Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 79 (1988)

Heft: 15

Artikel: Expertensysteme : eine Einführung

Autor: Imhof, K. / Amitrigala, R.

DOI: https://doi.org/10.5169/seals-904055

Expertensysteme – eine Einführung

K. Imhof und R. Amitirigala

Künstliche Intelligenz (KI) und ihr Teilbereich Expertensysteme sind Themen, die in den letzten Jahren ausgiebig für Diskussionsstoff gesorgt haben. Im vorliegenden Beitrag versuchen die Autoren, eine möglichst praktische und objektive Einführung in die Thematik der Expertensysteme zu geben.

L'intelligence artificielle et son domaine partiel les systèmes-experts sont des thèmes qui ces dernières années ont donné lieu à d'abondantes discussions. Dans le présent article, les auteurs essaient de donner une introduction aussi pratique et objective que possible à la thématique des systèmes-experts.

Adresse der Autoren

Karl Imhof, Dr. sc. techn., und *Rangie Amitirigala*, M.Sc., Asea Brown Boveri AG, Network Control, 5412 Gebenstorf.

Der Begriff *Expertensystem* ist nicht unumstritten. *Wissensbasierte Systeme* z.B. vermöchte das Arbeitsgebiet umfassender zu definieren: Wissen wird gesammelt, gespeichert und zu abgeleitetem Wissen verarbeitet. Da der Term Expertensysteme aber populärer ist, wird im folgenden daran festgehalten. Definitionen, die den Begriff Expertensystem festlegen, findet man in jedem einschlägigen Fachbuch. Da es sich jedoch um ein junges Arbeitsgebiet handelt und noch nicht alle Begriffe geklärt sind, finden sich kaum zwei gleiche Definitionen. Auch der untenstehende Versuch wird keine Ausnahme sein.

> Ein Expertensystem ist ein Computerprogramm, das gespeichertes Wissen eines engbegrenzten Fachgebietes verarbeitet, um eine gegebene Fragestellung zu beantworten.

Verarbeitung von Wissen

Schon beim Bau der ersten Computer hatte man sich darüber gestritten, ob diese Maschinen eigentlich Daten oder Wissen verarbeiten sollten. Die Umstände gaben den Ausschlag für Datenverarbeitung (d.h. Realisierung von numerischen Operationen). Und somit sind die heutigen, numerisch mächtigen Computer, vom Mainframe bis zum Supermini, das Produkt einer auf die Verarbeitung von Daten ausgerichteten Entwicklung.

Erst in der jüngeren Vergangenheit ist der zweite Aspekt, die Verarbeitung von Wissen, unter der Bezeichnung künstliche Intelligenz (KI) wiederum aufgegriffen worden. Während man in den Anfängen noch über *General Problem Solver* debattierte, hat man heute erkannt, dass mit der Enge des Arbeitsgebietes die Erfolgsaussichten steigen. Das heute noch unüberwindbare Hindernis für die Lösung von allgemeinen Aufgabenstellungen ist die Darstellung des allgemeinen Wissens. Der gesunde Menschenverstand lässt sich eben immer noch schwerlich in Regeln ausdrücken.

Unter den Sammelbegriff künstliche Intelligenz fallen Bereiche wie Erkennung und Verarbeitung von Sprache, Lernen, erfahrungsbezogene Verhaltensänderung (bei vorgegebener Situation) und die Neuralen Netzwerke[1] (Neural Networks), um nur einige der bekannten Teilgebiete zu nennen. Expertensysteme wurden und werden heute noch stark von solchen verwandten Bereichen beeinflusst. Die Darstellung von Wissen und das Lernen sind nur zwei Beispiele dazu.

Unrealistische Erwartungen

Mit KI und damit auch mit den Expertensystemen werden oft unrealistische Erwartungen verbunden. Neben einer unglücklichen Übersetzung des Begriffes KI aus dem Englischen (Artificial Intelligence), die beim Laien unrealistische Erwartungen weckt, hängt dies mit der intensiven Kopplung von Forschung und Entwicklung zusammen. Die Ideen und Prototypen aus dem Forschungslabor werden oft zu schnell als neue Produkte vermarktet. Für die sonst übliche Konsolidierung einer neuen Technik bleibt meist keine Zeit.

[1] Neurale Netzwerke versuchen die Verhaltensweise des menschlichen Gehirns zu modellieren.

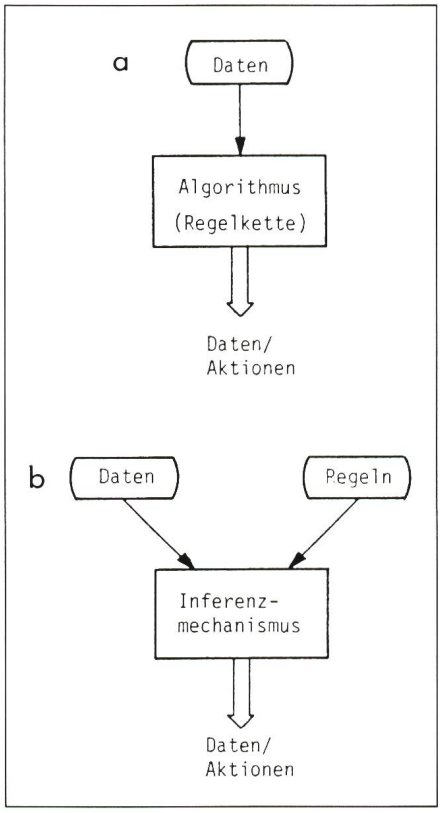
Spezielle Programmiersprachen und Software-Werkzeuge

