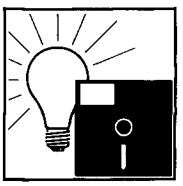


Juristische Programme der „zweiten Generation"? Gedanken zur Programmierung wissensbasierter Systeme in TurboProlog 2.0 (Teil 1)

Jürgen Oechsler

Überblick

Welchem Anwender juristischer Software kam nicht einmal der Traum, seinen Computer mit einer jener nervenaufreibenden, undurchschaubaren Akten zu füttern und einfach abzuwarten, bis der Drucker unter sägendem Lärm die Vollendung eines Kurzgutachtens ankündigt. Vergegenwärtigt man sich das zur Zeit verfügbare Angebot an juristischer Software, scheint es dagegen eher so, als würden derart goldene Eier für absehbare Zeit nur in den Tagträumen geplagter Rechtsanwender gelegt werden. So bleibt bei aller Euphorie über anfängliche Arbeitserleichterungen die ernüchternde Gewißheit zurück: Mag der Computer auch lästige Such- und Schreibarbeit vereinfachen, eine Datenbank mit Mandanten verwalten oder mit eiserner Strenge Terminkalender führen, sein Wirkungskreis bleibt stets auf den technischen Ablauf juristischer Arbeit beschränkt, zu inhaltlichen Fragestellungen vermag er keinen Beitrag zu leisten. „Computer Aided Subsumtion"(!) – so zeigt dieser erste Befund – erfordert einen neuen Zugang zu den Bedürfnissen des Anwenders und damit auch neue Denkweisen bei der Erstellung von Programmen. Nichts liegt daher näher, als sich mit alternativen Programmierkonzepten vertraut zu machen. Eine Sprache wie TURBO PROLOG 2.0 fordert dabei jeden heraus, der bereit ist, juristische Sprech- und Denkweise vor neuen Fragestellungen zu analysieren. So ist PROLOG, dies macht der Name bereits deutlich, ein Werkzeug zur PROgrammierung LOGischer Probleme; mit logischen Fragestellungen ist aber auch der Rechtsanwender täglich konfrontiert, nämlich – mirabile dictu – im Rahmen juristischer Subsumtion.

Neue Denkweisen sind gefragt

Subsumtion in Prolog

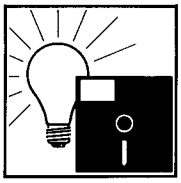
Unabhängig von Bewertungen und Interessenabwägungen bei der Lösung eines Einzelfalles, läßt sich die juristische Subsumtionsleistung zunächst als einfacher Schlußfolgerungsmechanismus charakterisieren. Schlüsse lassen sich aufgrund sehr unterschiedlicher Strategien ziehen. Dem kriminalistisch interessierten Leser sind vielleicht die Methoden von Sherlock Holmes gegenwärtig, dem es bekanntlich gelang, durch gründliche Untersuchung eines Spazierstocks auf die Identität eines nächtlichen Besuchers zu schließen[1]. Der logische Schluß setzt hier eine umfangreiche Aufklärung aller verfügbaren Elemente des Sachverhaltes voraus. Diese Strategie wird Vorwärtsverkettung[2] genannt: Die Reihe der aufeinander aufbauenden logischen Glieder führt von der Tatsachenebene aus auf die Wertungsebene, innerhalb derer, erst als letzter Schritt, ein bestimmtes Ergebnis gefunden wird. Vor Aufklärung des Sachverhaltes ist keine Bestimmung der Zielrichtung möglich, in die Schlüsse gezogen werden könnten. Eine frühzeitige Festlegung ist sogar schädlich, weil sie unter Umständen zur Nichtberücksichtigung wichtiger Sachverhaltselemente führen könnte. Diese Strategie ist daher stets dann hilfreich, wenn die Anzahl möglicher Schlußfolgerungen zu Beginn des Prozesses praktisch unbegrenzt ist. In dieser Lage befindet sich der Jurist jedoch nicht. Das Gesetz kennt nur eine begrenzte Zahl von Rechtsfolgen, die an ganz bestimmte Tatbestandsvoraussetzungen anknüpfen. Entsprechend sind auch nicht alle Umstände eines Lebenssachverhaltes wichtig. Ein entscheidender Teil der juristischen Arbeit liegt gerade darin, die wesentlichen Elemente des Sachverhaltes zu selektieren. Der Jurist arbeitet

Subsumtion = einfacher Schlußfolgerungsmechanismus

Eine Strategie: Vorwärtsverkettung

1 Zur weiterführenden Vertiefung sei dem interessierten Leser hier „The Hound of the Baskervilles" von Arthur C. Doyle empfohlen.

