Intelligenz für Ihren C 64!

Die Programmiersprache Prolog ist besonders geeignet, um »intelligente« Programme zu entwickeln. Sie ist in der »Künstlichen Intelligenz«-Forschung berühmt und wird zur Entwicklung von Expertensystemen gebraucht. Diese Sprache wurde für den C 64 um Ton- und Grafikbefehle erweitert.

Prolog 64 ist eine spezielle Prolog-Implementation für den C 64. Sie ist kompatibel zu den meisten Prolog-Interpretationen. Prolog 64 ist in das Betriebssystem des C 64 eingebettet und nutzt dessen 64 KByte RAM optimal aus. Spracherweiterungen unterstützen die Grafikmöglichkeiten, die Tonerzeugung und das Dateisystem des C 64.

Was ist Künstliche Intelligenz?

Ein »intelligenter« Computer soll sich unterhalten können, Sprache und Bilder verstehen, Probleme selbständig lösen und wissen, wie es auf der Welt so zugeht. Die Wissenschaft, die Computern dies alles beibringen will, nennt sich Künstliche Intelligenz (KI). Die zentrale Frage, um die es in der KI geht, ist folgende: »Wie kann das, was Menschen wissen, im Computer dargestellt und verarbeitet werden?« Das Schlagwort Wissensrepräsentation bezeichnet genau dieses Kernproblem. Die Probleme der Künstlichen Intelligenz können mit den bisherigen Programmiermethoden nicht mehr gelöst werden. Man braucht geeignete Methoden, um die Dinge der realen Welt (beispielsweise Personen, Gegenstände, Gesetzmäßigkeiten, Zusammenhänge) auf dem Computer darzustellen. Der Computer soll ja die Realität kennenlernen, denn nur wenn er über die Welt, in der die Menschen leben, Bescheid weiß, kann er »intelligent« agieren. Ein solcher Computer »weiß« zum Beispiel:
»Bäume sind Pflanzen.«
»Bäume sind grün.«

Damit hat er Informationen über Dinge, nämlich Bäume.
»Menschen brauchen Nahrung, weil sie sonst verhungern.«
»Autos fahren nur, wenn sie genug Benzin im Tank haben.«

Diese Gesetzmäßigkeiten muß man auch als Computer einfach kennen.
»Boris Becker ist ein bekannter Tennisspieler. Daher berichten die Zeitungen oft über ihn.«

Diese Information sagt etwas über eine Person (Boris Becker) aus und klärt zusätzlich einen Zusammenhang (weil er berühmt ist, schreibt man über ihn).

So wie eben beschrieben, kann Wissen über die reale Welt aussehen. Nun braucht man geeignete Methoden, um dieses Wissen auf einem Computer darzustellen. Daher wurden neue Sprachen und Konzepte entwickelt, mit denen diese komplexen Aufgaben zu lösen sind. Lisp und Prolog sind bekannte KI-Sprachen, die speziell für solche Zwecke entwickelt wurden.

Aber neue Programmiersprachen allein reichen nicht aus. Auch der Aufbau von Programmen mußte neu durchdacht werden. Ein Basic-Programm besteht aus den Computeroperationen auf der einen Seite. Auf der anderen Seite stehen die Eingabedaten, mit denen das Programm arbeitet. KI-Programme arbeiten nicht mehr mit Zahlen, sondern mit Informationen in Form von Regeln. Diese Regeln werden wie die Basic-Eingabedaten außerhalb des Programms in einer Datei zusammengefaßt. Eine Regel könnte so aussehen:
IF das Auto hat genug Benzin im Tank THEN es fährt

