Expertensysteme im Recht

— Eine Einführung —

M. Lusti

1. Einleitung

"A computer is like a violin. You can imagine a novice trying first a phonograph and then a violin. The latter, he says, sounds terrible. That is the argument we have heard from our humanists and most of our computer scientists. Computer programs are good, they say, for particular purposes, but they aren't flexible. Neither is a violin, or a typewriter, until you learn how to use it."

Diese optimistische Meinung über Einsatzmöglichkeiten von Computerprogrammen stammt aus einem Aufsatz von Marvin Minsky, einem der Pioniere der Künstlichen Intelligenz. Der Aufsatz trägt den Titel "Why Programming Is a Good Medium for Expressing Poorly-Understood and Sloppily-Formulated Ideas". Ziel der folgenden Ausführungen ist es, an Beispielen zu veranschaulichen, wie weit Minsky's These auf die Programmierung von Rechtsnormen zutrifft. Abschnitt 2 führt Nichtinformatiker in das Gebiet der Expertensysteme ein. Abschnitt 3 geht nach einem Überblick auf ausgewählte Probleme bei der Entwicklung juristischer Expertensysteme ein, und Abschnitt 4 zeigt, daß trotz aller Fortschritte der Hardware- und Softwaretechnologie der „Justizautomat" von Max Weber eine Fiktion bleibt. Ein Glossar am Ende des Aufsatzes umschreibt für den Nichtinformatiker einige im Text verwendete Fachbegriffe.

2. Expertensysteme

Ein *Expertensystem* ist ein Computerprogramm, das in einem engen Bereich Probleme löst, die sonst das Wissen menschlicher Experten voraussetzen. Solche Programme helfen zum Beispiel bei der Diagnose und Therapie bestimmter Krankheiten, bei der Suche nach Mineralvorkommen und beim Entwurf elektronischer Schaltungen (Bilder 1 und 2):

Name	Anwendung
DELTA	berät bei der Diagnose und Reparatur von Defekten an Lokomotiven
DENDRAL	identifiziert Molekülstrukturen aufgrund von massenspektrographischen Daten
DIPMETER ADVISOR	leitet aus Bohrversuchen Erkenntnisse über den geologischen Aufbau der Bohrstelle ab
MACSYMA	vereinfacht algebraische Ausdrücke, integriert symbolisch und löst Gleichungen
MYCIN	berät Ärzte bei der Auswahl von Antibiotika für die Behandlung bestimmter Infektionskrankheiten
XCON	stellt aufgrund von Kundenwünschen mögliche Konfigurationen von Computersystemen eines bestimmten Typs zusammen

Bild 1: Einige Beispiele bekannter Expertensysteme

Anwendung	Beispiele
Diagnose	von Maschinenfehlern und Krankheiten
Entwurf	von elektronischen Schaltungen und molekularen Strukturen
Überwachung	von Instrumenten in einem Reaktor
Ausbildung	in der Bedienung von Computerprogrammen

Bild 2: Einige Anwendungsbereiche von Expertensystemen

Gegenstand von Expertensystemen sind weniger algorithmisierbare Aufgaben, als Probleme, die sich nur (oder schneller) mit heuristischem Erfahrungswissen lösen lassen. Im Gegensatz zu konventionellen Programmen stellt ein Expertensystem sein Wissen meist in einer Form dar, die es erlaubt, Lösungswege zu begründen und neues Wissen leicht zu integrieren.

Die bekanntesten Expertensysteme stammen aus den Naturwissenschaften (zum Beispiel aus Medizin, Chemie und Geologie). Mit den wachsenden Erfahrungen im Bau von solchen Programmen und der Verfügbarkeit von mächtigen und benutzerfreundlichen Entwicklungswerkzeugen wächst der Anreiz, Expertensysteme auch auf andern Gebieten, insbesondere in Wirtschaft und Verwaltung, zu entwickeln. Ein frühes Beispiel ist *TAXADVISOR* (MIC82, MIC84), ein Programm, das Erfahrungen mit dem medizinischen System MYCIN (vgl. Bild 1) auf die Investitions- und Steuerberatung überträgt. Ausgewählte Aspekte von TAXADVISOR sollen einige Grundmuster von Expertensystemen veranschaulichen.

Gegenstand von TAXADVISOR ist die langfristige Finanzplanung eines privaten Haushalts. Diese Aufgabe erfordert vor allem Expertenwissen aus dem Steuerrecht und dem Bank- und Versicherungswesen. TAXADVISOR ist in EMYCIN implementiert. EMYCIN ("Empty MYCIN") ist ein *Werkzeug* zur Entwicklung von Expertensystemen, das jene Komponenten von MYCIN verwendet, die unabhängig vom konkreten Anwendungsbereich (Diagnose und medikamentöse Therapie bestimmter Infektionskrankheiten) sind (BUC84, Bild 3). EMYCIN ist eines von vielen Expert System Tools. [BAR83], [WAT85] und [HAR85] geben einen Überblick über den State of the Art solcher Entwicklungswerkzeuge.

Der Entwickler eines konkreten Expertensystems baut mit Hilfe von EMYCIN eine Wissensbank von *Tatsachen und Regeln* über den Gegenstandsbereich des Systems, in unserm Beispiel über die langfristige Finanzplanung privater Haushalte. EMYCIN stellt dem Endbenutzer (zum Beispiel einem Steuerberater und dessen Kunden) während einer Konsultation Fra-

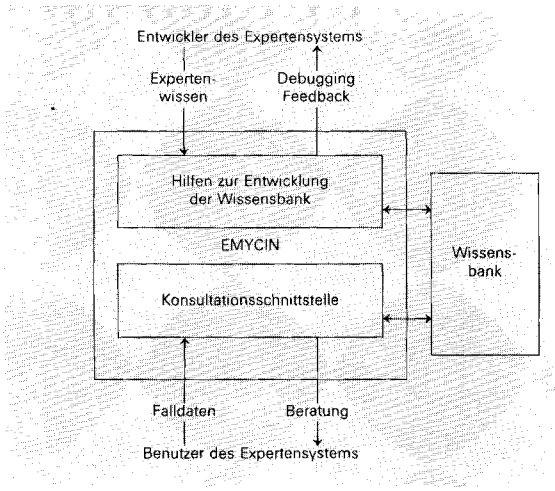

Bild 3: Die Interaktion zwischen EMYCIN und dem Entwickler bzw. Endbenutzer eines Expertensystems (nach MEL84 303)

gen zum konkreten Fall und leitet aus der Wissensbank und den gesammelten Falldaten Empfehlungen ab. Der Endbenutzer kann jederzeit eine Begründung der gestellten Fragen und der abgeleiteten Empfehlungen verlangen.