Bei der Entwicklung von Expertensystemen werden oft spezielle Computersprachen wie *Lisp* oder *Prolog* verwendet[2]. Ein solches Computerprogramm wird dann als Expertensystem bezeichnet. Es handelt sich dabei um ein auf ein bestimmtes Fachgebiet spezialisiertes oder um ein «leeres» Expertensystem, ein sogenanntes Shell, in das Fachwissen noch geladen werden muss. Die Sprache, welche das Expertensystem Shell versteht, ist dann meistens spezifisch für das entsprechende Produkt und setzt, wenn auch Expertensystemsprachen üblicherweise benutzerangepasst sind, eine entsprechende Programmiererfahrung voraus. Die Figur 1 zeigt die unterschiedlichen Entwurfsstadien beim Erstellen von konventionellen Programmen und Expertensystemen.

Wie unterscheiden sich nun konventionelle Programmiersprachen von Sprachen zur Verarbeitung von Wissen? In konventionellen Sprachen muss der Programmierer im Algorithmus Schritt für Schritt definieren, wie ein Problem gelöst wird (Fig.2a). Das Wissen steckt somit implizit im Algorithmus (geringe Flexibilität). Bei einem Expertensystem gibt der Programmierer an, was für Wissen zum Lösen eines Problems benötigt wird. Die schrittweise Verknüpfung dieses Wissens ist dann Sache des Inferenzmechanismus (Fig. 2b). Dies bedeutet, dass sich der Benützer eines Expertensystems auf einer höheren Abstraktionsebene bewegt. Diesen Unterschied demonstriert das Beispiel in Tabelle I.

Anatomie eines Expertensystems

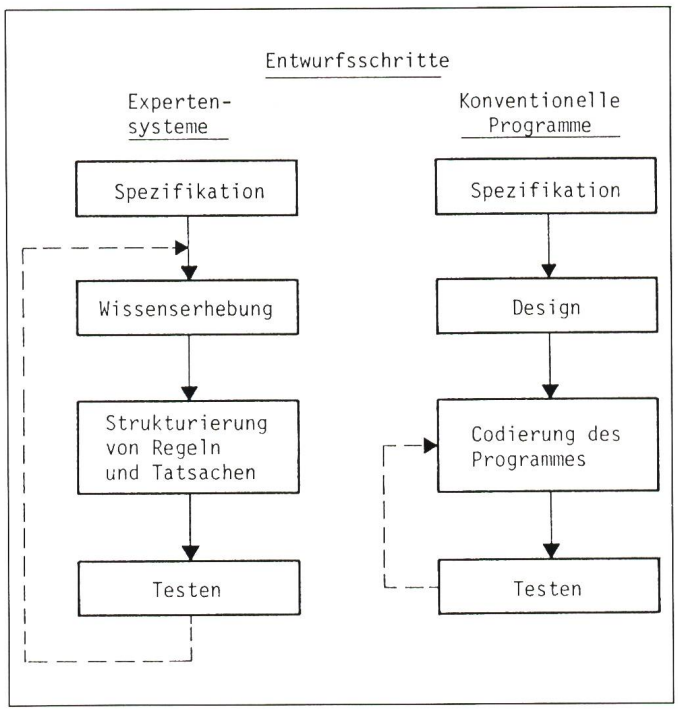
Alle Expertensysteme enthalten mindestens die drei Komponenten Wissensbasis, Inferenzmechanismus und Benützerschnittstelle. Die *Wissensbasis* stellt das gesammelte Wissen über ein gegebenes Gebiet dar. Der *Inferenzmechanismus* ist das Werkzeug, das aus dem momentanen Wissen neues, abgeleitetes Wissen erzeugt. Die *Benützerschnittstelle* ermöglicht dabei eine möglichst benützerfreundliche Kommunikation zwischen System und Benützer. Die Figur 3 gibt eine Übersicht über die drei Teile. Wissen in Form von Regeln und Tatsachen, das aus diversen Quellen zusammengesucht wurde, wird in die Wissensbasis abgelegt. Der Benützer konfrontiert nun das System mit Fragen über das spezifische Arbeitsgebiet via eine entsprechende Schnittstelle. In der Folge sucht der Inferenzmechanismus die Wissensbasis nach entsprechenden Antworten in Form von Tatsachen oder weiterführenden Regeln ab. Das Resultat wird mit mehr oder weniger intensiver Erklärung präsentiert.



Figur 2 Vergleich von Daten- und Wissensverarbeitung

Grundsätzlich können daten- und wissensverarbeitende Systeme dieselben Probleme lösen, allerdings nicht mit derselben Effizienz. Bei einem datenverarbeitenden System steckt das Wissen, d.h. die gegenseitige Zuordnung von Daten, im Algorithmus selbst. Das System ist bei Veränderung des Wissens sehr unflexibel (Umprogrammierung). Beim wissensverarbeitenden System ist das Wissen in strukturierter Form in die Wissensbasis ausgelagert. Der Algorithmus (Inferenzmechanismus) bleibt bei Veränderung des Wissensumfanges unverändert.

a Konventionelle Datenverarbeitung
b Bearbeitung von Wissen in Expertensystemen



Figur 1
Das Erstellen eines konventionellen Programmes im Vergleich zum Füllen eines Expertensystems.

