Schmidt, Reinhart; Ralfs, Dirk

KPS-Prolog: E. Problemlösungssystem auf Basis von Turbo-Prolog

Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, No. 225

Provided in Cooperation with:
Christian-Albrechts-University of Kiel, Institute of Business Administration

Nr. 225

KPS/Prolog
- Ein Problemlösungssystem auf Basis von Turbo Prolog -

Reinhart Schmidt
Dirk Ralfs

Dezember 1988

Prof. Dr. Reinhart Schmidt
cand. rer.nat. Dirk Ralfs

Institut für Betriebswirtschaftslehre
der Christian-Albrechts-Universität
zu Kiel

Inhaltsverzeichnis

1. Einleitung

Die Entwicklung wissensbasierter Systeme befindet sich in einem stürmischen Aufschwung. Dies betrifft zunächst das Anwachsen der Shells (Programmierumgebungen), die eine benutzerfreundliche Erstellung und Nutzung eines Expertensystems ermöglichen (vgl. Überblicke in [4] [15] [25]. Es betrifft weiterhin Anwendungen in den verschiedensten Bereichen - vom Prototyp bis zum voll einsatzfähigen System (vgl. die Darstellungen bei Mertens et al. [13].

Die Leistungsfähigkeit von Shells ist im Hinblick auf die Erfüllung einzelner Anforderungen recht unterschiedlich. Es kommt also entscheidend darauf an, ob das Anforderungsprofil des Anwendungsgebietes mit dem entsprechenden Profil der Shell in Übereinstimmung gebracht werden kann. Unter dem Gesichtspunkt der Anpassungsfähigkeit an neue Aufgabenstellungen, vor allem im Bereich der Forschung über neue Anwendungen wissensbasierter Systeme, ist es zweckmäßig, wenn Entwickler und Wissensingenieure die Möglichkeit haben, in einer Programmiersprache Systemkomponenten zu verändern oder neue zu schaffen.

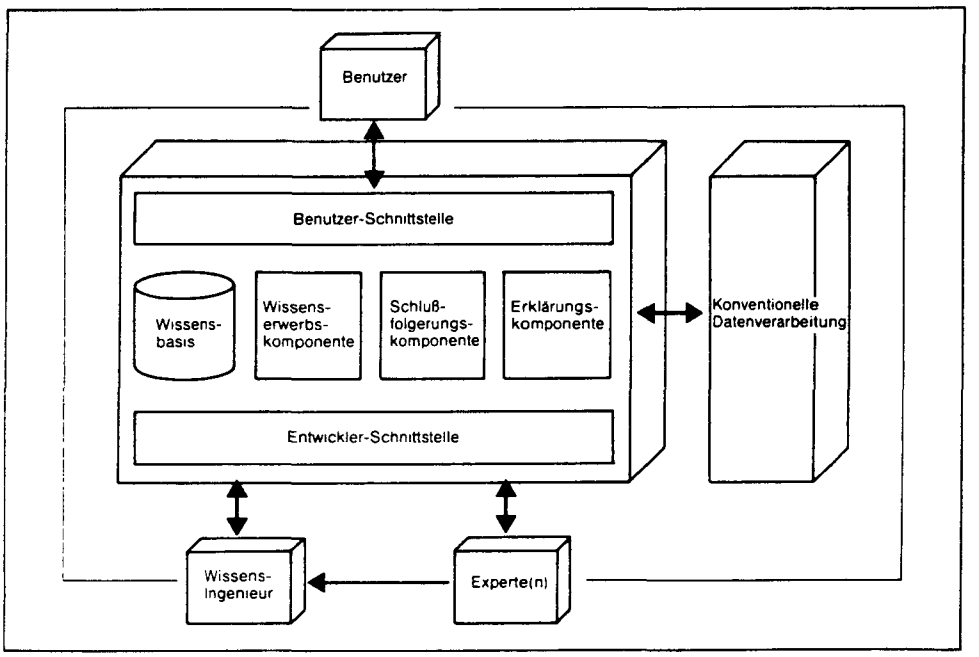
Am Institut für Betriebswirtschaftslehre der Universität Kiel wurde deshalb nach Sichtung der Eigenschaften vorhandener Shells ein eigenes System konzipiert und implementiert. Das entstandene System KPS/Prolog zeichnet sich durch eine besondere Erweiterungsfähigkeit seiner Komponenten seitens des Wissenschaftlers wie auch durch seine Benutzerfreundlichkeit für den Wissensingenieur aus.

Im folgenden wird dazu zunächst auf den Aufbau von Expertensystemen eingegangen. Es wird sodann der Aufbau des entwickelten Systems beschrieben. Schließlich wird der Einsatz als Problemlösungssystem diskutiert.

2. Aufbau von Expertensystemen

2.1. Elemente eines Expertensystems

Die Elemente eines Expertensystems sind in Abb. 1 dargestellt.



Quelle: IBM Deutschland GmbH.

Abb. 1: Bestandteile eines wissensbasierten Systems

Den Kern bildet die Wissensbasis, die das dem System angepaßte Expertenwissen enthält. Die Wissenserwerbskomponente ermöglicht die Zuführung von neuem Wissen. Die Schlußfolgerungskomponente (Inferenzkomponente) erlaubt es, Schlüsse zu ziehen und nach Lösungen zu suchen. Die Erklärungskomponente dient der Erläuterung von Fragestellungen und Fragefolgen sowie der Begründung von Antworten. Zum Umsystem sollten drei Schnittstellen vorhanden sein:

1. Die Entwickler-Schnittstelle erlaubt es dem Wissensingenieur in mehr oder weniger benutzerfreundlicher Weise, die Wissensbasis zu füllen und die genannten

Komponenten zu ändern bzw. zu ergänzen. Dazu gehört auch, daß die Wissenseingabe so weit möglich - auf Konsistenz geprüft wird.

2. Die Benutzer-Schnittstelle gestattet es, daß ein in dem Problemkreis mehr oder weniger sachverständiger Benutzer, der obendrein über keinerlei Programmierkenntnisse verfügt, von dem System bei Problemlösungen unterstützt werden kann.

3. Die Schnittstelle zur konventionellen Datenverarbeitung ermöglicht die Nutzung vorhandener Programm- und Datenbestände. Diese Schnittstelle hat eine sehr große Bedeutung, weil sich in den Anwendungsbereichen umfangreiche Daten- und Programmbestände gebildet haben, die prinzipiell integriert werden können und sollten.

Die Entwicklung von KPS/Prolog wurde von der Forderung veranlaßt, die Entwickler-Schnittstelle sowie die Schnittstelle zur konventionellen Datenverarbeitung nach eigenen Vorstellungen und damit sehr flexibel gestalten zu können. Außerdem sollte bei der Benutzer-Schnittstelle Flexibilität dahingehend geschaffen werden, daß von einer stärker formatierten Artikulationsform des Benutzers tendenziell auf einen Dialog in natürlicher Sprache übergegangen werden kann.

2.2. Formen der Wissensrepräsentation

Die Literatur unterscheidet vor allem drei Ansätze, wie das Wissen in wissensbasierten Systemen repräsentiert werden kann [6] [25]:

1. Der Frame-Ansatz ist durch die inklusive Unterordnung von Objekten, die bestimmte Attribute (Merkmale) mit bestimmten Werten (Merkmalsausprägungen) aufweisen, gekennzeichnet. Dabei können Eigenschaf-

ten übergeordneter Objekte ganz oder teilweise nach unten vererbt werden.

2. Semantische Netzwerke bilden das Wissen in Form von Knoten, die über Kanten miteinander verbunden sind, ab. Die wesentlichen Verknüpfungen sind dabei die Zugehörigkeit eines Objektes zu einer Objektklasse ("ist ein") und die Eigenschaft eines Objektes ("hat ein").

3. Der regelbasierte Ansatz beschreibt das Wissen in Form einer Auflistung von Bedingungen und Konsequenzen.

Es erscheint wünschenswert, verschiedene Formen der Wissensrepräsentation miteinander zu verbinden, um die Vorteile der jeweiligen Darstellung auszunutzen. Dies ist in KPS/Prolog versucht worden.

2.3. Turbo Prolog als Implementierungssprache

Prolog (Programming in Logic) ist keine prozedurale, sondern eine deskriptive Programmiersprache (vgl. zu Prolog vor allem [2] [18] [21] [24]). Es wird nicht der Lösungsweg angegeben, sondern die Lösungseigenschaften werden mittels Fakten und Regeln beschrieben. Dies stellt einen großen Vorteil bei der Erstellung von Expertensystemens dar, weil der Programmierer sich - mit Ausnahme des reinen Dialogablaufes - tatsächlich auf die Beschreibung des zu lösenden Problems beschränken kann. Die Lösungssuche übernimmt Prolog im Wege der Rückwärtsverkettung (Back Tracking). Schlußfolgerungen werden dabei aufgrund von Regeln gezogen, die einerseits gespeichertes Wissen, andererseits formuliertes Problem beinhalten.

Einen großen Vorteil von Prolog stellt die Eigenschaft dar, daß eine Verarbeitung von Listen (z. B. Listen von Attributen und entsprechenden Werten) unterstützt wird.

Der Vorteil einer Programmierung in Prolog liegt nicht nur in der theoretischen Eleganz, der Mächtigkeit von Prädikaten und der damit verbundenen Übersichtlichkeit, sondern auch in dem ständigen Zugang zu dem Inneren des Systems über die darunterliegende Programmiersprache Prolog selbst.

Turbo Prolog (vgl. dazu [3] [5] [8]-[12] [14] [17] [19] [22] [23]) als eine PC-Version erlaubt eine weitgehende Modularisierung des Programmsystems. Es ist offen gegenüber Änderungen, was den Vorteil hat, daß die Wissensbasis dynamisch erweitert werden kann. Außerdem kann auf externe Dateien zugegriffen werden, und andere Programme - auch in anderen Programmiersprachen - können eingebettet werden. Damit kann Turbo Prolog zugleich einen umfassenden Rahmen für die Ver- waltung und Nutzung unterschiedlicher Programmbestände bilden.

Um zu einer Lösung eines Problems in Turbo Prolog zu kommen, gibt es zwei Möglichkeiten:

1. Das Beweisziel ("Goal") kann vom Benutzer direkt im Dialog eingegeben werden. Dies hat den Vorteil höchster Flexibilität des zu beweisenden Sachverhalts (zu lösenden Problems), hat aber den Nachteil, daß der Benutzer die Prolog-Syntax beherrschen und die speziellen Namensgebungen des Programms kennen muß.

2. Die Schaffung eines programminternen Abschnitts "Goal" vermeidet die genannten Nachteile. Dabei kann die Spezifikation des zu führenden Beweises über ein oder mehrere Menüs abgefragt werden, oder der Benutzer kann die Zielanforderungen verbal artikulieren. Im letzteren Fall können Kommandos oder Kommandofolgen gegeben werden, oder es kann je nach Leistungsfähigkeit des Systems versucht werden, eine Interpretation von fast natürlicher Sprache zuzulassen.

