ProVisor

Eine PROLOG-Visualisierung

Benutzerhandbuch

Version 1.0

TH Darmstadt
6. Mai 1993

Auftraggeber: FG Praktische Informatik
StR G. Röhner

Auftragnehmer: Software-Entwicklungsgruppe
Schneewittchen und die sieben Zwerge

Betreuer: Prof. Dr. W. Henhapl

Schneewittchen und die sieben Zwerge sind: Colette Elcacho
Konrad Berg
Thomas Goetze
Ferenc Kurucz
Michael Metzger
Jürgen Schirmer
Marc Volz
Jan Zurek

Inhaltsverzeichnis

Abbildungsverzeichnis

1. Einführung

ProVisor ist eine Visualisierungsumgebung, die den Aufbau von PROLOG–Programmen und deren Ausführung graphisch darstellt. In dieser Umgebung sind neben dem Visualisierungsteil ein PROLOG-Interpreter mit eingeschränktem Sprachschatz, ein Debugsystem und ein Editor integriert. Die Oberfläche ist an bekannte Systeme wie Turbo–Pascal 6.0 angelehnt.

PROLOG ist eine logische Programmiersprache, die aus dem Bereich der künstlichen Intelligenz herausgefunden hat und immer größere Akzeptanz auch in der Fort– und Weiterbildung findet. Erfahrungsgemäß bereitet dieses Sprachkonzept Verständnisschwierigkeiten. Im Rahmen einer Lehrerfortbildung durch HIBS[1] und HILF[2] am Institut für Praktische Informatik der TH Darmstadt entstand daher der Wunsch nach einer Visualisierung von PROLOG. Diesem Wunsch wird mit ProVisor entsprochen.

1.1 Was bietet ProVisor?

ProVisor stellt eine umfassende Entwicklungsumgebung zur Verfügung:

- In ProVisor wird der Ablauf eines PROLOG–Programms dynamisch dargestellt. Dazu werden ein Detail–, ein Unifikations– und ein Übersichtsmodus angeboten, die in getrennten Fenstern betrachtet werden können. In diesen Modi werden Fehlschläge, Erfüllungen, Reerfüllungen oder auch der Cut besonders hervorgehoben. Im Unifikationsmodus sind darüber hinaus die Instantiierungen sichtbar und können so genau verfolgt werden.
- Neben dem Programmablauf können auch einzelne Terme sowohl statisch als auch dynamisch visualisiert werden. Diese Darstellung kann ebenfalls in getrennten Fenstern betrachtet werden, wobei die statische Termvisualisierung unabhängig vom aktuellen Programmlauf erfolgt.
- Der integrierte PROLOG–Interpreter enthält fast den vollständigen Sprachschatz nach dem Edinburgh–Standard. Da sich dieses Programm in erster Linie an die Anfänger von PROLOG wendet, wurde auf die Implementierung vordefinierter Prädikate verzichtet. Der vorliegende Interpreter ermöglicht aber, derartige Prädikate selbst zu erstellen.

[1] Hessisches Institut für Bildung und Schulwesen
[2] Hessisches Institut für Lehrerfortbildung

- Ein integrierter Debugger mit einem vollständigen Breakpointsystem ermöglicht selektive Betrachtungen von Programmabläufen und dynamischen Termdarstellungen. Das Breakpointsystem erlaubt einen gezielten Programmstopp während des Programmlaufs.
- Weitere Eigenschaften der Entwicklungsumgebung sind
  - eine vollständige graphische Oberfläche,
  - ein Multi–Dateien–Editor, der sich über die Tastenkürzel des Common User Access (*CUA*) bedienen läßt,
  - freie Fensterpositionierung
  - sowie eine vollständige Mausunterstützung.

1.2 Erforderliche Hard– und Software

ProVisor läuft auf allen Rechnern der IBM-Computerfamilie, einschließlich XT, AT und PS/2 sowie auf allen voll kompatiblen 286-, 386-, oder 486-Computern. Als Mindestausstattung der Hardware werden

- 640KByte RAM
- eine Festplatte mit mindestens 500 KByte freiem Speicherplatz
- ein Diskettenlaufwerk
- ein 80–Spalten–Bildschirm

benötigt. Als Betriebssystem wird eine MS-DOS–Version 3.31 oder höher erwartet.

1.3 Die Handbücher

Die Handbücher sind so gegliedert, daß durch Querverweise in den verschiedenen Kapiteln genau die Informationen gefunden werden können, die gesucht werden.

- Das *Benutzerhandbuch* ist größtenteils im Stil eines *Tutorials* gehalten. Es erklärt die Installation von ProVisor, beschreibt die Arbeit mit der integrierten Entwicklungsumgebung, unter anderem anhand von Beispielsitzungen, und beschäftigt sich kurz mit den Grundlagen der Programmiersprache PROLOG.

! → **Das Benutzerhandbuch darf nicht als Lehrbuch für PROLOG verstanden werden.** Auf entsprechende Literatur wird im Anhang hingewiesen.

- Das *Programmhandbuch* gibt eine formale Beschreibung des implementierten Sprachumfangs und erläutert ausführlich die Bedienung und Handhabung der Visualisierung und des integrierten Debuggers.

1.4 Die Typographie der Handbücher

Zur Verdeutlichung von Zusammenhängen werden in diesen Handbüchern mehrere unterschiedliche Schriftarten verwendet:

- `Schreibmaschine` – steht für Eingaben des Benutzers, Programmlistings und Ausgaben, die von Beispielprogrammen erzeugt werden;
- **Fettschrift** – kennzeichnet reservierte Wörter von PROLOG in Listings;
- (Alt) – für die Tasten der Tastatur, wie zum Beispiel "ein Druck auf die Tasten (Alt) (F3) " ;
- ⓣ, ⊖ – für die Pfeiltasten zur Bewegung des Cursors und andere Spezialtasten.
- ! → – dient zur Markierung von wichtigen Sätzen, Absätzen oder Warnungen.
- → X – ist ein Querverweis, wobei *X* für die Nummer auf das verwiesene Kapitel steht.

1.5 Lieferbedingungen und Haftungsausschluß von ProVisor

! → Das Softwarepaket ProVisor ist in seiner ausführbaren Version *Public Domain* und darf daher als vollständiges Paket beliebig vervielfältigt und weitergegeben werden. Das Benutzerhandbuch und das Programmhandbuch dürfen ohne vorherige Zustimmung des Eigentümers in Teilen oder im ganzen kopiert, fotokopiert, reproduziert und in maschinenlesbare Form gebracht oder auf andere Weise vervielfältigt werden.

Änderungen am Programm dürfen aber nur in Rücksprache und mit Genehmigung des Eigentümers vorgenommen und veröffentlicht werden.

Das Softwarepaket ProVisor wird »wie besehen« kostenlos weitergegeben. Der Eigentümer und die Programmentwickler übernehmen keinerlei Garantien für die Verwendungsfähigkeit des Softwarepakets ProVisor zu irgendeinem bestimmten Zweck; jede Haftung für direkte, indirekte, verursachte oder gefolgte Schäden, die durch Verwendung dieses Programms entstehen könnten, ist ausgeschlossen.

1.6 Geschützte Warenzeichen

Wir nehmen in diesem Buch auf mehrere Warenzeichen Bezug, die in den Handbüchern nicht mehr als solche gekennzeichnet sind:

- Turbo Pascal ist ein geschütztes Warenzeichen von Borland International;
- MS-DOS ist ein geschütztes Warenzeichen von Microsoft Corporation;
- IBM PC, XT, AT sind geschützte Warenzeichen von International Business Machines (IBM);

1.7 Der Eigentümer

Das im Rahmen eines Praktikums entstandene Software–Paket ProVisor befindet sich im Eigentum von:

```
Technische Hochschule Darmstadt
Fachbereich Informatik
Fachgruppe Praktische Informatik
Magdalenenstraße 11C
6100 Darmstadt
```

Bei Fragen und Problemen bitten wir Sie, zuerst die Handbücher sorgfältig durchzulesen. Wenn danach Fragen offenbleiben, wenden Sie sich bitte an

```
StR G. Röhner
Technische Hochschule Darmstadt
Fachbereich Informatik
Fachgruppe Praktische Informatik
Magdalenenstraße 11C
6100 Darmstadt
```

2. Installation und Konfiguration von ProVisor

Das ProVisor–Paket verfügt über ein einfaches automatisches Installations– und Konfigurationsprogramm mit dem Namen SETUP. Dieses Programm legt die benötigten Verzeichnisse an und erstellt eine Initialisierungsdatei mit dem Namen PROVISOR.PTH. Weiterhin werden die zum Arbeiten mit ProVisor notwendigen Dateien auf die Festplatte kopiert. Daneben kann aber auch eine manuelle Installation und Konfiguration vorgenommen werden, wenn eine individuelle Einrichtung des Pakets ProVisor mit SETUP nicht möglich erscheint.

Dieses Kapitel enthält folgende Informationen:

- Lieferumfang von ProVisor
- automatische und manuelle Installierung von ProVisor auf Ihr System
- automatische und manuelle Änderungen der Konfiguration

2.1 Lieferumfang von ProVisor

Vor der Installation von ProVisor sollten Sie sich über die Vollständigkeit des vorliegenden Software–Pakets vergewissern. Dies allein gewährleistet zwar keinen Schutz vor vireninfizierten Programmen, aber eine im Umfang abweichende Version sollte mit gebotener Vorsicht behandelt werden.

Das Originalpaket enthält folgende Dateien:

- Systemdateien
  - **PROVISOR.EXE** – der residente Teil von ProVisor. Er bleibt im Speicher und enthält die wichtigsten Datenbereiche des transienten Teils.
  - **PROVISOR.OVR** – der transiente Teil, der die komplette Oberfläche, Editor, Interpreter, Visualisierung und weiteren Moduln enthält.
  - **SETUP.EXE** – das Installationsprogramm.
  - **SYSTEM.PRO** – enthält einige Definitionen von Systemprädikaten.
- Graphiktreiber und Zeichensatzdateien

- **CGA.BGI** – Graphiktreiber für CGA– und MCGA–Modus.
- **EGAVGA.BGI** – Graphiktreiber für EGA– und VGA–Modus.
- **HERC.BGI** – Graphiktreiber für Hercules–, Monochrome– oder MDA–Modus.
- **ATT.BGI** – Graphiktreiber für AT&T-Graphikmodus.
- **PC3270.BGI** – Graphiktreiber für 3270 PC–Graphikmodus.
- **IBM8514.BGI** – Graphiktreiber für IBM 8514.
- **GOTH.CHR** – Vektorzeichensatz für CGA–, MCGA–, EGA– oder VGA–Modus.
- **LITT.CHR** – wie GOTH.CHR.
- **SANS.CHR** – wie GOTH.CHR.
- **TRIP.CHR** – wie GOTH.CHR.
- **8X8.BMP** – Bitmapzeichensatz für Hercules– oder MDA–Modus.
- **8X14.BMP** – wie 8X8.BMP.
- **8X16.BMP** – wie 8X8.BMP.

• PROLOG-Beispieldateien

- **FAMILY.PRO** – einfaches Beispiel von Familienbeziehungen. → Kapitel 3.
- **FAC.PRO** – Berechnung der Fakultät.
- **INSERTSO.PRO** – Insertionsort-Algorithmus.
- **BEINE.PRO** – Mit diesem Programm kann man alle möglichen Kombinationen von Spinnen und Käfern in einer Kiste unter Vorgabe der Beinanzahl erfragen. Käfer haben 6 und Spinnen haben 8 Beine. Z.B: kiste(24, Spinnen, Käfer).
- **APPEND.PRO** – Erzeugung einer Liste
- **SELECT.PRO** – Beispielprogramm aus [SHA88], Seite 53
- **RANGE.PRO** – Liefert die Liste der Zahlen von M bis N, wobei M < N vorrausgesetzt wird.
- **VORFAHR.PRO** – Beispielprogramm aus [ROE92]
- **GROSSMUT.PRO** – Beispielprogramm aus [ROE92]

– **MEMBER.PRO** – Das Programm prüft, ob ein Element E in einer Liste X enthalten ist.

- Letzte Informationen

  – **README.TXT** – enthält Hinweise zur Installation und letzte Änderungen.

