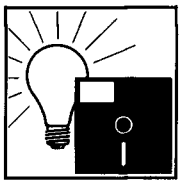


Juristische Programme der „zweiten Generation"? Gedanken zur Programmierung wissensbasierter Systeme in Turbo Prolog 2.0 (Teil 2)

Jürgen Oechsler

Wissensbasis

Die Konstruktion der eigentlichen Wissensbasis birgt vergleichbare Probleme. So wäre es nicht damit getan, das Beispielprogramm einfach um Regeln für alle Vertragsarten und alle Einwendungen anzureichern. Der Regelapparat würde unüberschaubar groß, die Suchwege für Backtracking endlos lang.

Das Grundproblem: Die Wissensrepräsentation

Von daher liegt eines der Grundprobleme moderner Expertensysteme[12] in der Repräsentation von Wissen.

Kennzeichnend für juristisches Wissen: Hohe Verschachtelungstiefe

Die Repräsentation juristischen Wissens wirft zudem spezifische Probleme auf. Juristisches Wissen ist durch eine hohe Verschachtelungstiefe gekennzeichnet: Viele Regeln werden erst dann relevant, wenn andere bejaht oder verneint wurden, die ihrerseits weiterverweisen. Stünden alle Regeln gleichberechtigt nebeneinander, würden sie vom Suchlauf des Backtracking im Rahmen jeder Teilprüfung neu durchsucht. Angesichts der damit verbundenen Wartezeiten könnte der Anwender über der maschinellen Subsumtion rasch graue Haare bekommen. Das ideale System sollte daher beispielsweise der Frage, ob der Käufer alle Kaufpreisraten mit seinem Taschengeld getilgt hat, nur dann nachgehen, wenn feststeht, daß es sich einerseits beim Käufer um einen beschränkt Geschäftsfähigen handelt, der einen Abzahlungskaufvertrag geschlossen hat, und daß andererseits der Vertrag nicht bereits aus anderen Gründen unwirksam ist (§ 110 BGB).

Ein weiteres Problem: Die „Vernetzung" der Rechtssätze

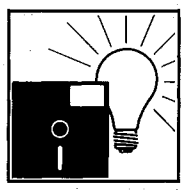
Die Problematik der Repräsentation juristischen Wissens verschärft sich durch den Umstand, daß zahlreiche juristische Regeln auf anderen aufbauen. Der eben erwähnte Rechtssatz zu § 110 BGB nimmt Bezug auf andere Rechtssätze, die sich über den Abzahlungskauf, die Erfüllung eines Vertrages usw. verhalten. Wollte man in den Regelkörper eigene Prädikate über den Abzahlungskauf und seine Erfüllung integrieren, müßte man beim Abfassen jeder Regel von vorne beginnen, das Rad also stets neu erfinden, ganz abgesehen davon, daß die Wissensbasis unnötig aufgebläht würde.

Juristische Expertensysteme müssen „begründungstransparent" sein

Ein juristisches Expertensystem sollte dem Anwender idealerweise zeigen, wie es zu einer Schlußfolgerung gelangt: Dies geschieht üblicherweise dadurch, daß dem Anwender die angewandten Regeln in ihrer Prüfungsreihenfolge aufgezeigt werden. Die bloße Angabe eines Rechtssatzes ist für juristische Anforderungen dabei eher dürftig, erlaubt sie doch nicht die Überprüfung seiner Richtigkeit. Der Jurist erwartet stets die Rechtsgrundlage einer Regel, ihre Begründung, soweit es sich nicht gerade um eine Wiederholung des Gesetzeswortlautes handelt, sowie weiterführende Belegstellen aus Rechtsprechung und Literatur. Ohne diese Bestandteile wird das Expertensystem zur Glaubensangelegenheit. Das Prologsystem bedarf ihrer zur Abarbeitung zwar nicht; trotzdem muß es sie als zusätzliche Last mit verwalten.

Angesichts dieser Problemstellung liegt wohl eine der Herausforderungen an die Rechtsinformatik darin, für juristische Bedürfnisse geeignete Datentypen zu finden. Doch gibt

12 Als Expertensystem bezeichnet man ein wissensbasiertes System, in dem das Wissen eines Experten so gespeichert ist, daß es einem Anwender auf Anfrage zur Verfügung steht. Kinnebrock (o. Fn. 2), S. 125); Schildt (o. Fn. 2), S. 49; Weiskamp/Hengl (o. Fn. 2), S. 209 ff. sowie vor allem das interessante Beispiel in Hashim/Seyer, Turbo Prolog – Advanced Programming Techniques, S. 3 ff.