Die Mächtigkeit von Prolog verbunden mit der Flexibilität durch den Einsatz der speziellen Sprache Turbo Prolog hat es erlaubt, KPS/Prolog im Wege eines Prototyping kontinuierlich und relativ schnell zu erstellen. Auch kann auf vorhandene Elemente von Entscheidungsunterstützungssystemen (DSS) oder auf komplette Prototypen, die am Institut für Betriebswirtschaftslehre vorhanden sind, im Wege einer Integration zurückgegriffen werden, so daß der Funktionsumfang im Sinne eines betriebswirtschaftlichen Problemlösungssystems schnell ansteigen kann.

3. Aufbau des Systems KPS/Prolog

Wissensrepräsentation, Wissenserwerb und Erklärungen von Lösungen sind in KPS/Prolog aufgrund eigener Vorstellungen gestaltet worden. Es ist versucht worden, den Frame-Ansatz mit dem regelbasierten Ansatz zu verbinden und gleichzeitig Elemente eines semantischen Netzwerkes zu berücksichtigen.

3.1. Die Struktur des deklarativen Wissens

Die Repräsentation des deklarativen Wissens in KPS/Prolog wird anhand von Abb. 2 veranschaulicht.

Das gesamte Wissen einer Domäne wird als in Bereiche gegliedert unterstellt. Diese Bereiche können sich auf mehr oder weniger umfangreiche Wissenskomplexe - je nach Entscheidung des Benutzers - beziehen. Die Bereiche können einen Bereichsnamen tragen. Das Wissen einzelner oder ggf. mehrerer Bereiche wird in externen Dateien gespeichert und je nach Wahl des bei der Problembeschreibung angesprochenen Bereichskomplexes in den Kernspeicher eingelesen. Bereiche können hierarchisch gegliedert sein, wodurch klassische Formen der Wissensrepräsentation - man denke an die Biologie oder an die Gliederung von Lehrbüchern - gut abgebildet werden können.

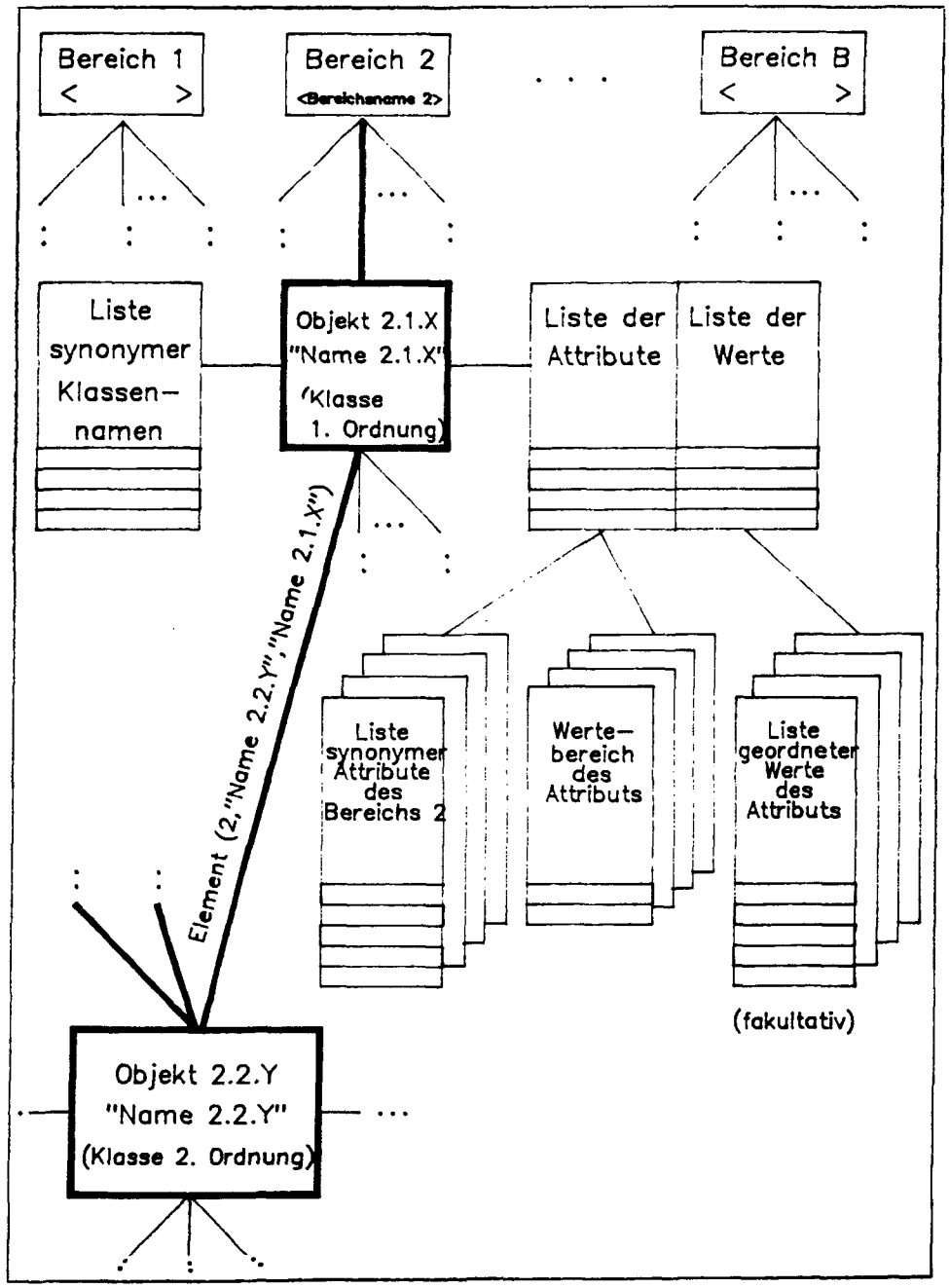


Abb. 2: Struktur des deklarativen Wissens in KPS/Prolog

Das Wissen eines Bereichs besteht aus Objekten und ihren Eigenschaften sowie eventuell einer hierarchischen Bereichsuntergliederung. Die Objekte weisen einen Objektnamen auf. Damit der Benutzer seine eigene Ausdrucksweise für Objekte verwenden kann, kann es für jedes Objekt eine Liste synonymer Namen geben. Dies ist besonders wichtig, wenn das wissensbasierte System den Benutzer ohne viele Rückfragen verstehen können soll.

Das Konzept synonymer Bezeichnungen erlaubt zugleich die Übersetzung in eine andere natürliche Sprache, allerdings mit der Einschränkung, daß es sich hier nur um Bezeichnungen handelt.

Jedem Objekt kann eine Liste von Attributen (Merkmalen) zugeordnet werden. Zu jedem Attribut einer solchen Liste kann es eine Liste synonymer Attribute - bezogen auf den speziellen Bereich - geben. Dadurch wird es möglich, daß der Benutzer bereichsspezifische Sprechweisen für Attribute benutzen kann, ohne daß die Funktion von Attributen als Schnittstelle zwischen unterschiedlichen Wissensbereichen ausgeschlossen wird.

Jedem Attribut eines Objekts ist ein Wert zugewiesen. Die Wertangabe richtet sich dabei nach dem Wertebereich des zugehörigen Attributs, es kann sich um nichtnumerische oder numerische Werte handeln. Numerische Werte sind in einer natürlichen Ordnung; für nichtnumerische Werte kann angegeben werden, ob sie ungeordnet oder geordnet sein sollen. Über eine Liste von Werten kann für Attribute mit nichtnumerischem Wertebereich eine Ordnung definiert werden. Dadurch wird ein Vergleich von qualitativen Ausprägungen ermöglicht.

Undefinierte Werte könnten grundsätzlich dadurch zugelassen werden, daß Objekte mit undefinierten Werten bei der Lösungssuche übergangen werden. Auch könnte anstelle einer Ausprägung "undefiniert" ein Programmname stehen, der es erlaubt, den Wert über ein separates Programm zu ermitteln.

Ein Objekt kann im Sinne einer Unterordnung Element eines anderen Objekts sein. Dabei sind innerhalb eines Bereichs auch mehrfache Unterordnungen möglich, so daß "Kind" Element von "Mutter" und "Vater" sein kann. Die (automatische) Vererbung von Eigenschaften auf untergeordnete Elemente ist in dem System bisher nicht realisiert, weil sich bei einer Mehrfachunterordnung Vererbungsprobleme ergeben können. Diese Probleme könnten

allerdings über Prioritätsregeln gelöst werden oder zu dem Wert "undefiniert" führen.
Die Wiedergabe des deklarativen Wissens in Prolog-Syntax ist Abb. 3 zu entnehmen. Für die Bereiche <Person>, <Verkehrsmittel> und <Reisezweck> werden Attribute definiert. Anhand des Prädikates "attribut" kann man dort erkennen, welche Attributnamen in jedem der drei Bereiche vorkommen. Für Verknüpfungen zwischen Bereichen ist von Bedeutung, daß ein bestimmter Attributname in mindestens zwei Bereichen vorkommt. Die Objekte der drei Bereiche sind unter dem Prädikat "objekt" mit ihren Namen verzeichnet. Außerdem finden sich in der dem Objektnamen folgenden Liste die Werte der Attribute des Objekts; dabei muß die Reihenfolge in der Liste der Reihenfolge in dem entsprechenden Prädikat "attribut" entsprechen.

```
ebenezahl(1)
bereichzahl(3)

ebenetext(1,"-")
bereichtext(1,"Person")
bereichtext(2,"Verkehrsmittel")
bereichtext(3,"Reisezweck")

attribut(1,1,["Status","Alter","Geld"])
attribut(1,2,["Geld","Alter","Entfernung","Tempo"])
attribut(1,3,["Entfernung","Tempo","Status"])

objekt(1,1,"Inge",["Schüler","12","wenig"])
objekt(1,1,"Hans",["Student","22","mittel"])
objekt(1,1,"Otto",["Rentner","62","viel"])
objekt(1,1,"Anna",["Kind","08","wenig"])
objekt(1,1,"Lisa",["Angestellte","40","viel"])
objekt(1,1,"Paul",["Arbeiter","51","mittel"])
objekt(1,2,"Fahrrad",["wenig","06","klein","schnell"])
objekt(1,2,"Auto",["viel","18","mittel","schnell"])
objekt(1,2,"Bahn",["mittel","00","groß","langsam"])
objekt(1,2,"Bus",["mittel","00","mittel","mittel"])
objekt(1,2,"Flugzeug",["viel","00","groß","schnell"])
objekt(1,3,"Kurzurlaub",["klein","langsam","Student"])
objekt(1,3,"Fernreise",["groß","schnell","Angestellte"])
objekt(1,3,"Arbeit",["mittel","mittel","Arbeiter"])
objekt(1,3,"Einkauf",["klein","langsam","Rentner"])
objekt(1,3,"Schule",["klein","schnell","Schüler"])
objekt(1,3,"Uni",["mittel","mittel","Student"])

wertebereich("Status","nichtnumerisch","ungeordnet")
wertebereich("Alter","numerisch","geordnet")
wertebereich("Geld","nichtnumerisch","geordnet")
wertebereich("Entfernung","nichtnumerisch","geordnet")
wertebereich("Tempo","nichtnumerisch","geordnet")

ordnung("Entfernung",["klein","mittel","groß"])
ordnung("Tempo",["langsam","mittel","schnell"])
ordnung("Geld",["wenig","mittel","viel"])

synonym(1,"Geld",["Einkommen"])
synonym(2,"Geld",["Kosten"])
synonym(1,"Alter",["Lebensalter"])
synonym(2,"Alter",["Mindestalter"])
synonym(2,"Bahn",["Eisenbahn","Bundesbahn"])
synonym(3,"Uni",["Universität"])
```

Abb. 3: Speicherung von Fakten als in Turbo Prolog eingebettete Daten

Die Wertebereiche für jedes Attribut werden in dem Prädikat "wertebereich" spezifiziert. Durch das Prädikat "ordnung" wird für die Attribute, die nichtnumerisch und geordnet sind, eine Liste der geordneten Werte angegeben.
Unter dem Prädikat "synonym" wird in dem Beispiel für das Attribut "Alter" für den Bereich 1 das Synonym "Lebensalter", für den Bereich 2 das Synonym "Mindestalter" eingeführt. Die Liste der synonymen Attributnamen besteht in diesem Fall nur aus einem Element, bei dem Prädikat "synonym" für das Objekt "Bahn" weist die Liste jedoch beispielhaft zwei Elemente auf.