2.2 Installation

Da ProVisor nicht mit einem Komprimierungs– oder Archivierungsprogramm gepackt ist und auch nicht mit Schutzvorrichtungen versehen ist, kann das Paket automatisch mit dem Installationsprogramm SETUP oder manuell installiert werden. Für die manuelle Installation sind aber Grundkenntnisse des Betriebssystems MS–DOS oder eines hierzu kompatiblen Betriebssystems erforderlich. Diese Kenntnisse können im Rahmen dieser Beschreibung nicht vermittelt werden.

2.2.1 Das Installationsprogramm SETUP

Die Installation erfolgt am einfachsten mit dem Programm SETUP. In der nachfolgenden Beschreibung gehen wir der Einfachheit halber von einem Standard–System mit einer Festplatte aus, die die Laufwerksbezeichnung C: trägt und einem Diskettenlaufwerk mit Laufwerksbezeichnung A:. Die Installation kann aber von einem beliebigen Diskettenlaufwerk auf ein ebenso beliebiges Ziellaufwerk erfolgen, wenn genügend Speicherplatz vorhanden ist. Die Installation besteht dann aus folgenden Schritten:

- Starten Sie Ihren Computer. Wenn das DOS–Prompt erscheint, legen Sie die Diskette in das Diskettenlaufwerk A: und geben den Befehl:

  ```
  A:SETUP
  ```

  Dadurch wird SETUP gestartet und meldet sich der »Begrüßungsbildschirm«, der nach einem Druck auf (Return) wieder verschwindet.

- SETUP fragt nun, ob eine Installation oder die Änderung einer bestehenden Konfiguration vorgenommen werden soll. Die Voreinstellung richtet sich danach, ob im aktuellen Verzeichnis eine Datei mit Namen PROVISOR.PTH gefunden wurde und diese das korrekte Format aufweist. (Bei Voreinstellung Konfiguration → 2.3). Für die Installation wird der Menüpunkt *Installation* gewählt und mit (Return) abgeschlossen.

- Nun erscheint eine Anfrage nach dem Quellaufwerk, von dem aus die Installation gestartet werden soll. SETUP gibt hierbei das Laufwerk A: vor. Außerdem wird nach dem Ziellaufwerk gefragt, auf das ProVisor

eingerichtet werden soll und das als Vorgabe die Laufwerksbezeichnung C: hat. Bestätigen Sie diese Vorgaben oder Ihre Änderungen mit (Return) .

! → Der Wechsel zum nächsten Menü kann einige Sekunden dauern, da zuerst eine Überprüfung der Eingaben vorgenommen wird. Bei einem erneuten Druck auf (Return) wird daher nicht die Anzeige des nächsten Menüs beschleunigt, sondern dieses Menü übersprungen!

- Nachdem Sie *Installation* gewählt haben, zeigt SETUP ein weiteres Menü, in dem das Stammverzeichnis von ProVisor aufgelistet ist sowie zwei Schalter, über die eine einfache, selektive Installation ermöglicht wird:

Abbildung 2.1: Stammverzeichnis und Schalter für selektive Installation

Mit den beiden Schaltern kann man bestimmen, ob

- die Graphiktreiber einschließlich Zeichensatzdateien und
- die Beispieldateien

ebenfalls installiert werden.

! → Bei den Graphiktreibern ist zu beachten, daß ProVisor ohne BGI-Graphiktreiber nicht läuft. Wenn Sie auf Ihrem Computer diese Treiberprogramme noch nicht installiert haben, müssen die BGI-Graphiktreiber mit den Systemdateien kopiert werden.

SETUP gibt als Pfadname C:\PROVISOR vor, es kann aber ein beliebiger Pfad eingegeben werden, sofern er den Konventionen des Betriebssystems genügt.

! → Wenn Sie eine selektive Installation vornehmen wollen, dürfen Sie nicht den Namen des Stammverzeichnisses mit (Return) bestätigen. SETUP wechselt dann zum nächsten Menü und geht gegebenenfalls von einer vollständigen Installation aus. Wechseln Sie stattdessen von der Eingabezeile zu den Schaltern mit (TAB) .

Die Schalter kann man mit ⊕, ⊖ selektieren und mit der Leertaste ein– bzw. ausschalten.

Sind alle Eingaben erfolgt, werden sie mit Return bestätigt.

- Nun erscheint ein Menü, in dem die (Unter–)Verzeichnisse für die Beispieldateien und die Graphiktreiber mit Zeichensatzdateien anzugeben sind. Nach Vorgabe von SETUP werden hierzu eigene Unterverzeichnisse angelegt, in die die entsprechenden Dateien kopiert werden.

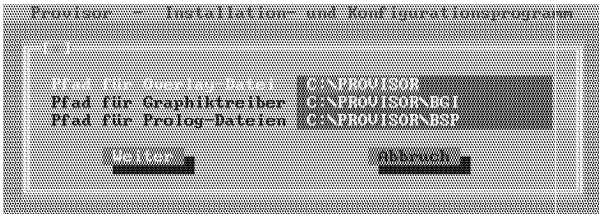

Abbildung 2.2: Eingabe der (Unter–) Verzeichnisse

! → Eine Ausnahme bildet die erste Eingabezeile, in der der Pfad für PROVISOR.OVR einzugeben ist. Die Vorgabe ist der Pfadname des Stammverzeichnisses von ProVisor und sollte nur dann geändert werden, wenn die Overlay–Datei in eine RAM–Disk geladen werden soll.

Es ist dann die Laufwerksbezeichnung der RAM–Disk einzugeben. Beachten Sie, daß ProVisor das Kopieren in die RAM–Disk nicht vornimmt. Auch werden durch SETUP nicht die hierfür notwendigen Änderungen in der CONFIG.SYS Ihres Systems vorgenommen oder eine entsprechende Batchdatei erzeugt. Diese Änderungen und die Erzeugung der Batch–Datei müssen selbst vorgenommen werden.

! → Wenn Sie eine selektive Installation gewählt haben und keine Installation der Graphiktreiber wollen, müssen Sie einen alternativen Pfad auf bereits installierte BGI-Treiber eingeben. Wenn Ihr Computer mit einer Hercules– oder MDA–Graphikkarte ausgestattet ist, müssen aus dem ProVisor–Paket die Dateien mit der Endung ≫.BMP≪ in das Verzeichnis der BGI-Treiber manuel kopiert werden.

- In diesem Menü werden die Einstellungen für die Verteilung des Puffers auf die Overlay–Datei, dem Editor und die Verwaltung des Fenstersystems vorgenommen.

SETUP bietet hierzu zwei Standardvorgaben an und die Möglichkeit einer individuellen Einstellung. Beachten Sie, daß die von uns gewählten Vorgaben rein subjektiv sind und nur unseren Erfahrungswerten entsprechen, die wir auf den zur Entwicklung dieses Programmpakets

Abbildung 2.3: Einstellung des Puffers mit Standardvorgabe

benutzten Computern gesammelt haben.

Selektieren Sie einen Schalter mit ⊕, ⊖ und bestätigen Sie Ihre Wahl mit (Return) .

- Nach Wahl der individuellen Einstellung gelangen Sie in das folgende Menü:

Abbildung 2.4: Individuelle Einstellung des Puffers

Geben Sie nun für jede Eingabezeile einen entsprechenden Wert ein. Die zulässigen Eingabewerte liegen dabei in folgenden Intervallen:

- **Puffer für PROVISOR.OVR :** 64 – 256
- **Puffer für Editor :** 28 – 128
- **Puffer für die Fenster :** 24 – 64

Wechseln Sie zwischen den Eingabezeilen mit (TAB) oder (SHIFT) + (TAB) und bestätigen Sie abschließend Ihre Eingaben mit (Return) .

- Nach Einstellung des Puffers werden die Verzeichnisse und PROVISOR.PTH erzeugt und ProVisor eingerichtet. Abschließend erscheint ein »Abschiedsbildschirm«, der mit (Return) verlassen wird. Nun befinden Sie sich im Stammverzeichnis von ProVisor und können das Programm mit dem Befehl

```
PROVISOR
```

aufrufen.

2.2.2 Die manuelle Installation

Die manuelle Installation setzt einige Kenntnisse des Betriebssystems Ihres Rechners voraus, insbesondere die Erzeugung von Verzeichnissen und ASCII–Dateien sowie das Kopieren von Dateien. Mit diesen Kenntnissen wird die Installation wie folgt durchgeführt:

- Erzeugen Sie zunächst auf Ihrem Ziellaufwerk eine Ihren Vorstellungen entsprechende Verzeichnisstruktur.
- Kopieren Sie nun die Dateien in die entsprechenden Verzeichnisse.

! → Die Systemdateien (→ 2. 1) sollten in ein Verzeichnis kopiert werden. Wenn Sie, anstelle der im Paket enthaltenen BGI-Graphiktreiber, eigene, bereits installierte Treiber benutzen wollen, beachten Sie bitte, daß für den Betrieb im MDA– oder Hercules– Graphikmodus die Zeichensatzdateien mit der Endung ≫.BMP≪ benötigt werden. Diese sind dann von Diskette möglichst in das Verzeichnis der BGI–Graphiktreiber zu kopieren.

- Nun ist im Verzeichnis der Systemdateien die Datei PROVISOR.PTH zu erzeugen. Diese Datei hat folgendes Aussehen:

  `OVR = <`*Verzeichnis für PROVISOR.OVR*`>`
  `BGI = <`*Verzeichnis für die BGI–Graphiktreiber*`>`
  `LOG = <`*Verzeichnis für die Protokolldateien*`>`
  `BMP = <`*Verzeichnis für die BGI–Graphiktreiber*`>`
  `PRO = <`*Verzeichnis für die PROLOG–Beispieldateien*`>`
  `OVRBUFFER = <`*Puffer für PROVISOR.OVR*`>`
  `EDITBUFFER = <`*Puffer für den Editor*`>`
  `WINDOWBUFFER = <`*Puffer für die Verwaltung der Fenster*`>`

  Anstelle der eckigen Klammern, einschließlich ihres Inhalts, sind hier die entsprechenden Pfadnamen einzugeben.

  Am einfachsten kann PROVISOR.PTH dadurch erzeugt werden, daß das Programm SETUP aufgerufen wird. Anstelle *Installation* wird der Menüpunkt *Konfiguration* gewählt, über den PROVISOR.PTH erzeugt wird (→ 2. 3).

! → Der Pfadname für OVR sollte normalerweise mit dem Pfadnamen für LOG übereinstimmen. Diese Angabe ist nur für den Fall vorgesehen, daß PROVISOR.OVR in eine RAM-Disk geladen werden soll. Dies kann zu einer Beschleunigung der Ausführung von ProVisor führen. Hierzu ist eine entsprechende Batch–Datei zu erzeugen, da ProVisor das Laden in eine RAM-Disk nicht automatisch vornimmt. Außerdem muß gegebenenfalls die Datei CONFIG.SYS Ihres Systems geändert

werden, um eine solche RAM-Disk zu erzeugen.

Die zulässigen Eingabewerte für Pufferzuteilung liegen in folgenden Intervallen:

- Puffer für PROVISOR.OVR : **64** – **256**
- Puffer für Editor : **28** – **128**
- Puffer für die Fenster : **24** – **64**

- Befinden Sie sich nun im Stammverzeichnis mit den Systemdateien, können Sie ProVisor mit dem Befehl

  `PROVISOR`

  aufrufen.

2.3 Konfiguration

Nachträgliche Änderungen, wie z. B. Änderung der Pufferzuteilung oder der Verzeichnisstruktur, können ebenfalls mit dem Programm SETUP vorgenommen werden. Es läßt sich auch eine defekte oder fehlende Datei PROVISOR.PTH neu erzeugen. Hierzu sind folgende Schritte notwendig:

- Starten Sie das Installations– und Konfigurationsprogramm vom Verzeichnis der Systemdateien (→ 2. 1) mit dem Befehl

  `<`*Laufwerk:\Pfad*`>` `SETUP ,`

  wobei die geklammerten Parameter optional sind. Es erscheint wieder der »Begrüßungsbildschirm«, der mit (Return) verlassen wird.

- In dem nun erscheinenden Menü wählen Sie die Option *Konfiguration* und bestätigen diese Wahl mit (Return) . Wenn die Datei PROVISOR.PTH im aktuellen Verzeichnis nicht gefunden wurde, erscheint eine Fehlermeldung, ob diese Datei erzeugt werden soll, was Sie gegebenenfalls mit (Return) bestätigen.

- In den nachfolgenden Menüs haben Sie nun die Möglichkeit, die von Ihnen gewünschten Änderungen vorzunehmen (→ 2.2).