2 Siehe nähere Darstellung bei Kinnebrock, Professionelles Programmieren mit Turbo Prolog, 1988 S.135 ff.; Schildt, Professionelles Turbo Prolog, 1988, S.53 f.; Weiskamp/Hengl, Künstliche Intelligenz: KI-Programmierung mit Turbo Prolog, 1989, S.105 ff. Schließlich zeigt auch der Lisp-Klassiker Winston/Horn, Lisp 1987, S.305 ff. eine Implementierung in Lisp.



daher bekanntlich „von der Rechtsfolge zur Voraussetzung“: Lückenlose Aufklärung des Sachverhaltes ist wenig hilfreich, wenn ein großer Teil der aufgeklärten Sachverhaltselemente keine Rechtsfolgen zeitigen kann. Der Schlußfolgerungsmechanismus der Subsumtion beruht daher auf Rückwärtsverkettung[3]: Der Name impliziert bereits die Strategie. Begonnen wird mit einem hypothetischen Ergebnis, dessen Voraussetzungen anhand des Sachverhaltes näher zu überprüfen sind. Der Jurist wendet diese gemeinhin als Gutachtenstil bekannte Strategie meist im Rahmen seiner Vorüberlegungen an: Das hypothetisches Ergebnis leitet die Prüfung in der Form eines Obersatzes ein. Die weitere Arbeit besteht darin, die Voraussetzungen zu prüfen, an die die Richtigkeit des Obersatzes geknüpft ist.

Die Frage

„Hat Berta gegenüber Anton einen Anspruch auf Zahlung des Kaufpreises aus § 433 II BGB;“

läßt sich ohne weiteres in der Sprache Turbo Prolog 2.0 formulieren[4]:

```
anspruch_auf(kaufpreis,berta,anton,"§ 433 II BGB").
```

Ziel von Prolog: Beweis (oder Widerlegung) eines „Ziels“

Der Obersatz in Prolog heißt goal (englisch, Ziel): Seine Richtigkeit oder Unrichtigkeit zu beweisen, ist einzige Aufgabe eines Prolog-Programmes. Bemühen wir unsere Kenntnisse über die Tatbestandsvoraussetzungen des § 433 II BGB noch etwas weiter, so hängt das Schicksal des Kaufpreisanspruches davon ab, ob zwischen Anton und Berta ein Kaufvertrag geschlossen wurde und Anton keine Einwendungen bzw. Einreden gegen den aus dem Kaufvertrag entstehenden Kaufpreisanspruch hat.

Dieser Rechtssatz heißt in Prolog übersetzt:

```
anspruch_auf(kaufpreis,Verkäufer,Käufer,"§ 433 II BGB")
        if kaufvertrag(Verkäufer,Käufer,Vertrag)
        and not(einwendungen(Käufer,Vertrag)).
```

Charakteristika der rückwärtsverkettenden Schlußfolgerungsstrategie

An dieser Prologregel wird die Wirkungsweise der rückwärtsverkettenden Schlußfolgerungsstrategie deutlich: Zu Beginn steht die Rechtsfolge „anspruch_auf(....)“, die technisch auch „Regelkopf“ genannt wird. Der Eintritt der Rechtsfolge bzw. die Richtigkeit des Regelkopfes ist von zwei Tatbestandsvoraussetzungen abhängig, die den Regelkörper bilden:

„kaufvertrag(Verkäufer,Käufer,...)“

und

„not(einwendungen(Käufer,Vertrag))“.

Die zunächst befremdende Mischung englischer und deutscher Sprachelemente beruht darauf, daß „if“, „and“ und „not“ vorgegebene Schlüsselwörter des Prologsystems sind, während die übrigen Konstrukte frei vom Programmierer benannt werden dürfen.