KPS/Prolog geht davon aus, daß ein Name eindeutig zuzuordnen ist. Polysemantische Wörter müssen also zu eindeutigen Namen gemacht werden. Dies ist grundsätzlich möglich, indem benutzerspezifische oder benutzergruppenspezifische Wissensbasen geschaffen werden. Voraussetzung dafür ist die vorherige Identifikation des Benutzertyps durch das System.

3.2. Die Struktur des prozeduralen Wissens

Unter prozeduralem Wissen wird in KPS/Prolog das Wissen verstanden, das über Regeln zusammengefaßt der Problemlösung dient. Zu lösende Probleme sind also durch Regeln in Prolog-Syntax zu repräsentieren. Der Wissensingenieur braucht jedoch diese Syntax nicht zu beherrschen, weil in dem System die Regeln als Daten verwaltet werden und so mit Hilfe der Wissenserwerbskomponente Regeln hinzugefügt oder abgeändert werden können.

In allen Regeln dieser Art werden Beziehungen zwischen verschiedenen Bereichen des Wissens hergestellt. Über Operatoren werden Werte entsprechender Attribute von Objekten aus verschiedenen Bereichen verglichen. Daraus folgt ein Ergebnis "wahr" oder "falsch".

Eine zweistellige Regel REGELNAME zwischen zwei Bereichen BEREICH1 und BEREICH2 wird definiert durch eine Liste von Attributen ATTR1,...,ATTRn und eine Liste von Operatoren OP1,...,OPn.

Ein Objekt X aus BEREICH1 und ein Objekt Y aus BEREICH2 erfüllen die zweistellige Regel REGELNAME, wenn für alle i=1,...,n
für den Wert XWERTi des Attributs ATTRi von Objekt X und den Wert YWERTi des Attributs ATTRi von Objekt Y gilt:

XWERTi OPi YWERTi.

Bei nichtnumerischen Wertebereichen wird bei diesem Vergleich auf die in der Wissensbasis definierte Ordnung zurückgegriffen.

Mehrstellige Regeln werden unter Zuhilfenahme von bereits definierten zweistelligen Regeln definiert. Damit sind beliebig komplizierte Verknüpfungen zwischen Objektbereichen darstellbar.

Eine n-stellige Regel REGELNAME betrifft eine Liste von Bereichen BEREICH1,...,BEREICHn und wird definiert durch eine Liste NR1,...,NRm von Nummern zweistelliger Regeln.

Eine Liste von Objekten X1,...,Xn, wobei für alle i=1,...,n Objekt Xi aus Bereich BEREICHi stammt, erfüllt die n-stellige Regel REGELNAME, wenn für alle j=1,...,m die betreffenden Objekte aus den in der zweistelligen Regel mit der Nummer NRj angesprochenen Bereichen diese zweistellige Regel erfüllen.

Sowohl für die zweistelligen wie auch für die mehrstelligen Regeln gilt, daß das prozedurale Wissen nicht über zwei- oder mehrdimensionale Faktentabellen abgebildet wird, sondern daß eine Schreibweise gefunden wird, bei der die konkrete Ausgestaltung von in Frage

kommenden Objekten, Attributen und Werten dem deklarativen Teil der Datenbasis entnommen und mit der abstrakten Regel verbunden wird. Abb. 4 zeigt den Aufbau von Regeln in KPS/Prolog in Prolog-Syntax.

```
zweibezzahl(4)
mehrbezzahl(1)

zweibeztext(1,"ist für","alt genug")
zweibeztext(2,"kann sich","leisten")
zweibeztext(3,"ist","angemessen")
zweibeztext(4,"ist für","geeignet")
mehrbeztext(1,"sollte benutzen für",["sollte","benutzen","für"])

zwei(1,1,2,["Alter"],[">="])
zwei(2,1,2,["Geld"],["ist mindestens"])
zwei(3,2,1,["Geld"],["ist gleich"])
zwei(4,2,3,["Entfernung","Tempo"],["ist gleich","ist mindestens"])
mehr(1,[1,2,3],[1,2,4])
```

Abb. 4: Speicherung von Regeln als in Turbo Prolog eingebettete Daten

3.3. Die Gestaltung der Schnittstellen

Gemäß Abb. 1 sind drei Schnittstellen zu schaffen. Für den Benutzer wie für den Wissensingenieur ist die Präsentation des Wissens am Bildschirm gleichermaßen relevant. Also muß die Möglichkeit zur Wissensdarstellung in beiden Schnittstellen gegeben sein. Die Wissensakquisition ist für den Wissensingenieur oder auch für den Experten zu gestalten. Bei entsprechender Zugangsberechtigung kann es auch dem Benutzer ermöglicht werden, dem System bestimmtes Wissen mitzuteilen, etwa eigene Bezeichnungen zu vergeben. Dies wird unumgänglich, wenn man ein lernendes System etablieren will.

Die Befragung der Wissensbasis muß primär im Hinblick auf Benutzerbedürfnisse ausgerichtet sein, jedoch wird auch der Wissensingenieur von Zeit zu Zeit eine Befragung vornehmen, um bei dem Systementwickler ggf. Änderungen des Systems zu veranlassen oder Fehler in der Konstruktion von Wissensbasen zu entdecken. Dies gilt in gleicher Weise für den Experten, der anhand bestimmter ermittelter Lösungen vielleicht zu dem

Ergebnis kommt, daß die Wissensbasis zu verändern ist, weil Probleme unvollständig formuliert sind.

In KPS/Prolog existiert eine einheitliche Benutzeroberfläche. Es ist leicht möglich, Zugangsberechtigungen und Benutzeridentifikationen vorzusehen, so daß eine individuelle Schnittstelle entsteht. Die dritte Schnittstelle gemäß Abb. 1, nämlich die Schnittstelle zur konventionellen Datenverarbeitung, ist in KPS/Prolog systemintern vorhanden. Benutzer, Wissensingenieur und Experte haben die Möglichkeit, an dafür vorgesehenen Stellen Programmnamen oder Datenfile-Namen anzugeben, wodurch entsprechende externe Programme ausgeführt oder Datenbestände aktiviert werden.

Im folgenden werden Wissensdarstellung, Wissensakquisition und die Befragung der Wissensbasis (Problemlösung) anhand von Abbildungen generierter Bildschirme beschrieben.

3.3.1. Wissensdarstellung

Die Benutzung von KPS/Prolog beginnt mit der Auswahl einer Wissensbasis (vgl. Abb. 5), die über einen systeminternen Befehl "consult" eingelesen werden soll.

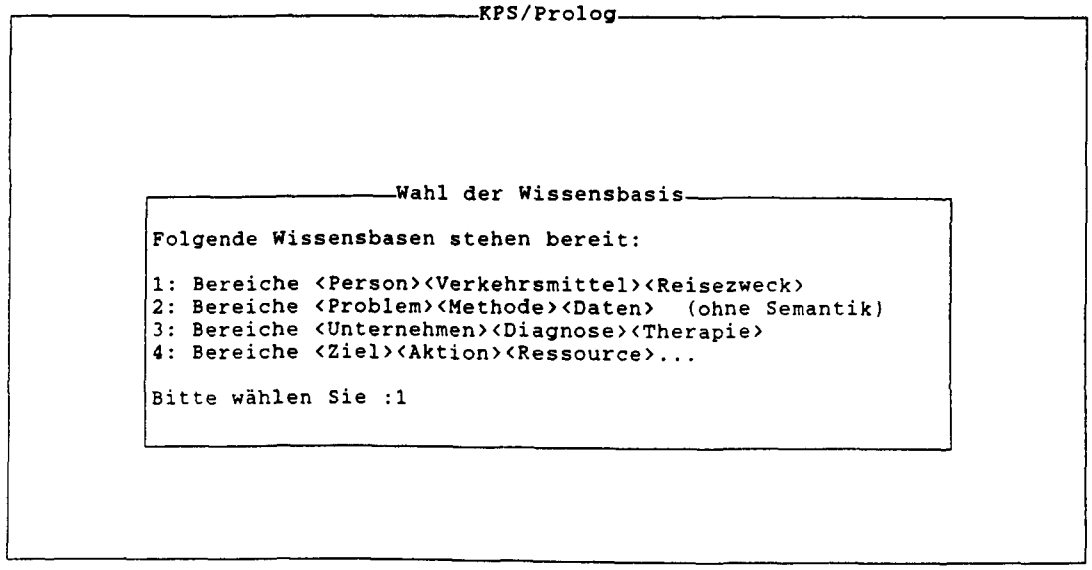


Abb. 5: Auswahl der zu bearbeitenden Wissensbasis

Dabei sind im vorliegenden Beispiel vier Wissensbasen (Domänen) ansprechbar, wobei jede Wissensbasis in diesen Fällen aus drei Bereichen besteht.

KPS/Prolog bietet dem Benutzer eine menügesteuerte Auswahl unter einer Vielzahl möglicher Ausführungspukte (siehe unten). Zur Implementierung der hierarchisch gegliederten Menüs bietet sich in Turbo Prolog die folgende Vorgehensweise an:

Überschriften und mögliche Alternativen in Menüs können als Daten betrachtet werden. Deshalb sind sie zweckmäßigerweise aus dem eigentlichen Programm auszulagern. Bei geeigneter Verwaltung und Definition von Datenbankprädikaten in Turbo Prolog-Syntax werden dann beim Start von KPS/Prolog die Menüdaten "konsultiert". Mit Hilfe des im Programm verbliebenen Rumpfes zur Menügestaltung wird unter Übergabe der Menünummer, die eventuell von der vorherigen Wahl auf höherer Ebene abhängt, das gewünschte Menü mit Überschrift und durchnumerierten Alternativen dargeboten, und es wird die aktuelle Wahl des Benutzers als Ausgabeparameter abgeliefert.

Als vorteilhaft erweist sich dabei neben der erheblichen Verkürzung des Programmcodes die Möglichkeit, weitere Menüs oder Menüpunkte zu integrieren, ohne das Programm neu übersetzen zu müssen. Außerdem steigt die Übersichtlichkeit bei hierarchischer Menüuntergliederung, da sämtliche Texte in einem File zusammengefaßt und von anderen Programmbestandteilen getrennt sind. Die hier benutzte Form der Menüdarbietung ist natürlich nicht auf diesen Zusammenhang und eine Gliederungstiefe von drei Ebenen beschränkt.