[2] Eine konventionelle Sprache, obwohl nicht ausgerichtet auf die Verarbeitung von Wissen, könnte, wenn auch aufwendiger, den gleichen Zweck erfüllen.

KI- contra klassische Programmierung

Einen Einblick in den Unterschied zwischen klassischer Programmierung (Pascal) und KI-Programmierung (Prolog) gibt das nachfolgende Beispiel, in welchem zwei Personen und ihre Beziehung zueinander definiert werden.

In Pascal

```
TYPE    person = RECORD (*Beschreibung einer Person*)
          name: PACKED ARRAY [1 .. 12] OF CHAR; (*Name der Person*)
          boss: ↑ person; (*Pointer zu Boss der Person*)
          next: ↑ person; (*Pointer zur nächsten Person*)
        END;
VAR     a,b: ↑ person; (*Person a und b*)
BEGIN
...
a↑.name  := 'HANS';   (*Name der Person a*)
b↑.name  := 'MARIA';  (*Name der Person b*)
b↑.boss  := a;   (*Boss von Person b ist Person a*)
...
```

In Prolog

```
boss (HANS, MARIA)  /*Es gibt zwei Objekte mit dem Namen 'Hans'*/
                    /*und 'Maria', dazwischen eine Relation 'boss'*/
```

Tabelle I

Die Wissensbasis

Sie enthält das Wissen über das aktuelle Arbeitsgebiet. In gewissen Systemen können verschiedene Wissenbasen nebeneinander existieren. Wissensbasen können abgelegt und später wieder in das Expertensystem geladen, d.h. aktiviert werden. Eine Wissensbasis und damit das Wissen kann grob aufgeteilt werden in Daten (Tatsachen) und Regeln.

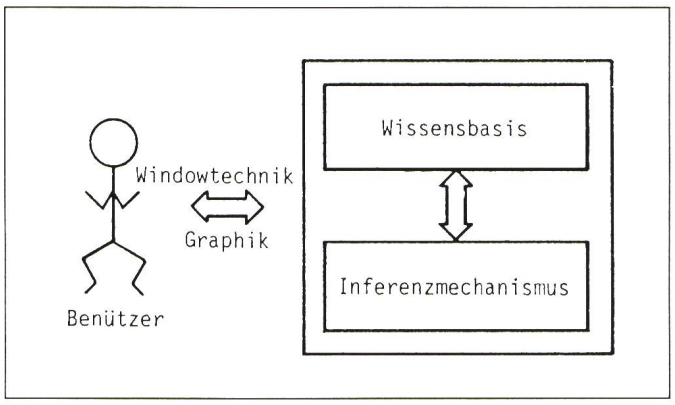
Daten

Die Wahl einer geeigneten Wissensdarstellung ist ein kritischer Aspekt. Bei wenig komplexem Wissen werden die Daten unstrukturiert in Dateien oder Datenbanken abgelegt. Bei komplexeren Systemen wird eine geeignete Datenstruktur verwendet. Neben *Frames* (vorgegebene Rahmen) für die Darstellung von Objekten existieren weniger gebräuchliche Methoden wie *semantische Netzwerke* für die Darstellung engverknüpfter Fakten, z.B. Sprachsätze, und Scripts zur Repräsentation von zeitlich sich verändernden Situationen.

Ein Frame enthält Informationen, die ein Objekt, z.B. eine Pumpe, vollständig beschreiben. Dazu gehören im allgemeinen nicht nur die passiven Daten, sondern auch die Funktionen, die das Verhalten des Objektes modellieren. Die Berechnung der momentanen Pumpenleistung aus Sensordaten und die Überwachung auf entsprechende Grenzwertüberschreitungen sind typische Beispiele dafür.

Das Frame ist der Grundbaustein für das Konzept des *modellbasierten Schlussfolgerns*. Modellierung ist eine natürliche und dem Ingenieur vertraute Vorgehensweise bei der Lösung von Problemen. Anstatt Wissen implizit zu speichern, werden die fraglichen Objekte modelliert; die Frames enthalten somit das entsprechende Wissen explizit. Ein Beispiel dazu: Bei der impliziten Darstellung werden Regeln wie «Falls mein Auto am Morgen nicht anspringt, ist wahrscheinlich die Batterie zuwenig geladen» verwendet. Bei der expliziten und anfänglich aufwendigeren Darstellung werden zuerst die drei Frames *Auto, Anlasser* und *Batterie* definiert. Eine der beliebig vielen Relationen zwischen den drei Frames widerspiegelt die obige Regel.

Das Arbeiten mit Frames oder Objekten, das *objektorientierte Programmieren* (Tab. II), ist ein eigener Zweig in der KI. Dieser neue Programmierstil ist mehr als eine neue Sprache. Es ist eine vielversprechende Methode zur effizienten Lösung von Programmieraufgaben. Moderne Expertensysteme sind durch diesen KI-Zweig wesentlich beeinflusst worden.



Figur 3
Der Aufbau eines Expertensystems
Die Wissensbasis enthält Tatsachen und Regelwissen, aus denen der Inferenzmechanismus neues Wissen ableitet.