Einige technische Gesichtspunkte

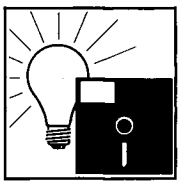
Konstruktionen wie

```
anspruch_auf(kaufpreis,berta,anton,"§ 433 II BGB")
```

und

3 Siehe zur Rückwärtsverkettung: Kinnebrock (o.Fn.2), S.129 ff.; Schildt (o.Fn.2), S.54 sowie Weiskamp/Hengl (o.Fn.2), S.102 ff.

4 Die Prologzeilen sind aus Anschaulichkeitsgründen bewußt etwas vereinfacht gewählt. So ließe sich hier durchaus zu Recht einwenden, daß etwa die Höhe des Kaufpreises mindestens in die Klausel mit einfließen müßte. Der Leser ist herzlich dazu aufgerufen, dies selbst zu perfektionieren.



```
kaufvertrag(Käufer,Verkäufer,Vertrag)
```

bilden den Grundbestandteil eines Prolog-Programmes und werden als „Prädikate" bezeichnet. Prädikate sind Ausdrücke – besser noch Funktionen – die vom Prologsystem darauf geprüft werden, ob sie wahr oder falsch sind. Der vor die Klammer gezogene Teil der Konstruktion („anspruch_auf", „kaufvertrag") verleiht dem Prädikat den Namen und repräsentiert zugleich die Beziehung zwischen den nachfolgend in Klammern eingeschlossenen Teilausdrücken; diese letzteren bezeichnet man als Argumente oder Objekte des Prädikats. Die Möglichkeit, die Beziehung einzelner Objekte zueinander in Prädikatform darstellen zu können, bezeichnet man als Prädikatenlogik[5]. Diese Darstellungsweise lehnt sich eng an die menschliche Vorstellungswelt an; denn auch menschliches Denken besteht zu einem großen Teil darin, Beziehungen zwischen verschiedenen Anschauungsobjekten herzustellen. Die Prädikatenlogik erleichtert daher bereits durch ihre Grundstruktur die Darstellung von Sprache und Wissen der Menschen.

„Prädikate" in Prolog

Das Prädikat

```
kaufvertrag(anton,berta,"Vertrag vom 27.12.1988")
```

stellt eine Beziehung zwischen Anton, Berta und dem Datum eines bestimmten Vertrages her. Gegenstand der Beziehung ist das Zustandekommen eines Kaufvertrages dieses Datums zwischen den Parteien.
Auf den ersten Blick ungereimt erscheint die unterschiedliche Groß- und Kleinschreibung der Objekte. Der Unterschied ist jedoch bedeutungstragend: Objekte, die mit Kleinbuchstaben beginnen, sind Konstanten, also in ihrem Wert unveränderlich. So soll durch das Prädikat

Nicht ungereimt: Groß- und Kleinschreibung

```
anspruch_auf(kaufpreis,Verkäufer,Käufer,"§433 II BGB")
```