! → Die vorgenommenen Änderungen an der Verzeichnisstruktur werden nur in PROVISOR.PTH eingetragen. Die Erzeugung der Verzeichnisse sind aber manuell vorzunehmen.

Natürlich lassen sich Änderungen an PROVISOR.PTH auch manuell vornehmen. Beachten Sie aber dabei genau das Format der Datei (→ 2.2.2). Dieses Format wird mit SETUP gewährleistet, weshalb wir Änderungen an PROVISOR.PTH nur mit diesem Programm empfehlen.

3. Bedienung der Oberfläche

3.1 Beschreibung der Oberfläche

3.1.1 Die Komponenten

In der Arbeitsumgebung zu ProVisor finden Sie drei sichtbare Komponenten: Das Auswahlmenü in der obersten Zeile, die eigentliche Arbeitsfläche und darunter die Statuszeile. Viele Wahlpunkte aus dem Menü dienen zum Öffnen von Dialogfenstern. Bevor wir dazu übergehen, die einzelnen Wahlpunkte detailliert zu beschreiben, befassen wir uns mit den generischen Komponenten.

3.1.2 Die Menüzeile und die einzelnen Menüs

Sämtliche Menü-Befehle sind über die Menüzeile erreichbar. Die Menüzeile ist immer sichtbar. Wenn das Auswahlmenü aktiviert ist, sehen Sie einen hervorgehobenen dargestellten Menütitel; dieser zeigt das aktuell *gewählte* Menü an.

Wenn Sie einen Menü-Befehl mit Fortsetzungspunkten (...) wählen, öffnet sich ein Dialogfenster. Wählen sie hingegen einen Befehl, dem ein Pfeil folgt, gelangen Sie in ein weiteres Menü (ein Pop-Up-Menü). Menü-Befehle ohne Zusatzzeichen führen ihre Aufgabe sofort nach Auswahl aus. Im folgenden wird die Anwahl der Menü-Befehle über die Tastatur erläutert:

1. Drücken Sie (F10) . Dadurch wird die Menüzeile aktiviert.
2. Die Auswahl eines Menüs erfolgt mit den Pfeiltasten und (Return) oder durch die Eingabe des Tastenkürzels, d.h. den hervorgehobenen Buchstaben des gewünschten Menütitels. Mit (Esc) brechen Sie eine Option ab.

   Das Edit-Menü können Sie beispielsweise von der Menüzeile aus mit (E), von jeder beliebigen anderen Stelle aus mit (Alt) (E) aufrufen.
3. Wählen Sie innerhalb des Menüs nochmals mit Hilfe der Pfeiltasten einen Befehl an oder geben Sie das Tastenkürzel ein (den hervorgehobenen Buchstaben des Befehls).

ProVisor führt dann entweder den Befehl aus, öffnet ein Dialogfenster oder öffnet ein weiteres Menü.

Sämtliche Befehle lassen sich auch mit der Maus anwählen:

1. Öffnen Sie das Menü, indem Sie den Menütitel anklicken.
2. Klicken Sie den Befehl an.

Sie können den Mauszeiger vom Menütitel aus auch geradlinig zum gewünschten Menü-Befehl ziehen und dann die Maustaste loslassen. (Dieser Vorgang wird abgebrochen, wenn Sie den Mauszeiger aus dem Menü ziehen und ihn dann loslassen.)

Die Menü-Befehle selbst sind nur dann verfügbar, wenn ihre Anwahl auch sinnvoll ist.

3.1.2.1 Tastenkürzel

ProVisor bietet eine Reihe von Möglichkeiten zur schnellen Anwahl von Menü-Befehlen. Der schnellste Weg für Maus-Benutzer sieht folgendermaßen aus:

Halten Sie die Maustaste gedrückt, ziehen Sie den Mauszeiger vom Hauptmenü zum gewünschten Menü-Befehl und lassen Sie die Maustaste dann los; der Menü-Befehl wird augenblicklich ausgeführt. Noch schneller erfolgt die Anwahl eines Menü-Befehls, wenn Sie die in der Statuszeile dargestellten Tastenkürzel anklicken.

Tastatur-Benutzern stehen für denselben Zweck zahlreiche Tastenkürzel zur Verfügung. Sie erreichen bzw. aktivieren Menü-Befehle durch das gleichzeitige Drücken der (Alt) -Taste und des hervorgehobenen Buchstabens. Ist ein Menü aktiviert, dann brauchen Sie lediglich den hervorgehobenen Buchstaben des gewünschten Menü-Befehls (oder das nebenstehende Tastenkürzel) zu drücken. Innerhalb von Dialog-Fenstern arbeiten Sie auf die gleiche Weise. In den nachstehenden Tabellen finden Sie die gebräuchlichsten Tastenkürzel:

Taste	Menü-Befehl	Funktion
F2	Datei/Speichern	Speichert die Datei, die sich im aktiven Editor-Fenster befindet.
F3	Datei/Öffnen	Öffnet das Dialog-Fenster *Datei* zum Laden einer Datei.
F4	Datei/Consult	Öffnet das Dialog-Fenster *Datei* zum Konsultieren einer Datei.
Alt F4	Datei/Reconsult	Öffnet das Dialog-Fenster *Datei* zum Rekonsultieren einer Datei.

Taste	Menü-Befehl	Funktion
F5	Fenster/Ganzer Bildschirm	Vergrößert das aktive Fenster auf die Größe des Hauptfensters.
F6	Fenster/Nächstes	Wechselt zum nächsten Fenster und aktiviert es.
F7	Visualisierung/Einzelschritt	Setzt den Programmlauf im Einzelschrittmodus fort.
F8	Visualisierung/Überspringen	Setzt den Programmlauf fort, ohne schrittweises Verfolgen der untergeordneten Prädikate.
F9	Visualisierung/Weiter	Setzt den Programmlauf fort, bis eine Lösung gefunden wird.
F10		Aktiviert die Menüzeile.
Alt X	Datei/Beenden	Beendet ProVisor.

Tabelle 3.1: Allgemeine Tastenkürzel

Taste(n)	Menü-Befehl	Funktion
Ctrl Del	Bearbeiten/ Löschen	Löscht den markierten Text aus dem Fenster, ohne ihn in das Clipboard zu kopieren.
Ctrl Ins	Bearbeiten/ Kopieren	Kopiert den markierten Text in das Clipboard.
Shift Del	Bearbeiten/ Ausschneiden	Kopiert den markierten Text in das Clipboard und löscht die Markierung.
Shift Ins	Bearbeiten/ Einfügen	Kopiert den Text aus dem Clipboard ins aktive Fenster.
Alt C	-	Konsultiert den Text, der sich im aktiven Editor-Fenster befindet.
Alt R	-	Rekonsultiert den Text, der sich im aktiven Editor-Fenster befindet.

Tabelle 3.2: Editor Tastenkürzel

Taste(n)	Menü-Befehl	Funktion
Alt F3	Fenster/Schließen	Schließt das aktive Fenster.
Alt F5	Fenster/Interpreter-fenster an/aus	Schaltet die Anzeige des Interpreterfensters an bzw. aus.
F5	Fenster/Ganzer Bildschirm	Vergrößert/verkleinert das aktive Fenster (Zoom).
F6	Fenster/Nächstes	Wechselt zum nächsten Fenster und aktiviert es.
Ctrl F5	Fenster/Größe	Ändert Größe und Position des aktiven Fensters. Zum Verändern der Position können die Cursortasten, zum Verändern der Größe Shift und die Cursortasten verwendet werden.

Tabelle 3.3: Tastenkürzel zur Fensterverwaltung

Taste	Menü-Befehl	Funktion
F7	Visualisierung/Einzelschritt	Setzt den Programmlauf im Einzelschrittmodus fort.
F8	Visualisierung/Überspringen	Setzt den Programmlauf fort ohne schrittweises Verfolgen der untergeordneten Prädikate.
F9	Visualisierung/Weiter	Setzt den Programmlauf fort bis eine Lösung gefunden wird.
Ctrl Back	Visualisierung/Stoppen	Hält den momentanen Programmlauf an.
Ctrl Del	Visualisierung/Abbrechen	Bricht den momentanen Programmlauf ab.

Tabelle 3.4: Tastenkürzel während des Interpreterlaufs

3.1.3 Fenster in ProVisor

Die Arbeit in ProVisor erfolgt fast ausschließlich in *Fenstern*. Ein Fenster ist ein Teil des Bildschirmes, den Sie verschieben, vergrößern, verkleinern, teilen, mit anderen Fenstern verdecken, öffnen und schließen können.

Die Anzahl der Fenster, die Sie öffnen können, hängt nur vom verfügbaren Speicher ab — *aktiv* ist allerdings immer nur ein einziges. Als aktives Fenster wird immer das Fenster bezeichnet, in dem Sie gerade arbeiten.

Zwar gibt es mehrere Fenstervarianten, die meisten davon haben aber folgende Elemente gemeinsam:

- einen Titel
- ein Schließfeld
- zwei Rollbalken
- ein Eckfeld zur Änderung der Größe
- ein Zoom-Feld
- eine Fensternummer

Das aktive Fenster erkennen Sie an der doppelten Umrahmung. Es verfügt jeweils über ein Schließfeld, ein Zoom-Feld, zwei Rollbalken und ein Eckfeld, über das die Fenstergröße geändert werden kann. Überlappen sich zwei Fenster, ist immer das Fenster im Vordergrund aktiv.

Das Editorfenster zeigt zusätzlich die aktuelle Zeilen- und Spaltennummer in der unteren linken Ecke an. Wurde der Text im Editor-Fenster geändert, erkennen Sie dies an dem Zeichen "*" links neben den Spalten- und Zeilennummern.

Die Abbildung 3.1 zeigt ein typisches Fenster:

Das *Schließfeld* finden Sie in der oberen linken Ecke. Durch Anklicken bzw. mit *Fenster/Schließen* oder mit Alt F3 können Sie ein Fenster schließen.

Der *Titel*, der oberste horizontale Balken, enthält den Namen des Fensters und die Fensternummer. Durch doppeltes Anklicken können Sie das Fenster auf volle Größe vergrößern bzw. die ursprüngliche Größe wiederherstellen, durch ein einfaches Anklicken und Halten können Sie das Fenster verschieben.

Das *Zoom-Feld* eines Fensters finden Sie in der oberen rechten Ecke. Wird in dieser Ecke als Sinnbild ein Pfeil nach oben angezeigt, können Sie den Pfeil anklicken und das Fenster zu seiner maximalen Ausdehnung vergrößern. Ein Pfeil mit zwei Spitzen als Sinnbild zeigt an, daß das Fenster bereits in seiner

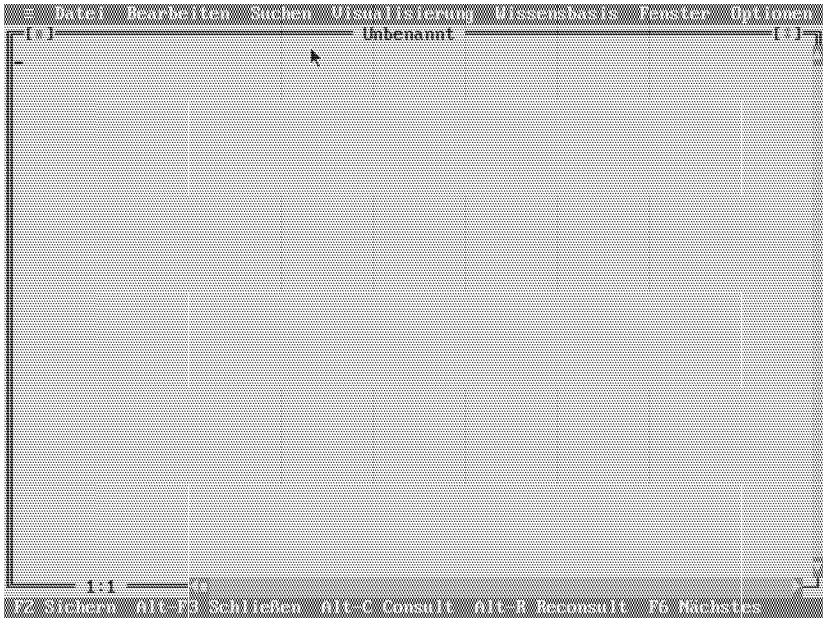

Abbildung 3.1: Typisches Fenster von ProVisor

maximalen Größe dargestellt wird. In diesem Fall wird das Fenster nach dem Anklicken dieses Pfeiles auf seine ursprüngliche Größe zurückgesetzt. Wenn Sie die Tastatur benutzen, dann vergrößern oder verkleinern Sie ein Fenster mit *Fenster/Ganzer Bildschirm* oder durch Betätigen der F5 - Taste.