Die Vorgehensweise bei der Menübehandlung ist in Abb. 6 verdeutlicht.

```
REGEL ZUR MENÜGESTALTUNG IN KPS/PROLOG
(Inputparameter Z1,Z2,Z3, Outputparameter Z4)

menu(Z1,Z2,Z3,Z4) if
     übertext(Z1,Z2,Z3,Ütext),
     nl,write(Ütext),nl,nl,
     menutext(Z1,Z2,Z3,I,Itext),
     write(I,": ",Itext),nl,
     fail;
     nl,write("Bitte wählen Sie: "),
     readint(Z4).
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

AUFRUF VON MENÜ 311

menu(3,1,1,Z4)
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

AUSSCHNITT AUS MENÜDATEN BETREFFEND MENÜ 311
(Menü auf Ebene 3 bei Wahl von 1 auf Ebene 1, von 1 auf Ebene 2)

übertext(3,1,1,"Welche Bezeichnungen möchten Sie gezeigt bekommen?")
menutext(3,1,1,1,"Bezeichnungen für Ebenen")
menutext(3,1,1,2,"Bezeichnungen für Bereiche")
menutext(3,1,1,3,"Bezeichnungen für zweistellige Regeln")
menutext(3,1,1,4,"Bezeichnungen für mehrstellige Regeln")
```

Abb. 6: Menügestaltung in KPS/Prolog

In Abb. 7 wird der hierarchische Menüaufbau durch die Fenstertechnik deutlich; außerdem wird die jeweilige Tiefe des Menüs am oberen Fensterrahmen angezeigt.

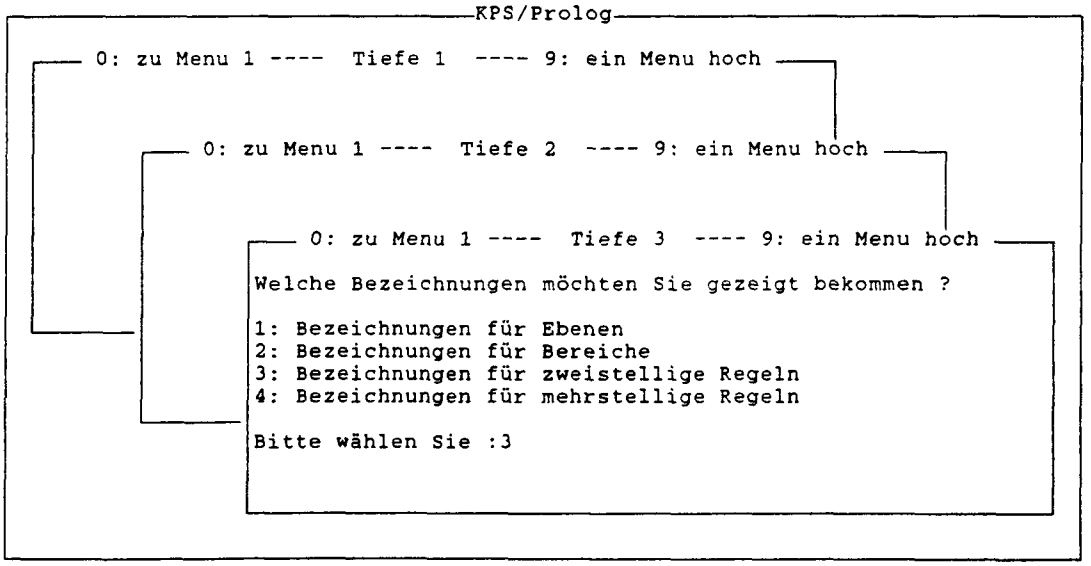


Abb. 7: Menüdarbietung in KPS/Prolog

Zum besseren Verständnis kennzeichnet KPS/Prolog bei sämtlichen Bildschirmausgaben Bezeichnungen für Ebenen, Bereiche und Regeln mit spitzen Klammern (<.>), während Wissensinhalte wie Objekte, Attribute und Werte sowie Operatoren durch Hochkommata ('.') hervorgehoben

werden. Dadurch wird sehr deutlich, daß der Anteil an in KPS/Prolog fest programmierten Ausgabetexten gering ist und der Benutzer fast nur mit den für das bearbeitete Wissensgebiet festgelegten Begriffen angesprochen wird.

Ein einfaches Beispiel soll zur Veranschaulichung der Benutzung von KPS/Prolog dienen. Es beschäftigt sich mit dem Wissensgebiet <Person> <Verkehrsmittel> <Reisezweck>.
Abb. 8 zeigt einen Ausschnitt aus dem hierarchischen Menüaufbau, soweit die Anzeige der Wissensbasis betroffen ist.

```
1 Wissensbasis      1 Bezeichnungen     1 Ebenen
  zeigen                                2 Bereiche
                                        3 Zweistellige Regeln
                                        4 Mehrstellige Regeln

                    2 Definitionen      1 Objekte eines Bereichs
                                        2 Unterordnungen zwischen
                                          Objekten eines Bereichs
                                        3 Zweistellige Regeln
                                        4 Mehrstellige Regeln

                    3 Eigenschaften     1 Synonyme für Objekte
                                        2 Attribute und Werte von
                                          Objekten
                                        3 Synonyme für Attribute
                                        4 Wertebereiche (Ordnungen)
                                          von Attributen
```

Abb. 8: Hierarchische Gliederung im Menüzweig "1: Wissensbasis zeigen"

```
KPS/Prolog
Merkmale von Objekt zeigen
Merkmale von 'Auto':

  Attribut          Wert
-------------------------
  'Geld'          : 'viel'
  'Alter'         : '18'
  'Entfernung'    : 'mittel'
  'Tempo'         : 'schnell'

Return drücken
```

Abb. 9: Eigenschaften eines Objekts

Abb. 9 zeigt die Auflistung von Attributen und Werten des Objekts 'Auto'.

Abb. 10 zeigt für das Attribut 'Tempo' die Eigenschaften des Wertebereichs und die vorgesehene Ordnung auf dem Wertebereich.

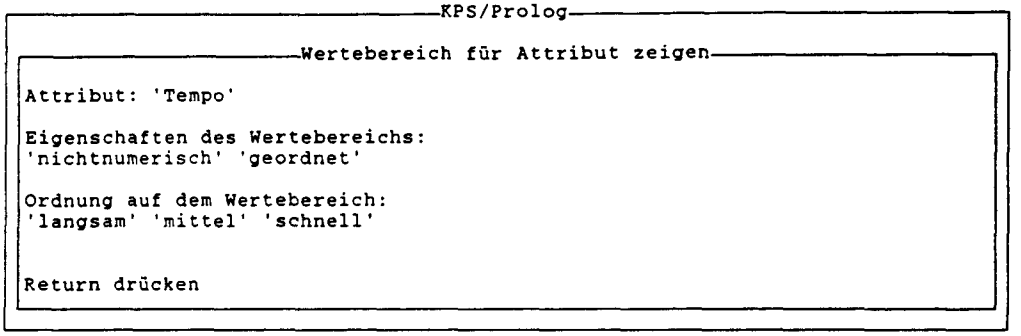


Abb. 10: Wertebereich eines Attributs

Der Shell-Charakter von KPS/Prolog tritt besonders anhand von Abb. 11 hervor, in der Bezeichnungen für vorhandene zweistellige Regeln angegeben sind.
Zur Annäherung an natürliche Sprache ist die Regelbezeichnung bei zweistelligen Regeln in zwei Teile aufgespalten: <Person><kann sich leisten><Verkehrsmittel> wird zu <Person><kann sich><Verkehrsmittel><leisten>.

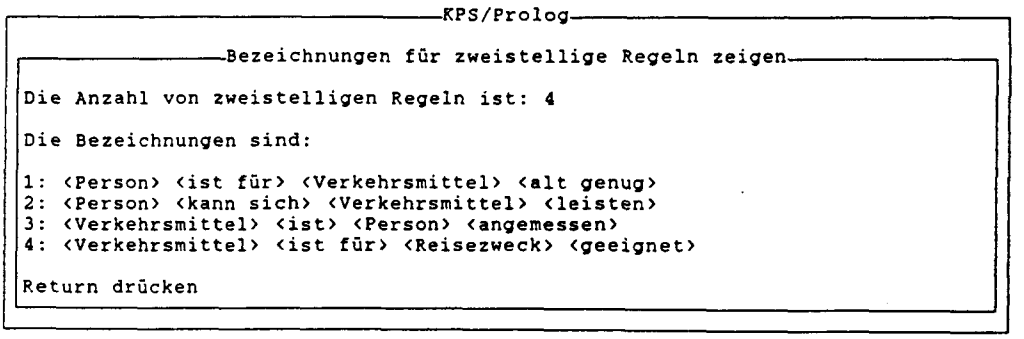


Abb. 11: Bezeichnungen zweistelliger Regeln

In Abb. 12 wird die Struktur des prozeduralen Wissens für den Benutzer offengelegt. Die Regel <ist für><alt genug> stellt eine Verknüpfung zwischen den Bereichen <Person> und <Verkehrsmittel> her. Sie verlangt einen Vergleich der Werte des Attributs 'Alter' unter Verwendung des für numerische Wertebereiche zulässigen Operators '>='.

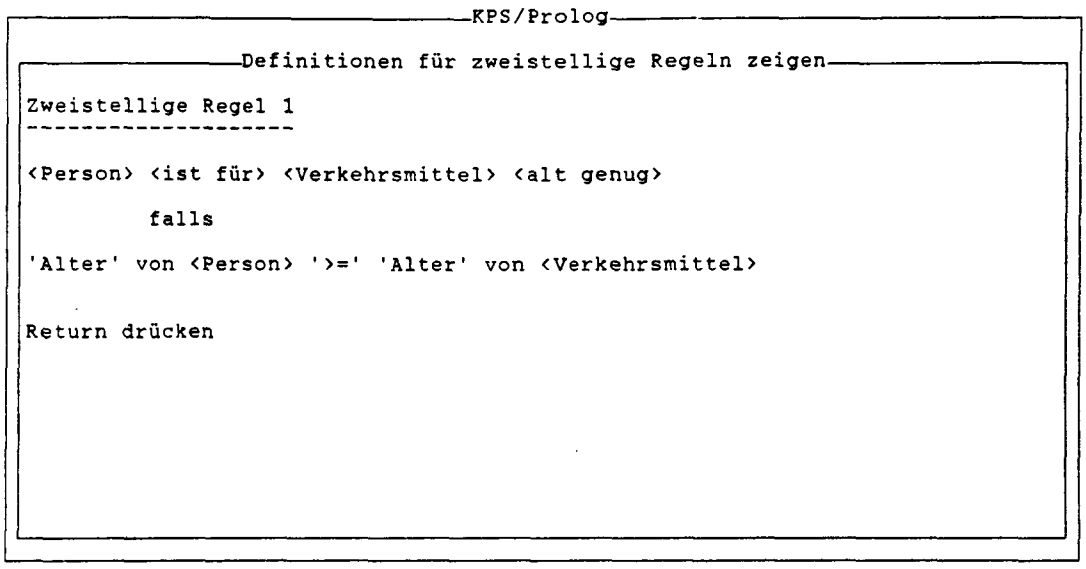


Abb. 12: Definition einer zweistelligen Regel

Abb. 13 zeigt die Definition einer dreistelligen Regel, wobei die Gültigkeit der genannten zweistelligen Regeln zwischen entsprechenden Objekten zu analysieren ist. Entsprechend der oben geschilderten Aufspaltung einer Regelbezeichnung sind bei n-stelligen Regeln n Bezeichnungsteile vorgesehen, in Abb. 13: <sollte><benutzen> <für>.

