Programmierparadigmen: Logische Programmierung

Ivan Hell

8. Juni 2012

1 Einleitung

Bei der logischen Programmierung handelt es sich um ein Programmierparadigma, welchem die mathematische Logik zugrunde liegt. Diese Art der Programmierung wird vor allem in Bereichen wie künstliche Intelligenz, Expertensysteme, Datenbank-Management oder automatisierte Beweise verwendet.

2 Allgemein

Betrachtet man ein herkömmliches, imperatives Programm, so besteht dieses aus einer Menge von Instruktionen. Ein solches Programm beginnt mit dem Ausführen des ersten Befehls und arbeitet dann Schritt für Schritt alle weiteren Anweisungen ab. Es wird also vom Programmierer genau festgelegt, wie man zur Lösung eines Problems gelangt. Bei einem logischen Programm hingegen bestimmt der Programmierer nicht das „wie“, sondern hinterlegt eine Ansammlung von Bedingungen und Aussagen, welche als „Wissen“ betrachtet werden. Das Programm ist deshalb zunächst keinem eindeutigen Problem zugeordnet. Erst durch das Festlegen einer Problembeschreibung bzw. Frage, bekommt das Programm einen Zweck und das darin enthalten Wissen wird vom System zur Beantwortung der gestellten Frage verwendet. Ein logisches Programm könnte zum Beispiel folgende drei Aussagen enthalten:

```
1. Anakin ist Lukes Vater.
2. Anakin ist Leias Vater.
```

```
3. Zwei Menschen sind Geschwister,
      wenn sie den selben Vater haben.
```

Auf die Behauptungen, „Leia und Luke sind Geschwister.“ und „Anakin ist Lukes Vater“, würde das System durch Nutzung dieses Wissens, in beiden Fällen mit Ja, also wahr antworten. In Anbetracht dieser Tatsachen wird ersichtlich, dass logische Programmiersprachen, zur Gruppe der deklarativen, also beschreibenden Sprachen gehören. Im Allgemeinen kann man zwischen folgenden 3 Anweisungen unterscheiden:

- Fakten: Sie gelten als die einfachste Form einer Anweisung. Sie legen die Beziehungen zwischen Objekten fest und sind allgemein gültig. Betrachtet man das Beispiel von vorhin, so könnte man die darin enthaltene Aussage mit Hilfe folgender Fakten, logisch definieren:

  ```
  vater(anakin,luke).
  vater(anakin,leia).
  ```

  Damit wird definiert, dass die Beziehung *vater* sowohl zwischen den Objekten *anakin* und *luke*, als auch zwischen *anakin* und *leia* gilt. Demnach ist *anakin* in diesem Programm der Vater von *luke* und *leia*. Diese Beziehungen zwischen Objekten werden im Bezug auf die Prädikatenlogik auch Prädikate genannt.

- Anfragen: Mit Hilfe von Anfragen kann in der logischen Programmierung eine bestimmte Aussage auf ihren Wahrheitsgehalt überprüft werden. Sie überprüft anhand des Programms, ob eine Beziehung zwischen Objekten gilt oder nicht. Auf die beiden Anfragen:

  ```
  vater(anakin,han)?
  vater(anakin,luke)?
  ```

  würde das vorher definierte Programm jeweils mit flasch und wahr antworten. Um mächtigere Anfragen zu definieren, können auch in logischen Pogrammen Variablen verwendet werden. Diese Variablen beginnen, anders als die Objektnamen mit einem Großbuchstaben [3], und stehen für ein eindeutiges, jedoch noch nicht bekanntes Objekt. Durch das Absetzen der Anfrage „`vater(X,luke)?`“, wird das System sein „Wissen“ nach einem Objekt durchsuchen, zu welchem *luke*

in der Beziehung *vater* steht. Folglich wird das Programm, das Objekt *anakin* an die Variable X binden und somit dieses als Antwort liefern. Gilt das Prädikat für mehrere Objekte, so wird ein zufälliges zurückgegeben (abhängig vom System). Auch ist es möglich mehrere Anfragen, durch ein „Komma“ getrennt, miteinander zu verknüpfen (alle Teil-Anfragen müssen erfolgreich sein, damit die gesamte Anfrage als wahr bestätigt wird). Dabei werden alle verwendeten Variablen zwischen den Teil-Anfragen geteilt. Demnach gibt die Anfrage „`vater(X,luke),vater(X,leia)?`“ den Vater von *luke* und *liea* zurück.

- Regeln: Die dritte und wohl wichtigste Anweisung wird dazu verwendet, um neue Beziehungen zwischen Termen, von bereits definierten Prädikaten, festzulegen. Diese Relationen werden durch eine logische Implikation dargestellt:

  `geschwister(X,Y)` $\leftarrow$ `vater(Z,X),vater(Z,Y).`

  Die soeben definierte Regel bzw. Beziehung *geschwister*, validiert eine Aussage, bei welcher X und Y Geschwister sind. Dazu wird einfach überprüft, ob ein Objekt existiert, zu welchem sowohl X als auch Y in der Relation *vater* stehen. Wird nun zum Beispiel die Anfrage „`geschwister(luke,liea)?`“ an das System gestellt, so wird diese anhand des vorher definierten *Wissens* als wahr bestätigt.

Mithilfe dieser Anweisungen ist es möglich, das Konzept eines logischen Programms zu definieren, wobei gilt:

Definition: „Ein logisches Programm ist eine endliche Menge von Regeln.“ [3]

3 Geschichte

Die Geschichte der logischen Programmierung reicht bis in die 1960 Jahre zurück und ist eng mit der, der mathematischen Logik verbunden. Zur damaligen Zeit existierten bereits einige Prototypen von sogenannten Fragen-Beantwortenden-Systemen. Viele bedeutende Logiker versuchten daraufhin, die Idee einer deklarativen Sprache, sowie die Theorie der automatischen Deduktion und konstruktiven Logik zu vereinen. Unter diesen Umständen,

definierten Alain Colmerauer und Robert Kowalski im Jahre 1972 die erste logische Programmiersprache: Prolog (**Pro**grammation en **Log**ique). Prolog zählt heute zu den wohl bekanntesten logischen Programmiersprachen.

4 Beispiel in Prolog

Um zum Abschluss einen Eindruck davon zu bekommen, wie nun ein komplettes logisches Programm aussehen könnte, wurde im folgenden Beispiel das Lied „*99 Bottles of beer*“, in der Programmiersprache *PROLOG* implementiert:

```
beer:- beer(99).
beer(0):- write('No more bottles of beer on the wall, '),
        printBottles(0),write(' of beer.'),nl,
        write('Go to the store and buy some more, '),
        write('99 bottles of beer on the wall.'),!.
beer(N):- printBottles(N),write(' of beer on the wall, '),
        printBottles(N),write(' of beer.'),nl,
        NN is N-1,
        write('Take one down and pass it around, '),
        printBottles(NN),write(' of beer on the wall.'),
        nl,nl,beer(NN).
printBottles(1):- write('1 bottle'),!.
printBottles(0):- write('no more bottles'),!.
printBottles(N):- write(N),write(' bottles').
```

Literatur

[1] Chitta Baral and Michael Gelfond. Logic programming and knowledge representation. 1994.

[2] J. W. Lloyd. *Foundations of logic programming*. Springer Verlag, 1987.

[3] Leon Sterling und Ehud Shapiro. *The Art of Prolog*. MIT Press, 1994.

[4] Peter Rechenberg und Gustav Pomberger. *Informatik-Handbuch*. Carl Hanser Verlag, 2006.