Regeln

Eine Regel besteht im wesentlichen aus einer Prämisse, einem Bedingungsteil, und einer Schlussfolgerung, einem Aktionsteil:

IF «Prämisse» THEN «Schlussfolgerung»

Die Prämisse kann dabei aus einer beliebigen Anzahl von logischen und numerischen Bedingungen zusammengebaut werden. Der Schlussfolgerungsteil veranlasst Aktionen wie die Ausgabe von Informationen oder das Ändern von Daten. Im letzteren Fall können neue Vorbedingungen für andere Regeln erfüllt werden, die zu weiteren Aktionen führen. Eine Verkettung von Regeln ist die Folge. Die Regeln spezifizieren, wie das System reagieren soll, ohne dass detailliertes Wissen über den sequentiellen Ablauf der Regeln bekannt sein muss.

Die Abgrenzung zwischen Regel- und Datenwissen ist oft fliessend. Daten können durch Regeln ersetzt werden und umgekehrt, zum Beispiel kann die momentane Pumpenleistung als Datum direkt vorliegen, oder beim Zugriff zu diesem Datum wird eine Regel aktiviert, die die momentane Leistung ermittelt.

Das unsichere Wissen

Eine Regel leitet Wissen aus gegebenem Wissen ab. Das gegebene Wissen

Objektorientiertes Programmieren

Will ein Absender einen Brief, ein Paket oder ein Telegramm an einen gewünschten Ort senden, muss er ihn am richtigen Ort, üblicherweise bei der Post, aufgeben. Unabhängig von der Sendung ist ein solcher Übermittlungsdienst für die Ablieferung beim Empfänger besorgt. Während Art und Weise des Transportes für den Kunden unwichtig ist, ist anderseits der Inhalt der Sendung für die Post nicht wesentlich. Will man die Funktion eines solchen Übermittlungsdienstes auf einer abstrakten Ebene behandeln, so können die Objekte *Sendung* und *Endstellen* unterschieden werden. Der Oberbegriff *Sendung* ist aufgeteilt in die Objektklassen *Briefe, Pakete* und *Telegramme*. Während gewisse Eigenschaften sich auf untergeordnete Klassen vererben, sind andere spezifisch für eine bestimmte Klasse. So hat z.B. jede Sendung einen Absender und einen Empfänger, die Unterklasse *Paket* benötigt jedoch eventuell noch eine Zolldeklaration. Die Objekte der Klasse *Endstellen* wissen, wie Sendungen zum Empfänger gelangen. Für diesen Zweck tauschen sie Meldungen miteinander aus.

Beim objektorientierten Programmieren besteht die Welt des Programmierers aus solchen hierarchisch strukturierten, erbberechtigten *Objekten* und aus *Meldungen*. Die Objekte sind nicht nur die (passiven) Daten, sondern auch alle Funktionen, die auf das Objekt und seine Umgebung angewendet werden können (z.B. Übermittlung). Für die nötige Kommunikation zwischen diesen sorgen Meldungen. Die Implementation (Realisierung und Programmierung) der Objekte ist für den Benutzer nicht sichtbar; er kennt nur die Spezifikationen. Er weiss somit auch nicht, auf welchem Weg seine Sendung von A nach B gelangt. Eine Implementationsänderung ist für ihn nicht relevant.

In konventionellen Programmiersprachen bearbeiten die aktiven Prozeduren die passiven Daten: Eine Prozedur nimmt ein Datum, manipuliert es und legt den modifizierten Wert wieder ab. Wesentlich dabei ist die algorithmische Verarbeitung. Beim objektorientierten Vorgehen steht die Modellierung der Aufgabenstellung durch Objekte und Meldungen im Vordergrund: Alle Aufgaben, die wir mit Hilfe des Computers lösen, sei es ein Gleichungssystem, die Buchhaltung oder eine Prozessüberwachung, ist mit diesen zwei Sprachmitteln beschreibbar. Die Trennung von Implementation und Benützung von Objekten bedeutet, dass man auf einer höheren Abstraktionsstufe als bei konventionellen Sprachen arbeitet. Die hierarchische Vererbung von Objekteigenschaften erlaubt, dieselben Softwareeinheiten für mehrere ähnliche Aufgaben einzusetzen. Die Triebkraft hinter dem neuen Programmierstil ist der Wille zur Effizienzsteigerung (wie etwa beim Übergang von Assembler zu höherer Programmiersprache), wobei zu betonen ist, dass es sich dabei weniger um die Einführung neuer Programmiersprachen als um den Beginn einer neuen Denkweise handelt.

Tabelle II

sowie die Regel selber ist mit einer gewissen Unsicherheit behaftet. Diese Unsicherheit kann durch widersprüchliches Wissen oder durch Wissen, dessen Gültigkeit nur mit einer gewissen Wahrscheinlichkeit angenommen wird, verursacht sein. Im ersten Fall spricht man von nichtmonotonem (Non Monotonic Reasoning) und im zweiten Fall von wahrscheinlichkeitsbasiertem Schlussfolgern (Probabilistic Reasoning). In fortgeschrittenen Systemen kommen oft beide Methoden gleichzeitig zum Einsatz.

Beim wahrscheinlichkeitsbasierten Schlussfolgern werden jedem Datum und jeder Regelaussage entsprechende Wahrscheinlichkeitsfaktoren zugeordnet. Dabei kommen unterschiedlich genaue Berechnungsmethoden zum Einsatz. Ein erfolgversprechender Ansatz in diesem Gebiet ist die sog. Fuzzy Logic. Anstelle der Werte *wahr* und *nicht wahr* werden Werte wie *selten, manchmal, oft, meistens, fast immer* verwendet. Mit solchen Prädikaten wird Wissen als wahr, als wahrscheinlich oder als möglich qualifiziert.