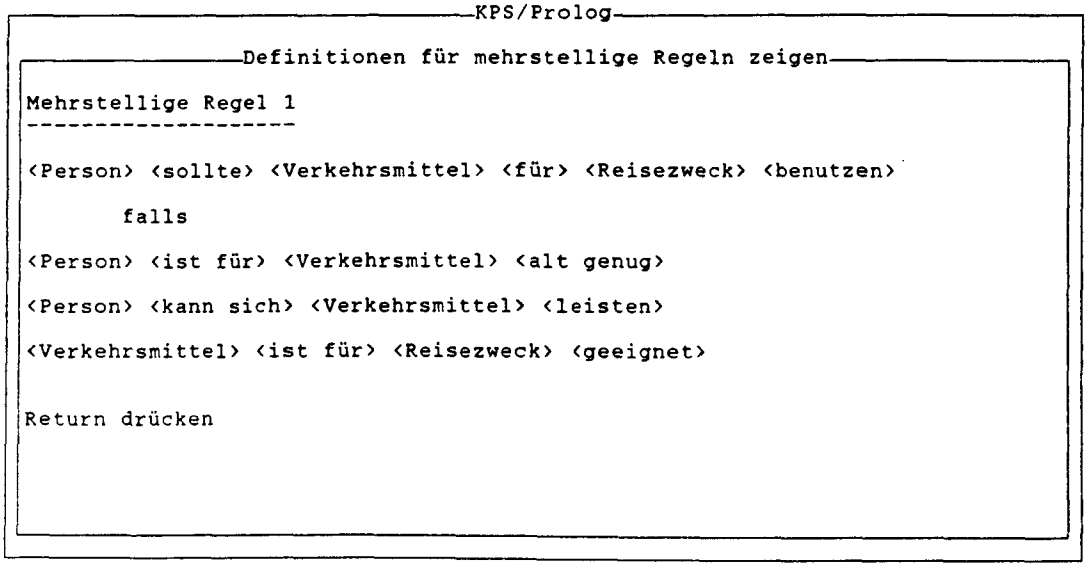


Abb. 13: Definition einer mehrstelligen Regel

3.3.2. Wissensakquisition

Für die Wissensakquisition wird die in KPS/Prolog vorhandene Wissenserwerbskomponente benutzt. Die Editorfunktion wird also von der Benutzeroberfläche übernommen, obwohl systemintern die Eingaben in eine Datenstruktur überführt werden, die in Prolog-Syntax gehalten ist.

Abb. 14 zeigt die möglichen Veränderungen bzw. Ergänzungen der Wissensbasis. Es können Bezeichnungen für Ebenen, Bereiche und Regeln geändert werden. Definitionen können für Objekte, Unterordnungen und Regeln editiert werden. Eigenschaften von Objekten können über Synonyme, Attribute und Werte sowie über Synonyme von Attributen, Wertebereiche von Attributen und Ordnungen auf Wertebereichen eingegeben werden.

2 Wissensbasis verändern	1 Bezeichnungen	1 Ebenen 2 Bereiche 3 Zweistellige Regeln 4 Mehrstellige Regeln
	2 Definitionen	1 Objekt eintragen 2 Objekt streichen 3 Unterordnungen zwischen Objekten einrichten 4 Unterordnungen zwischen Objekten aufheben 5 Zweistellige Regeln umdefinieren 6 Mehrstellige Regeln umdefinieren
	3 Eigenschaften	1 Synonyme für Objekte 2 Attribute oder Werte von Objekten 3 Synonyme für Attribute 4 Wertebereiche von Attributen 5 Ordnungen auf Wertebereiche

Abb. 14: Hierarchische Gliederung im Menüzweig "2: Wissensbasis verändern"

Abb. 15 zeigt, wie der Regeltext <ist für><alt genug> durch <darf><benutzen> ersetzt und damit dem Sprachgebrauch des Benutzers angepaßt wird. Dadurch können benutzerspezifische Versionen derselben Wissensbasis geschaffen werden.

```
KPS/Prolog
Bezeichnung einer zweistelligen Regel verändern

Vorher:

<Person> <ist für> <Verkehrsmittel> <alt genug>

Neue Bezeichnung der zweistelligen Regel eingeben:

<Person> <TEXT1> <Verkehrsmittel> <TEXT2>

TEXT1 ? darf
TEXT2 ? benutzen

Jetzt:

<Person> <darf> <Verkehrsmittel> <benutzen>

Return drücken
```

Abb. 15: Änderung der Bezeichnung
für eine zweistellige Regel

3.3.3. Befragung der Wissensbasis

Die Befragung der Wissensbasis geschieht in der derzeitigen Version von KPS/Prolog auch über hierarchische Menüs (vgl. Abb. 16). Diese Menüs sind von den Menüinhalten her zunächst abstrakt und damit allgemeingültig gehalten.

```
3 Wissensbasis befragen
    1 Einzelne Bereiche
        1 Objekte mit bestimmten Eigenschaften
        2 Untergeordnete Objekte
        3 Übergeordnete Objekte
        4 Attribute eines Bereichs
        5 Werte eines Attributs
    2 Zweistellige Regeln
        1  1. Regel
        2  2. Regel
        3  3. Regel
        .       .
        :       :
    3 Mehrstellige Regeln
        1  1. Regel
        2  2. Regel
        3  3. Regel
        .       .
        :       :
    4 Liste von Objekten
        1 Zweistellige Regeln zu Liste von Objekten
        2 Mehrstellige Regeln zu Liste von Objekten
    5 Liste von Bereichen
        1 Zweistellige Regeln zu Liste von Bereichen
        2 Mehrstellige Regeln zu Liste von Bereichen
```

Abb. 16: Hierarchische Gliederung im Menüzweig
"3: Wissensbasis befragen"

Die Bezeichnungen der einzelnen Menüpunkte können aber ohne weiteres anwendungsspezifisch durch für den Benutzer verständlichere Bezeichnungen ersetzt werden.Der Benutzer muß zur Zeit u.a. noch die Begriffe "Ebene", "Bereich", "Objekt", "Eigenschaft", "Attribut", "Wert", "Operator", "Regel" in ihrer Bedeutung einordnen können. Hilfsfunktionen können hier Benutzerängste abbauen.

Abb. 17 zeigt nun zunächst eine Anfrage, die sich auf einen bestimmten Bereich beschränkt, nicht aber auf die Verknüpfung von Bereichen bezieht. Es geht also um eine herkömmliche Aufgabe des Retrieval, wie sie schon seit langem in Management-Informationssystemen wahrgenommen wird. Ein wissensbasiertes System muß natürlich auch solche Anfragen gestatten.

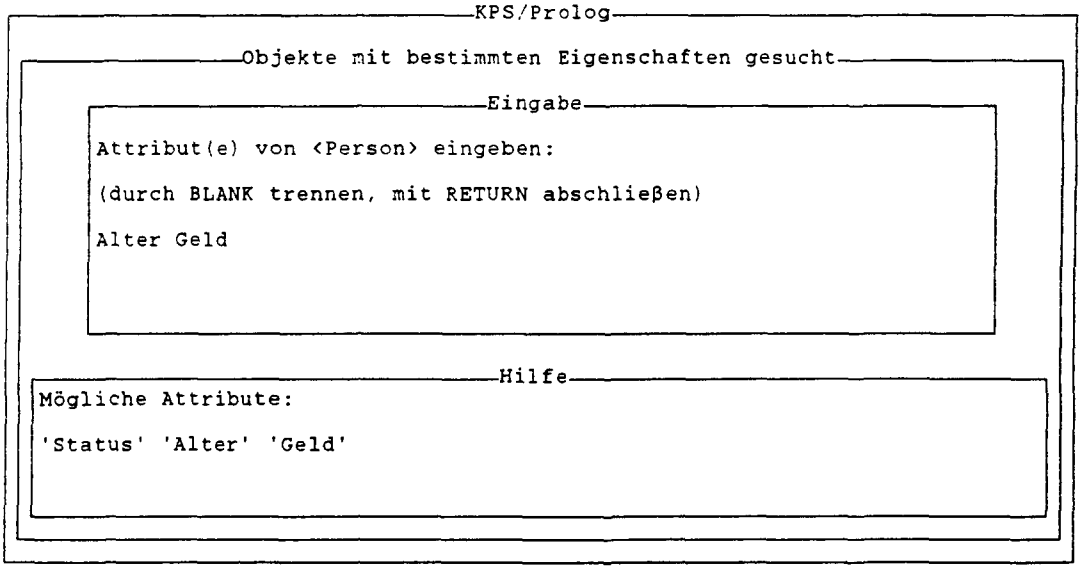


Abb. 17: Vergabe von Attributen zur Selektion von Objekten

In Abb. 17 wird deshalb gezeigt, wie Objekte mit bestimmten Merkmalen gesucht werden können. Der Benutzer stellt Anforderungen an die Werte der Attribute 'Alter' und 'Geld', um dadurch bestimmte Personen zu ermitteln. Das System bietet ihm im unteren Teil des Bildschirms eine Hilfestellung, indem es aufzeigt, welche Attribute gemäß dem augenblicklichen Systemzustand vorhanden sind.

In Abb. 18 wird der Benutzer dann gebeten, zum Attribut 'Alter' einen Operator und einen Wert anzugeben. Auch hier wird ihm wieder geholfen, indem ihm bei jeder Eingabe, also in Abb. 18 zweimal, ein Hilfsfenster geboten wird. Dieses Hilfsfenster enthält bei der ersten Eingabe die für den Wertebereich des Attributs 'Alter' zulässigen Operatoren und bei der zweiten Eingabe die Art des Wertebereichs, so daß der Benutzer in diesem Falle einen numerischen Wert einzugeben hat.

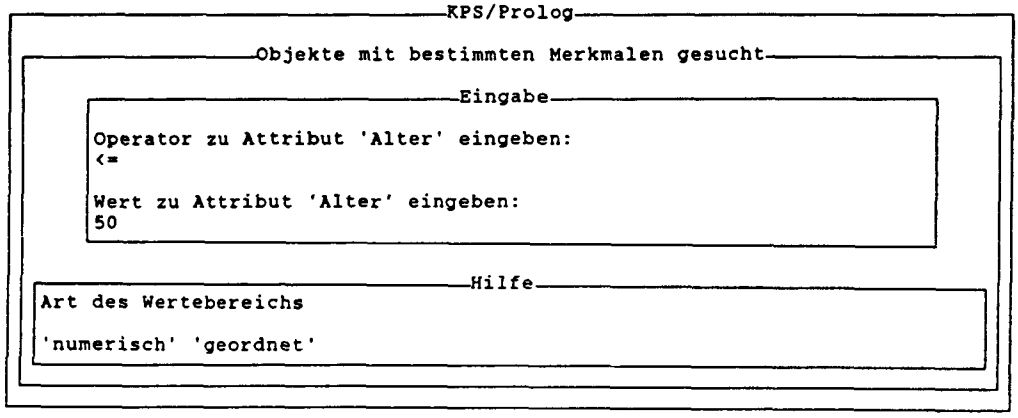


Abb. 18: Vorgabe von Operator und Wert zum Attribut 'Alter'

In Abb. 19 dagegen hat der Benutzer eine Eingabe für ein nichtnumerisches Attribut zu tätigen. Bei der Behandlung von Attributen mit nichtnumerischem Wertebereich zeigt sich der Vorteil einer logik-orientierten Programmiersprache.