Gefragt: Dem juristischen Denken angemessene Datentypen

Ein mögliches Werkzeug: Frames

es bereits heute Strukturen, die in die Zukunft weisen. Genannt seien hier die vor allem von Erweiterungen der Sprache Lisp her bekannten Frames[13], die sich auch in Prolog modellieren lassen[14]. Frames (englisch für „Rahmen") sind abgeschlossene, kleine, in sich untergliederte Welten. In diesen Untergliederungen, Slots genannt, speichern sie ihr Wissen. Frames können auf weitere Frames (Söhne) verweisen und diese in den Lösungsprozeß mit einbeziehen. Dadurch wird es beispielsweise möglich, daß eine Regel aus dem thematischen Bereich des § 110 BGB eine weitere Regel über den Abzahlungskauf, die in einer anderen Datenbank gespeichert ist, aufruft. Damit erübrigt es sich im Bereich des § 110 BGB, eigene Routinen über den Abzahlungskauf zu schreiben.

Durch die gegenseitigen Verweisungen entstehen wiederum Netze, die das Prologsystem zielstrebig auf dem Lösungspfad weiterleiten können. Die erwähnte Regel zu § 110 BGB kann dann nur noch über einen „Vater"-Frame „Taschengeld" aufgesucht werden, der seinerseits nur über den Frame „Einwilligung des gesetzlichen Vertreters" erreicht werden kann, welcher seinerseits ein „Kind" eines weiteren Frames ist, der innerhalb des thematischen Bereiches der §§ 108 ff. BGB angesiedelt werden muß. Diese hierarchische Ordnung des Wissens vermeidet unnötige Prüfungen bzw. Suchen und entspricht der zielgerichteten juristischen Fallprüfung: Probleme werden nur diskutiert, wenn sie auf dem Weg zur Lösung von Interesse sind. Die Abarbeitung von Regeln gerät nicht mehr zu einem wahllosen Herumstochern in einem Berg von Informationen, wie dies im Beispielsprogramm dem Grunde nach angelegt ist.

Mit Frames modellierbar: Eine hierarchische Ordnung des Wissens

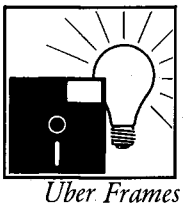
Schließlich könnte ein solcher Frame auch eigene Slots für Rechtsgrundlage, Begründung und Belegstellen beinhalten, die stets von der eigentlichen Regel aus erreichbar wären, so daß auch diese Probleme einer Lösung zugeführt werden könnten.

```
FRAME

Slot 1: Rechtssatz
Vereinbart ein beschränkt Geschäftsfähiger einen Abzahlungskauf
und erfüllt er die Kaufpreisraten mit seinem Taschengeld, bleibt
der Kaufvertrag solange schwebend unwirksam, bis alle Raten mit
dem Taschengeld getilgt sind.

Slot 2: „Vater"-Frames
Frame: Taschengeld

Slot 3: Verweise
Zum Abzahlungskauf befrage die Datenbank „Abzahlungsgeschäfte".

Zur Erfüllung befrage die Datenbank „Einwendungen" und dort
den Frame „Erfüllung".

Slot 4: Rechtsgrundlage
§ 110 BGB

Slot 5: Argumentationsstand
Der Wortlaut des § 110 BGB „bewirkt" legt es nach natürlichem
Sprachempfinden nahe, daß die Leistung durch den beschränkt
Geschäftsfähigen ganz erbracht werden muß.

Die besondere Gefährlichkeit der Abzahlungsgeschäfte

Slot 6: Belege und weiterführende Literatur
Palandt ...
```

Turbo Prolog unterstützt mit einem eigenen Datentyp, der Liste, die Anhäufung umfangreicher und heterogener Informationen innerhalb eines Slots: Die Liste ist eine Aneinanderreihung von Elementen, innerhalb derer gesucht, eingefügt und gelöscht werden kann.

13 Siehe etwa die Kurzvorstellung von PC-Scheme bei Roland Fischer, Objektorientierte Erweiterungen – Modegag oder neuer Standard?, Toolbox 89, Heft 10, S. 24 ffi (28 ff) sowie Winston/Horn (o. Fn. 2), S. 351 ffi, die allerdings objektorientierte Programmierung in Common Lisp behandeln.

14 Weiskamp/Hengl (o. Fn. 2), S. 187.



Über Frames zur Objektorientierung

Charakteristika des Frame-Konzeptes, wie die gegenseitigen Beziehungen der in sich abgeschlossenen kleinen Datenwelten, die Übernahme von Wissen usw., weisen in Richtung des objektorientierten Paradigmas mit seinen Forderungen nach Klassifizierung von Objekten, Einkapselung und Vererbung usf.[15].