Beim nichtmonotonen Schlussfolgern muss zu jeder Aussage die Verkettung der Regeln mit den entsprechenden Ausgangsdaten abgespeichert werden, die das abgeleitete Wissen unterstützen oder eventuell verwerfen (Truth Maintenance System). Damit können beim Auftauchen von neuen Tatsachen, die ein gegebenes Wissen erhärten oder in Frage stellen, die Wahrscheinlichkeiten für alles daraus abgeleitete Wissen modifiziert werden.

Der Inferenzmechanismus

Wissen aus Daten abzuleiten erfordert die Anwendung von Logik. Dieses logische Schlussfolgern wird als *Inferenz* bezeichnet. Inferenzmechanismen werden von uns tagtäglich angewendet. Dies geschieht immer dann, wenn aus unseren Wahrnehmungen Schlussfolgerungen gezogen werden. Ein Beispiel dazu: «Falls die Wohnungstüre geschlossen ist, benötige ich einen Schlüssel.» Gleiche Schlussfolgerungsverfahren werden beim Beweisen von mathematischen Formeln angewendet.

Einer der bekannten Inferenzmechanismen ist der sogenannte *Modus Ponens*. Er ist der algorithmische Grundbaustein von vielen Expertensystemen. Ein einfaches, erklärendes Beispiel dazu:

1. Jeder Onkel ist männlich.
2. Hans ist ein Onkel.

Als Schlussfolgerung aus 1 und 2 ergibt sich: Hans ist männlich.

Der Inferenzmechanismus basiert auf einer allgemeinen Problemlösungsstrategie. Dabei werden mögliche Zwischenzustände, die näher in Richtung Lösung liegen, erzeugt, getestet, weiterverfolgt oder verworfen. Ein solches systematisches Suchverfahren ist unten skizziert.

1. Wähle einen Kandidaten (einen Zustand) aus der Liste aus und wende alle möglichen logischen Schlussfolgerungen darauf an.
2. Trage die neuerzeugten Kandidaten in die Liste ein.
3. Falls einer der neuen Kandidaten dem Zielzustand entspricht, ist eine mögliche Lösung gefunden, sonst gehe zurück zu Schritt 1.

Dieser Mechanismus führt zu einem Suchbaum, der je nach Auswahl des nächsten Kandidaten unterschiedlich durchlaufen wird. Die Knoten sind dabei die Kandidaten, und die Übergänge repräsentieren die dazu verwendeten Inferenzen (Figur 4).

Die Selektion des nächsten Kandidaten, z.B. eine Schachbrettkonfiguration, und die Auswahl der darauf anzuwendenden Regeln (Conflict Resolution), z.B. der nächste Schachzug, ist oft der eigentliche *Kern des Inferenzverfahrens*. Sogenannte *Metaregeln* bestimmen dann, nach welchen Kriterien die Auswahl der Kandidaten und Regeln getroffen wird. Sowohl Strategien wie das Ausführen der Regeln, die zuletzt ins Set der erfüllten Regeln eingetragen wurden, als auch problemspezifische Kriterien kommen dabei zum Einsatz. In gewissen Inferenzmechanismen sind solche Metaregeln direkt einprogrammiert. In anderen

Systemen kann der Benützer entsprechende Regeln zusätzlich definieren. Bei einigen Suchstrategien werden die Teillösungen bewertet. Mögliche Kriterien dazu sind die verbleibende Distanz zur Lösung oder der bereits geleistete Aufwand für die Erzeugung der Teillösung.

Beim Suchen eines möglichen Weges zwischen Problemstellung und Lösung gibt es grundsätzlich zwei unterschiedliche Stossrichtungen. Bei der *vorwärtsgerichteten* Methode startet man bei der Problemstellung, bei der rückwärtsgerichteten Strategie bei der Lösung selber. Dazu sucht man Wissen, das die vorgeschlagene Lösung unterstützt oder verwirft. Die Methoden werden oft auch gemischt angewendet. Der Entscheid, welcher Weg optimal ist, hängt von der Aufgabenstellung ab. Das Kriterium dazu ist die Anzahl möglicher Startpunkte. Je weniger mögliche Verzweigungspunkte vorhanden sind, desto kleiner wird der Suchaufwand für einen möglichen Weg zwischen Problem und Lösung sein. Ein Beispiel: Bei der Suche der zu benützenden Bahnlinien von Zürich nach Disentis wird man sicher in Disentis starten[3].

Die Benützerschnittstelle

Die Benützerschnittstelle besorgt den Austausch von Informationen zwischen dem Benützer und dem Expertensystem, wie Informationsabfrage, Darstellen von Schlussfolgerungsabläufen und Präsentation von Resultaten. An der Fähigkeit, Ein- und Ausgabeinformationen zu visualisieren, wird oft das ganze System beurteilt, d. h. die Effizienz und Natürlichkeit der Informationsfrage und ihre Darstellung ist vielfach entscheidend für die Akzeptanz des neuen Werkzeuges.

Mögliche Eingabemedien sind neben der konventionellen Tastatur Mittel wie Maus und Leuchtstift. Informationseingabe ist auf zweierlei Art möglich: entweder als Antwort auf eine automatische Abfrage durch den Inferenzmechanismus oder als explizit programmierte Eingabe. Für die Resultatpräsentation kann Graphik-Software zugezogen werden, welche erlaubt, selbst komplexe Information einfach darzustellen.