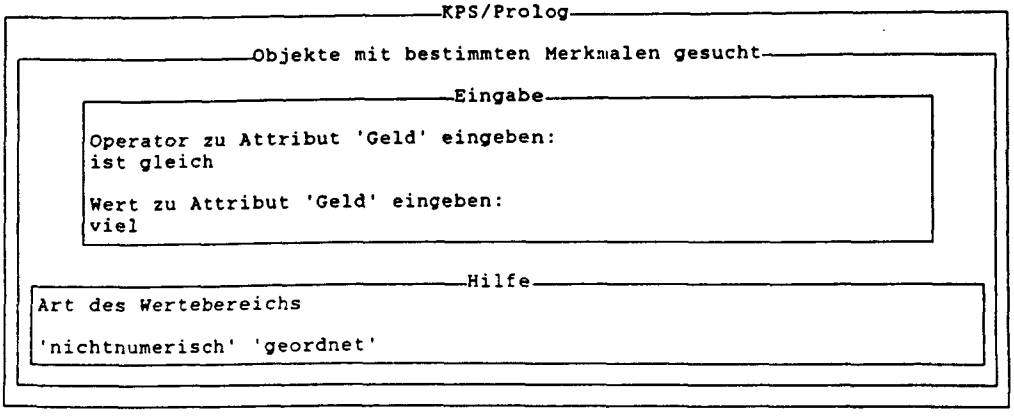


Abb. 19: Vorgabe von Operator und Wert zum Attribut 'Geld'

Die Antwort des Systems ergibt sich aus Abb. 20. Es wurde das Objekt 'Lisa' gefunden. Der Benutzer erhält eine entsprechende Begründung. In Form dieser Begründung zeigt sich erneut der Vorteil eines wissensbasierten und in einer logikorientierten Programmiersprache implementierten Systems gegenüber einem herkömmlichen Management-Informationssystem, das eine so ausformulierte Begründung schwer geben kann.

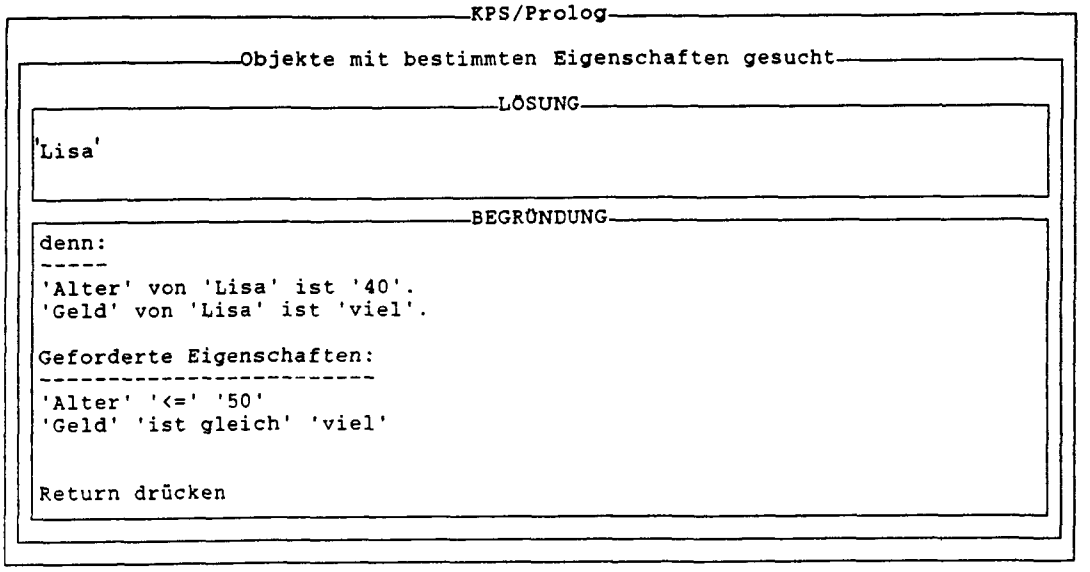


Abb. 20: Lösung einer Anfrage im Hinblick auf ein Objekt mit bestimmten Eigenschaften

Das eigentlich Interessante bei Anfragen an wissensbasierte Systeme sind aber die Anfragen an definierte Regeln. Solche Anfragen beinhalten das Ziel einer Problemlösung, weshalb für wissensbasierte Systeme hier der Begriff "Problemlösungssystem" synonym verwendet wird. Problemlösung ist also gleichzusetzen mit der Suche nach Objekten, die Regeldefinitionen erfüllen.

Abb. 21 zeigt die Namen der gespeicherten zweistelligen Regeln, die natürlich auch in einer den Benutzer noch verständlicheren Weise als "Probleme" hätten bezeichnet werden können. Die Verbindung zur Logik der Regeldefiniton kommt jedoch über die in Abb. 21 enthaltene Bezeichnungsform besser zum Ausdruck.

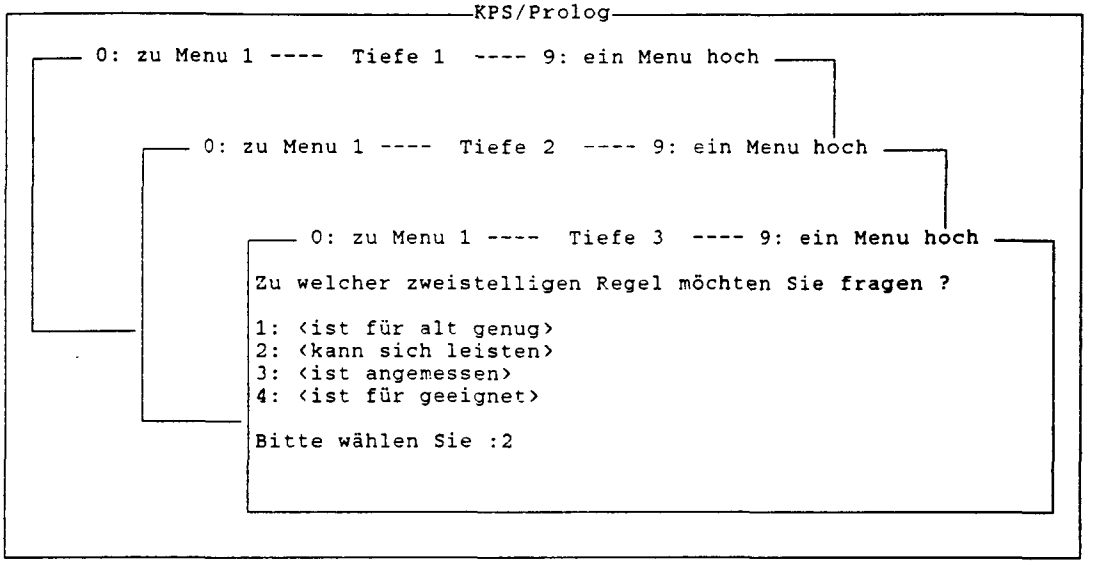


Abb. 21: Anfrage an eine zweistellige Regel

Hat der Benutzer sich für eine Fragestellung im Hinblick auf <kann sich><leisten> entschieden, so zeigt ihm das System gemäß Abb. 22 die möglichen Fragestellungen, wobei deutlich wird, daß Objekte jeweils bekannt oder unbekannt sein können. Dies bedeutet, daß in Prolog die Variablen entweder gebunden oder frei sind. Hier liegt die eigentliche Stärke einer Programmierung des Systems in Prolog, da keine Unterscheidung dieser Fälle vorgenommen werden muß. Die Lösungsuche hängt nicht davon ab, welche Variablen frei und welche gebunden sind.

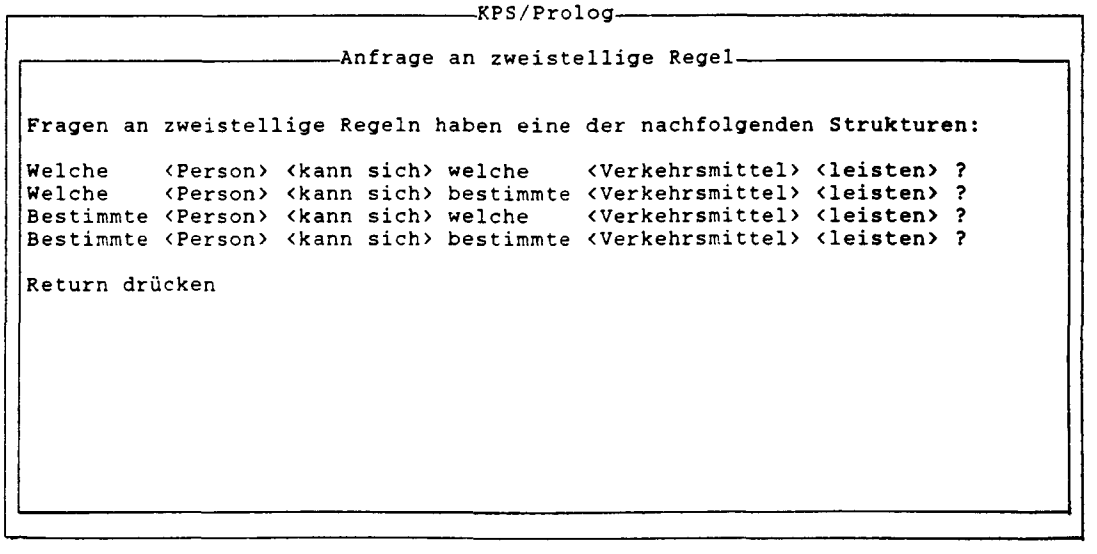


Abb. 22: Mögliche Anfragestrukturen
bei zweistelligen Regeln

In Abb. 23 wird nunmehr im Wege einer sequentiellen Eingabe von dem Benutzer die weitere Problemstrukturierung erfragt. Er entscheidet sich dafür, den Personennamen vorzugeben und nach Verkehrsmitteln zu suchen, die sich diese bestimmte Person leisten kann.