Rollbalken sind horizontale oder vertikale Balken am rechten bzw. unteren Fensterrand, die an beiden Enden durch einen Pfeil begrenzt sind. Sie enthalten eine Positionsmarke, die die Position des Fensterausschnittes relativ zum Gesamtinhalt anzeigt. Einen Rollbalken können Sie mit der Maus benutzen, um den Inhalt eines Fensters zu "rollen". Klicken Sie an einem beliebigen Ende des Balkens das Pfeil-Symbol an, so wird immer eine Zeile bzw. eine Spalte gerollt. Kontinuierliches Rollen erfolgt, wenn Sie die Maustaste gedrückt halten und die Positionsmarke ziehen. Möchten Sie jeweils genau eine Seite rollen, klicken Sie die schattierten Bereiche neben der Positionsmarke an. Außerdem können Sie die Positionsmarke zu einem beliebigen Punkt auf dem Rollbalken ziehen, wenn sie schnell an einer anderen Position innerhalb des Fensters, bezogen auf die augenblickliche Stellung der Positionsmarke, gelangen möchten.

Das Feld zum *Verändern der Größe* finden Sie in der unteren rechten Ecke (an dieser Stelle ist der Fensterrahmen nur mit einer einfachen Linie gezeichnet). Bewegen Sie diese Ecke, so wird das Fenster größer bzw. kleiner. Über die Tastatur ändern Sie die Größe mit *Größe* aus dem *Fenster*-Menü oder mit

Ctrl F5. Zum Verändern der Position können die Cursortasten benutzt werden. Zum Ändern der Größe benutzen Sie Shift und die Cursortasten.

3.1.4 Verwaltung der Fenster

In folgender Tabelle finden Sie einen Kurzüberblick über die Fenster in ProVisor. Sie können alle Aktionen sowohl mit der Maus als auch mit der Tastatur durchführen.

Ergebnis	Arbeitsmethoden
Öffnen eines Editorfensters	Wählen Sie *Datei / Öffnen* oder drücken Sie F3.
Schließen eines Fensters	Wählen Sie im Fenster-Menü *Schließen*, drücken Sie Alt F3, oder klicken Sie das Schließfeld an.
Aktivieren eines Fensters	Klicken Sie das Fenster an einer beliebigen Stelle an, oder wählen Sie *Fenster / Nächstes* oder F6 und aktivieren Sie damit das nächste Fenster.
Verschieben des aktiven Fensters	Verschieben Sie den Titelbalken, oder drücken Sie Ctrl F5 (*Fenster / Größe/Verschieben*) und verwenden Sie die Pfeiltasten zum Positionieren des Fensters. Drücken Sie anschließend Return .

Ergebnis	Arbeitsmethoden
Verkleinern des aktiven Fensters	Verschieben Sie die Ecke zum Verkleinern, oder wählen Sie *Fenster/Größe* und drücken Sie (Shift), solange Sie die Pfeiltasten verwenden. Drücken Sie dann (Return). Wollen Sie ein Tastenkürzel verwenden, drücken Sie (Ctrl)(F5) und arbeiten dann mit (Shift) und den Pfeiltasten.
Vergrößern des aktiven Fensters	Klicken Sie das Zoom-Feld in der oberen rechten Ecke an, klicken Sie den Titel zweimal an, wählen Sie *Fenster/Ganzer Bildschirm*, oder drücken Sie (F5).

Tabelle 3.5: Fenster Manipulieren

3.1.5 Die Statuszeile

Die Statuszeile befindet sich in der untersten Zeile des ProVisor-Bildschirms. Die Funktionen der Statuszeile:

- Sie zeigt die aktuell verfügbaren Tastenkürzel für die Menü-Befehle, die Sie auch in der Statuszeile anklicken bzw. eingeben können, statt sie im Menü zu wählen.
- Sie meldet, was das Programm gerade macht. Beim Sichern einer Datei wird beispielsweise "Sichere *Dateiname*" ausgegeben.

Die Statuszeile ändert sich, wenn Sie die Fenster wechseln oder bestimmte Menü-Befehle aufrufen.

3.2 Dialog-Fenster

Menü-Befehle mit Fortsetzungspunkten (...) öffnen ein Dialog-Fenster mit mehreren Optionen und Einstellungen.

Für Einstellungen in Dialog-Fenstern stehen Ihnen die folgenden fünf verschiedenen Bildschirm-Steuerungen zur Verfügung: Schaltfelder, Markierungsfelder, Aktionsschalter, Eingabefelder und Listen. Bei Farbmonitoren werden die einzelnen Elemente des Dialog-Fensters in verschiedenen Farben dargestellt. Die Abbildung 3.2 zeigt ein typisches Dialog-Fenster.

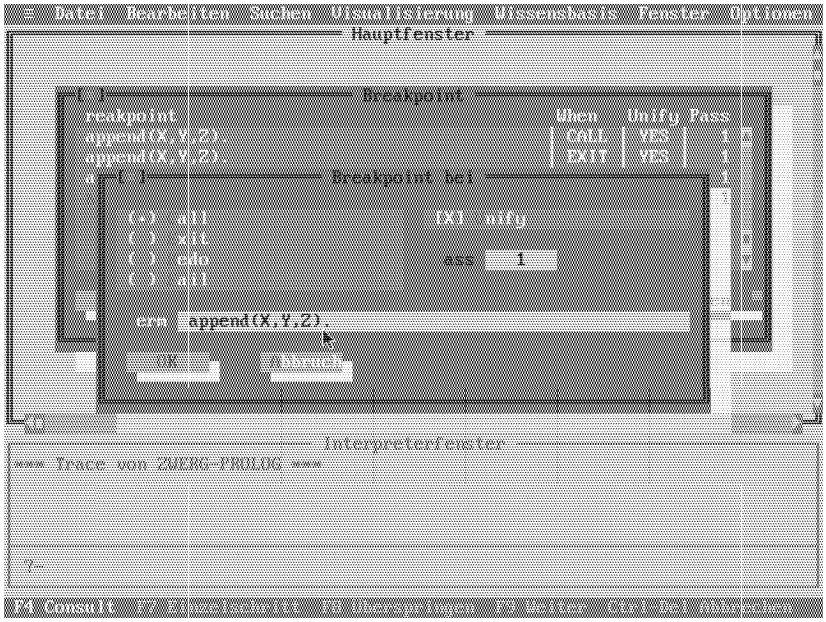

Abbildung 3.2: Typisches Dialog-Fenster

In diesem Dialog-Fenster finden Sie zwei *Aktionsschalter*: *OK* und *Abbruch*. Der Schalter *OK* bewirkt die Umsetzung der getroffenen Auswahl in ProVisor. Mit *Abbruch* verlassen Sie das Dialog-Fenster wieder und verwerfen die Eingaben. (Esc) ist ein Tastenkürzel für *Abbruch* (auch wenn kein *Abbruch*-Aktionsschalter erscheint).

Die Betätigung der Aktionsschalter erfolgt entweder durch Anklicken mit der Maus oder über die Tastatur durch Drücken der (Alt) -Taste und dem hervorgehobenen Buchstaben: Mit (Alt) (O) bestätigen Sie z.B. den *OK*-Aktionsschalter. Die Tasten (Tab) (vorwärts) bzw. (Shift) (Tab) (rückwärts) dienen innerhalb eines Dialog-Fensters zur Aktivierung der einzelnen Elemente. Aktivierte Elemente werden immer hervorgehoben dargestellt. Mit (Tab) springen Sie zum nächsten Aktionsschalter; mit (Return) wird er bestätigt.

Im Dialog-Fenster der Abbildung ist die Voreinstellung *OK*, d.h. Sie wählen den Aktionsschalter durch Drücken von (Return) . Bei Monochrom-Systemen wird die Voreinstellung mit Pfeilen, bei Farbmonitoren hervorgehoben angezeigt. Beachten Sie, daß durch die Anwahl eines Aktionsschalters mit (Tab) die Voreinstellung geändert wird.

3.2.1 Markierungsfelder und Schaltfelder

Gewählte *Markierungsfelder* werden durch ein **X** gekennzeichnet. Ein leeres Feld bedeutet also, daß die Option ausgeschaltet ist. Ein Markierungsfeld wird durch Anklicken des Feldes bzw. des zugehörigen Textes gewählt, oder indem Sie (Tab) drücken (bis das Markierungsfeld hervorgehoben wird) und anschließend (Leert.) drücken, oder indem Sie (Alt) und den hervorgehobenen Buchstaben eingeben.

Sie können beliebig viele Markierungsfelder auswählen. Markierungsfelder werden manchmal auch zu einer Gruppe zusammengefaßt. Ist die Gruppe mit (Tab) angewählt, markieren Sie mit den Cursortasten das gewünschte Element und schalten es mit der (Leert.)-Taste ein bzw. aus. Bei Monochrom-Bildschirmen werden die aktiven Markierungsfelder bzw. Gruppen von Markierungsfeldern mit einem Chevron-Symbol, (») dargestellt. Nach Drücken von (Tab) springt das Symbol zur nächsten Gruppe von Markierungsfeldern oder Schaltfeldern.

Schaltfelder sind im Gegensatz zu Markierungsfeldern prinzipiell *immer* in Gruppen zusammengefaßt. Der entscheidende Unterschied zu Markierungsfeldern besteht darin, daß die Wahl *eines* Schaltfeldes die Wahl eines anderen Schaltfeldes innerhalb der gleichen Gruppe ausschließt. Die Wahl erfolgt mit der Maus durch Anklicken des Schaltfeldes bzw. des zugehörigen Textes oder mit der Tastatur durch Drücken der (Alt) -Taste und des hervorgehobenen Buchstabens (Sie können auch (Tab) drücken, bis die Gruppe hervorgehoben wird und dann mit den Cursortasten das Schaltfeld anwählen. Mit (Tab) oder (Shift) (Tab) verlassen Sie die Gruppe mit dem neu gewählten Schaltfeld wieder).

3.2.2 Eingabefelder und Listen

Dialog-Fenster können auch Eingabefelder enthalten. Die meisten Editier-Tasten haben auch in diesem Fenstertyp ihre gewohnte Funktion (z.B. die Cursortasten (Home), (End) und (Ins)). Sollten Sie beim Eintippen einmal bis ganz an den Rand des Eingabefeldes gelangen, dann wird der Fensterinhalt automatisch weitergerollt. Außerdem erscheinen, falls das Eingabefeld mehr Text beinhaltet als im dafür reservierten Bildschirmbereich dargestellt werden kann, an den Enden des Fensters Pfeilspitzen als Symbole, die zum Rollen und Verschieben des Textes mit der Maus dienen.

Ein ↓ -Symbol auf der rechten Seite des Eingabefeldes gibt an, daß zu diesem Feld eine Eingabeaufzeichnungsliste existiert, welche durch Anklicken des ↓ -Symbols oder über die Taste (↓) geöffnet wird. In der Liste finden Sie die letzten von Ihnen eingegebenen und verwendeten Texte. Die Aufzeichnungsliste des *Suchen*-Feldes zeigt z.B. Begriffe, die Sie bereits gesucht

haben. Wählen Sie einen der alten Einträge, editieren Sie diese Zeile nötigenfalls, und bestätigen Sie Ihre Wahl mit (Return) . Mit (Esc) brechen Sie die Funktion ab und verlassen die Eingabeaufzeichnungsliste.

Ein weiterer Bestandteil vieler Dialog-Fenster ist eine Liste mit variabler Länge. Sie können durch eine Liste rollen und Elemente auswählen, ohne dabei das Dialog-Fenster zu verlassen.

Ein Listen-Fenster aktivieren Sie mit dem hervorgehobenen Buchstaben aus dem Titel des Listen-Fensters, durch Anklicken oder durch Drücken der (Tab) -Taste, solange bis das Fenster hervorgehoben ist. Den Inhalt der Liste bewegen Sie mit Hilfe der Positionsmarke oder mit den Cursortasten.