Als Eingabe- und Ausgabemedium kommen auch Datenbanken, Texteditoren, Sensoren und Steuerungskomponenten in Frage. In solchen Fällen ist das Expertensystem Kernstück eines sog. integrierten Systems. Ein einfaches Beispiel dazu:

In einem Versandhaus ist das Bestellwesen automatisiert. Lagerbestand sowie Kundenkartei sind in entsprechenden Datenbanken abgelegt. Bei der Abwicklung einer Bestellung werden aus diesen Datenbanken Informationen für die Rechnungsstellung abgerufen. Die Regeln des integrierten Expertensystems bestimmen Mengen- und Kundenrabatte sowie zusätzliche Kosten für Versand und Porto in Abhängigkeit von Zustellart und Ort.

Antworten von Expertensystemen sind glaubwürdiger und verständlicher, wenn sie begründet werden. Damit befasst sich die *Erklärungsmethodik*. Die einfachste und bekannteste Methode ist, die Kette der aktivierten Regeln, die sog. Argumentationskette, darzustellen. Wegen der Menge der angesprochenen Regeln kann dies allerdings sehr unübersichtlich werden. Andere Systeme beantworten Fragen wie z. B., was passieren würde, wenn gewisse Daten geändert oder fehlen würden (hypothetisches Schlussfolgern) oder warum eine bestimmte Schlussfolgerung nicht in Frage kommt.

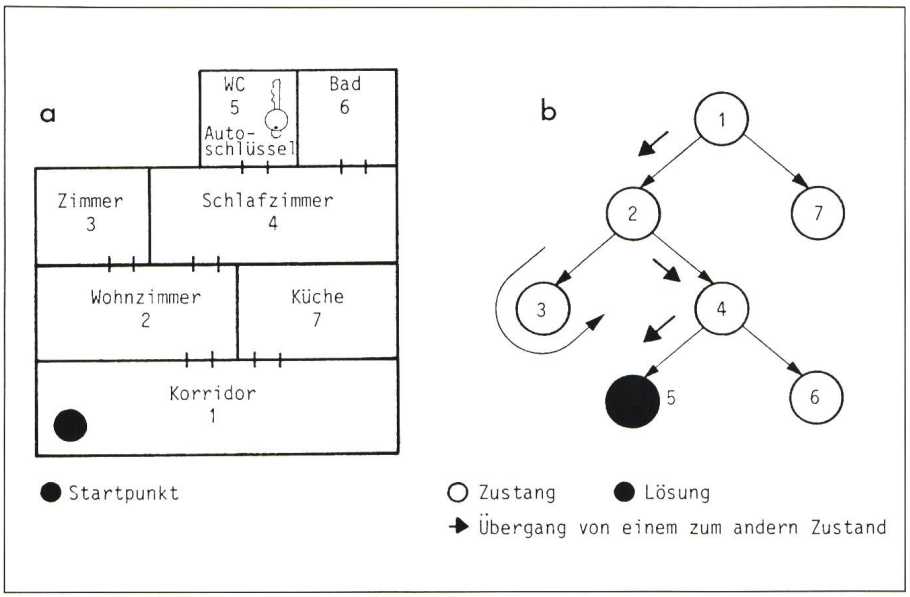
Typische Anwendungsgebiete

Überall dort, wo Wissen heuristisch verarbeitet wird, d.h. wo systematisch aus gegebenem Wissen Schlussfolgerungen zu ziehen sind, ist der Einsatz von Expertensystemen vielversprechend. In der Praxis haben sich darum folgende typische Gebiete herauskristallisiert:

Interpretation und Diagnose: Eine gemessene und gegebene Situation wird erfasst und untersucht. Aus der Analyse können Schlussfolgerungen und Aktionen abgeleitet werden. Beispiele: medizinische Diagnose, Überwachen eines Prozesses.

Vorhersage: Herleiten von möglichen Konsequenzen aus einer gegebenen Situation unter Zuhilfenahme von ähnlichen Vorfällen, statistischen Zahlen usw. Beispiele: Wettervorhersage, Risikoanalyse bei der Prüfung von Krediten.

Planung und Entwurf: Bestimmungen der örtlichen und zeitlichen Anordnung von Objekten und Aktivitä-



Figur 4 Suchstrategie

a Problemstellung: Wo ist mein Autosschlüssel

b Lösungsweg: Transformation der Realität in den entsprechenden Suchbaum und Abarbeiten des Suchbaumes nach einer vorgewählten Strategie.

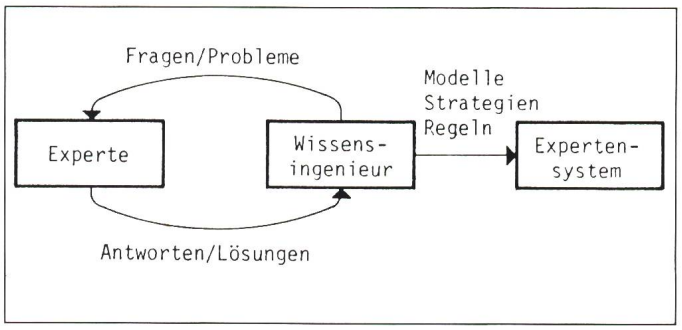
[3] Regeln werden vom Inferenzmechanismus direkt interpretiert oder vor der Verarbeitung kompiliert. Hier zeigen sich Parallelen zu einem konventionellen Programm. Bei der Verwendung von kompilierten Regeln handelt es sich im wesentlichen um ein konventionelles Suchprogramm, das mit Regeln vom Benützer gesteuert wird.

ten unter Befolgung von entsprechenden Bedingungen. Beispiele: Terminplanung, Synthese von Computersystemen.