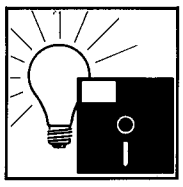
von vornherein nur ein Anspruch auf den Kaufpreis geprüft werden; andere Anspruchsziele interessieren nicht, weshalb das Objekt „Kaufpreis" mit einem Kleinbuchstaben beginnt. Objekte, die mit Großbuchstaben beginnen, sind Variablen, also Platzhalter: Diese haben zu Beginn des Programms noch keinen Wert, sondern nehmen diesen günstigstenfalls während der Programmausführung an. An die Stelle von „Verkäufer" und „Käufer" sollen im Idealfall „Berta" und „Anton" treten, wenn das Prädikat anspruch_auf(...) sich als wahr erweist, der Anspruch also besteht. Dem folgend beginnen die Objekte „Verkäufer" und „Käufer" mit Großbuchstaben. Die Anspruchsgrundlage § 433 II BGB wird schließlich in Anführungszeichen gesetzt, um vom Prologsystem als einheitliche Zeichenkette (String) verstanden zu werden, was wegen der Leerstellen innerhalb des Ausdrucks sonst nicht möglich wäre. Auch die Anspruchsgrundlage steht also von Anfang an fest und wird während des Programmablaufes nicht mehr verändert; es handelt sich ebenfalls um eine Konstante. Schließlich kann anstelle eines Objektnamens noch ein einfacher Unterstrich „_" stehen: Dieser kennzeichnet das Objekt schlicht als unbenannte Variable, die während des Programmablaufs jeden Wert annehmen kann.
Auffälliger noch als diese Details ist wohl der Umstand, daß die Namen der Prädikate und ihrer Objekte frei nach sprachlichen und logischen Vorstellungen des Programmierers gewählt werden können. Prolog tritt als deklarative Sprache mit dem Anspruch auf, dem Programmierer eine ideale Umgebung zu schaffen, innerhalb derer er sich nur noch um die Lösung der eigentlichen logischen Probleme zu bemühen braucht; die Widrigkeiten, die mit der Umsetzung des Programms in ausführbaren Code verbunden sind, sollen ihm weitestgehend vom System abgenommen werden. So verläßt die Programmierung zu keiner Zeit die begrifflich logische Welt menschlicher Vorstellung, was das Abstraktionsniveau gegenüber herkömmlichen Programmiersprachen erhöht.

Das Ideal der deklarativen Sprache

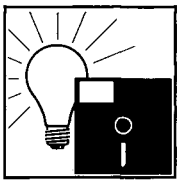
Ein Beispiel

Als minimale Demonstration der Vergleichbarkeit zwischen juristischer Subsumtion und der in Turbo Prolog unterstützten Rückwärtsverkettung mag in der weiteren Betrachtung das abgedruckte Beispiel dienen.

5 Weiskamp/Hengl (o.Fn.2), S.96.



```
/************ Beispiel für Subsumtion unter Prolog ************/
/***************** Festlegung der Datentypen! *****************/

domains

person                = symbol
leistungshandlung     = symbol
leistungsgegenstand   = symbol
anspruchsziel         = symbol
vertrag               = symbol
anspruchsgrundlage    = symbol

/***************** Festlegung der Prädikate! ******************/

predicates

anspruch_auf(anspruchsziel,person,person,anspruchsgrundlage)
vertrag_zwischen(person,person,vertrag)
kaufvertrag(person,person,vertrag)
schuldet(person,leistungshandlung,
                                leistungsgegenstand,person,vertrag)
einwendungen(person,vertrag)
erfüllung(person,vertrag)
leistet(person,leistungshandlung,
                                leistungsgegenstand,person,vertrag)

/*****  Äquivalent zum juristischen Obersatz:                  ***/
/*****  Hat Berta gegenüber Anton einen Anspruch auf Zahlung ***/
/*****  des Kaufpreises aus § 433 II BGB ?                     ***/

goal

anspruch_auf(kaufpreis,berta,anton,"§ 433 II BGB")
        and write("Der Anspruch besteht !")
or
        write("Der Anspruch besteht nicht !").

/*********** Folgende 4 Rechtssätze werden beherrscht: *********/

clauses

/***Regel Nr.1: Ein Anspruch auf Zahlung des Kaufpreises be-  **/
/***           steht, wenn zwischen den Parteien ein Kaufver-**/
/***           trag zustandegekommen und der Käufer dem An-  **/
/***           spruch des Verkäufers auf Zahlung des Kauf-   **/
/***           preises keine Einwendungen bzw. Einreden ent- **/
/***           gegenhalten kann !                            **/

anspruch_auf(kaufpreis,Verkäufer,Käufer,"§ 433 II BGB")
       if  kaufvertrag(Käufer,Verkäufer,Vertrag)
       and not(einwendungen(Käufer,Vertrag)).

/***Regel Nr.2: Bei einem Vertrag handelt es sich um einen    **/
/***           Kaufvertrag, wenn der Verkäufer Übereignung   **/
/***           der Kaufsache und der Käufer Zahlung des      **/
/***           Kaufpreises schuldet !                        **/

kaufvertrag(Käufer,Verkäufer,Vertrag)
       if  vertrag_zwischen(Käufer,Verkäufer,Vertrag)
       and schuldet(Verkäufer,übereignung,_,Käufer,Vertrag)
       and schuldet(Käufer,zahlung,_,Verkäufer,Vertrag).
```



```
/***Regel Nr.3: Der Käufer kann gegenüber dem Kaufpreisan-    **/
/***           spruch Einwendung  der Erfüllung geltend      **/
/***           machen !                                      **/
einwendungen(Käufer,Vertrag) if erfüllung(Käufer,Vertrag).

/***Regel Nr.4: Der Kaufvertrag ist erfüllt, wenn der Käufer  **/
/***           die geschuldete Zahlung erbracht hat und dies **/
/***           mit Rücksicht auf den zwischen den Parteien   **/
/***           bestehenden Kaufvertrag erfolgte !            **/

erfüllung(Käufer,Vertrag)
      if  schuldet(Käufer,zahlung,Summe,Verkäufer,Vertrag)
      and leistet(Käufer,zahlung,Summe,Verkäufer,Vertrag).

/************* Hier folgt der Lebenssachverhalt ! *************/

vertrag_zwischen(anton,berta,"Vertrag vom 27.12.1988").
schuldet(berta,übereignung,pkw,anton,"Vertrag vom 27.12.1988").
schuldet(anton,zahlung,"300,-DM",berta,"Vertrag vom 27.12.1988").
leistet(anton,zahlung,"300,-DM",berta,"Vertrag vom 28.5.1988").
```