3.2.3 Zusätzliche Eigenschaften des Editors

In nachstehender Zusammenfassung erfahren Sie, welche Funktionen der Editor über die einfache Texteingabe hinaus enthält:

- Mausunterstützung
- Die Tasten (Shift) (↑)(↓)(→)(←) zum Markieren von Text
- Editor-Fenster, deren Größe veränderbar ist, die sich überlappen können und die Sie auch verschieben können
- Multidatei-Fähigkeiten, wodurch Sie mehrere Dateien gleichzeitig öffnen können
- Ein editierbares Clipboard, mit dessen Hilfe Sie Texte ausschneiden, kopieren und von einem Fenster in ein anderes kopieren können

3.3 Arbeiten mit ProVisor

Nachdem Sie ProVisor gestartet haben, erscheint die für ProVisor typische Bildschrimaufteilung. Den größten Teil des Bildschirms nimmt das (noch leere) Hauptfenster ein, im unteren Bildschirmbereich befindet sich das Interpreterfenster. In der Statuszeile erscheint noch für einige Sekunden die Meldung "*Bitte warten*", bis die Initialisierung abgeschlossen ist. Das Interpreterfenster selber ist wiederum zweigeteilt. Im unteren Bereich steht eine Eingabezeile für Anfragen an den PROLOG-Interpreter zur Verfügung, im oberen Bereich werden die Ausgaben des Interpreters protokolliert.

Hauptfenster und Interpreterfenster zusammen bedecken immer den gesamten Bildschirm. Sie können weder verschoben noch geschlossen werden. Die einzige Möglichkeit das Hauptfenster zu vergrößern, besteht darin, das Interpreterfenster über den Menüpunkt *Fenster/Interpreterfenster an/aus* oder die Tastenkombination (Alt) (F5) auszublenden.

Eine weitere Besonderheit des Hauptfensters ist, daß es nicht in den Vordergrund springt, wenn es ausgewählt wird. Daher müssen Sie alle anderen Fenster schließen, wenn Sie nur das Hauptfenster sehen wollen.

Drücken Sie nun (F3) zum Öffnen des Dialog-Fensters *Datei/Öffnen*. Der Cursor befindet sich nun in dem Eingabefeld für den Dateinamen. Geben Sie hier `Nummer1` ein und drücken Sie anschließend die (Return) -Taste (die Extension .PRO brauchen Sie nicht einzugeben, da sie von ProVisor automatisch hinzugefügt wird). Nun können Sie damit beginnen, Ihr erstes Programm zu schreiben:

```
vater(peter,anna).
vater(peter,klaus).
```

Zuerst einmal ein paar Hinweise zu diesen Zeilen:

- Jede Programmzeile wird mit (Return) abgeschlossen.
- für die Korrektur von Tippfehlern stehen Ihnen die Cursortasten (←), (→), (↑) und (↓) zur Verfügung, (Backsp.) löscht jeweils das Zeichen, das sich links des Cursors befindet. Korrekturen sind auch dann noch möglich, wenn die betreffende Zeile bereits mit (Return) abgeschlossen wurde.

Überprüfen Sie Ihre Eingaben noch einmal genau (beide Zeilen enden mit einem Punkt, alle Wörter sind klein geschrieben).

3.3.1 Konsultieren

Drücken Sie die Tasten (Alt) (C). Für eine kurze Zeit erscheint nun ein Fenster, das Sie darüber informiert, daß der Text konsultiert (in die Datenbasis des Interpreters übernommen) wird.

Wenn Sie Tippfehler gemacht haben, dann erscheint eine entsprechende Fehlermeldung im Interpreterfenster. Editieren Sie in diesem Fall ihren Text noch einmal und Sie verbessern den angegebenen Fehler und drücken Sie erneut (Alt) (C).

Ihr Programm wurde nun in die Datenbasis des Interpreters übernommen. Das können Sie überprüfen, indem Sie sich die Wissensbasis des Interpreters anzeigen lassen. Ein Druck auf (F10) beendet den Editor und bringt Sie zur Menüleiste von ProVisor zurück. Aktivieren Sie nun mit (W) oder (Alt) (W) das Menü *Wissensbasis* und wählen Sie mit einem weiteren Druck auf (A) den Punkt *Anzeigen* innerhalb dieses Menüs.

Im Interpreterfenster erscheinen nun Ihre eingegebenen Zeilen.

Nun verschieben Sie das Editor-Fenster *NUMMER1.PRO* an den rechten Rand. Drücken Sie dazu F10 , um in die Menüleiste zu kommen. Aktivieren Sie mit F oder Alt F den Menüpunkt *Fenster* und wählen dort mit G den Punkt *Größe*. Alternativ zu der ganzen Prozedur hätten Sie auch die Tasten Ctrl F5 drücken können.

Der Rahmen des Editorfensters hat sich verändert und das Fenster kann mit den Cursortasten verschoben werden. Durch Drücken der End –Taste verschieben Sie jetzt das Editorfenster ganz nach rechts. Mit Return beenden Sie das Verschieben.

Als nächstes muß das Interpreterfenster aktiviert werden, damit eine Anfrage an den Interpreter gestellt werden kann. Drücken Sie die F6-Taste.

Dadurch wurde das aktive Fenster geändert, das Editorfenster ist nun deaktiviert und im Interpreterfenster blinkt der Cursor. Geben Sie nun

```
vater(X,anna).
```

ein (achten Sie dabei auf die Groß-/Kleinschreibung und den Punkt am Ende der Zeile) und drücken Sie Return .

Die Anfrage hat den Interpreter gestartet und man sieht im Hauptfenster die Anfrage in graphischer Form.

Drücken Sie nun entweder

1. die Taste F9 oder

2. wählen Sie in der Menüleiste den Menüpunkt *Visualisierung/Weiter* mit Alt V und danach W aus.

Der Interpreter sucht nun nach einer Lösung und meldet nach kurzer Zeit

```
X = peter
yes
```

Es erscheint ein Dialog-Fenster, in dem Sie angeben können, ob nach weiteren Lösungen gesucht werden soll. Drücken Sie N um anzugeben, daß Sie an keiner weiteren Lösung interessiert sind.

Damit ist der Interpreterlauf beendet (bis die nächste Anfrage eingegeben wird).

3.3.2 Die Speicherung des Quelltextes

Damit unser Meisterwerk nicht verlorengeht, wenn Sie ProVisor irgendwann einmal verlassen, muß es als Datei gespeichert werden: Dazu muß zuerst das Editor-Fenster aktiviert werden. Drücken Sie solange (zweimal) die (F6)-Taste, bis das Editor-Fenster aktiv ist. Drücken Sie nun (F2) oder wählen Sie *Speichern* aus dem *Datei*-Menü.

3.3.3 Funktionen, die nicht über die Menüleiste erreichbar sind

Einige Funktionen können nicht über die Menüleiste angesprochen werden. Im folgenden erhalten Sie eine Übersicht über diese Funktionen.

3.3.3.1 Anfragen stellen

Anfragen an den Interpreter stellen Sie, indem Sie das Interpreterfenster aktivieren (mit der Maus anklicken oder solange (F6) drücken, bis das Interpreterfenster aktiv ist) und die Anfrage in der Eingabezeile eingeben. Mit den Tasten (↑) und (↓) können Sie auf vorherige Eingaben zurückgreifen. Anfragen müssen mit einem Punkt enden, sonst erhalten Sie eine entsprechende Fehlermeldung. Beenden Sie die Eingabe der Anfrage mit (Return) und der Interpreter beginnt damit, die Anfrage zu erfüllen.

Das Interpreterfenster selbst ist in zwei Bereiche unterteilt. Es gibt eine Eingabezeile und ein Ausgabefenster. Die Eingabezeile ist aktiviert, wenn der Cursor dort blinkt. Das Ausgabefenster ist aktiviert, wenn die Rollbalken zum Verschieben des Ausgabetextes sichtbar sind. Zum Wechseln zwischen beiden Bereichen drücken Sie die (Tab) -Taste oder klicken den gewünschten Bereich mit der Maus an.

! → Die Eingabezeile kann nur aktiviert werden, wenn der Interpreter sich nicht in einem Programmlauf befindet, d.h. wenn keine Anfrage bearbeitet wird.

3.3.3.2 Konsultieren und Rekonsultieren eines eingegebenen Textes

Ist das aktive Fenster ein Editor-Fenster, dann ändert sich die Statuszeile. In der Statuszeile befinden sich dann die Funktionen *Consult* und *Reconsult*, die mit (Alt) (C) bzw. (Alt) (R) oder durch Anklicken mit der Maus aktiviert werden können.

! → Diese Funktionen können nur aufgerufen werden, wenn der Interpreter keine Anfrage bearbeitet.

4. Einführung in ProVisor

ProVisor ist mehr als ein Visualisierungsprogramm für PROLOG — dieser Kern ist in eine Entwicklungsumgebung eingebunden, die einen PROLOG–Interpreter, einen Editor, Mechanismen zur Fehlersuche, zum Laden und Speichern von Texten und noch einiges mehr enthält. Sämtliche Funktionen dieser Umgebung sind von einem Punkt aus erreichbar: dem Hauptmenü.

Das ProVisor–Programmpaket ist in erster Linie für schulische Zwecke gedacht. Dieses Kapitel gibt daher anhand eines Beispiels eine einführende Beschreibung der Anwendungsmöglichkeiten von ProVisor innerhalb der integrierten Entwicklungsumgebung.

Um Ihnen einen einführenden Überblick zu verschaffen, haben wir das Kapitel in fünf Abschnitte unterteilt:

- der erste Abschnitt gibt eine kurze Einführung in PROLOG, um insbesondere Anfängern der Programmiersprache PROLOG die Möglichkeit zu geben, das nachfolgende Beispielprogramm zu verstehen.

! → Dieser Abschnitt darf nicht als Ersatz für ein Lehrbuch über PROLOG verstanden werden.

- im zweiten Abschnitt werden Start und Beendigung von ProVisor beschrieben sowie der Umgang mit dem Hauptmenü. Eine ausführliche Beschreibung der Bedienung der Oberfläche konnten Sie im vorhergehenden Kapitel lesen (→ Kap. 3).

- der dritte und vierte Abschnitt beschäftigt sich mit dem Erzeugen, Editieren und Laden von PROLOG–Programmdateien; der Sprachumfang des integrierten PROLOG–Interpreters werden ausführlich im *Programmhandbuch*(→ Kap. 3) beschrieben.

- der letzte Abschnitt gibt eine Einführung in die Visualisierung. Es werden die grundsätzlichen Visualisierungsmöglichkeiten besprochen. Eine Detailbeschreibung finden Sie ebenfalls im Programmhandbuch (→ Kap. 4).

4.1 Kleine Einführung in PROLOG

PROLOG basiert auf Konzepten der logischen Programmierung und gehört in die Gruppe der sogenannten nichtprozeduralen Sprachen. Weitere wichtige und interessante Vertreter dieser Gruppe sind LISP — der Klassiker unter

den nichtprozeduralen Sprachen —, SQL — eine relationale Datenbankabfragesprache — oder auch PLANNER, eine Spezialsprache der künstlichen Intelligenz.

All diesen Sprachentwicklungen liegt die Idee zugrunde, einem Rechner nicht mehr mitzuteilen, *wie* er ein Problem lösen soll, sondern ihm zu sagen *was* für ein Problem man hat. Das *wie* soll nunmehr von der Maschine allein besorgt werden.

Verwendet man klassische prozedurale Sprachen zum Lösen eines Problems, so werden in der Regel die implementierten Algorithmen die notwendigen Informationen und den Kontrollfluß des Algorithmus, der schließlich zur Problemlösung führt, miteinander verschränken. Oftmals sind aber notwendige Informationen nicht explizit in Datenstrukturen vorhanden, sondern nur implizit in Schleifen oder Sprüngen versteckt. In PROLOG werden nun diese beiden Aspekte des Problemlösens voneinander getrennt:

- alle problemspezifischen Informationen werden in einer Wissensbasis gesammelt,
- die Auswertung erfolgt durch eine problemunabhängige Beweismaschine.

Das Konzept der logischen Programmierung besteht nun darin, daß das problemspezifische Wissen in einer speziellen Form des logischen Prädikatenkalküls 1. Stufe[1], den sogenannten *Hornklauseln* beschrieben wird. Logische Programmierung heißt somit: anwendungsspezifisches Wissen wird durch Klauseln der formalen Logik repräsentiert. Ein PROLOG–Programm ist demnach eine Sammlung von *Klauseln*, die von einer PROLOG–Maschine interpretiert werden.