Eine EMYCIN-Regel hat die Form „WENN Voraussetzung DANN Folgerung", wobei die Voraussetzung aus einer Konjunktion von Bedingungen besteht, die wahr oder falsch sein können. Bild 4 zeigt ein (vereinfachtes) Beispiel einer Regel aus TAXADVISOR und den durch die Regel initiierten Dialog mit dem Endbenutzer.

Bild 4 geht davon aus, daß der Benutzer bzw. das System von Bedingungen und Folgerungen mit Sicherheit sagen kann, ob sie zutreffen oder nicht. In EMYCIN und anderen Werkzeugen zur Entwicklung von Expertensystemen kann der Benutzer, seine Antworten mit einer (subjektiven) Wahrscheinlichkeits-Schätzung charakterisieren. Aufgrund eines Wahrscheinlichkeitsmodells versieht das System eine Folgerung mit einem aus den Wahrscheinlichkeiten der Bedingungen abgeleiteten *Vertrauensfaktor* (zum Beispiel einem Wert zwischen +1 (sicher) und −1 (unmöglich)).

Obwohl das Gebiet der Expertensysteme noch jung ist, ist die Literatur bereits kaum mehr zu überblicken (für Einführungen und Sammelreferate siehe etwa [WAT85], [HAR85], [RAU84] und [HAY83]). Eine Einschätzung der Zukunft, vor allem der praktischen Bedeutung von Expertensystemen ist schwierig. Die gegenwärtige Euphorie auf dem Gebiet der Anwendungen von Methoden und Werkzeugen der Künstlichen Intelligenz, insbesondere der Expertensysteme wird von einigen Fachleuten kritisch kommentiert. So relativiert ein Aufsatz mit dem Titel "The *overselling* of expert systems" den Erfolg bekannter Expertensysteme unter anderem damit, daß diese Programme von ausgezeichneten Leuten mit gewaltigen Ressourcen (Zeit und Geld) in nichtstandardisierten Entwicklungsumgebungen geschaffen wurden [MAR84]. Die folgenden Ausführungen zu Expertensystemen im Recht sind.

a) Regel
WENN
(1) der Kunde und (oder) dessen Ehepartner bereit sind (ist), Vermögen zu treuen Händen für mindestens 10 Jahre oder bis zum Tod des Treuhänders zu übergeben UND
(3) der Kunde und (oder) dessen Ehepartner in einer höheren Einkommensklasse als der Treuhänder sind (ist) UND
(5) der Kunde und (oder) dessen Ehepartner in der Lage sind (ist), auch ohne dieses Vermögen — selbst bei Arbeitsunfähigkeit oder Arbeitslosigkeit — den Lebensunterhalt zu bestreiten UND
...
DANN
ist eine Übergabe von Vermögen zu treuen Händen empfehlenswert.

b) Dialog mit dem Benutzer (Antworten des Benutzers in Großbuchstaben)
(1) Sind (ist) der Kunde und (oder) dessen Ehepartner bereit, Vermögen zu treuen Händen für mindestens 10 Jahre oder bis zum Tod des Treuhänders zu übergeben?
**JA
(3) Sind (ist) der Kunde und (oder) dessen Ehepartner in einer höheren Einkommensklasse als der Treuhänder?
**JA
(5) Sind (ist) der Kunde und (oder) dessen Ehepartner in der Lage, auch ohne dieses Vermögen — selbst bei Arbeitsunfähigkeit oder Arbeitslosigkeit — den Lebensunterhalt zu bestreiten?
**JA
...
Ich empfehle ein Treuhandverhältnis.

Bild 4: Eine Regel aus TAXADVISOR und der dadurch initiierte Dialog mit dem Benutzer (vereinfacht nach [MIC84], 156 und [MIC85], 306)

denn auch vor diesem Hintergrund eher Hinweise auf Forschungsaktivitäten in einem frühen Stadium als eine Beschreibung verbreitet eingesetzter Systeme.

3. Expertensysteme im Recht

Obwohl Expertensysteme auf den Gebieten der Rechtssetzung und -anwendung selten sind und in der Regel Forschungsprototypen geblieben sind, gelten vor allem ausgewählte Bereiche der Rechtsanwendung als vielversprechende Kandidaten für Expertensysteme (Bild 5):

Anwendung	Beschreibung
Interpretation, Vorhersage	Interpretation von Rechtsnormen, Simulation von Änderungen von Rechtsnormen (z. B. im Sozialversicherungsrecht oder in der Mieterschutzgesetzgebung)
Dokumentenerstellung	Erstellen juristischer Dokumente, z. B. von Testamenten und Verträgen
Vorbereitung der Rechtsprechung	Strukturierung von Falldaten, Schätzung von Streitwerten, Festlegen von Verhandlungsstrategien
Überwachung	Inspektion von Rechtsdatenbanken und Meldung von Änderungen, die den Benutzer betreffen
Ausbildung	Intelligente Tutorensysteme in der juristischen Ausbildung (z. B. Simulation von juristischen Entscheidungsprozessen)

Bild 5: Anwendungsbereiche von Expertensystemen im Recht (in Ergänzung von [WAT85], 224 ff.)

Die Beispiele in Bild 6 zeigen zwei *Forschungsschwerpunkte:* (2), (3), (6) und (9) konzentrieren sich auf die Darstellung einfacher Vorschriften des geschriebenen Rechts und ihre Anwendung auf bestimmte Falldaten. (1), (5), (8), (10) und (11) versuchen, auch schwer operationalisierbare und außerjuristische Entscheidungsfaktoren einzubeziehen. Der folgende Abschnitt beschreibt den ersten, der übernächste Abschnitt am Beispiel von (10) den zweiten Anwendungsschwerpunkt.