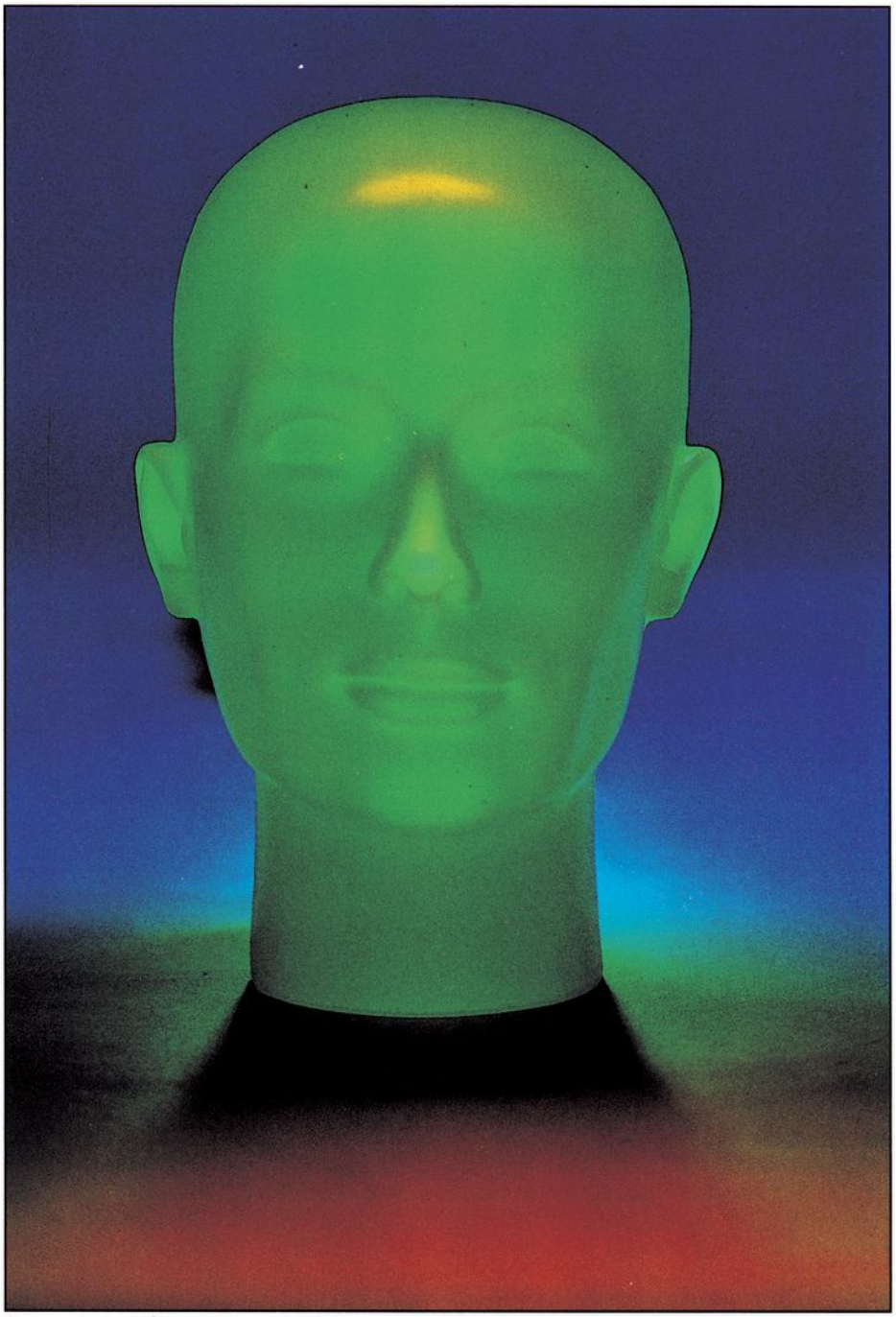
Diese IF-THEN-Form gibt es in Basic auch. In unserer Regel haben wir aber keinen Befehl, der sagt, was der Computer tun soll! Die Regel sagt nur aus, wie ein Auto reagiert, wenn es genug Benzin im Tank hat.

In einem Basic-Programm würde man im Programm den Befehl IF Benzin>0 THEN GOTO Autofährt schreiben. In einem KI-Programm werden solche Informationen aus dem Programm rausgezogen und in einer eigenen Datei abgelegt. Diese Ansammlung von Wissen nennt man »Wissensbasis« und ein KI-Programm, das darauf arbeitet, heißt »wissensbasiertes« Programm oder im Spezialfall Expertensystem. — »Wissensbasis« und »Expertensystem« sind ganz wesentliche Fachbegriffe in der KI-Forschung. Sie werden in Tabelle 1 kurz erläutert. — Die wissensbasierten Programmsysteme und die Expertensysteme gehören zu den bekanntesten Konzepten, mit denen sowohl Wissen über Objekte und Zusammenhänge als auch Metawissen (Regeln darüber, wie man Wissen anwendet) in Programmen dargestellt werden kann. Mit diesen Methoden kann auch sogenanntes »vages« Wissen verarbeitet werden. »Vage« ist alles, was man nicht mit 100prozentiger Sicherheit weiß. Man wirft zum Beispiel eine Münze und weiß:

Mit 50prozentiger Wahrscheinlichkeit werfe ich Kopf. Aber genauso wahrscheinlich ist es, daß eine Zahl geworfen wird. Expertensysteme zeichnen sich unter anderem dadurch aus, daß sie auf solch »vagem« Wissen arbeiten.

