Logische Programmierung

1. Einleitung

Elmar Eder

22. März 2021

Verschiedene Paradigmen der Programmierung

Imperative Programmierung

- Einfache imperative Programmierung
- Prozedurale Programmierung
- Objektorientierte Programmierung

Deklarative Programmierung

- Logische Programmierung
- Funktionale Programmierung

Unterschiede

Imperative Programmierung

- Befehls/Prozedur-Aufruf verändert Zustand. Seiteneffekt
- Ergebnis eines Aufrufs hängt vom momentanen Zustand ab.
- Zweimaliger Aufruf kann verschiedene Ergebnisse liefern.

```
x = x+1; x = x+1;
```

Unterschiede

Deklarative Programmierung

- Keine Seiteneffekte
- Mathematische Beschreibung durch
  - Prädikate (= Relationen) (Logische Programmierung)
  - Funktionen (Funktionale Programmierung)
- Ein Prädikats-/Funktions-Aufruf liefert immer dasselbe Ergebnis – unabhängig vom momentanen Zustand der Maschine
- Programm leichter zu verstehen
- Korrektheit leichter zu zeigen

Logische Programmiersprachen

- Prolog (älteste, einfachste und verbreitetste)
- Erweiterungen und Multi-Paradigmen-Sprachen:
- Mercury
- Gödel
- Oz
- constraint logic programming
- concurrent logic programming
- higher order logic programming
- ...

Geschichte

Mathematische Logik

- Aristoteles 4.Jh. vor Chr.
- Gottfried Wilhelm Leibniz 17.Jh. Entscheidungsproblem
- Gottlob Frege 1879 Begriffsschrift
- Bertrand Russell 1902: Begriffsschrift widersprüchlich
- Russell, Whitehead 1910–1913 Principia Mathematica
- David Hilbert
- Gerhard Gentzen 1934 Sequenzenkalkül, Schnittelimination
- Kurt Gödel 1930 Vollständigkeitssatz
- Kurt Gödel 1931 Unvollständigkeitssatz, Unentscheidbarkeit

Geschichte

Automatisches Beweisen

- Robinson 1965 Resolutionskalkül

Prolog

- Programmation en Logique
  Programmieren in Logik
- Anfang der 70er Jahre: Colmerauer
- Warren's Abstract Machine 1983

Prolog: Einige der Einsatzgebiete

- Künstliche Intelligenz
- Automatisches logisches Schließen
- Expertensysteme, regelbasierte Systeme
- Datenbanken
- Automatische Programmsynthese, Programmverifikation
- Problemlösungs- und Planungssysteme
- Roboter
- Entwurfsysteme
- Sprachverarbeitung, Sprachverstehen, Übersetzen
- Rapid Prototyping
- Symbolisches Rechnen, Computeralgebra
- Computerspiele
- Web-Anwendungen

Was ist logische Programmierung?

Dinge, die beschrieben werden

- Objekte (Individuen, Gegenstände)
- Prädikate (Relationen, Beziehungen) zwischen Objekten

Programmierung

- Logisches Programm
  logische Beschreibung eines Teils der Welt
- Anfragen an das System

System der logischen Programmierung

- Ein automatischer Theorembeweiser
- soll beweisen, dass die Aussage der Anfrage logisch aus dem Programm folgt.

Beispiel

Dinge, die beschrieben werden

- Objekte: Steinbock, Gemse, Hai, Edelweiß, Enzian, Seetang, Gebirge, Meer
- Prädikate: „ist ein Tier“, „ist eine Pflanze“, „lebt in“

Welche Prädikate treffen auf welche Objekte zu?

Definition der Prädikate

- Der Steinbock ist ein Tier.
- Die Gemse ist ein Tier.
- Der Hai ist ein Tier.

- Das Edelweiß ist eine Pflanze.
- Der Enzian ist eine Pflanze.
- Der Seetang ist eine Pflanze.

- Der Steinbock lebt im Gebirge.
- Die Gemse lebt im Gebirge.
- Der Hai lebt im Meer.
- Das Edelweiß lebt im Gebirge.
- Der Enzian lebt im Gebirge.
- Der Seetang lebt im Meer.

Das Prolog-Programm — lebt.pl

```
ist_tier(steinbock).
ist_tier(gemse).
ist_tier(hai).

ist_pflanze(edelweiss).
ist_pflanze(enzian).
ist_pflanze(seetang).

lebt_in(steinbock,gebirge).
lebt_in(gemse,gebirge).
lebt_in(hai,meer).
lebt_in(edelweiss,gebirge).
lebt_in(enzian,gebirge).
lebt_in(seetang,meer).
```

Stelligkeit eines Prädikats

- Jedes Prädikat bezieht sich auf eine fixe Anzahl von Objekten.
- Diese Zahl heißt die Stelligkeit des Prädikats.

- `ist_tier` bezieht sich auf 1 Objekt (z.B. `steinbock`).
- Daher ist `ist_tier` 1-stellig.
- In Prolog heißt das Prädikat `ist_tier/1`.

- `lebt_in` bezieht sich auf 2 Objekte (z.B. `steinbock` und `gebirge`).
- Daher ist `lebt_in` 2-stellig.
- In Prolog heißt das Prädikat `lebt_in/2`.