4.1.1 Terme, Fakten, Regeln und Klauseln

PROLOG besitzt eine einheitliche zugrundeliegende Datenstruktur: den Term. Syntaktisch gibt es in PROLOG nur Terme — Eingabedaten und Hornklauseln sind nur spezielle Formen von Termen. Terme kann man sich als markierte Bäume vorstellen. Sie werden in der Regel in Präfix–Notation notiert[2]:

<Term> :- (<Regel>)| <Fakt> | <Variable> | <Term> {, <Term>})

[1] Es ist sicherlich nicht notwendig, daß Sie mit dem Prädikatenkalkül vertraut sind oder sich damit vertraut machen müssen. Der Hinweis auf das Prädikatenkalkül ist nur als Information gedacht. Wenn Sie sich weiter für dieses logische Konzept interessieren, finden Sie eine Einführung bei Nillson[NIL81].

[2] Die genaue Syntax für Terme finden Sie im Programmhandbuch, Kapitel 3.

Beispiele für Terme sind:

```
hugo
1
plus(sieben,zwei)
```

Die »Blätter« in solchen Termen heißen *Atome*, sofern es sich nicht um Zahlen oder Variablen handelt. In unserem Beispiel tauchen die Atome "hugo", "sieben" und "zwei" auf. Atome besitzen keinen Wert wie z. B. in prozeduralen Sprachen, sondern stehen für sich selbst. Atome, die nichtnumerische Zeichen enthalten oder mit einem Großbuchstaben beginnen sollen, werden in Hochkommata eingeschlossen. Die Bezeichner am Anfang von zusammengesetzten Termen heißen Funktoren. Für den Term *plus(sieben,zwei)* ist "plus" der Funktor und "sieben", "zwei" seine Argumente.

Fakten sind spezielle Formen von Hornklauseln. Sie dienen zur Darstellung logischer Aussagen, die ohne weitere Bedingung wahr sind. Ein Fakt ist syntaktisch ein Term, gefolgt von einem Punkt. Dieser ist ein Funktor ohne Argument und damit ohne Klammern. Beispiel für Fakten sind:

```
ist_Mutter_von(conny,bastian).
ist_Mutter_von(conny,corinna).
ist_Mutter_von(lotte,moritz).
ist_Mutter_von(lotte,anna).
ist_Mutter_von(lotte,theresa).
ehemann(peter,kathi).
ehemann(claus,conny).
ehemann(ulrich,lotte).
```

Das Fakt "`ist_Mutter_von(conny,bastian).`" ist zu lesen:

```
Conny ist Mutter von Bastian
```

und das Fakt "`ehemann(claus,conny).`" entsprechend:

```
Claus ist Ehemann von Conny
```

Fakten können bereits ein vollständiges PROLOG–Programm bilden.

Regeln sind logische Aussagen, die im Gegensatz zu Fakten an gewissen Voraussetzungen gebunden sind. Typische Regeln sind in unserer Miniwelt:

```
ist_Vater_von(Vater,Kind) :-
   ehemann(Vater,Mutter),ist_Mutter_von(Mutter,Kind).
ist_Kind_von(Kind,Vorfahr) :-
   ist_Mutter_von(Vorfahr,Kind).
ist_Kind_von(Kind,Vorfahr) :-
   ist_Vater_von(Vorfahr,Kind).
```

Das vordefinierte Atom ":-" bedeutet soviel wie "wenn". Der Term links von ":-" — der sogenannte Klauselkopf — ist wahr, wenn die Terme rechts von ":-" — dem Klauselrumpf — wahr sind. Das Komma, das die Terme im Klauselrumpf voneinander trennt, ist hierbei als "und" zu lesen.

Besondere Terme sind die Variablen. Sie beginnen mit einem Großbuchstaben und werden nicht mit Hochkommata geklammert. Variablen sind Platzhalter für weitere Terme. Kommt dieselbe Variable mehrfach in einer Regel vor, so kann stets nur derselbe Term für die Variable substituiert werden. Variablen in verschiedenen Regeln sind dagegen auch bei Namensgleichheit verschieden. In der Regel

```
ist_Kind_von(Kind,Vorfahr) :-
   ist_Mutter_von(Vorfahr,Kind).
```

haben wir als Variablen die Terme "Kind" und "Vorfahr". Die Regel besagt:

```
Ein Kind ist Kind von einem Vorfahr, wenn der Vorfahr
   Mutter des Kindes ist.
```

Diese durchaus sinnvolle Aussage wird in unserer Miniwelt durch eine weitere Regel ergänzt, durch die dem Kind auch ein Vater zugeordnet wird.

Variablen können nicht nur in Regeln, sondern auch in Fakten auftreten. So besagt das Fakt

```
ehemann(ulrich,X).
```

daß Ulrich polygam sei, da er nach diesem Fakt mit allen Frauen — und Männern! — verheiratet ist.

Fakten und Regeln werden unter dem Begriff Klauseln zusammengefaßt. Offenbar sind Fakten Klauseln mit fehlendem Klauselrumpf. Klauseln können nach dem Funktor der linken Seite und der Zahl der Parameter gruppiert werden. Eine Menge von Klauseln mit demselben Funktor und identischer Zahl von Argumenten heißt ein *Prädikat*.

4.1.2 Der PROLOG–Interpreter

Nachdem wir im vorhergehenden Abschnitt geklärt haben, wie eine Wissensbasis entsteht, stellt sich nun die Frage, was ein PROLOG–Interpreter mit dieser anfängt.

Dem PROLOG–Interpreter werden nun verschiedene Fragen gestellt, die er durch Durchsuchen der Wissensbasis und Anwendung bestimmter Regeln zu beantworten sucht. Wollen wir beispielsweise wissen, ob Lotte die Mutter von Theresa ist, stellen wir folgende Anfrage an den PROLOG–Interpreter:

```
?-ist_Mutter_von(lotte,theresa).
```

Falls die Information in der Wissensbasis zu finden ist, wird der Interpreter mit "yes", andernfalls mit "no" antworten. Die Wissensbasis wird dabei sequentiell *von vorne nach hinten* durchsucht.

! → Aus dieser sequentiellen Suche folgt, daß die Reihenfolge der Klauseln in PROLOG sehr wichtig ist.

Anfragen können auch Variablen enthalten. "Nenne mir alle Kinder von Lotte" sieht so aus:

```
?-ist_Mutter_von(lotte,X).
```

Bei der Durchsuchung der Wissensbasis wird überprüft, ob Köpfe von Klauseln oder Fakten die Anfrage *matchen*, d.h. dieselbe Struktur wie der Anfrageterm haben. Dabei können Variablen, die im Anfrageterm vorkommen, *instantiiert* werden. Die Anfrage "`?-ist_Mutter_von(lotte,X).`" matcht zunächst das Faktum

```
ist_Mutter_von(lotte,moritz).
```

Dabei wird die Variable X mit dem Atom "moritz" instantiiert. Der PROLOG-Interpreter gibt dann aus:

```
X = moritz
```

Wird der Suchprozeß fortgesetzt, führt dies zu folgender Ausgabe:

```
X = anna
```

danach zu

```
X = theresa
```

und schließlich

```
no
```

d.h. es konnte keine matchende Klausel mehr gefunden werden.

Dieser Suchprozeß ist ein Spezialfall der allgemeinen Vorgehensweise des PROLOG-Interpreters. Wir wollen nun wissen, wer die Eltern von Theresa sind und stellen daher folgende Anfrage:

```
?-ist_Kind_von(theresa,X).
```

In der Wissensbasis findet der Interpreter nun die Klausel

```
ist_Kind_von(Kind,Vorfahr) :-
   ist_Mutter_von(Vorfahr,Kind).
```

Er *instantiiert* nun das Atom "theresa" des *matchenden* Klauselkopfs mit der Variablen "Kind" aus dem Klauselrumpf. Die Variable "X" aus dem Klauselkopf wird dagegen mit der Variablen "Vorfahr"aus dem Klauselrumpf *unifiziert*. Nun versucht der Interpreter einen *Match* für

```
ist_Mutter_von(X,theresa).
```

und findet in der Wissensbasis das Fakt

```
ist_Mutter_von(lotte,theresa).
```

womit die Anfrage erfolgreich erfüllt werden konnte. Die Variable "X" wird durch den PROLOG-Interpreter zu "lotte" instantiiert und die Klausel "`ist_Kind_von(Kind,Vorfahr) :- ist_Mutter_von(Vorfahr,Kind).`" markiert.

Nun soll ein weiterer *Match* gesucht werden. Das nächste Ziel ist nun die markierte Regel, also

```
ist_Kind_von(theresa, lotte)
```

Der Interpreter versucht nun für dieses Ziel eine weitere Erfüllung, gelangt zu

```
ist_Mutter_von(lotte, theresa)
```

Das Ziel

```
ist_Mutter_von(X, theresa)
```

kann jedoch nicht erfüllt werden, d.h. der Interpreter findet keine weitere Mutter für "Theresa". Das Ziel ist *fehlgeschlagen*, wodurch ein bestimmter Prozeß — das *Backtracking* — ausgelöst wird: der Interpreter versucht "ist_Kind_von(theresa, lotte)" auf andere Weise zu erfüllen. Die von dem Interpreter intern und für den Anwender unsichtbaren Markierungen der Klauseln, die er erfolglos zu lösen versuchte, werden zurückgenommen. Dabei sucht der Interpreter gleichzeitig nach weiteren Klauseln, die er *matchen* kann. Der Interpreter findet nun in der Wissensbasis:

```
ist_Kind_von(Kind,Vorfahr) :-
   ist_Vater_von(Vorfahr,Kind).
```

Die Instantiierung von "X" zu "lotte" wird nun zurückgenommen und der Interpreter versucht eine *Re–Erfüllung* für

```
ist_Kind_von(theresa,X).
```

Nun läuft der Suchprozeß für dieses Regel genauso ab wie zuvor. "Kind" wird wieder zu "theresa" instantiiert und "Vorfahr" zu "X" unifiziert. Um eine Erfüllung für das Ziel

```
ist_Vater_von(X,theresa).
```

zu erhalten, sind die beiden Teilziele

```
ehemann(X,Mutter)
```

! → und[3]

```
ist_Mutter_von(Mutter,theresa)
```

zu erfüllen. Der Interpreter findet für das erste Teilziel das Fakt

```
ehemann(peter,kathi)
```

kann aber mit diesen Instantiierungen keine Erfüllung des zweiten Teilziels erreichen. Das erste Teilziel wird dann mit

```
ehemann(claus,conny)
```

instantiiert, was ebenfalls zu einem Fehlschlag des zweiten Teilziels führt. Die nächste Instantiierung des ersten Teilziels führt nun zu

```
ehemann(ulrich,lotte)
```

mit

```
ist_Mutter_von(lotte,theresa)
```

als Instantiierung von "`ist_Mutter_von(Mutter,theresa)`". Der Interpreter gibt dann als Antwort

```
X = ulrich
```

womit wir laut Wissensbasis "Lotte" und "Ulrich" als Eltern identifiziert haben. Eine weitere Suche des Interpreters nach Elternteilen von "Theresa" ergibt schließlich auch *no*, womit die Suche beendet ist.

Dieses Beispiel macht die allgemeine Vorgehensweise des Interpreters deutlich. Für ein vorgegebenes Ziel wird zunächst eine matchende Regel in der Wissensbasis gesucht. Die neuen Teilziele im Rumpf einer eventuell gefundenen Regel werden dann von *links* nach *rechts* zu erfüllen versucht. Dabei wird die Wissensbasis *sequentiell* durchsucht. Jedes aktivierte (Teil-)Ziel — also die, die noch nicht fehlgeschlagen sind — ist mit einem Zeiger in der Wissensbasis assoziiert, der auf die letzte erfolgreich gematchte Klausel zeigt. Anfangs deutet dieser Zeiger vor diese Klausel, nach einem endgültigen Fehlschlag hinter der Klausel. Wenn ein Teilziel fehlgeschlagen ist, wird der sogenannte *Backtracking*–Prozeß initiiert, d.h. der Interpreter versucht eine *Re–Erfüllung* für das Ziel, das in der Klausel links vom fehlgeschlagenen Ziel steht. Die Markierungen werden dabei nach vorne bewegt. Wenn das am

[3] Das "," entspricht, wie wir bereits gesagt haben, der Konjunktion

weitesten links stehende Ziel fehlschlägt, wird das Backtracking beim Aufrufer fortgesetzt. Während des Suchprozesses werden Variablen instantiiert und unifiziert. Ist ein Ziel fehlgeschlagen, werden die zuletzt vorgenommenen Instantiierungen und Unifikationen wieder zurückgenommen. Kann ein Ziel erfüllt werden, gibt der Interpreter als Antwort *yes* aus. Wenn weitere Lösungen gewünscht werden, wird der *Backtracking*–Prozeß dort aufgenommen, wo er unterbrochen wurde.