Das Programm gliedert sich in vier Abschnitte:
- domains,
- predicates,
- goal,
- clauses.

Dabei sind „domains" und „predicates" für das Verständnis der logischen Zusammenhänge von untergeordneter Bedeutung. In „domains" werden neue Datentypen anhand der von Turbo Prolog 2.0 vorgegebenen Standarddatentypen vereinbart. Der Abschnitt „predicates" enthält die Vereinbarung aller Prädikattypen, die im Programm vorkommen werden und erfüllt dadurch eine Aufgabe, die für den Compiler[6] wichtiger ist als für den Programmierer[7]. Aus juristischer Sicht interessant wird es erst im bereits vorgestellten Abschnitt „goal", der den zu prüfenden Obersatz enthält:

```
anspruch_auf(kaufpreis,berta,anton,"§ 433 II BGB")
```

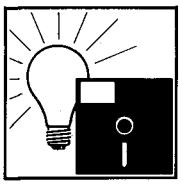
Natürlich ist damit die Programmierarbeit nicht abgeschlossen. Das Programm muß wenigstens rudimentär mit zusätzlichem Wissen ausgestattet werden. Dieses Wissen besteht aus Regeln
(Rechtssätzen) und Fakten (zu prüfenden Sachverhalten). Beides findet sich in Gestalt von Prädikaten im Abschnitt „clauses". Dieser enthält üblicherweise das Regel- und Faktenwissen eines Programms. So beginnt auch das Beispiel mit einer bescheidenen „Wissensbasis" von 4 Regeln. Regel Nr. 1 trifft Aussagen über die Tatbestandsvoraussetzungen eines Anspruches auf Zahlung eines Kaufpreises. Ein Vertrag wird durch Regel Nr. 2 anhand seiner Hauptleistungspflichten als Kaufvertrag eingeordnet. Dem Kaufpreisanspruch kann aber eine Einwendung entgegenstehen: Regel Nr. 3 kennt in gröbster Vereinfachung nur die Einwendung der Erfüllung, deren Tatbestandsvoraussetzungen sich in Regel Nr. 4 finden.

Regeln und Fakten

Im letzten Teil des Abschnittes „clauses" befindet sich der zu prüfende Sachverhalt, welcher selbsterklärend sein dürfte.
Das Programm beginnt seine Aufgabe im Abschnitt „goal". Der Auftrag lautet:

```
anspruch_auf(kaufpreis,berta,anton,"§ 433 II BGB") ...
```

6 Übersetzer in Maschinensprache.
7 Turbo Pascal-Programmierer dürfen sich zu Recht an das Interface einer Unit erinnert fühlen, während der C-Kenner den Bezug zur Funktion von Header-Dateien herstellt: Ganz verschont Turbo Prolog den Programmierer vor den Anforderungen der zu programmierenden Maschine also nicht!