Quelle (Name des Systems)	Gegenstand	Methode	Implementierung	Adressaten
(1) BIN80 (SARA)	Analyse von Kannvorschriften	Frames	Norwegian Research Center for Computers and Law	(Forschung)
(2) COR84, KOW85	British Nationality Act von 1981	logische Programmierung	micro-PROLOG Imperial College London	(Forschung)
(3) KEM84	Nichterfüllung von Außenhandelsliefer-verträgen	vernetzte Entscheidungstabellen	Werkzeuge der konventionellen Programmierung Hochschule für Ökonomie, Berlin (DDR)	Kaufmann des Außenhandels
(4) KRU84 (DSCAS)	"differing site condition claims"	Produktionsregeln	ROSIE Rand Corporation	Bauunternehmen
(5) MCC82 (TAXMAN I, II)	Auslegung auch unbestimmter Rechtsbegriffe am Beispiel von steuerrechtlichen Normen	Frames, Variation prototypischer Rechtsnormen	Rutgers University AIMDS	(Forschung)
(6) HAM83a, HAM83b	Sozialversicherung (Dept. of Health and Social Security Benefit, DHSS)	logische Programmierung	micro-PROLOG APES (A Prolog Expert System Shell)	DHSS offices
(7) MEL75 (LEGAL ANALYSIS SYSTEM)	absichtliche Körperverletzung		PSL (Preliminary Study Language) MIT	(Forschung)
(8) MIC82 (TAXADVISOR)	Vermögensplanung unter Berücksichtigung von Steuerrecht (Vermögen über $175000)	vorwärts verkettete Regeln	EMYCIN University of Illinois	Steuerberater
(9) SCH85 (Tax Advisor)	Steuersubjekte von Beteiligungsvermögen	logische Programmierung	Prolog-86	Steuerberater, Anwalt
(10) WAT80 (Legal Decisionmaking System)	Produktehaftung	Produktionsregeln	ROSIE Rand Corporation	Versicherung
(11) WAT85 (System for Asbestos Litigation)	Haftung für Asbestschäden	Produktionsregeln	ROSIE Rand Corporation	Versicherung

Bild 6: Beispiele von Expertensystemen im Recht

3.1 Die logische Programmierung von Rechtsnormen

Versuche, die Rechtssprache zu formalisieren und der Programmierung zu erschließen sind auch außerhalb des Gebiets der Expertensysteme unternommen worden (siehe z.B. [COO80], [FOH85]). Im Zusammenhang mit dem Bau von Expertensystemen stehen vor allem Arbeiten, welche Rechtsnormen in der Sprache der logischen Programmierung abbilden. Die *logische Programmierung* [LLO84] erlaubt es, aus Aussagensystemen von der Mächtigkeit der Prädikatenlogik erster Ordnung automatisch (durch ein Computerprogramm) Folgerungen abzuleiten. Die bekannteste logische Programmiersprache ist *Prolog* [CLO84]. Die folgenden Beispiele illustrieren die Darstellung von Rechtsnormen in einem Prolog-Dialekt.

Ein logisches Programm kann Rechtsnormen interpretieren, die im Prädikatenkalkül erster Ordnung darstellbar sind. Ein einfaches Beispiel (nach [COR84] und [KOW85]) soll dies veranschaulichen. Paragraph 1.1 des British Nationality Act von 1981 lautet: "A person born in the United Kingdom after commencement [of the act, der Verf.] shall be a British citizen if at the time of birth his father or mother is (a) a British citizen or (b) settled in the United Kingdom". Teil (a) dieser Definition der britischen Staatszugehörigkeit läßt sich zum Beispiel wie folgt als *Aussage eines logischen Programms* darstellen (Die folgenden Beispiele sind in micro-PROLOG geschrieben ([CLA84]):

x wird (Britischer Staatsbürger) nach-Norm (1.1a) zum-Zeitpunkt t IF

x geboren-in UK AND
x geboren-zum-Zeitpunkt t AND
t ist-nach-Inkrafttreten AND
y ist-Vater-oder-Mutter AND
(y ist (Britischer Staatsbürger) nach-Norm p zum-Zeitpunkt t OR
y ist-niedergelassen-in UK zum-Zeitpunkt t)

Eine Aussage enthält Konstanten (z.B. UK für United Kingdom), Variablen (z.B. x für eine beliebige natürliche Person) und Prädikate (z.B. geboren-in). Die Form einer Aussage ist „Folgerung IF Voraussetzung". Die Voraussetzung besteht aus Bedingungen, die durch AND oder OR verbunden sind. Mehrere solche Aussagen bilden ein logisches Programm. Der *Folgerungsteil* entspricht in juristischer Interpretation der Rechtsfolge, der *Voraussetzungsteil* dem (generellen) Tatbestand. Durch Anwendung einer Schlußregel, zum Beispiel dem modus ponens, kann aus einer Aussage dieser Form und einem (singulären) Lebenssachverhalt ein Schluß abgeleitet werden.

Um abzuklären, ob eine bestimmte natürliche Person x britischer Staatsbürger ist, prüft das Prolog-Übersetzungsprogramm die Bedingungen der Voraussetzung. Es klärt zum Beispiel durch Befragung des Benutzers oder einer Datenbank ab, ob x im Vereinigten Königreich geboren wurde. Eine Bedingung kann auch auf eine weitere Regel des logischen Programms verweisen. Um die letzte Bedingung zu prüfen, wendet der Prolog-Übersetzer unter anderem die folgende Regel an:

y ist (Britischer Staatsbürger) nach-Norm p zum-Zeitpunkt t1 IF
x wird (Britischer Staatsbürger) nach-Norm p zum-Zeitpunkt t2 AND
t2 ist-früher-oder-gleichzeitig-wie t1 AND
not x hat-verloren (Britische Staatsbürgerschaft)

Der Übersetzer versucht, auch die Bedingungen dieser Regel über den Benutzer, eine Datenbankabfrage oder die Anwendung zusätzlicher Regeln zu beweisen. Eine Frage an ein logisches Programm (z.B. „Wird XY zum Zeitpunkt YZ britischer Staatsbürger?") beantwortet ein Prolog-Übersetzer also, indem er eine Regel sucht, deren linke Seite (Teil vor dem Wort IF) mit der Frage übereinstimmt und dann für die gefundene Regel prüft, ob deren Bedingungen zutreffen.