Änderungen im Wissensbestand

Wie wird man mit Veränderungen der Wissensbasis fertig?

Unabhängig von der Entscheidung für einen Datentyp stellt sich ein weiteres prologinternes Grundlagenproblem wissensbasierter Systeme: Es wurzelt im Bedürfnis nach ständiger Aktualisierung und damit Veränderung der Wissensbasis. Der Wert eines Programms, das Fälle aufgrund eines überholten Meinungsstandes löst, bedarf keiner weiteren Erörterung.

Ein Weg: Dynamische Datenbanken

Neue Regeln müssen daher hinzugefügt und alte entnommen werden können. Prädikate aber, die andere Prädikate verändern, bezeichnet man gemeinhin als Metaprädikate, da sie sozusagen über der Stufe des gewöhnlichen Prädikates angesiedelt sind, andere Prädikate damit gerade auf die Stufe von Objekten „herabsetzen"[16]. Turbo Prolog 2.0 unterstützt eine dynamische Veränderung der Regelbasis zur Laufzeit durch Metaprädikate nicht. Das im Lieferumfang von Turbo Prolog 2.0 enthaltene Miniaturmodell eines Expertensystems geht einen anderen Weg: Es verwendet dynamische Datenbanken. In diesen liegt das Wissen nicht mehr in Prädikatform vor, sondern als gewöhnlicher Datensatz, der von anderen Prädikaten aus aufgerufen werden kann. Dafür spricht zunächst die einfache Handhabung. Im übrigen läßt sich auf diese Weise der allen Metaprädikaten gemeinsame „Hunger" nach Speicherplatz begrenzen. Schließlich kennen moderne Programmierparadigmen wie der objektorientierte Ansatz keine strenge Trennung zwischen Daten und Regeln.

Dagegen besteht jedoch die Gefahr der fehlenden Flexibilität: Daten die von Prädikaten aus aufgerufen werden, sind auf den Aufbautypus der Aufrufprädikate festgelegt: Sie müssen aus Feldern bestehen, die das aufrufende Prädikat verarbeiten kann. Muß die Struktur der Wissensrepräsentation geändert werden, wünscht also der Anwender beispielsweise einen zusätzlichen Slot, der Tarifverträge aufnehmen soll, so läßt sich dies wohl nur über eine Änderung der Regelbasis lösen.

Im Lieferumfang von Turbo Prolog 2.0 befindet sich der Quelltext eines vollständigen Prolog-Interpreters, der selbst in Turbo Prolog 2.0 geschrieben ist und die wichtigsten Metaprädikate beherrscht[17]. In der weiterführenden Literatur finden sich weitere interessante Ansätze[18].

Die Programmierwerkzeuge

Mit edit() zum eigenen Editor

Im übrigen läßt das System Turbo Prolog 2.0 den Programmierer in keiner Weise im Stich. Im Gegenteil ist der Lieferumfang so umfassend wie in keiner anderen Borland-Sprache. Dies beginnt bereits bei einem ausgereiften Window-Konzept, das sowohl ein Scrollen innerhalb der Fenster gestattet wie nachträgliche Größenänderungen. Als Beispielprogramm findet sich ein einfacher „Pop Up"-Menu-Generator, der den Programmierer für's erste von den Kümmernissen des gotoxy() befreit. Mit dem Prädikat edit() läßt sich gar der Editor in eigene Anwendungen integrieren, ein geradezu futuristisches Feature, das kommende objektorientierte Systeme anzukündigen scheint, die es dem Programmierer gestatten werden, Tabellenkalkulationen und Textverarbeitungsprogramme aus eigenen Programmen heraus anzusprechen.

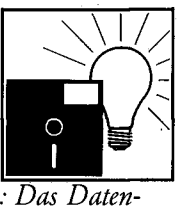
---

15 Siehe dazu etwa Josef Mittendorfer, Smalltalk – Hintergründe der objektorientierten Programmierung, mc 89, Heft 3, S. 54 ff; Klaus Zerbe, Vorbild Smalltalk – Objektorientierte Programmierung vor dem Durchbruch, c't 89, Heft 4, S. 120 ff sowie die in allen Computerzeitschriften erschienen Reviews zu den objektorientierten Dialekten Turbo Pascal 5.5, Quick Pascal und Zortech C++.

16 Vgl. Hashim/Seyer (o. Fn. 11) S. 3 ff.

17 Turbo Prolog 2.0, Referenz Anhang K.

18 Hier sei vor allem das Experiment von Hashim/Seyer zur Lektüre empfohlen (o. Fn. 11) S. 3 ff sowie die grundlegende Arbeit von Gustav Neumann: Metaprogrammierung und Prolog, 1988.