Was ist an Prolog so anders?

Prolog wurde etwa 1970 in Marseille entwickelt. Ähnlich wie Lisp, die wohl bekannteste Sprache der Künstlichen Intelligenz, unterscheidet Prolog sich grundlegend von Sprachen wie Basic und Pascal. Prolog ist ebenso wie Basic eine interaktive Sprache. Die Entwicklung und Ausführung von Prolog-Programmen erfolgt im Dialog mit dem



Computer. Das ist aber auch schon die einzige Gemeinsamkeit von Basic und Prolog. Denn diese Sprache beruht auf einem radikal neuen Konzept. Der Programmierer braucht sich nicht mehr um Algorithmen zur Lösung seines Problems zu kümmern, sondern muß genau angeben, worin sein Problem besteht.

In herkömmlichen Programmiersprachen, wie auch zum Beispiel in Basic, bestimmt der Programmentwickler die Reihenfolge der Computeroperationen. Er legt sie nämlich mit den Programmbefehlen fest. In Prolog-Programmen wird nicht mehr das »wie« spezifiziert, sondern das »was«. Prolog besitzt keine Sprachelemente, die festlegen, in welcher Reihenfolge der Computer die Programmoperationen ausführt. Solche Anweisungen sind in Basic IF/THEN, ELSE, FOR, WHILE und GOTO. Mit solchen Kontrollbefehlen sagen wir dem Computer »mache zuerst das, dann mache das«. Ein Prolog-Programm dagegen gleicht mehr einer ungeordneten Ansammlung von Wissen. Mit einfachen Wenn-Dann-Befehlen und mit Fakten werden Sachverhalte beschrieben. Dem Computer wird so gesagt, was er über seine »Welt« wissen muß. Man nennt solche Programmiersprachen, die dem Computer nicht sagen, in welcher Reihenfolge er eine Folge von Problemen bearbeiten soll, »nichtalgorithmisch«. In nichtalgorithmischen Sprachen wie zum Beispiel Prolog wird durch ein Programm nur das Problem beschrieben. Wir teilen dem Computer wahre Fakten (Tatsachen) über ein Problem mit und sagen ihm, wie er sie zu interpretieren hat. Und nun soll endlich an einem ganz einfachen Beispiel gezeigt werden, wie solche Fakten (Bild 1 gibt eine genauere Erklärung des Begriffs) in Prolog aussehen können.

Prolog lernt Tiere kennen.

Wir geben ein: »Ein Hund ist ein Tier.« »Eine Katze ist ein Tier.« und »Eine Kuh ist ein Tier.«

```
tier(hund).
tier(katze).
tier(kuh).
```

Der Punkt hinter jeder Zeile ist wichtig! Prolog erkennt daran das Ende einer Eingabe.

Nehmen wir an, unser Prolog-Programm »wüßte« nur diese drei Fakten, die wir ihm eingegeben haben. Wir fragen nun das Programm nach dem, was es weiß:
»Ist ein Hund ein Tier?«.

```
?-tier(hund).
```

Das Prologsystem antwortet mit:

```
yes.
```

»Ist eine Katze ein Tier?«

```
?-tier(katze).
yes.
```

»Ist ein Wolf ein Tier?«

```
?-tier(wolf).
no.
```

Auf die letzte Anfrage kann Prolog nur mit »no« antworten, da dem System ja noch nicht bekannt ist, daß der Wolf auch ein Tier ist. Ein »no« ist in diesem Sinne immer als ein »ich weiß es (noch) nicht« zu verstehen.