Ein Vergleich zwischen dem natürlichsprachlichen Wortlaut der Bestimmung aus dem Nationality Act und ihrer Abbildung in einer logischen Programmiersprache zeigt, daß die programmiersprachliche Version explizit erwähnt, was ein fachkundiger Leser implizit berücksichtigt. Die computerlesbare Fassung führt ein zusätzliches zeitliches Prädikat („zum-Zeitpunkt") ein und berücksichtigt ausdrücklich, daß jemand nach verschiedenen Rechtsnormen britischer Staatsbürger werden kann (Prädikat „nach-Norm"). Neben der expliziten Formulierung von Sachverhalten, die der menschliche Leser aus dem Textzusammenhang, mit „gesundem Menschenverstand" oder Fachwissen, ersieht, unterliegen logische Programme den folgenden *Einschränkungen:*

— Rechtsnormen, die nicht in der Prädikatenlogik erster Ordnung darstellbar sind, lassen sich in der Regel nicht in einer logischen Programmiersprache wie Prolog formulieren.
— Aus Gründen der Speicher- und Laufzeiteffizienz verwirklichen konkrete Übersetzer logischer Programmiersprachen das Ideal der logischen Programmierung oft nur unvollständig.

Nach diesen grundsätzlichen Überlegungen zur Darstellung und automatischen Interpretation von Rechtsnormen und -tatsachen in der Sprache der logischen Programmierung geht der zweite Teil dieses Abschnitts auf eine praktische Implementation ein: Peter Hammond [HAM83] hat anfangs der 80er Jahre ein einfaches Expertensystem zu einem engen Bereich des Sozialversicherungsrechts in micro-PROLOG implementiert. Das System informiert über staatliche Unterstützungsleistungen aufgrund von *DHSS Regulations* (DHSS steht für Department of Health and Social Security). Bild 7 zeigt einen (vereinfachten) Ausschnitt aus einer möglichen Interaktion zwischen dem Expertensystem und einem Beamten, der wissen will, ob ein Antragsteller x einen Anspruch auf Unterstützungsleistung hat (Angaben des Benutzers in Großbuchstaben):

```
(1) HAT x ANSPRUCH AUF UNTERSTÜTZUNGSLEISTUNG?
(2) Wie alt ist x?
    23
(3) Hat x eine Niederlassungsbewilligung?
    JA

(4) Wie groß sind die Ersparnisse von x?
    2600
(5) x hat keinen Anspruch auf Unterstützungsleistung!
    WESHALB?
(6) Die folgende Bestimmung ist nicht erfüllt:
    „x hat-Anspruch-auf unterstützung IF
       not x ist-minderjährig AND
       x ist-niederlassungsberechtigt AND

       x benötigt-finanzielle-Hilfe AND
       ...
    „not x ist-minderjährig"
       trifft zu
    „x ist-niederlassungsberechtigt"
       trifft zu

    aber „x benötigt-finanzielle-Hilfe" trifft nicht zu.
    WESHALB?
(7) „x benötigt-finanzielle-Hilfe" würde aus der folgenden Bestimmung folgen:
    „x benötigt-finanzielle-Hilfe IF
       ersparnisse-von x LESS zulässige-Ersparnisse"
    aber „ersparnisse-von x LESS zulässige-Ersparnisse" trifft nicht zu.
    WESHALB?
(8) ersparnisse-von x: 2600
    zulässige-Ersparnisse: 2500
```

Bild 7: Ausschnitt aus einer möglichen Interaktion zwischen dem DHSS-System und einem Beamten, der wissen will, ob ein Antragsteller x einen Anspruch auf Unterstützungsleistung hat (Angaben des Benutzers in Großbuchstaben. Vereinfachtes und übersetztes Beispiel aus [HAM83a] und [HAM83b]

Nach der Frage WESHALB erklärt das Expertensystem, wie es eine bestimmte Antwort abgeleitet hat. *Erklärungsmöglichkeiten* erhöhen die Akzeptanz der Antworten beim Benutzer und sind vor allem dann sinnvoll, wenn das System einen Benutzer berät, der fähig ist, abgeleitete Antworten zu relativieren und zu ergänzen. Bild 8 beschreibt gebräuchliche Erklärungsmuster:

Erklärungsmuster	Beschreibung
HOW	WIE bin ich zu einer bestimmten Antwort gekommen
WHY	WESHALB stelle ich eine bestimmte Frage an den Benutzer
WHY NOT	Weshalb bin ich NICHT zu einer bestimmten Antwort gekommen
WHAT IF	Wie WÄRE die Antwort bei andern Voraussetzungen

Bild 8: Gebräuchliche Erklärungsmuster von Expertensystemen

Die Erklärungskomponente eines Expertensystems soll Erklärungen der sich ändernden Wissensbank anpassen können. Wenn die Wissensbank zum Beispiel um Tatsachen und Regeln erweitert worden ist, müssen Erklärungen — ohne zusätzliche Programmierung — die neuen Tatsachen und Regeln berücksichtigen. Dies ist dann einfach möglich, wenn die Wissensbank Expertenwissen in einer Form darstellt, die ohne oder nach geringfügigen Transformationen vom Benutzer verstanden wird. Die in den Schritten (6)—(8) von Bild 7 erwähnten Regeln und Prädikate sind praktische un-

veränderte Bestandteile eines Programms in micro-PROLOG. Die benutzernahe und änderungsfreundliche (modulare) Wissensdarstellung gehört zu den wichtigsten Vorteilen spezieller Werkzeuge zur Entwicklung von Expertensystemen. Konventionelle Methoden und Werkzeuge erschweren die schrittweise Entwicklung und Anpassung von Wissensbanken.

Syntax und Semantik der hier vorgestellten Bestimmungen aus dem British Nationality Act und den DHSS Regulations sind vergleichsweise einfach. Beispiele komplexerer Normen aus dem amerikanischen Steuerrecht finden sich im Prolog-Programm von Schlobohm [SCH85]: Das Programm hilft bei der Interpretation von Section 318(a) des amerikanischen *Internal Revenue Code* von 1954. Die Bestimmungen definieren die Steuersubjekte von Beteiligungen am Vermögen juristischer Personen.

Schwerpunkt der in diesem Abschnitt zitierten Arbeiten ist der Nachweis, daß sich Teile des geschriebenen Rechts in der Sprache der logischen Programmierung abbilden lassen. Der nächste Abschnitt beschreibt einen Versuch, komplexe Vorgehensweisen von Versicherungsexperten bei der Beurteilung von Produkthaftungs-Fällen in einem Expertensystem abzubilden.