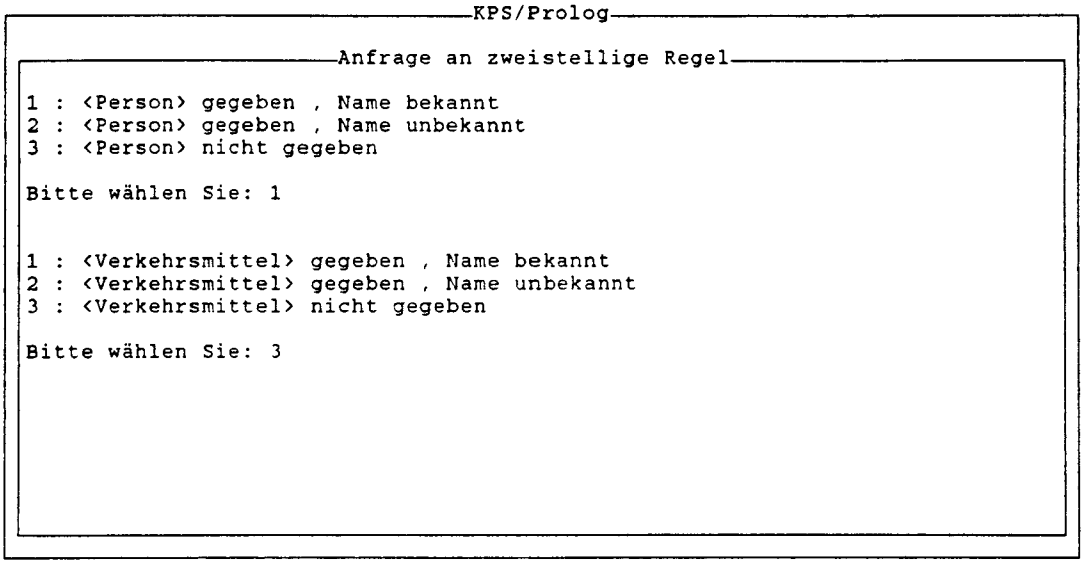


Abb. 23: Auswahl einer Anfragestruktur bei einer zweistelligen Regel

In Abb. 24 wird die Person benannt, wobei das System dem Benutzer in einem Hilfsfenster wieder anzeigt, welche Objekte in diesem Bereich überhaupt gespeichert sind.

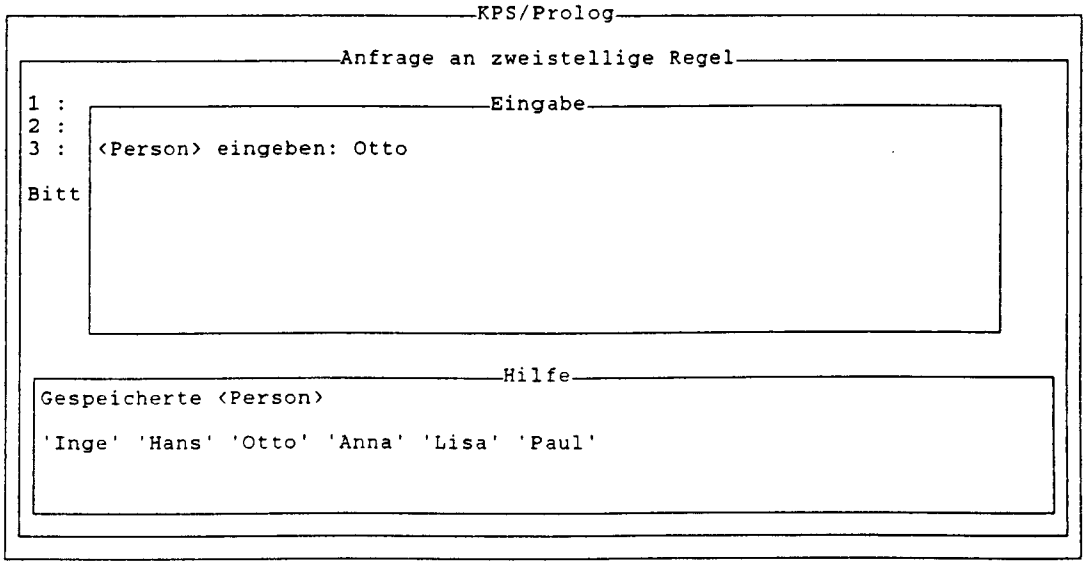


Abb. 24: Spezifikation einer Person

In Abb. 25 ist dann eine Lösung des Problems mit entsprechender Begründung wiedergegeben. In KPS/Prolog wird die Begründung dabei so gegeben, daß auch der unerfahrene Benutzer diese Begründung versteht.

```
KPS/Prolog
Anfrage an zweistellige Regel
LÖSUNG
'Otto' <kann sich> 'Bahn' <leisten>
BEGRÜNDUNG
denn:
-----
'Geld' von 'Otto' ist 'viel',
'Geld' von 'Bahn' ist 'mittel'.

und:
---
<Person> <kann sich> <Verkehrsmittel> <leisten>, falls
'Geld' von <Person> 'ist mindestens' 'Geld' von <Verkehrsmittel>

Return drücken
```

Abb. 25: Lösung einschließlich einer Begründung mit Eigenschaften und mit Definition der angesprochenen zweistelligen Regel

Die Voreinstellung des Systems kann so erfolgen, daß solche Objekte angelistet werden, die keine Lösung des Problems darstellen (vgl. Abb. 26). Dies hat den Vorteil, daß der Benutzer den Wissensingenieur oder Experten im Hinblick auf die Qualität der Wissensbasis ansprechen kann.

```
KPS/Prolog
Anfrage an zweistellige Regel
FALSCH IST:
'Geld' von 'Anna' 'ist mindestens' 'Geld' von 'Auto'
BEGRÜNDUNG
denn:
-----
'Geld' von 'Anna' ist 'wenig',
'Geld' von 'Auto' ist 'viel'.


Return drücken
```

Abb. 26: Begründung einer Nicht-Lösung mit Eigenschaften bei Anfrage an zweistellige Regel

In KPS/Prolog werden alle Objektpaare geliefert, die nach Einsetzen in die Regel zu der Aussage "wahr" führen. Es können also mehrere Lösungen auftreten.

Eine besondere Variante der Begründung durch KPS/Prolog ist in Abb. 27 enthalten. Damit für die Fälle vorgesorgt wird, in denen der Benutzer mit der erhaltenen Lösung unzufrieden ist, kann auch von dem System verlangt werden, daß es angibt, welche Attribute der beteiligten Objekte für die Problemlösung nicht herangezogen wurden. Dies kann den Benutzer zu anderen Fragestellungen oder den Wissensingenieur bzw. Experten zur Neuformulierung des Regelwerks veranlassen.

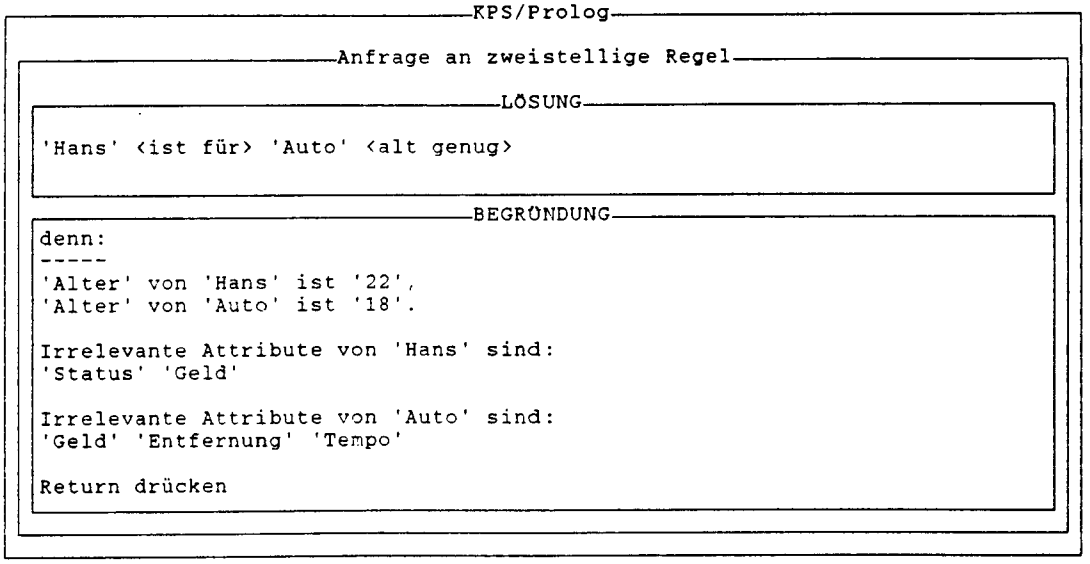


Abb. 27: Lösung und Begründung mit Eigenschaften sowie irrelevanten Attributen zur Anfrage an eine zweistellige Regel

In Abb. 28 wird eine Anfrage an eine dreistellige Regel wiedergegeben. Dabei wird allerdings nicht erkennbar, daß im Falle einer Eingabe "gegeben" sofort über ein dann erscheinendes Fenster die Spezifikation erfragt wird. Im Falle der Abb. 28 hat der Benutzer 'Hans' als <Person> und 'Einkauf' als <Reisezweck> festgelegt. Die dadurch repräsentierte Problemstellung lautet: "Welches Verkehrsmittel sollte Hans für den Einkauf benutzen ?"

```
KPS/Prolog
Anfrage an mehrstellige Regel

1 : <Person> gegeben , Name bekannt
2 : <Person> gegeben , Name unbekannt
3 : <Person> nicht gegeben

Bitte wählen Sie: 1


1 : <Verkehrsmittel> gegeben , Name bekannt
2 : <Verkehrsmittel> gegeben , Name unbekannt
3 : <Verkehrsmittel> nicht gegeben

Bitte wählen Sie: 3


1 : <Reisezweck> gegeben , Name bekannt
2 : <Reisezweck> gegeben , Name unbekannt
3 : <Reisezweck> nicht gegeben

Bitte wählen Sie: 1
```

Abb. 28: Auswahl einer Anfragestruktur bei einer mehrstelligen Regel

In Abb. 29 antwortet das System mit einer Lösung und der laut Parametereinstellung gewünschten Begründung.

```
KPS/Prolog
Anfrage an mehrstellige Regel
LÖSUNG

'Hans' <sollte> 'Fahrrad' <für> 'Einkauf' <benutzen>

BEGRÜNDUNG
denn:
-----
<Person> <sollte> <Verkehrsmittel> <für> <Reisezweck> <benutzen>

     falls

<Person> <ist für> <Verkehrsmittel> <alt genug>
<Person> <kann sich> <Verkehrsmittel> <leisten>
<Verkehrsmittel> <ist für> <Reisezweck> <geeignet>

Return drücken
```

Abb. 29: Lösung und Begründung mit Definition zur Anfrage an eine mehrstellige Regel

3.3.4. Parametereinstellungen

Bei fortgesetzter oder wiederholter Benutzung des Systems durch denselben Benutzer ist es zweckmäßig, wenn bestimmte Eingaben nicht jeweils neu getätigt

werden müssen. Deshalb enthält KPS/Prolog eine ganze Reihe von Möglichkeiten, solche Voreinstellungen vorzunehmen (vgl. Abb. 30). Grundsätzlich ist es möglich, diese Parametereinstellungen benutzerfreundlich für jeden Benutzer abzuspeichern, so daß eine komfortable, individuelle Benutzerführung und -bedienung erreicht werden kann.