Instruktion: Hilfsmittel zur Leitung, Kontrolle und Überprüfung eines Schülers.

Führung: Kontrollieren des Einsatzes von mehreren Teilsystemen, um ein übergeordnetes Ziel zu erreichen. Beispiele: Hilfsmittel für das Management, Führen eines technischen Prozesses.

Figur 5
Transformation von Expertenwissen in Expertensystemwissen



Das Bereitstellen von Wissen

Erst die Aufbereitung von Wissen in eine wohldefinierte Form macht ein Softwaresystem zu einem Expertensystem. Damit ist eine der heikelsten Phasen in der Entwicklung eines Expertensystems angesprochen. Mit Ausnahme von einfachen Hilfsmitteln existieren kaum Werkzeuge, welche Akquisition und Darstellung von Wissen rationalisieren. Wissen kann von diversen Orten kommen. Neben den unproblematischen Quellen wie Text- und Handbücher findet sich das Wissen vor allem bei den entsprechenden *Fachexperten*. Deren Befragung und die Umwandlung der erhaltenen Information in eine für Expertensysteme brauchbare Form ist ein iteratives und somit aufwendiges Unterfangen (Fig. 5). Es werden dazu mehr und mehr speziell geschulte Personen eingesetzt (Knowledge Engineers). Wie man sich leicht vorstellen kann, sind dabei die psychologischen Probleme oft anspruchsvoller als die technischen. Die Neuartigkeit des Werkzeugs und ein Widerstand, eigenes Wissen preiszugeben, sind die (oft unbewussten) Hauptgründe für diese Schwierigkeiten.

Es muss auch bedacht werden, dass menschliche Experten ihr Wissen in einer vorverarbeiteten Weise speichern. Bei einem bereits bekannten Problem wird eine Kette von gespeicherten Schlussfolgerungen aktiviert. Dies geschieht eher intuitiv als bewusst. Das heisst, ein bekanntes Problem wird direkt mit einer Lösung assoziiert. Dies führt zum folgenden Experten-Paradoxon:

> Je kompetenter ein Fachexperte ist, desto weniger ist es ihm möglich, sein Problemlösungswissen weiterzugeben.

Somit ist es selten wirksam, Experten direkt nach Regeln zu befragen. Die Antworten sind meistens zu generell und enthalten nicht identifizierte Abstraktionen. Diverse Methoden, wie das Lösen von spezifischen Fällen, Beobachten des Experten bei seiner Arbeit, Konfrontation des Expertensystemablaufs mit dem Schlussfolgern des Fachexperten, haben sich in der Praxis bewährt.

Die Befragung des Experten, gefolgt von einer Überarbeitung bzw. Erweiterung des vorhandenen Regelsatzes, des Wissens, ist ein iteratives Verfahren. Dies kann sich über Wochen oder Monate erstrecken. Zur Illustration dazu Ausschnitte aus der Befragung (F) eines «Wetterfrosches» (A):

Erstes Interview:

F: Warum wird es morgen regnen?
A: Oh! Der Nordhimmel ist rot heute abend.
F: Was ist, wenn der Himmel im Nordosten rot ist?
A: Egal. Solange es im Norden so ist, wird es regnen.
F: Ist das alles?
A: Ja, sicher.

Regel 1: IF «Abendrot am Nordhimmel» THEN «Es regnet morgen»

Zweites Interview:

...
F: Die Regel fürs Regnen am nächsten Morgen stimmt noch?
A: Oh! Wenn die kleinen Vögel dort tief fliegen am Abend.
F: Wie ist es mit dem Abendrot?
A: Ja, das auch oder wenn die Vögel tief fliegen.

Regel 1: IF «Abendrot am Nordhimmel» OR «Vögel fliegen tief» THEN «Es regnet morgen»

Expertensystemmarkt

Der Markt für Expertensysteme wird heute sehr optimistisch eingeschätzt. Diagnosensysteme, Risikobeurteilung, Materialauswahl und Beratungssysteme sind dabei die erfolgreichsten Anwendungen. Optimistische Studien prognostizieren eine jährliche Zuwachsrate von etwa 50%. Um die Jahrtausendwende wird ein 25%-Marktanteil der Expertensysteme in der Computerbranche erwartet.

Bei den Herstellern wird der momentane Markt noch kritisch beurteilt. Der Schritt aus den roten Zahlen ist oft schwieriger, als es die entsprechenden Marktstudien vermuten lassen. Während kleine und klar limitierte Systeme sich als nützliche Hilfsmittel erwiesen haben, sind die grossen und professionellen Systeme noch voll von Problemen.

Bei den *kleinen Shells* handelt es sich um Produkte, die meistens auf PCs implementiert sind. Die Preise variieren typischerweise zwischen 100 und 2000 Schweizer Franken. Solche Kleinsysteme bestehen oft nur aus einem Inferenzmechanismus mit limitiertem Instruktionsset und einer eigenen formalen Struktur für das Wissen (Wissensbasis). Die häufigsten Einsatzmöglichkeiten ergeben sich bei der Diagnose, bei Testsystemen für weiter führende Entwicklungen oder beim Einstieg in die neue Technik.