3.2 Phasen der Entwicklung eines komplexen Expertensystems

Die Entwicklung eines praktisch einsetzbaren Expertensystems beansprucht beträchtliche personelle, zeitliche und finanzielle *Ressourcen.* Die Entwicklungszeit früher Systeme wie MACSYMA und DENDRAL betrug über 30, jene von XCON, einem der reifsten und meistgebrauchten Systeme, etwa 8 Mannjahre (vgl. Bild 1). Obwohl inzwischen fortgeschrittenes Know How und die Verfügbarkeit von Werkzeugen die Entwicklungszeit verkürzt haben, beträgt der Aufwand für viele Systeme immer noch mehrere Mannjahre (siehe [WAT85] für eine detailliertere Erörterung des Entwicklungsaufwandes). Nachdem der letzte Abschnitt vor allem das Problem der Darstellung von Rechtsnormen in Expertensystemen angeschnitten hat, sollen die folgenden Ausführungen die Entwicklung eines komplexen Expertensystems an einem hypothetischen Beispiel veranschaulichen. Das Beispiel idealisiert die Phasen der Entwicklung eines Systems, das Versicherungsexperten bei der Beurteilung von Schadenersatzansprüchen im Bereich der *Produktehaftung* unterstützt. Es fußt auf Erfahrungen, die eine Forschungsgruppe der Rand Corporation (eines der bedeutendsten privaten Forschungsinstitute der USA) gesammelt hat ([WAT80]; [WAT84]; [WAT85] 133 f., 162 ff.).

a) Definition des Problems

Ausgangspunkt der Entwicklung eines Expertensystems könnte das folgende Problem sein:

Der Umfang der Schadenersatzforderungen an die Versicherungsgesellschaft XY hat in den letzten Jahren dramatisch zugenommen, während die Zahl der erfahrenen Schadeninspektoren zurückging. Die Gesellschaft sucht deshalb nach einer Möglichkeit, die Bearbeitung der Fälle zu standardisieren und die Inspektoren zu entlasten.

Die Definition des Projektumfangs ist für die Entwicklung von Expertensystemen entscheidend. Einerseits erfordert das Ziel der Praxistauglichkeit einen Mindestumfang, anderseits ist die Projektgröße und -komplexität durch die verfügbaren Ressourcen (Wissen, Zeit, Geld) beschränkt. Unserer Versicherungsgesellschaft bieten sich zum Beispiel folgende Kriterien zur *Eingrenzung des Projektumfangs* an:

- Haftungsgründe bzw. Versicherungsarten (z. B. Werkeigentümerhaftung)
- Schadenarten (z. B. ziffernmäßig nachgewiesener Schaden oder zu schätzender Schaden, Schadenersatz i.e.S. oder Genugtuung)
- Bearbeitungsphase (z. B. Schadenberechnungsphase oder Schlichtungsphase)

Nach Abklärung dieser Fragen durch eine Projektgruppe aus einem internen Fachexperten und einem externen Knowledge Engineer (Fachmann in der Entwicklung von Expertensystemen) beschränkt die Gesellschaft den Problembereich auf die Berechnung des Streitwertes in Fällen von Produktehaftung (Bild 9):

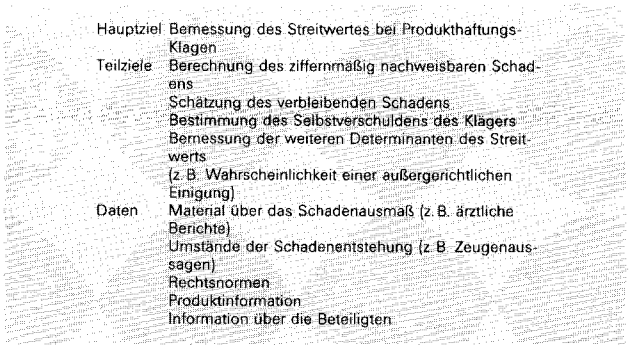

Bild 9: Hypothetische Projektbeschreibung für ein Expertensystem zur Streitwertberechnung bei Produkthaftungsklagen

b) Erwerb des Expertenwissens

Nachdem sich der Knowledge Engineer und der Versicherungsexperte in das Fachgebiet des andern eingearbeitet haben und eine erste Fassung der Projektbeschreibung vorliegt, extrahieren die beiden aus schriftlichen Quellen und den Erfahrungen des Fachexperten die Objekte und Beziehungen, die bei Streitwertberechnungen von Bedeutung sind. Eine wichtige Rolle spielen dabei Dossiers bereits abgeschlossener Fälle. Zur Veranschaulichung dieser Phase betrachten wir einen konkreten *Fall* von Produkthaftung ([WAT84], 68 ff.):

Beim Öffnen einer (geschenkten) Champagnerflasche löst sich der Korken vorzeitig, nachdem der Kläger die metallene Haltevorrichtung des Korkens teilweise geöffnet hatte. Der Korken traf ein Auge. Die Verletzung verursachte große Schmerzen und erforderte eine Operation im lokalen Krankenhaus. Nach

einer vierstündigen Operation war ungewiß, ob der Patient die Sehkraft auf dem verletzten Auge wieder erlangen würde. Das Befinden des Klägers ist inzwischen stabil: Der größte Teil seiner Sehkraft ist wieder hergestellt; die Folgen der Verletzung erfordern allerdings das ständige Tragen einer Brille. Die Wahrscheinlichkeit, später an grünem Star zu erkranken ist um 5—10% größer.

Der Geschädigte ist 30 Jahre alt und arbeitet als Sportansager bei einer lokalen Radiostation und bewirbt sich zur Zeit um eine Arbeit beim Fernsehen. Er trank selten Alkohol und öffnete zum ersten Mal eine Champagner-Flasche. Beim Öffnen der Flasche richtete er den Korken gegen sich. Die Haltevorrichtung des Korkens war angerostet. Für die Experten der Klägers lag der Grund dafür in der fehlerhaften Herstellung der Haltevorrichtung.

Der Geschädigte wird durch einen überdurchschnittlichen und erfahrenen Anwalt vertreten. Der Fall steht in 6—12 Monaten bei einem Gericht an, das den Ruf hat, bei Klagen wegen Produktehaftung den Kläger zu begünstigen.

Die Versicherungsgesellschaft versucht, aufgrund dieser (und weiterer) Informationen den Streitwert zu bemessen und mit dem Kläger zu einer außergerichtlichen Einigung zu gelangen.