Prinzipiell kann man einen PROLOG-Intepreter als einen Theorembeweiser betrachten. Eine Anfrage stellt dann ein Theorem dar, das der Interpreter zu widerlegen versucht. Scheitert die Widerlegung, so ist die Anfrage erfüllt, d.h. das Theorem aus der Wissensbasis beweisbar.

4.2 Starten und Beenden von ProVisor

Ziel dieses und der nachfolgenden Abschnitte in diesem Kapitel ist, daß Sie

- mit der Anwendung von ProVisor vertraut werden,
- einen Einblick in die Möglichkeiten von ProVisor erhalten,
- die Visualisierung von PROLOG verstehen lernen und
- das Beispiel aus dem vorhergehenden Abschnitt (→ Kap. 4.1, S. 28) besser nachvollziehen können.

Es ist daher sinnvoll, wenn Sie die folgende Beschreibung an Ihrem Computer nachvollziehen.

Zum Programmstart von ProVisor wechseln Sie in das Verzeichnis mit den Systemdateien (→ Kap. 2.1, S. 5). Das Programm können Sie nun mit dem Befehl

```
PROVISOR
```

starten.

Der sich nun darstellende Bildschirm (siehe Abb. 4.1) kann in fünf Bereiche unterteilt werden:

1 : Über die Menüzeile können Sie sämtliche Menübefehle erreichen (→ Kap. 3.1.2, S. 13).

2 : In das Hauptfenster wird die Visualisierung dargestellt. Standardmäßig ist für das Hauptfenster der Unifikationsmodus eingestellt. Sie können aber über den Menüauswahl *Visualisierung/Darstellung...* den Darstellungsmodus ändern.

! → Das Hauptfenster kann nicht geschlossen werden.

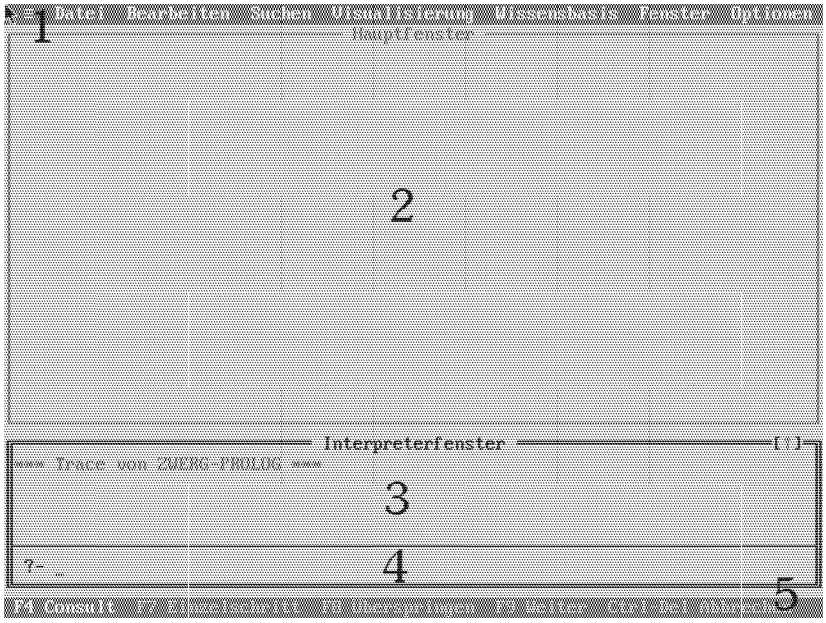

Abbildung 4.1: Bildschirm von ProVisor

3 : In den oberen Teil des Interpreterfensters wird bei einem Programmlauf das *Trace-Protokoll* eingeblendet. Über dieses Teilfenster kann man sich auch nach einem Programmlauf das Protokoll ansehen. Dazu müssen Sie mit der Maus in das Teilfenster klicken oder über F6 bzw. Shift F6 das Fenster aktivieren. Dann können Sie mit Hilfe der Cursortasten ↑ und ↓ das Trace-Protokoll betrachten.

! → Das gesamte Interpreterfenster (d.h. einschließlich der nachfolgend beschriebenen Eingabezeile) läßt sich über die Tastenkombination Alt F5 aus– bzw. auch einschalten. Damit haben Sie die Möglichkeit, die Visualisierungsfläche zu vergrößern.

4 : In den unteren Teil des Interpreterfensters stellen Sie Ihre Anfragen an ein zuvor *konsultiertes* Programm.

5 : Die Statuszeile enthält die aktuell verfügbaren Tastenkürzel. Weiterhin werden Meldungen ausgegeben, die Sie darüber informieren, was ProVisor gerade macht.

Neben den standardmäßig eingerichteten Fenstern (Haupt– und Interpreterfenster) können Sie beliebig viele Editor– und Graphikfenster öffnen.

! → Eine Beschränkung gilt für die Fenster zur Darstelllung des Programmlaufs: hier können höchstens zwei weitere Fenster neben dem Hauptfenster geöffnet werden. Diesen drei Fenstern können Sie nun die drei zur Verfügung stehen-

den Darstellungsmodi zuordnen — wenn Sie wollen, in allen drei Fenstern den gleichen Modus.

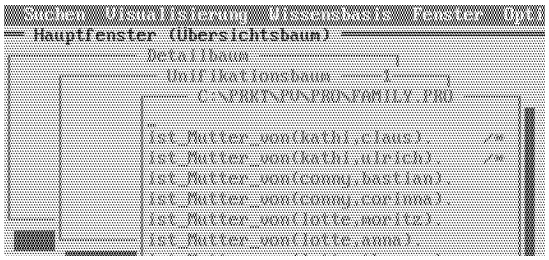

Abbildung 4.2: Die verschiedenen Fenster in ProVisor

Wie Sie in Bild 4.2 sehen können, enthalten alle Graphikfenster eine Überschrift, die den Darstellungsmodus der jeweiligen Fenster angeben.

Die Editorfenster haben als Überschrift den Pfadnamen der Datei, die sie beinhalten.

ProVisor können Sie nun auf zwei Wege wieder verlassen:

- über die Menüzeile wählen Sie den Menüpunkt *Datei/Beenden*
- und mit der Tastenkombination (Alt) (X).

Bevor Sie aber eine dieser beiden Möglichkeiten wahrnehmen, begleiten Sie uns durch die folgenden Abschnitte.

4.3 Schreiben eines Beispielprogramms

Nachdem Sie nun ProVisor gestartet haben, wollen Sie sicherlich auch ein Programm editieren. Zunächst beginnen wir mit einer existierenden Datei, die Sie aus der Einführung in PROLOG (→ Kap. 4.1, S. 28) kennen: FAMILY. PRO.

Sie starten hierzu das Menü *Datei/Öffnen* über die Taste (F3) und erhalten ein Auswahlfenster wie in Bild 4.3 dargestellt.

Selektieren Sie nun die gewünschte Datei — für unsere Beispielsitzung also FAMILY.PRO — oder geben Sie direkt den Namen in die Eingabezeile des Dateiauswahlmenüs ein. Nun können Sie (Return) drücken und auf Ihrem Bildschirm erscheint einen Augenblick später ein Editorfenster mit dem Inhalt der selektierten Datei.

Dieses Fenster nimmt zunächst nur einen Teil der Fläche des Hauptfensters ein und ist daher zum Editieren zu klein. Drücken Sie daher die Taste (F5), wodurch das Editorfenster nun die gesamte Fläche des Hauptfen-

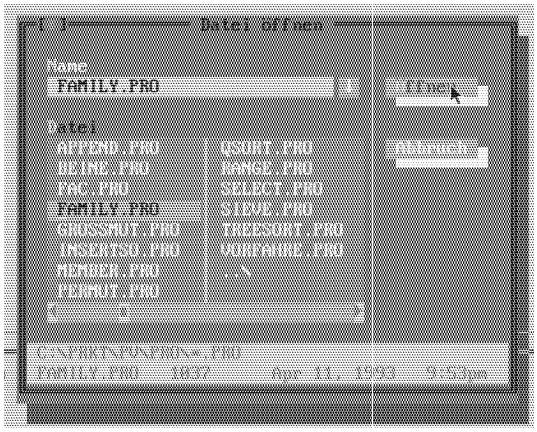

Abbildung 4.3: Dateiauswahl für den Editor

sters einnimmt. Wenn Ihnen auch das nicht ausreicht, schalten Sie über (Alt) (F5) das Interpreterfenster aus und drücken erneut auf (F5) . Sie erhalten nun ein Editorfenster, wie es in Bild 4.4 abgebildet ist.

Sie können nun, wie Sie das von Ihrer Arbeit in anderen Entwicklungsumgebungen gewohnt sind, in dieser Datei editieren:

Angenommen, Sie wollen FAMILY.PRO — unser Beispiel — um das Fakt ergänzen, daß Jan und Natascha verheiratet sind. Nun paßt "eheleute" als Funktor nicht in die Miniwelt von FAMILY.PRO, daher fügen Sie die Aussage "Jan ist Ehemann von Natascha" in die Wissensbasis ein. Dazu formen Sie diese Aussage in das Fakt

```
ehemann(jan,natascha).
```

um. Da Jan und Natascha noch keine Kinder haben, sind Sie mit der Erweiterung von FAMILY.PRO fertig. Diese wertvollen Informationen sollten natürlich nicht verloren gehen, weshalb die Datei über die Taste (F2) gesichert wird.

Auf diese Weise können Sie jedes PROLOG–Programm erweitern oder ergänzen.

Wenn Sie nun selbst ein Programm schreiben wollen, öffnen Sie über den Menüpunkt *Datei/Neu* ein leeres Editorfenster. In dieses Fenster können Sie nun Ihren Programmtext schreiben. Nach Beendigung sollten Sie das Sichern nicht vergessen. Mit der Taste (F2) öffnet sich nun ein Menüfenster, das Sie nach dem Namen der von Ihnen erstellten Datei fragt. Nachdem Sie den Dateinamen eingegeben haben, wird die Datei im aktuellen Verzeichnis gespeichert. Dieses ist für den Editor das Verzeichnis, in dem ProVisor die PROLOG–Beispieldateien sucht (→ Kap. 2.2, S. 7).

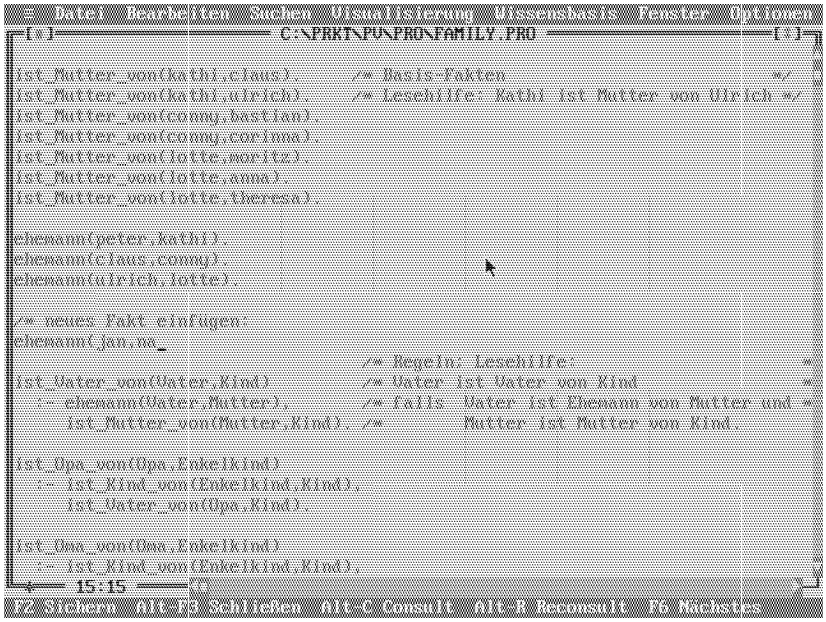

Abbildung 4.4: FAMILY.PRO im Editorfenster

4.4 Konsultieren des Beispielprogramms

Jetzt haben Sie bereits unser FAMILY–Beispiel in den Editor geladen und editiert und vielleicht sogar ihr erstes PROLOG–Programm in dieser Umgebung geschrieben, wollen nun aber allmählich die Visualisierung eines PROLOG–Programms sehen.