So läuft in etwa eine Prolog-Session ab. Eine Menge von Fakten und Regeln wird eingegeben, wie wir es in unserem Beispiel in ganz kleinem Rahmen getan haben. Die Regeln und Fakten können auch als Sätze (wie ein Basic-Programm) von einer Datei geladen werden. Danach kann der Benutzer Fragen an das System stellen, auf die Prolog im einfachsten Fall mit »yes« oder »no« antwortet. Dies ist natürlich noch keine anspruchsvolle Anwendung von Prolog. Die Fähigkeiten von Prolog sind sehr viel umfassender, als hier gezeigt werden kann.

Aber das folgende Programm zeigt anschaulich, wie die bekann-

ten »Türme von Hanoi« in Prolog implementiert werden können.

```
196              hanoi
197       Die Tuerme von Hanoi
198 */
210 hanoi (N) :-
211      moves(N,left,centre,right).
220 moves(0,_,_,_) :- ! .
230 moves(N,A,B,C) :-
240     M is N-1,
250     moves(M,A,C,B),
260     inform(A,B),
270     moves(M,C,B,A).
280 inform(X,Y) :-
281     write([move,a,disc, from,
        the,X,pole,to,the,Y,pole]).
282     nl.
```

Prolog wird vor allem dort eingesetzt, wo Symbole verarbeitet werden. Für numerische Datenverarbeitung, also Berechnungen und die Verarbeitung von Zahlen, ist diese Sprache nicht entworfen worden. Typische Anwendungen von Prolog sind:

- der Aufbau von Wissensbasen (Tabelle 1) für Expertensysteme oder intelligente Datenbanksysteme
- Verarbeitung natürlicher Sprache; sie umfaßt das Erkennen natürlicher Sprache und die Gesprächsführung durch das Programm
- Bilderkennung und -verarbeitung (Szenenanalyse)
- der Entwurf kompletter Expertensysteme (Tabelle 1)
- rapid prototyping (Tabelle 1)

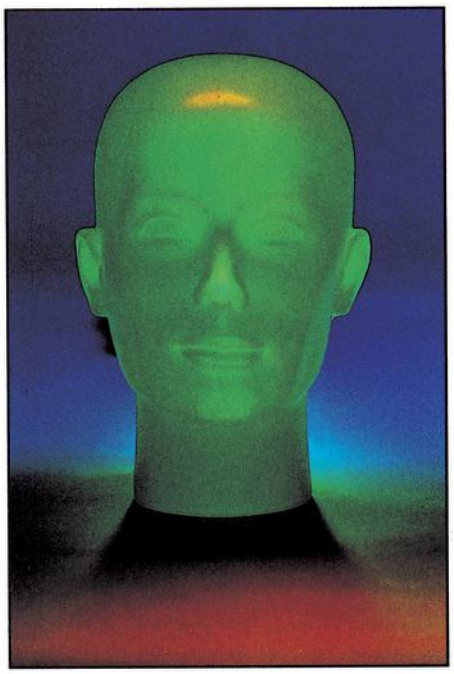
Bedienung und Handbuch

Prolog 64 bietet dem Programmierer eine gelungene Kombination der neuen Befehle dieser Programmiersprache und der Befehle, die jedem C 64-Besitzer bekannt sind. So kann man in einem Prolog-Lauf beliebige Basic-Programme ausführen oder nachträglich laden. Aus dem aktiven Prolog-System kann man jederzeit mit dem Befehl »EOF« oder dem EOF-Zeichen »SHIFT/PFUND« auf den Basic-Bildschirm zurückschalten. Mit »STOP/RESTORE« wird wieder zum Prolog-System zurückgeschaltet.

Die Schnittstelle zu Basic

Andersherum geht's auch: Prolog können Sie von einem Basic-Programm aus mit dem Befehl »SYS 49152« aufrufen. Will man Prolog von einem eigenen Programm aus laden, so geht das ebenfalls ohne große Probleme.

In 40 Sekunden wird das Prolog-System von der Diskette geladen (»LOAD "PROLOG",8« und »RUN«).



Intelligenz für Ihren C 64!

Nach dem »LOAD« können Monitor- und Klangfarben verändert werden. Nun sind noch genau 19703 Byte frei, nachdem Prolog 64 unter Basic geladen wurde. Der Speicher wird vom Prolog-System aufgeteilt. 16 KByte sind für Daten, 16 KByte für Grafik reserviert. Für den Stack werden 34,5 KByte (mit Grafik 21 KByte) und für den Grafik-RAM-Speicher werden 8 KByte belegt.