Nach Durchsicht weiterer Dossiers rekonstruieren der Knowledge Engineer und der Versicherungsexperte gemeinsam den Entscheidungsprozeß zur Festsetzung des Streitwertes. Sie identifizieren dabei folgende *Entscheidungsfaktoren* (Bild 10):

Allgemein	Beispiele
1. SCHADEN	
a) ziffernmäßig nachweisbarer Schaden	Spitalkosten, Lohnausfall
b) geschätzter Schaden	Schmerzensgeld, verminderte Berufschancen
2. RECHTLICH relevante Gründe für die HERABSETZUNG des Schadenersatzes	
a) in der Person des Beklagten (v.a. Verschulden)	Herstellungsfehler bei der Haltevorrichtung
b) in der Person des Klägers (v.a. Selbstverschulden)	leichte Fahrlässigkeit beim Öffnen der Flasche
c) übrige Herabsetzungsgründe	
3. VERHANDLUNGSPOLITISCHE Gründe für die Herabsetzung des Streitwertes	
a) Eigenschaften des Klägers und seiner Vertreter	Fähigkeiten der Anwälte und Gutachter, Dringlichkeit des Benisses des Klägers nach Schadenersatz
b) Eigenschaften des verantwortlichen Gerichtes	Einstellung und Fähigkeiten der Richter

Bild 10: Entscheidungsfaktoren bei der Festsetzung des Streitwertes durch den Versicherer

Bild 10 zählt lediglich Konzepte auf, welche bei der Bemessung des Streitwertes eine Rolle spielen. Der schwierigere Teil der Streitwertbemessung besteht in der Identifikation und Quantifizierung von *Beziehungen* zwischen den Entscheidungsfaktoren und der Größe des Streitwerts, insbesondere in der Operationalisierung der Bemessungsobjekte und -verfahren. Der folgende Abschnitt über die Darstellung des erworbenen Expertenwissens zeigt an Fakten und Regeln der Wissensbank einige einfache Operationalisierungsansätze.

c) Formale Darstellung und Implementation der Wissensbank

Abschnitt 3.1 hat an Beispielen veranschaulicht, wie ein logisches Programm Expertenwissen darstellen kann. Die zitierten Arbeiten der Rand Corporation wurden hingegen mit ROSIE, einem speziell für den Bau von Expertensystemen konzipierten Werkzeug entwickelt ([BAR83], 321—326). Solche *Werkzeuge* sind oft benutzerfreundlicher als eine logische Programmiersprache wie Prolog oder andere symbolische Programmiersprachen wie Lisp und verkürzen deshalb die Entwicklungsdauer. Ihre Fähigkeit, sich an unterschiedlichste Aufgabenstellungen anzupassen ist allerdings geringer; insbesondere stützen sie sich in der Regel nicht — wie zum Beispiel Prolog — auf eine mächtige theoretische Grundlage.

ROSIE kann Expertenwissen als quasi-natürlichsprachliche Fakten und Regeln darstellen. Diese *Form der Wissenschaftsdarstellung* begünstigt insbesondere die Kommunikation mit nicht-informatikkundigen Fachexperten. Bild 11 nennt einige Beispiele von ROSIE-Statements für unser Produkthaftungs-Beispiel. Um die benutzernahe (bzw. programmiersprachen-ferne) Form der Statements zu unterstreichen, sind die Beispiele als ausführbarer ROSIE-Code und nicht in der deutschen Übersetzung dargestellt. Die quasi-natürlichsprachliche Form der Statements darf allerdings nicht darüber hinwegtäuschen, daß ROSIE weit davon entfernt ist, natürliche Sprache zu „verstehen". Die Zahl und Struktur der erlaubten Sprachmuster ist beschränkt. Muster und entsprechende Beispiele in Bild 11 sind etwa:

— ⟨Element⟩ did [not] ⟨Verb⟩ [⟨Element⟩] {⟨Präpositionselement⟩} in Statement (3)
— ⟨Element⟩ is [not] ⟨Adjektiv⟩ {⟨Präpositionselement⟩} in Statement (4)

(Winkelklammern bedeuten Variablen, eckige Klammern fakultative Wörter und geschweifte Klammern 0 oder mehr Wiederholungen, ⟨Element⟩ kann zum Beispiel eine Zahl oder eine Folge von Wörtern sein)

```
a) Einige Fakten (Tatsachen, engl. assertions)
(1) Assert the plaintiff does have (a chance of 'contracting glaucoma')
        and that chance was caused by the plaintiff's injury
        and let the value of that chance be 10%.
(2) Assert the plaintiff's injury does require (the plaintiff to wear glasses).
(3) Assert the plaintiff did not wear glasses before the injury.
(4) Assert the plaintiff's appereance is important for work.
(5) Assert glaucoma is a serious illness.
b) Einige Regeln
(6) If the plaintiff does have (a chance of 'contracting glaucoma')
        and that chance was caused by the plaintiff's injury
        and the value of that chance is greater than 5%
        and the value of that chance is less than or equal to 15%,
    increase the future trauma factor by $30000.
(7) If the plaintiff did not wear glasses before the injury
        and the plaintiff's injury does require (the plaintiff to wear glasses),
        and the age (of the plaintiff) at (the time of the injury) )25
        and the plaintiff's appereance is important for work,
    increase the disfigurement factor by $5000.
```

Bild 11: Darstellung von Expertenwissen über Produkthaftung in ROSIE (nach [WAT84], 67 und [WAT85], 170f.)

d) Bewertung und Überarbeitung

Der Weg vom Demonstrations-Prototyp bis zum praktisch einsetzbaren System führt über mehrere *Tests und Verfeinerungen* der Wissensbank. Die Zahl der Regeln kann dabei von einigen Dutzend auf mehrere Hundert anwachsen (XCON (vgl. Bild 1) enthält zum Beispiel mehr als 3000 Regeln).

Nachdem der Knowledge Engineer einen Demonstrations-Prototyp mit Regeln der Art von Bild 11 entwickelt hat, füttert er ihn mit den Daten des Champagner-Falls und führt ihn dem Versicherungsexperten vor. Um das System auch auf andere Fälle anzuwenden, verallgemeinern die beiden dann die bestehenden Fakten und Regeln und fügen neue hinzu. Die Statements (5) und (6) von Bild 11 werden zum Beispiel durch die Einführung folgender Begriffe verallgemeinert: "serious illness" und "value of 'contracting the illness'" (Bild 12).

```
(5') Assert each of glaucoma, epilepsy and heart disease is a
     serious illness.
(6') If the plaintiff does have (a chance of 'contracting a serious
     illness')
        and that chance was caused by the plaintiff's injury
        and the value of that chance is greater than 5%
        and the value of that chance is less than or equal to
        15%,
     increase the future trauma factor by 30% of
        (the value of 'contracting the illness').
(8)  If the product was dangerous to a substantial number of
     people
        and the plaintiff was injured by the product
        and the product is represented by the defendant
        and (the defendant did not warn of the danger
           or the warning was not complete
           or the warning was insufficient)
        and the normal use of the product was both intended
        and foreseeable,
     Assert the product was defective for failure to warn.
```