„Backtracking"

Die Prüfung erfolgt nun in einem Suchlauf, Backtracking genannt, in dem alle Voraussetzungen dieses Prädikates anhand bekannter Regeln und Fakten geprüft werden. Diese befinden sich bekanntlich im Abschnitt „clauses". Die Suche beginnt dort von links nach rechts und von oben nach unten. Bereits bei Regel Nr. 1 wird das Programm fündig:

```
anspruch_auf(kaufpreis,Verkäufer,Käufer,"§ 433 II BGB") if ...
```

Prolog „subsumiert" ...

Der Kopf von Regel Nr. 1 – die Rechtsfolge – gehört demselben Prädikattyp an wie das zu prüfende Prädikat, das den Obersatz enthält. Der nächste Schritt besteht nun darin, die genaue Übereinstimmung beider Regelköpfe festzustellen, weil sich nur so ermitteln läßt, ob der Regelkörper für die Fallösung von Interesse ist. Prolog „subsumiert" daher den geprüften Obersatz unter die allgemeine Regel Nr. 1; dieser Prozeß wird als Unifikation bezeichnet:

```
anspruch_auf (kaufpreis,Verkäufer,Käufer,"§ 433 II BGB")...
                    |          |        |          |
anspruch_auf (kaufpreis, berta,     anton,"§ 433 II BGB")...
```

Übereinstimmung findet sich bereits zwischen den jeweils ersten und vierten Objekten der Prädikate: „kaufpreis" und „§ 433 II BGB". Lediglich die Objekte „Verkäufer" und „Käufer" sowie „berta" und „anton" scheinen einander beziehungslos gegenüberzustehen. Dieser Eindruck täuscht hingegen: Die Objekte „Verkäufer" und „Käufer" im Prädikat der Regel Nr. 1 sind Variablen, also zunächst inhaltslose Platzhalter, die erst einen Wert annehmen müssen. Die Variableneigenschaft – dies sei der Deutlichkeit halber noch einmal wiederholt – ist dabei an der Großschreibung ablesbar. Als juristisches Äquivalent der Variable könnte man den abstrakten Oberbegriff ansehen, der stets durch einen Sachverhalt konkretisiert, mit Leben erfüllt werden muß.
Die abstrakten Begriffe des Regelkopfes „Verkäufer" und „Käufer" nehmen die Werte „berta" und „anton" an, da im übrigen zwischen beiden Prädikaten Übereinstimmung besteht. Diese Art der Subsumtion wird – wie bereits erwähnt – als Unifikation oder

„Unifikation" in Prolog

auch Bindung bezeichnet, weil die Variable „Verkäufer" an den feststehenden Ausdruck „anton" gebunden wird. Diese Bindung besteht hingegen nur so lange, wie sich auch die Teile des Regelkörpers an die Objekte „anton" und „berta" binden lassen.
Der Suchlauf des Programms, das Backtracking, setzt sich deshalb im Körper von Regel Nr. 1, sozusagen der Tatbestandsseite, fort. Dort wird der Kaufpreisanspruch vom Abschluß eines Kaufvertrages abhängig gemacht:

```
kaufvertrag(Käufer,Verkäufer,Vertrag)
```

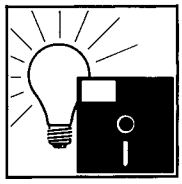
wobei dem Anspruch zusätzlich keine Einwendungen/Einreden des Käufers gegenüberstehen dürfen:

```
not(einwendungen(Käufer,Vertrag))
```