Das Handbuch zum Programm

Das Handbuch zum Programm bietet neben der Bedienungsanleitung nur eine kurze Beschreibung von Prolog. Wer Prolog lernen und in dieser Sprache Programme entwickeln will, muß sich zusätzlich das Standardwerk von Clocksin und Mellish anschaffen. In diesem Buch wurde 1981 das Kern-Prolog definiert und dieser sogenannte »Edinburgh«-Standard liegt den heutigen Prolog-Implementationen zugrunde. Auch Prolog 64 basiert auf dem Kern-Prolog.

Mitgelieferte Bibliotheken

Mitgeliefert wird eine Beispielsitzung, anhand derer man die ersten Versuche mit der neuen Programmiersprache relativ sicher durchführen kann. Dies ist auch nötig! Prolog ist halt völlig anders als die üblichen Programmiersprachen und man muß sich erst an seine Besonderheiten (zum Beispiel: jede Eingabe muß mit einem Punkt abgeschlossen werden) gewöhnen. Jeder, der lange in Basic (oder anderen algorithmischen Sprachen wie Pascal oder Fortran) programmiert hat, wird anfangs große Schwierigkeiten haben, sich auf die neue Programmierweise in Prolog einzustellen, weil er noch »in Basic denkt«.

Auf der mitgelieferten Diskette finden sich Bibliotheken für Grammatikregeln, grafische Routinen, Spritedefinitionen für das Demo-Programm, Musikroutinen, verschiedene mathematische Programme, Mengenoperationen und Suchverfahren, einen Precompiler für Grammatikregeln und einen Übersetzer von Prädikatenlogik in Klauselform (siehe Clocksin/Mellish), komfortable Ein-/Ausgabeoperationen und Faktenverwaltung auf Floppy und natürlich für ein Demo-Programm. Zusammen mit einer Bibliothek für einige Prolog-Befehle umfassen die Beispielprogramme 43,4 KByte. Die Listings dieser Beispiel-Files sind im Anhang des Handbuchs abgedruckt. Es ist alles da, was man braucht, um eine fremde Sprache kennenzulernen. Nun muß man nur noch loslegen und das Prolog-System ausprobieren.

Prolog für Sie!

Prolog ist eine sehr interessante Sprache für alle, die sich näher mit der Künstlichen Intelligenz beschäftigen wollen. Jeder C 64-Besitzer, der sich in diesem Bereich der Zukunft engagieren will, wird die Möglichkeit begrüßen, die Prolog 64 ihm bietet: Auf dem Computer, den er kennt, dem Commodore 64, kann er sich mit einer berühmten KI-Sprache anfreunden. Prolog 64 umfaßt die Sprachmöglichkeiten, die auch den Prolog-Programmierern auf Großrechnern zur Verfügung stehen. Für Verspielte sind die Grafik- und Tonerzeugungsmöglichkeiten des C 64 voll verfügbar. Und man kann jederzeit Basic-Programme und damit auch Routinen in Maschinensprache in Prolog-Programme einbauen! Zum Kennenlernen dieser Sprache ist die Prolog-Version Prolog 64 ideal. Und an Speicherplatzprobleme dürfte jeder C 64-Besitzer gewöhnt sein. Denn große Programmsysteme kann man so natürlich nicht entwickeln. Wenn Sie Interesse an Künstlicher Intelligenz haben, dann schreiben Sie uns!

Der erste Prolog-Interpreter für den C 64 ist mit deutschem Handbuch für 289 Mark erhältlich. (cg)

Info: Brainware GmbH, Kirchgasse 24, 6200 Wiesbaden, Tel: (0612l) 372011

Literatur: Clocksin und Mellish, Programming in Prolog, Springer Verlag, Berlin, Heidelberg, New York, 1985, ISBN 3-540-11046-1, 44 Mark

Expertensystem (expert system):
Expertensysteme sind »intelligente« Programme aus dem Bereich der Künstlichen Intelligenz. Ihre Aufgabe ist es, wie ein menschlicher Experte über ein bestimmtes Gebiet (möglichst) vollständig Bescheid zu wissen. Solche Anwendungsgebiete können in der Medizin (Diagnose, Behandlung von Tropenkrankheiten), der Technik (Konstruktion von Automotoren, Aufbau von Rechnerkonfigurationen) oder in der Geschichte liegen. Jedes Gebiet, in dem es menschliche Spezialisten gibt, ist geeignet.