Besonders ausgereift: Das Datenbankkonzept

Besonders ausgereift ist das Datenbankkonzept von Turbo Prolog 2.0, wohl als Ausgleich für die fehlenden Meta-Prädikate. Dem Programmierer stehen gleich zwei Systeme zur Verfügung. Eine interne Datenbank ist einfach zu handhaben, wird aber insgesamt ins Ram geladen und findet ihre Grenzen in der Gesamtgröße des adressierbaren Hauptspeichers. Komplizierter in der Handhabung, dafür aber universaler ist die externe Datenbank, die sich sowohl im Ram, als auch auf Diskette und im Expanded Memory ablegen läßt. Die Datensuche wird durch den schnellen „B+ Tree"-Algorithmus unterstützt. Der professionelle Zuschnitt wird etwa an Prädikaten deutlich, die die Programmierung von Schutzvorrichtungen für Stromausfall erlauben. Geradezu eine Provokation für C- und Pascal-Programmierer ist das vordefinierte Datenbankprädikat db_garbagecollect(Datenbank), das - so die Referenz - freien Speicherplatz in einer Datenbank zu größeren Stücken zusammenfügt. Wer sich in C oder Pascal einmal an einer Heap-Verwaltung versucht hat, wird spätestens hier hellhörig.

Vor Tippfehlern im Handbuch sei gewarnt

Das Handbuch bietet eine vollständige Einführung in Prolog, ist aber leider mit Tippfehlern durchsetzt: Dem Verfasser stellten sich erhebliche Selbstzweifel, als er daran scheiterte, das einleitende „Hallo Welt"-Programm zum Laufen zu bringen: Auch hier hatten sich Fehler eingeschlichen. Das Abtippen bleibt dem Anfänger aber weitgehend erspart. Die meisten Beispiele und Übungsaufgaben werden im Quelltext mitgeliefert. Hervorhebung verdienen die Beispielprogramme: Neben den obligatorischen Türmen von Hanoi findet der Anfänger einen kleinen Parser, eine natürlich-sprachliche Datenbank sowie das bereits erwähnte Expertensystem. Dieses vermag Tiere zu bestimmen, allerdings auf dem Niveau einer Frage wie: Der Name welches Tieres beginnt mit „P" und endet mit „inguin"? Die Miniatur zeigt aber eindrucksvoll das Zusammenspiel der einzelnen äußeren Komponenten eines Expertensystems und lädt zum Ausbau ein.

Nach Erfahrungen in C und Pascal ist die Programmierung in Prolog sehr direkt. Die Lösung der interessierenden logischen Probleme kann meist unmittelbar angegangen werden. Es ist nicht nötig, über endlose Seiten einen Prozedurapparat zu schreiben. Allerdings hat die Programmierung ihre Tücken: Iterationen in Prolog können meist nur rekursiv durchgeführt werden.

```
for  ( i_anfang = 0; i <= i_ende; i++) { printf("%d",i); i++; }
```

„heißt" in Prolog:

```
for (I_anfang,I_ende)
    if I_anfang <= I_ende
     and writef("%d",i)
     and I_zwischen = I_anfang + 1
     and for (I_zwischen,I_ende).
```

Schluß

Zum Schluß ein wenig Trost

Eine Erörterung, die fast ausschließlich Probleme aufwirft, ohne Lösungen anzubieten, sollte wenigstens zum Abschluß Trost spenden: Die Grundlagen zu einer Subsumtionsmaschine sind gelegt, vieles läßt sich programmtechnisch realisieren. Die Hindernisse sind jedoch noch breit gestreut: Zum großen Teil fehlt es an Grundlagenforschung, aber auch an Kapazitäten. Jedes juristische wissensbasierte System muß einen ganzen Berg an Detailwissen „erlernen". Die Erstellung umfassender Wissensbasen ist daher kaum eine Aufgabe für Hobbyisten, sondern eine kostenintensive Investition; ähnliches gilt für semantische Netze. Andererseits liegen die für den Programmierer interessanten Bereiche nicht im Abfassen einer Wissensbasis, sondern in den diese umlagernden Problemen der Wissensrepräsentation und der dynamischen Wissensverwaltung. Lösungen müssen sich vor allem an der juristischen Denkweise orientieren. Dadurch gewinnt das Wissen des Juristen unter Umständen stärker Bedeutung als die Programmierkenntnisse. Vor diesem Hintergrund kann dem interessierten Juristen der Einstieg in Turbo Prolog 2.0 empfohlen werden.