Die Objekte dieser beiden Prädikate „Käufer", „Verkäufer" und „Vertrag" sind an der Großschreibung als Variablen erkennbar. Das Backtracking versucht nun zu beweisen, daß auch die Prädikate des Regelkörpers mit den Objekten des Obersatzprädikates „anton" und „berta" eine zutreffende Aussage ergeben. Deshalb sind die Variablen „Käufer" und „Verkäufer" auch innerhalb der Prädikate „kaufvertrag(...)" und „einwendungen(...)" nicht mehr frei, sondern nehmen wie im Regelkopf jeweils für „Verkäufer" den Wert „berta" und für „Käufer" den Wert „anton" an. Auf diese Weise hat eine automatische Subsumtion stattgefunden, so daß der Regelkörper eigentlich nun folgenden Wert hat:

```
if kaufvertrag(berta,anton,Vertrag)
and not(einwendungen(anton,Vertrag).
```

Die Variable „Vertrag" allerdings bleibt zunächst mangels Entsprechung ungebunden. Die Notwendigkeit, die beiden Tatbestandsvoraussetzungen aus dem Regelkörper auf ihre Richtigkeit zu überprüfen, veranlaßt den Steuerungsmechanismus, mit der Suche



fortzufahren. Vergleichbar mit dem Verlauf einer juristischen Prüfung wird zunächst das erste der beiden Tatbestandsmerkmale aus dem Regelkörper geprüft:

```
kaufvertrag(anton,berta,Vertrag)
```

Das Backtracking hat sofort Erfolg: Denn über die Rechtsfolge „kaufvertrag" trifft Regel Nr. 2 eine Aussage. Mit anderen Worten stimmt der Prädikattyp im Regelkopf mit dem geprüften Prädikattyp überein („kaufvertrag(...)").
Prolog subsumiert das geprüfte Tatbestandsmerkmal von Regel Nr. 1 mit den gebundenen Werten unter den Regelkopf von Regel Nr. 2:

```
kaufvertrag(Käufer,Verkäufer,Vertrag)
            |        |        |
kaufvertrag(anton,   berta,   Vertrag)
```

Der weitere Ablauf entspricht dem bereits besprochenen. Innerhalb des Regelkörpers von Regel Nr. 2 werden die Variablen „Verkäufer" und „Käufer" gebunden; die Regeln nehmen folgenden Wert an:

```
if vertrag_zwisehen(anton,berta,Vertrag)
and schuldet(berta,übereignung,_,anton,Vertrag)
and schuldet(anton,zahlung,_,berta,Vertrag).
```

Die Unterstriche stehen hier für unbenannte Variablen, die jeden Wert annehmen können, nämlich „Kaufsache" und „Kaufpreis".
Die Prüfung des Zustandekommens eines Kaufvertrages nähert sich langsam ihrem Ende. Prolog sucht nach einem Prädikat, das dem ersten Prädikat im Regelkörper von Regel Nr. 2 vertrag_zwischen(...) entspricht. Die beiden nächsten Regeln des Abschnitts clauses passen von ihren Regelköpfen her nicht: Entsprechend werden sie übergangen. Erst im letzten Teil findet sich ein Lebenssachverhalt, der unter das Prädikat subsumiert werden kann, so daß auch die Variable „Vertrag" einen festen Wert annimmt:

Die Subsumtion nähert sich dem Ende ...

```
vertrag_zwischen(anton,berta,Vertrag)
                 |     |      |
vertrag_zwischen(anton,berta,"Vertrag vom 27.12.1988")
```

Lösungen finden sich schließlich auch für die beiden Prädikate „schuldet(...)":

```
schuldet(berta,Übereignung, _, anton,"Vertrag vom 27.12.1988")
         |       |       |    |                  |
schuldet(berta,übereignung,pkw,anton,"Vertrag vom 27.12.1988")
```

und

```
schuldet(anton,zahlung,     _   ,  berta,"Vertrag vom 27.12.1988")
         |       |       |       |              |
schuldet(anton,zahlung,"300,-DM",berta,"Vertrag vom 27.12.1988")
```

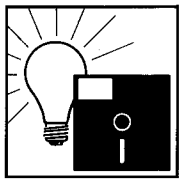
Regel Nr. 2 ist damit erfolgreich geprüft: Daraus folgt, daß auch das erste Tatbestandsmerkmal von Regel Nr. 1 – der Abschluß eines Kaufvertrages – bestätigt wurde. Das Backtracking wendet nun die Suche dem zweiten Tatbestandsmerkmal von Regel Nr. 1 zu – der Abwesenheit von Einwendungen. Das Prädikat