Expertensysteme bestehen aus mehreren Komponenten. Die Wissensbasis enthält das Expertenwissen, das auf geeignete Weise im Computer dargestellt wird. Der Aufbau dieser Wissensbasis ist das Kernproblem, das sich beim Aufbau eines Expertensystems stellt. Nicht nur Buchwissen soll aufgenommen werden, sondern auch Erfahrungswissen, das, was man erst durch langjährige Praxis an Tricks und Kniffen lernt. Ein Expertensystem arbeitet auf dieser Wissensbasis und zwar im Dialog mit seinem Benutzer. Diese Dialogkomponente ist ebenfalls typisch. Der Benutzer stellt dem Programmsystem eine Frage (»Welche Krankheit hat der Patient, wenn folgende Symptome auftreten: ...?« oder »Ich will für meine Schreinerei einen Computer und Software anschaffen. Was braucht man und was gibt es?« Nachdem der Computerexperte aufgrund seines gespeicherten Wissens und im Gespräch mit dem Fragenden alle nötigen Informationen gesammelt und eine Lösung des Problems gefunden hat, kann der Benutzer von der Erklärungskomponente Gebrauch machen. Das Expertensystem erklärt jeden einzelnen Schritt seiner Schlußfolgerungen. Dies sind die wesentlichen Bestandteile eines Expertensystems: eine Wissensbasis, die auch vages Wissen enthält, die Dialog- und die Erklärungskomponente.

rapid prototyping:
Dies ist eine Methode, die schon beim Entwickeln von Programmen logische Fehler im späteren Programm verhindern soll. Programme werden auf einer abstrakteren Ebene, als dies die Programmierebene ist, spezifiziert. Die Spezifikationssprache ist so konzipiert, daß Fehler schnell erkannt und oft schon automatisch behoben werden können. Ist die Spezifikation des Programms dann fehlerfrei, folgt die (teilweise wieder automatische) Programmierung in konventioneller Sprache. Auf diese Weise soll garantiert fehlerfreie Software entwickelt werden. Dies spart Kosten für Wartung und Service.

Wissensbasis (knowledge base):
In einer Wissensbasis werden Informationen gespeichert. Die übliche Form, in der sie dargestellt werden, ist:
WENN (IF) ... DANN (THEN) ...
WENN bestimmte Bedingungen zutreffen DANN kann man daraus (mit einer bestimmten Wahrscheinlichkeit) schlußfolgern, daß eine bestimmte Situation vorliegt, also:
»WENN der Patient raucht, DANN ist die Wahrscheinlichkeit, daß er zu dick ist, 5 Prozent niedriger als sonst«.

Tabelle 1. Fachtermini

Fakten sind Tatsachen über Objekte und ihre Beziehungen zueinander. Namen von Gegenständen, Personen und so weiter (Petra, Prolog) werden in Fakten kleingeschrieben. Die Beziehung oder die Aussage über Objekte steht vor der Klammer (sind, kennt). Geben wir zum Beispiel folgende Fakten über Prolog und Computerfans ein:

```
pr_sprache(prolog).
```

»Prolog ist eine Programmiersprache.«

```
kennt(petra,logo).
```

»Petra kennt Logo.«

```
kennt(petra arnd).
sind(arnd,petra,c_fans).
```

»Arnd und Petra sind Computerfans.«

Fragen sehen genauso aus wie Fakten, vor die »?-« gesetzt wurde. Wenn eine Frage an Prolog gestellt wird, durchsucht das System die Datenbank, die alle bekannten Fakten enthält. Prolog sucht ein Fakt, das der Frage entspricht. Existiert ein solches Fakt, dann antwortet Prolog auf die Frage des Benutzers mit »yes«, sonst mit »no«. Beispiel:

```
?-kennt(dr_bobo,indiana_joe).
no
```

»Kennt Dr. Bobo (den Hacker) Indiana Joe?« Prolog weiß nur das, was wir ihm oben eingegeben haben und sagt: Nein.

```
?-kennt(petra,logo).
yes
```

»Kennt Petra Logo?« Prolog sagt: Ja.

