, isaac).
  - Gebrauch von logischen Variablen
    - ?- father (abraham, X).

Logische Variablen

- **Variable**: unspezifiziertes Objekt
- **Abstraktion**, die für mehrere Abfragen steht
- Beispiel: Von wem ist abraham der Vater?
  - Alle Objekte durchgehen, bis eine Abfrage Ja ergibt
    - ?- father(abraham, lot).
    - ?- father(abraham, milcah).
    - ...
    - ?- father(abraham, isaac).
  - Gebrauch von logischen Variablen
    - ?- father (abraham, X).

Ist abraham Vater von irgendeinem Objekt?

Programmausführung: konstruktiver Beweis. Daher ist das Ergebnis: Ja mit X=isaac

Logische Terme

- Induktive Definition: Logischer Term
  - Konstanten und Variablen sind logische Terme
  - Zusammengesetzte Terme (“Strukturen“) sind Terme

- Zusammengesetzter Term
  - Funktor + Sequenz von mindestens einem Argument
  - Argumente sind Terme
  - Funktor ist charakterisiert durch:
    - Name
    - Arität (Anzahl der Argumente)

Fakten und Abfragen sind zusammengesetzte Terme

Zusammengesetzte Terme

- Terme, die keine Variablen enthalten werden "**Grundterm**" genannt
- Beispiele
  - s(0).
  - hot(milk).
  - name(john,doe).
  - list(a,list(b,nil)).
  - tree(tree(nil,3,nil),5,R).
  - foo(X).

Name=s, Arität=1, Funktor- s/1

Grundterm

Strukturen sind rekursiv

Kein Grundterm (**Nicht-Grundterm**)