Grosse Shells sind flexibler. Meistens enthalten sie mehrere Optionen zur Strukturierung und Manipulation von Wissen. Der Unsicherheitsgrad des Wissens kann mit mehreren Methoden dargestellt werden. Der Zugriff von einer KI-Sprache zu einer konventionellen Programmiersprache ist über eine entsprechende Schnittstelle möglich. Fakten werden in Files oder in

einer Datenbank abgelegt. Zum Füllen des Shells mit Wissen stehen Hilfsmittel wie Regeleditoren mit Windowtechnik zur Verfügung. Zusätzliche Werkzeuge wie Spreadsheet und Texteditoren machen solche Shells zu einem universellen Arbeitsmittel für die Verarbeitung von Wissen. Kee von Intellicorp Inc. and Guru, ein Produkt von Micro Data Base Systems Inc., sind nur zwei Beispiele für solche grossen Shells.

Bei *Guru* handelt es sich um eine Entwicklungsumgebung, bestehend aus relationaler Datenbank, Spreadsheet, Graphik, Textverarbeitung und Expertensystemkomponente. Verschiedene Benutzeroberflächen erlauben, unterschiedlich nahe am System zu arbeiten, z. B. über systemgeführte Menüs oder mittels direkter Kommandosprache. Bei der Regeleingabe gibt man an, von wo entsprechende Information bezogen werden soll. Die Gestaltung einer Konsultation mit dem Benützer wird nicht wie bei den meisten Produkten vom System geführt, sondern muss explizit vorgegeben werden, d. h. der Wissensingenieur bestimmt, mit welchen Fragen das System wonach zu fragen hat.

Kee ist ein Expertensystem, das sich stark auf Frames abstützt. Neben Programmieren in Regeln ermöglicht es objektorientiertes und prozedurales Arbeiten. Das auf Lisp basierende System erlaubt die Koexistenz von mehreren Wissensbasen. Kee unterstützt die Verarbeitung von unsicherem und inkonsistentem Wissen. Will man einen schnellen Ablauf erreichen, so kann ein Regelcompiler verwendet werden. Teil der Benützerschnittstelle ist ein Multiwindow-System und eine graphische Regel- und Erklärungskomponente.

Schlusswort

Das Gebiet der Expertensysteme ist ein junges, herausforderndes Betätigungsfeld. Da aber in den letzten Jahren mehr darüber philosophiert als geleistet wurde, hat das neue Werkzeug vielerorts an Kredit verloren. Expertensysteme wurden für alle möglichen Aufgaben angepriesen; es kann ja auch jede Aufgabe aus Objekten und Regeln zusammengesetzt werden. Nur sollte man die Grenzen des neuen Werkzeugs respektieren, und das ist in der Vergangenheit oft zuwenig geschehen. Der Einsatz von Expertensystemen ist limitiert: Falls ein Algorithmus oder ein spezifisches Problemlösungsverfahren vorhanden ist, sind Expertensysteme im allgemeinen fehl am Platz. Es gibt noch weitere Beschränkungen: Expertensysteme können sich nur schlecht an sich ändernde Bedingungen anpassen. Die Kreativität des menschlichen Experten fehlt gänzlich. Expertensysteme haben meistens ein sehr limitiertes Wissen; der «gesunde Menschenverstand» geht ihnen gänzlich ab. Falls jedoch die Aufgabenstellung nicht zu komplex ist, das Wissen von mehreren Experten übereinstimmend artikuliert werden kann und kein Allgemeinwissen benötigt wird, ist der Einsatz von Expertensystemen möglich. Falls zudem die Anwendung Gewinn verspricht, entsprechende Experten selten sind oder an mehreren bzw. exponierten Orten benötigt werden, ist der Einsatz sogar sinnvoll.

Ein kleiner Führer durch die Literatur

Im jungen Arbeitsgebiet der Expertensysteme sind die Grenzen zu verwandten Gebieten noch sehr vage abgesteckt. Diese noch offenen Grenzen widerspiegeln sich auch in der entsprechenden Literatur. Standbücher fehlen, und wichtige Beiträge findet man oft ausserhalb des KI-Gebietes. Trotzdem wurde versucht, eine geeignete Auswahl zu treffen. Neben diesen reinen KI- und Expertensystem-Zeitschriften gibt es natürlich zahlreiche weitere Publikationen in den verschiedensten Zeitschriften.

Literatur

[1] *E. Rich:* Artificial intelligence. New York a.o., McGraw-Hill, 1983.

[2] *D.A. Waterman:* A guide to expert systems. Reading/Massachusetts, Addison-Wesley, 1986.

[3] *F. Hayes-Roth, D.A. Watermann* and *P.R. Cohen:* Building expert systems. Reading/Massachusetts, Addison-Wesley, 1983.

[4] *A. Barr, E.A. Feigenbaum* and *P.R. Cohen:* Handbook of artificial intelligence. 3 volumes. London, Pitmann a.o., 1981.

[5] *P.H. Winston* and *B.K.P. Horn:* Lisp. Reading/Massachusetts, Addison-Wesley, 1984.

[6] *W.F. Clocksin* and *C.S. Mellish:* Programming in Prolog. Second edition. Berlin a.o., Springer, 1984.

[7] *B. Sawyer* and *D.L. Foster:* Programming expert systems in Pascal. New York, John Wiley, 1986.