Bild 12: Überarbeitung der Wissensbank durch Verallgemeinerung bestehender und Hinzufügen neuer Fakten und Regeln (nach [WAT85], 170f. und 123)

Durch Fortlaufendes schrittweises Testen und Verändern erreicht der Prototyp eine Größe und Komplexität, die eine völlige Überarbeitung nahelegen. Der Demonstrations-Prototyp wird zum Feld-Prototyp, der an wirklichkeitsnahen Fällen („im Feld") getestet wird. Nach dem erfolgreichen Feldtest wird das System schließlich von einer Gruppe von Programmierern in einer effizienteren Programmiersprache (als ROSIE) reimplementiert. Das Expertensystem ist dann — nach einigen Mannjahren Entwicklungsarbeit — bereit für den praktischen Einsatz.

4. Zusammenfassung

Die Abschnitte 1 und 2 haben an Beispielen aus dem Sozial- und Steuerrecht gezeigt, daß sich Sprachen der logischen Programmierung und andere Werkzeuge zur Entwicklung von Expertensystemen, gut dazu eignen, juristische Routineentscheide ohne Ermessensfreiheit zu automatisieren. Der relativ geringe Unterschied zwischen der Darstellung vieler Rechtsnormen in der juristischen Fachsprache und ihrer Abbildung in eine computerlesbare Form erleichtert nicht nur die Kommunikation zwischen dem Entwickler bzw. Verwalter eines Expertensystems und dem juristischen Fachexperten, sondern ermöglicht auch die Entwicklung komfortabler Erklärungsmöglichkeiten für den Endbenutzer.

Logische Programmiersprachen und Expert System Tools sind allerdings Werkzeuge, die erst seit kurzem Eingang in die Praxis finden. Die erwähnten juristischen Expertensysteme sind denn auch im wesentlichen noch nicht über das Stadium von Forschungsprototypen gediehen. Viele juristische Entscheidungen lassen sich außerdem nur *schwer oder überhaupt nicht automatisieren.* Ein Rechtssystem ist — zum Teil gewollt — lückenhaft und verweist ausdrücklich auf das Ermessen des Richters. Beispiele aus dem schweizerischen Recht sind ZGB 1 Abs. 2, OR 43/44 und OR 99 Abs. 3 (Herabsetzungsgründe bei der Bemessung des Schadenersatzes, vgl. Bild 10). Rechtsbestimmungen verweisen zudem oft auf außerrechtliche Normen. So erwähnen OR 19 etwa die „guten Sitten" und ZGB 2 „Treu und Glauben". Eine weitere Schwierigkeit sind — neben unbestimmten — schlecht und uneinheitlich definierte Rechtsbegriffe. Die Diskussion um die Abbildung einer Bestimmung aus dem British Nationality Act in Abschnitt 3.1 zeigt, daß ungenau und uneinheitlich definierten Begriffen nur durch explizite Zusatzinformation in der Wissensbank (z.B. Ausnahmekataloge) begegnet werden kann. Abschnitt 3.2 veranschaulicht am Beispiel der Schadenersatzbemessung die Operationalisierung unbestimmter außerrechtlicher Begriffe. Dort wo allerdings das Gesetz auf das Ermessen des Richters verweist, wo zum Beispiel der Richter „nach der Regel entscheiden [soll], die er als Gesetzgeber aufstellen würde" (ZGB 1 Abs. 2), ist eine Automatisierung undenkbar. Es wäre verhängnisvoll, „wenn künftig an die Stelle der in unserem Recht üblichen breiten Generalklauseln (‚nach den Umständen angemessen', ‚aus wichtigen Gründen' ...) eine enge und fein verästelte Kasuistik träte, die Erwägungen des Richters im Einzelfall überflüssig machen würde" ([FOR84], 12).

Glossar

ALGORITHMUS
siehe Heuristik

DEBUGGING
Nachdem in der Testphase geeignete Testfälle Programmfehler (engl. "bugs") identifiziert haben, werden die Fehler in der anschließenden Phase des Debugging repariert.

FRAME
Neben Fakten und Regeln verbreitetste Form der Darstellung von Expertenwissen. Ein Frame ist ein leeres Schema, das je nach Bedarf mit wechselnden Eigenschaftswerten und Anweisungen gefüllt werden kann.

HEURISTIK
Erfahrungsregel, welche die Zahl der möglichen Lösungen eines schlecht strukturierten Problems zwar einschränkt, aber nicht (wie ein Algorithmus) eine genau definierte Lösungsprozedur angibt

LOGISCHE PROGRAMMIERUNG
Im Gegensatz zu konventionellen prozeduralen Programmen schreiben logische Programme nicht vor, WIE der Computer eine bestimmte Menge hardware-naher Operationen einsetzen soll, um ein bestimmtes Problem zu lösen. Logische Programme beschreiben stattdessen eine Aufgabe in der Sprache

der formalen Logik. Ein Übersetzer, der ausgewählte Schlußverfahren der Logik kennt, leitet automatisch Lösungen ab, die mit dieser Beschreibung verträglich sind. Prolog ist eine erste Annäherung an das Ideal einer logischen Programmiersprache und erlaubt die Beschreibung von Aufgaben, die sich in der Prädikatenlogik erster Ordnung darstellen lassen.

PRÄDIKATENLOGIK ERSTER ORDNUNG
Teilsprache der formalen Logik, die Beziehungen zwischen Objekten mit Prädikaten (Namen für Eigenschaften und Beziehungen) und quantifizierenden Redeteilen (z.B. „alle") beschreibt.

PRODUKTIONSREGEL
siehe Regel

REGEL
Wissensdarstellung in der Form „WENN Voraussetzung DANN Folgerung". Regeln können rückwärts oder vorwärts verkettet werden.

RÜCKWÄRTSVERKETTUNG
Ableitungsmethode, bei der ein Expertensystem von der Folgerung ausgeht, die es beweisen soll. Soll zum Beispiel in der Aussage „C WENN A UND B" C bewiesen werden, dann versucht das System alle Regeln anzuwenden nnd alle Fakten zu sammeln (zum Beispiel vom Benutzer zu erfragen), welche die Voraussetzungen A und B beweisen. Gelingt es, A und B zu bestätigen, dann gilt auch C.