4 Parameter-einstellung verändern	1 Zu bearbeitende Wissensbasis	1	1. Wissensbasis
		2	2. Wissensbasis
		3	3. Wissensbasis
		.	.
		:	:
	2 Begründung für Lösungen zweistelliger Regeln	1	keine Begründung
		2	Begründung mit Eigenschaften
		3	Begründung mit Definition und Eigenschaften
		4	Begründung mit Eigenschaften und irrelevanten Attributen
	3 Begründung für Lösungen mehrstelliger Regeln	1	keine Begründung
		2	Begründung mit Definition
		3	Begründung mit Definition und Eigenschaften
	4 Begründung für Nicht-Lösungen	1	keine Begründung
		2	Begründung mit Eigenschaften
	5 Erläuterung von Anfrage-strukturen	1	Erläuterung erwünscht
		2	Erläuterung nicht erwünscht

Abb. 30: Hierarchische Gliederung im Menüzweig "4: Parametereinstellung verändern"

Jederzeit kann sich der Benutzer auf der obersten Menüebene die gegenwärtigen Parametereinstellungen anzeigen lassen (Abb. 31).

```
KPS/Prolog
Einstellung von Parametern

1 : Zu bearbeitende Wissensbasis
    Bereiche <Person><Verkehrsmittel><Reisezweck>

2 : Begründung für Lösungen zweistelliger Regeln
    Begründung mit Eigenschaften

3 : Begründung für Lösungen mehrstelliger Regeln
    Begründung mit Definition

4 : Begründung für Nicht-Lösungen
    keine Begründung

5 : Erläuterung von Anfragestrukturen
    Erläuterung nicht erwünscht

Return drücken
```

Abb. 31: Einstellung von Parametern

4. Einsatz als Problemlösungssystem

4.1. Ausgewählte betriebswirtschaftliche Anwendungen

In Abb. 32 ist zunächst grundsätzlich gezeigt, wie ein Ablauf des Problemlösungsprozesses strukturiert werden kann.

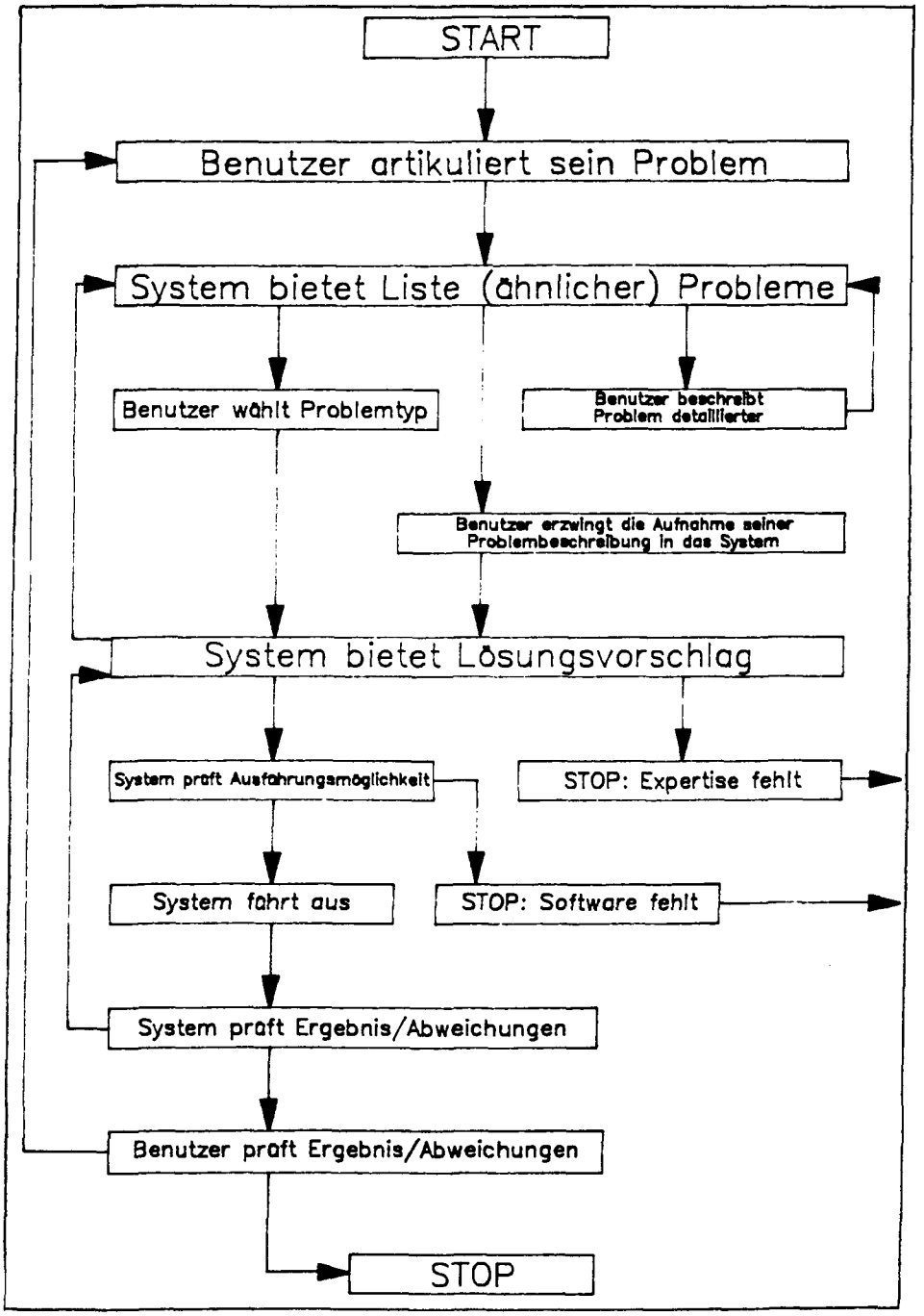


Abb. 32: Ablauf des Problemlösungsprozesses

Das Neue gegenüber vielen anderen Systemen ist, daß das System KPS/Prolog nicht nur Lösungsvorschläge bieten kann, sondern daß auch entsprechende Programme ausgeführt werden können, die über die Schnittstelle zur konventionellen Datenverarbeitung angestoßen werden (vgl. [20]). Nach Ausführung dieser Programme wird wiederum KPS/Prolog aktiviert, so daß auch ein mehrstufiger Problemlösungsprozeß mit einer sehr unterschiedlichen Folge von Teilproblemen unterstützt werden kann.

Weil das System am Institut für Betriebswirtschaftslehre der Universität Kiel entstanden ist, stehen zunächst betriebswirtschaftliche Anwendungen im Vordergrund, obwohl das System Anwendungen aller Art zuläßt.

Bisherige Anwendungen betreffen den Bereich der wissensbasierten Modellierung (vgl. [20]). Dabei wird eine wissensbasierte Modellierung mit der Ausführung von auf diese Weise generierten Unternehmensplanungsmodellen verbunden.

Außerdem sind Anwendungen im Bereich der strategischen Unternehmensplanung, der Unternehmensanalyse und der Auswahl von Finanzinnovationen geplant.

Schließlich sei erwähnt, daß sich KPS/Prolog gut eignet, um heterogene Programm- und Datenbestände zu verwalten (vgl. dazu auch [16]).

4.2. Erweiterungsmöglichkeiten

Erweiterungsmöglichkeiten bieten sich zunächst im Hinblick auf Konsistenzprüfungen bei Ergänzungen der Wissensbasis. Außerdem sind Anleitungen zur Konstruktion neuer Wissensbasen wünschenswert. Denn im Falle einer Neukonstruktion ist der Ablauf der erforderlichen Wissenseingabe durch die Struktur der Wissensrepräsentation in KPS/Prolog vorgegeben, so daß keine Editierungen über Änderungsmenüs erforderlich sind.

Weiterhin ist es im Hinblick auf größere Anwendungen zweckmäßig, Segmentierungen der Wissensbasis vornehmen zu können, so daß Kernspeicherbeschränkungen eingehalten werden können.

Besonders gewinnen dürfte der Einsatz von KPS/Prolog durch den Datenimport aus Lexika, insbesondere aus Fachlexika. Außerdem bietet die Mehrsprachigkeit unter Benutzung des Synonymansatzes interessante Systemausgestaltungen. Dies betrifft nicht nur Fremdsprachen, sondern auch die Verbindung zwischen unterschiedlichen Konversationsniveaus, z.B. zwischen Umgangs- und Fachsprache. Mehr Natürlichsprachlichkeit stellt daher eine große Herausforderung für die weitere Entwicklung des Systems dar.

Literaturverzeichnis

[1] Böhringer, B., Chiopris, C., Futo, I.: Wissensbasierte Systeme mit Prolog. Bonn 1988.

[2] Clocksin, W.F., Mellish, C.S.: Programming in Prolog. 3. Aufl., Berlin, Heidelberg 1987.

[3] Dieterich, G.: Kompaktführer Turbo-Prolog. Bonn 1987.

[4] Electronic Trend Publications (Hrsg.): PC-Driven Expert Systems. Saratoga o. J.

[5] Grothaus, M., Gust, H.: Turbo Prolog. Würzburg 1987.

[6] Harmon, P., Kjng, D.: Expertensysteme in der Praxis. 2 Aufl., München, Wien 1987.

[7] Haugg, F., Omlor, S.: Expertensysteme auf PCs: Entwicklung von Expertensystemen mit Turbo PROLOG. München, Wien 1987.

[8] Heimsoeth & Borland (Hrsg.): Turbo Prolog. München 1987.

[9] Heimsoeth & Borland (Hrsg.): Turbo Prolog Toolbox. München 1987.

[10] Herrmann, D.: Probleme und Lösungen mit Turbo-Prolog. Braunschweig 1988.

[11] Kinnebrock, W.: Turbo Prolog. München, Wien 1988.

[12] Knauss, W.: Turbo-Prolog. München, Wien 1987.

[13] Mertens, P., Borkowski, V., Geis, W.: Betriebliche Expertensystem-Anwendungen - Eine Materialsammlung. Berlin-Heidelberg-New York 1988.

[14] Nath, S.: Turbo Prolog: Features for Programmers. Portland 1986.

[15] o.V.: Expertensystem-shells für den PC. In: PC Magazin 1987, Nr. 30, S. 40-41.

[16] Reusch, P.: Modellverwaltung und Expertensystemkomponenten für betriebliche Informationssysteme. Mannheim, Wien, Zürich 1988.

[17] Robinson, P.R.: Using Turbo Prolog. Berkeley 1987.

[18] Savory, S.: Grundlagen von Expertensystemen. München, Wien 1988.

[19] Schildt, H.: Professionelles Turbo Prolog. Hamburg, New York 1987.

[20] Schmidt, R.: Zur Verbindung von wissensbasierter Modellierung und What-if-Planung. In: Operations Research Proceedings 1988, Berlin-Heidelberg-New York 1989 (in Druck).

[21] Schnupp, P.: Prolog. München 1986.

[22] Smith P.: Expert System Development in Prolog and Turbo-Prolog. Wilmslow 1988.

[23] Townsend, C.: Einführung in Turbo Prolog. Düsseldorf 1987.

[24] Walker, A. (Hrsg.), McCord, M., Sowa, J.F., Wilson, W.: Knowledge Systems and Prolog. Reading 1987.

[25] Wolfgram, D.D., Dear, T.J., Galbraith, C.S.: Expert Systems for the Technical Professional. New York et al. 1987.