Anfragen (queries)

Ist die Gemse ein Tier?

```
?- ist_tier(gemse).
yes
```

Ist die Gemse eine Pflanze?

```
?- ist_pflanze(gemse).
no
```

Ist der Hund ein Tier?

```
?- ist_tier(hund).
no
```

Closed world assumption

Prädikate treffen auf Objekte nur dann zu, wenn dies explizit im Programm steht oder logisch aus dem Programm folgt.

Anfragen

Ist der Hai ein Fisch?

```
?- ist_fisch(hai).
ERROR: toplevel: Undefined procedure: ist_fisch/1 ...
```

Einfaches Ziel

Prädikat$(Term_1, \ldots, Term_n)$

Beispiel

`ist_tier(gemse)` Prädikat: `ist_tier`, Term: `gemse`

Anfrage

`?-` $Ziel_1, \ldots, Ziel_n$ `.`

Variablen

Wer ist ein Tier?

```
?- ist_tier(Tier).
Tier = steinbock ;
Tier = gemse ;
Tier = hai.
?-
```

Objekte, Prädikate, Variablen

- `Tier` ist eine Variable.
- Variablen: mit großem Anfangsbuchstaben
- Eine Variable kann für etwas beliebiges stehen.
- Ein Objekt ist ein konkretes Ding.
- Ein Prädikat ist eine konkrete Relation.
- Objekte und Prädikate: mit kleinem Anfangsbuchstaben

Anfragen mit mehreren Variablen

Wer lebt wo?

```
?- lebt_in(Lebewesen,Lebensraum).
Lebewesen = steinbock, Lebensraum = gebirge ;
Lebewesen = gemse, Lebensraum = gebirge ;
Lebewesen = hai, Lebensraum = meer ;
Lebewesen = edelweiss, Lebensraum = gebirge ;
Lebewesen = enzian, Lebensraum = gebirge ;
Lebewesen = seetang, Lebensraum = meer.
?-
```

Welches Lebewesen X lebt im Lebensraum X?

```
?- lebt_in(X,X).
no
?-
```

- Eine Variable darf in einer Anfrage zweimal vorkommen.
- Dann steht sie beidemale für dasselbe Objekt.

Zusammengesetzte Anfragen

Welche Pflanze und welches Tier leben in welchem gemeinsamen Lebensraum?

```
?- ist_pflanze(Pflanze), ist_tier(Tier),
   lebt_in(Pflanze,Lebensraum), lebt_in(Tier,Lebensraum).
Pflanze = edelweiss, Tier = steinbock, Lebensraum = gebirge;
Pflanze = edelweiss, Tier = gemse, Lebensraum = gebirge ;
Pflanze = enzian, Tier = steinbock, Lebensraum = gebirge ;
Pflanze = enzian, Tier = gemse, Lebensraum = gebirge ;
Pflanze = seetang, Tier = hai, Lebensraum = meer.
?-
```

Das Komma (,) ist als und zu lesen. Konjunktion

Ein weiteres Beispiel

Dinge, die beschrieben werden

- Objekte: Franz, Max, Christine, Anna, Hans
- Prädikate (Relationen): männlich, weiblich, Kind von

Zwischen welchen Objekten bestehen welche Relationen?

Beschreibung der Relationen

- Max ist männlich.
- Franz ist männlich.
- Hans ist männlich.

- Christine ist weiblich.
- Anna ist weiblich.
- Eva ist weiblich

- Max ist Kind von Franz.
- Christine ist Kind von Franz.
- Max ist Kind von Anna.
- Christine ist Kind von Anna.
- Hans ist Kind von Christine.
- Eva ist Kind von Hans.

Das Prolog-Programm — verwandtschaft.pl

```
% maennlich(P): Die Person P ist maennlich.
maennlich(max).
maennlich(franz).
maennlich(hans).

% weiblich(P): Die Person P ist weiblich.
weiblich(christine).
weiblich(anna).
weiblich(eva).

% kind(K,P): K ist Kind der Person P.
kind(max,franz).
kind(christine,franz).
kind(max,anna).
kind(christine,anna).
kind(hans,christine).
kind(eva,hans).
```

Kommentare

- Von % bis Ende der Zeile
- Von /* bis */

Anfragen (queries)

Ist Christine Kind von Franz?

```
?- kind(christine,franz).
```

Für welche G,X,K ist K Kind von X, X Kind von G und G männlich?

```
?- kind(K,X), kind(X,G), maennlich(G).
K = hans,
X = christine,
G = franz
```

Fakten

Fakten

```
kind(max,franz).
```

Ein weiteres Beispiel:

Das Gleichheitsprädikat

```
gleich(X,X).
```

Regeln

G ist Großvater von K, wenn
K Kind von X,
X Kind von G
und G männlich ist.

Großvater

```
grossvater(G,K) :-
    kind(K,X),
    kind(X,G),
    maennlich(G).
```

`:-` ist zu lesen als wenn.

Person

```
person(P) :-
    maennlich(P).
person(P) :-
    weiblich(P).
```

Rekursion

- N ist Nachkomme von X, wenn
  N Kind von X ist.
- N ist Nachkomme von X, wenn
  K Kind von X und
  N Nachkomme von K ist.

Nachkomme

```
% nachkomme(N,X): N ist Nachkomme von X.
nachkomme(N,X) :-
    kind(N,X).
nachkomme(N,X) :-
    kind(K,X),
    nachkomme(N,K).
```

Das Prädikat `nachkomme/2` ist unter Bezug auf sich selbst definiert.