VORWÄRTSVERKETTUNG
Ableitungsmethode, bei der ein Expertensystem von einer Menge von Bedingungen ausgeht und alle Regeln anwendet, deren Voraussetzungsteil aus Teilmengen dieser Bedingungen besteht.

Literatur

[BAR83] *Barstow, D. R.* (et al.) (1983) Languages and tools for knowledge engineering, in [HAY83], 283—345

[BAU84] *Bauknecht, K., Forstmoser, P., Zehnder, C. A.* (eds.) (1984) Rechtsinformatik. Bedürfnisse und Möglichkeiten, Zürich, Schulthess

[BIN80] *Bing, J.* (1980) Legal Norms, discretionary rules and computer programs, in Niblett, B. (ed.) Computer science and law, Cambridge, England, 119—146

[BUC84] *Buchanan, B. G., Shortliffe, E. H.* (eds.) (1984) Rule-based expert systems. The MICIN experiments of the Stanford Heuristic Programming Project, Reading, Massachusetts, Addison-Wesley

[CLA84] *Clark, K. L., McCabe, F. G.* (1984) micro-PROLOG: Programming in logic, Englewood Cliffs, New Jersey, Prentice-Hall

[CIA82] *Ciampi, C.* (ed.) (1982) Artificial Intelligence and legal information systems, New York, North Holland

[CIA84] *Ciampi, C.* (1984) Evolution of systems and research using data processing in jurisprudence over the past 25 years, *C. Ciampi,* Inf. & Diritto, 10, 2, 81—222

[COO80] *Cook, S., Stamper, R.* (1980) LEGOL as a tool for the study of bureaucracy, in Lucas, H. (ed.) (1980) The information systems environment, Amsterdam, New Holland

[COR84] *Cory, H. T., Hammond, P., Kowalski, R. A., Kriwaczek, F., Sadri, F., Sergot, M. J.* (1984) The Britisch Nationality Act as a logic program, Logic Programming Research Reports, Dept. of Computing, Imperial College of Science and Technology, London, England

[CLO84] *Clocksin, W. F., Mellish, C. S.* (1984) Programming in Prolog, 2nd ed., Berlin, Springer

[FOH85] *Fohmann, L.* (1985) Die informationale Programmiersprache IPL. Studie zur Operationalisierung regelorientierter schlechtdefinierter nichtschematischer Entscheidungen, GMD Bericht 147, München, Oldenbourg

[FOR84] *Forstmoser, P.* (1984) Rechtsinformatik: Einführung aus juristischer Sicht, in [BAU84], 3—12

[HAM83a] *Hammond, P.* (1983) APES: A user manual, Dept. of Computing Report 82/9, Imperial college of Science and Technology, London

[HAM83b] *Hammond, P.* (1983) Representation of DHSS regulations as a logic program, Dept. of Computing Report 82/26, Imperial College of Science and Technology, London

[HAM84] *Hammond, P.* (1984) miero-PROLOG for expert systems, in [CLA84], 294—319

[HAY83] *Hayes-Roth, F., Waterman, D. A., Lenat, D. B.* (eds.) (1983) Building Expert Systems, Reading, Massachusetts, Addison-Wesley

[KEM84] *Kemper, M., Koitz, R.* (1984) Strukturtheoretische Reflexionen zur computergestützten Anspruchsermittlung (auf der Grundlage des Programmsystems DIALEX), Datenverarbeitung im Recht, 13, 3, 217—250

[KOW85] *Kowalski, R.* (1985) Logic programming, BYTE, 8/85, 161—177

[LLO84] *Lloyd, J. W.* (1984) Foundations of logic programming, Berlin, Springer

[LUS85] *Lusti, M.* (1985) PROLOG — Konzept und Anwendung einer ungewöhnlichen Programmiersprache, LOG IN, 5, 5/6, 40—44

[MAR84] *Martins, G. R.* (1985) The overselling of expert systems, Datamation, International Edition, 30, 18, 76—80

[MEL75] *Meldman, J. A.* (1975) A preliminary study in computer-aided legal analysis. PhD thesis, Department of Electrical Engineering and Computer Science, MIT

[MEL84] *van Melle, W., Shortliffe, E. H., Buchanan, B. G.* (1984) EMYCIN: A knowledge engineer's tool for constructing rule-based expert systems, in [BUC84], 302—313

[MIC82] *Michaelson, R. H.* (1982) A knowledge-based system for individual income and transfer tax planning, Ph.D. diss., University of Illinois

[MIC84] *Michaelson, R. H.* (1984) An expert system for federal tax planning, Expert Systems, 1, 2, 149—167

[MIC85] *Michaelson, R. H., Michie, D., Boulanger, A.* (1985) The technology of expert systems, BYTE 4/85, 303—312

[NOE85] *Noelke, U.* (1985) Das Wesen des Knowledge Engineering, in [SAV85], 109—123

[RAU84] *Rault, J. C.* (1984) Les systemes experts: perspectives industrielles, Bulletin de liaison de la recherche en informatique et automatique, no. 97, 9—23

[SAV85] *Savory, S. E.* (ed.) (1985) Künstliche Intelligenz und Expertensysteme. Ein Forschungsbericht der Nixdorf AG, 2., erg. Aufl., München, Oldenbourg

[SHA85] *Sharpe, W. P.* (1985) Logic programming for the law, in *Hammersley, P.* (ed.) Research and development in expert systems, Proceedings of the fourth technical conference of the British Computer Society specialist group on expert systems, University of Warwick, Dec. 1984

[SCH85] *Schlobohm, D.* (1985) Tax Advisor — A Prolog program analyzing income tax issues, Dr. Dobb's Journal, March 1985

[SER82] *Sergot, M. J.* (1982) Prospects for representing the law as logic programs, in *Clark, K. L., Taernlund, S. A.* (eds.) (1977) Logic Programming, London, Academic Press

[WAT80] *Waterman, D. A., Peterson, M. A.* (1980) Rule-based models of legal expertise, Proceedings of the First Annual National Conference on Artificial Intelligence, 1980

[WAT84] *Waterman, D. A., Peterson, M. A.* (1984) Evaluating eivil claims: an expert system approach, Expert Systems, 1, 1, 65—76

[WAT85] *Waterman, D. A.* (1985) A guide to expert systems, Reading, Massachusetts, Addison-Wesley