Nun zu prüfen: Fehlen von Einwendungen

```
not(einwendungen(Käufer,Vertrag))
```

ist erst dann erfolgreich, wenn die Variablen Käufer und Vertrag die bereits bekannten festen Werte „anton" und „Vertrag vom 27.12.1988" annehmen können und das Prädikat „einwendungen(...)" sich als falsch erweist.
Der Suchlauf läßt zunächst Regel Nr. 2 außer acht: Regelkopf und Rechtsfolge stimmen nicht mit dem jeweils Gesuchten überein. Regel Nr. 3, die einen Rechtssatz über

Erfüllung als ein Einwendungsfall



Einwendungen beinhaltet, ist hingegen einschlägig: Einwendung ist danach die Erfüllung.
Zur Erfüllung eines Vertrages verhält sich Regel Nr. 4:

```
schuldet(Käufer,zahlung,Summe,Verkäufer,Vertrag)
and
leistet(Käufer,zahlung,Summe,Verkäufer,Vertrag).
```

Die Variablen dieser Prädikate sind aber zum größten Teil nicht mehr frei, sondern haben im Wege der Subsumtion/Bindung die bekannten Werte erhalten:

```
schuldet(anton,zahlung,Summe,berta,"Vertrag vom 27.12.1988")
and
leistet(anton,zahlung,Summe,berta,"Vertrag vom 27.12.1988")
```

Prolog sucht nun im Sachverhalt, also gegen Ende des Abschnitts „clauses", nach Übereinstimmungen und trifft dabei auf folgende Klausel:

```
leistet(anton,zahlung,"300,-DM",berta,"Vertrag vom 28.5.1988")
```

Diese entspricht aber nicht dem gesuchten

```
leistet(anton,zahlung,"300,-DM",berta,"Vertrag vom 27.12.1988").
```

Das Objekt „Vertrag vom 28.5.1988" ist trotz der Großschreibung keine Variable, sondern eine feststehende Zeichenkette (String), was die Anführungszeichen deutlich machen. Eine Subsumtion des Lebenssachverhaltes unter die Regel Nr. 4 (Erfüllung) und damit unter die Regel Nr. 3 (Einwendung) scheitert daher. Juristisch zeigt sich dabei, daß die Leistung einer Zahlung in Höhe des geschuldeten Kaufpreises allein noch keine Erfüllung begründet, vielmehr muß sie mit Rücksicht auf die geschuldete Leistung erfolgen (Erfüllungstheorien). Dies führt innerhalb von Regel Nr. 1 zur Bejahung des zweiten Tatbestandsmerkmals

Entscheidend sind die Erfüllungstheorien

```
not(einwendungen(anton,"Vertrag vom 27.12.1988"))
```

denn weitere Einwendungen sind nicht ersichtlich.
Die Fallfrage läßt sich deshalb unter Regel Nr. 1 subsumieren: Der Anspruch besteht. Das System springt zurück zum Abschnitt „goal". Dort wird noch ein Schlußsatz auf dem Bildschirm ausgegeben, der das Ergebnis nach außen verkündet.

Ergebnis: Der Anspruch besteht

Der erste Eindruck, den das Beispiel hinterläßt, ist abermals ernüchternd. Die erträumte Subsumtionsmaschine hat sich in einen juristischen Pygmäen verwandelt, der genau einen Fall lösen kann.

Statt der „Subsumtionsmaschine" ein „juristischer Pygmäe"?

Von den zahlreichen Fragen, die das Beispiel zurückläßt, können hier nur einige wenige gestreift werden.
Ein grundlegendes Ärgernis ist das völlige Fehlen von Flexibilität innerhalb des Systems. Soll die Maschine gar einen ganzen Sachverhalt aufnehmen können, muß der Anwender Informationen in einer Weise eingeben können, die seiner Denkweise möglichst nahe kommt. Würde der Sachverhalt vor seiner Eingabe zunächst in eine schwer verdauliche Maschinensprache zerlegt, gingen bedeutungstragende Teile schon vor der eigentlichen Subsumtion verloren.

(wird fortgesetzt)