Variablen verwendet man in Fragen, um (alles) zu erfahren, was das Prolog-System über ein bestimmtes Objekt weiß. Variablen beginnen mit einem Großbuchstaben. Eine solche Variable heißt zum Beispiel »X« oder »Diesisteinbeliebigervariablenname«. Eine Variable bezeichnet kein Objekt. Sie wird dann verwendet, wenn man etwas sucht, das man nicht genau bezeichnen kann. Nehmen wir die Variable X und fragen, was Petra alles kennt (X bezeichnet das, was Petra kennt):

```
?-kennt(petra,X).
X=logo
```

ist die Antwort. Gibt man nach dieser ersten Antwort ein »;« (das logische »oder«) ein, so sucht das Prologsystem nach weiteren Objekten. Die nächste Antwort ist dann

```
X=arnd
```

Geben wir einfach »Return« ein, dann wird die Suche beendet.

Wenn Prolog eine Frage gestellt wird, die eine Variable enthält, durchsucht das Prolog-System alle seine Fakten nach einem Objekt, das die Variable ersetzen kann.

Konjugationen sind Verknüpfungen durch ein logisches »und«. Sie werden verwendet, wenn Fragen über kompliziertere Beziehungen zwischen Objekten gestellt werden sollen. Beispiel:
»Wer kennt Logo und Prolog?«
In Prolog heißt das:

```
?-kennt(X,logo),kennt(X,prolog).
```

Die Variable X steht für die Person, die wir suchen. Durch das »,« (= und) werden die Teile unserer Frage verknüpft. In unserer kleinen Beispieldatenbank finden wir leider niemanden, der beide Sprachen kennt. Aber auf die Frage
»Wer kennt Arnd und (die Programmiersprache) Logo?«:

```
?-kennt(X,arnd),kennt(X,logo).
```

findet Prolog in unserem kleinen Beispiel die Antwort:

```
X=petra
```

Regeln braucht man, wenn eine Tatsache für mehr als einen Fall gelten soll. Beispiel:
Wir wissen, daß Dr. Bobo das C 64-Spiel Summer Games kennt. Aber er kennt auch alle anderen Computerspiele, die auf dem C 64 laufen. Das heißt in Prolog:
»Wenn ein Spiel auf dem C 64 läuft, dann kennt Dr. Bobo es ganz sicher.«

```
läuft(Spiel,c-64):-kennt
(dr_bobo,Spiel).
```

»Daraus folgt« wird in Prolog durch »:-« bezeichnet.
Eine kompliziertere Regel ist die folgende:
»(x'*y + x*y') ist eine Ableitung von x*y, wenn x' Ableitung von x ist und y' Ableitung von y.« Die entsprechende Prolog-Regel ist:

```
ableitung(X*Y,X1*Y+Y1*X):-
ableitung(X,X1),
ableitung(Y,Y1).
```

Aus solchen Regeln und den oben beschriebenen Fakten besteht ein Prolog-Programm.

Backtracking ist eine Besonderheit von Prolog. Backtracking bedeutet »Zurückgehen und einen neuen Lösungsweg suchen«. Da ein Prolog-Programm aus vielen Regeln besteht, kann es mehrere Möglichkeiten geben, für eine Variable einen Wert zu finden. So kann das Prolog-System auf der Suche nach einer Lösung in einer Sackgasse landen. Prolog kann solche Sackgassen erkennen und wieder verlassen, indem der bisher gefundene Lösungsweg bis zur letzten Alternative rückgängig gemacht wird. Nun wird eine andere Möglichkeit ausprobiert. Ist auch diese nicht erfolgreich, wird die nächste Alternative ausprobiert, bis die Lösung gefunden ist.

Ein- und Ausgabe sind nützlich, wenn das Programm eine »Unterhaltung« mit dem Benutzer selbst beginnen soll. Haben wir zum Beispiel eine Datenbank programmiert, so muß der Benutzer bei jedem Schritt gefragt werden, was als nächstes gemacht werden soll.

Der Befehl put druckt das Zeichen, dessen ASCII-Code in Klammern angegeben wurde:

```
?-put(104),put(101),put(108),put(108),
put(111).
hello
```

ist das Ergebnis des Prologsystems.

Bild 1. Die elementarsten Grundlagen von Prolog