Bevor wir Sie diesem Erlebnis aussetzen können — und auch wollen! — , muß der PROLOG-Interpreter den Programmtext *konsultieren*, um die Wissensbasis zu erzeugen oder zu ergänzen. Hierzu drücken Sie einfach auf die Tastenkombination (Alt) (C) und die Meldungen "Lese Datei" und "Consulting" erscheinen nacheinander auf Ihrem Bildschirm. Jetzt können Sie mit der Programmvisualisierung beginnen. Der Interpreter hat nun aus dem Programm eine Wissensbasis erzeugt, auf die er nach Ihren Anfragen zugreift.

! → Wenn Sie eine bereits konsultierte Datei erneut mit (F4) konsultieren (oder aus einem Editorfenster heraus mit der Tastenkombination (Alt) (C)), wird der Inhalt der Datei an den Inhalt der Wissensbasis angehängt. Sie haben damit Ihr zweimal konsultiertes PROLOG–Programm auch zweimal in der Wissensbasis! Dieses führt zu falschen Programmabläufen, da jeder Term bei der sequentiellen Suche des Interpreters zweimal gefunden und jedesmal eine Erüllung versucht wird. Löschen Sie daher in einem solchen Fall die Wissensbasis über den Menüpunkt *Wissensbasis/Löschen* und

laden mit F4 erneut die gewünschte Datei.

Unabhängig von einer Konsultation aus einem Editorfenster heraus können Sie auch ein PROLOG-Programm direkt in die Wissensbasis laden. Drücken Sie hierzu die Taste F4 und es erscheint eine eigene Dateiauswahl. Über diese können Sie nun jede existierende PROLOG-Datei in die Wissensbasis laden.

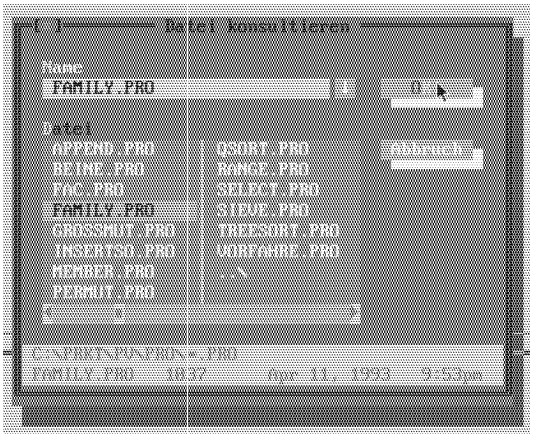

Abbildung 4.5: Dateiauswahl zum Konsultieren

Sie haben aber auch die Möglichkeit, ein bereits geladenes Programm zu konsultieren. Öffnen Sie hierzu, wie im vorhergehenden Abschnitt beschrieben, über die Taste F3 die Dateiauswahl des Editors und selektieren den Namen der zu editierenden Datei. Nachdem Sie Ihre Änderungen vorgenommen haben, müssen Sie nun die Tastenkombination Alt R zum Re–Konsultieren drücken. Damit werden die in der Datei vorgenommenen Änderungen auch in die Wissensbasis aufgenommen. Ein erneutes Konsultieren über Alt C würde dagegen den neuen Programmtext an den alten nur anhängen, nicht aber ersetzen.

4.5 Visualisierung des Beispielprogramms

Nach all dem Editieren, Sichern, Konsultieren und Re–Konsultieren kommen wir nun zur eigentlichen Aufgabe von ProVisor: der Visualisierung. Dieser Abschnitt ist hierzu in zwei Teile untergliedert:

- Im ersten Teil beschreiben wir die Visualisierung des Programmlaufs und
- im zweiten Teil gehen wir auf die Termvisualisierung ein.

4.5.1 Visualisierung des Programmablaufs

Nachdem Sie FAMILY.PRO in die Wissensbasis geladen haben, wechseln Sie in die Eingabezeile des Interpreterfensters. Dazu klicken Sie mit der Maus in die Eingabezeile oder wechseln in das Fenster mit der (F6) –Taste.

! → Ob die Eingabezeile aktiv ist, erkennen Sie am blinkenden Cursor. Ist die Eingabezeile nicht aktiv, erscheint auch kein Cursor in der Zeile.

In unserer schönen Miniwelt aus FAMILY.PRO interessiert uns nun, wer die Eltern der hübschen Theresa sind (→ Kap. 4.1, S. 28). Dazu geben Sie die Anfrage

```
ist_Kind_von(theresa,X).
```

in die Eingabezeile ein. Diese Anfrage hatten wir bereits in der "Einführung in PROLOG" gestellt. Der Funktor "?-", wie Sie ihn vielleicht aus diesem Beispiel in Erinnerung haben, wird durch die Eingabezeile standardmäßig gesetzt. Bestätigen Sie nun Ihre Eingabe mit (Return) und Sie werden Ihren Anfrageterm visualisiert im Hauptfenster sehen (siehe Abb. 4.6).

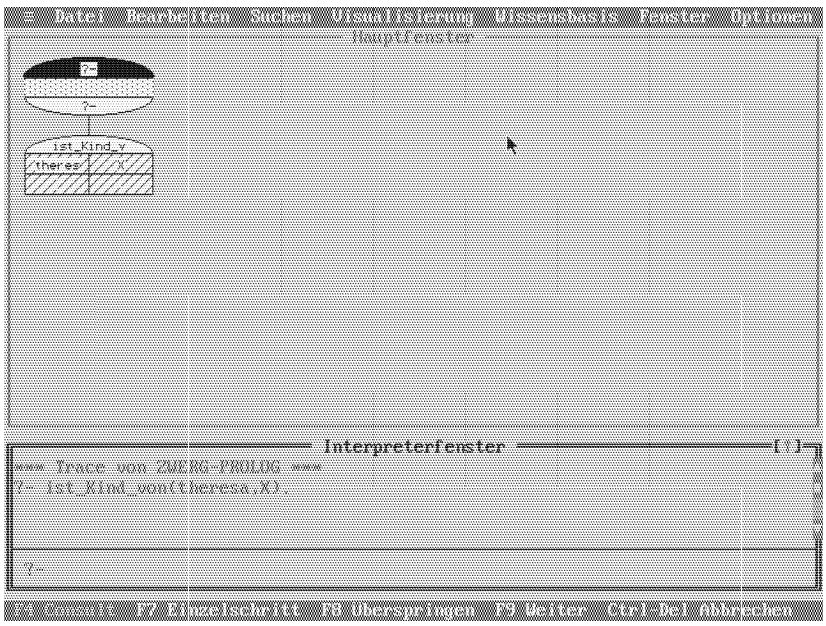

Abbildung 4.6: Visualisierung des Anfrageterms "?-ist_Kind_von(theresa,X)"

Sie haben nun mehrere Möglichkeiten, die Visualisierung fortzusetzen:

- Mit (F7) können Sie den Programmablauf im Einzelschrittmodus verfolgen, d. h. wie der Beweisbaum auf– und wieder abgebaut wird.

- Die Taste F8 ermöglicht das Überspringen ganzer Teilbäume, die sich unterhalb des gerade aktiven Knotens befinden. Der Programmlauf stoppt erst dann, wenn der Knoten, von dem diese Aktion gestartet wurde, wieder erreicht wird. Damit können Sie langwieriges Durchlaufen des Baumes umgehen, wenn Sie z. B. wissen oder vermuten, daß in diesem Teilbaum keine für Sie interessanten Instantiierungen und Unifikationen erfolgen.
- Mit F9 erfolgt erst dann ein Programmstop, wenn der Interpreter eine Lösung gefunden hat oder keine finden konnte. Diese Option ist dann von Nutzen, wenn Sie am Ergebnis und nicht an der Visualisierung interessiert sind.

Neben dem Programmablauf haben Sie die Auswahl verschiedener Darstellungsmodi:

- Im Detailmodus enthalten die Knoten die Namen der Klauselköpfe und die Liste der Argumente. Während des Programmlaufs werden die Variablen durch deren Instantiierungen ersetzt.
- Der Unifikationsmodus erhält die Variablen während eines gesamten Programmlaufs. In einer weiteren Zeile werden zusätzlich die Instantiierungen und Unifikationen der Argumente angezeigt. Der Unifikationsmodus ist im Hauptfenster voreingestellt, kann aber über den Menüpunkt *Visualisierung/Darstellung ändern...* jederzeit geändert werden.
- Der Übersichtsmodus ermöglicht die Gesamtansicht auch größerer Bäume. Dazu werden die Knoten nur sehr klein dargestellt, so daß nur noch graphische Informationen möglich sind.

Graphische Informationen sind unter anderem

- Inverse Darstellung des gerade aktiven Knotens
- Schraffur von Knoten, wenn sie noch nicht besucht sind
- dicke Kanten als Markierung des aktiven Pfades
- durchgestrichene Knoten für Fehlschlag eines (Teil–)Ziels
- Schattenbehaftete Knoten, wenn eine Re–Erfüllung versucht wird

Eine ausführliche Beschreibung finden Sie in Kapitel 4 des Programmhandbuchs.

In den drei möglichen Fenstern für den Programmlauf — einschließlich Hauptfenster — können die aufgeführten Modi beliebig gewählt werden. Ein Wechsel des Darstellungsmodus ist auch während eines Programmlaufs

über den Menüpunkt *Visualisierung/Darstellung ändern...* möglich.

Neben der Visualisierung des Programmablaufs wird parallel im Interpreterfenster das zugehörige Trace–Protokoll ausgegeben. Das Interpreterfenster läßt sich, wie bereits erwähnt, über Alt F5 ausschalten. Auf diese Weise kann man die Darstellungsfläche des Hauptfensters vergrößern. Das Trace–Protokoll wird aber weiterhin mitgeschrieben und läßt sich auch nachträglich über den Puffer des Interpreterfensters oder in einem Editorfenster aus der Datei TRACE.LOG lesen.

Nach soviel Überblick wollen wir nun das Beispiel weiterverfolgen.

Benutzen Sie zunächst den Einzelschrittmodus mit der Taste F7 und sehen, wie der Beweisbaum nach und nach aufgebaut wird. Der aktive Knoten ist so positioniert, daß möglichst alle Söhne in das Fenster passen. Leider ist dies, wie Sie es bereits an unserem Beispiel sehen, nicht immer möglich.

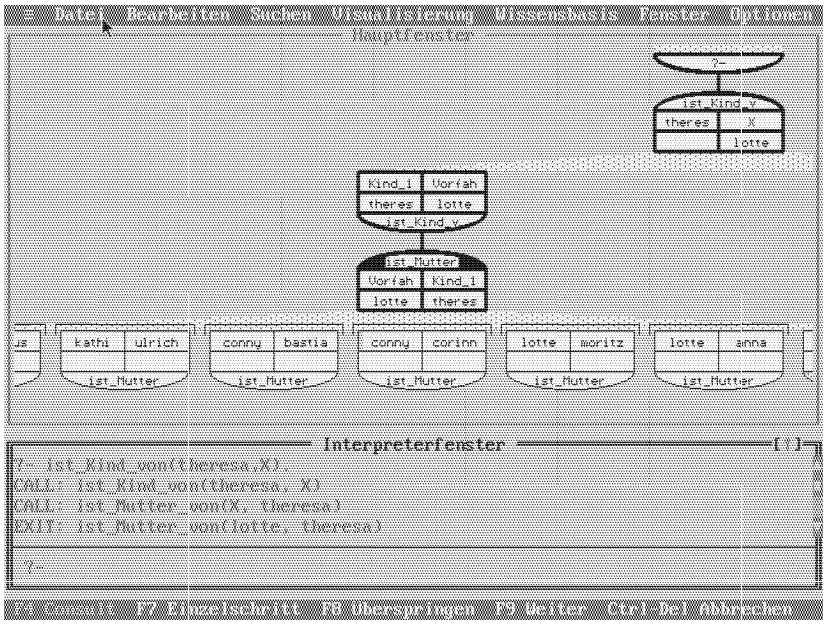

Abbildung 4.7: Expansion des Anfrageterms "?-ist_Kind_von(theresa,X)"

Wenn nun zwischen einem Blatt des Baumes — er repräsentiert das Faktenwissen der Wissensbasis — und dem zugehörigen Vaterknoten keine Instantiierung möglich ist, erhält das Blatt einen Deckel als Zeichen des Fehlschlags. Wie Sie in Bild 4.7 sehen können, konnte z. B. das Fakt "ist_Mutter_von(kathi,ulrich)." nicht mit dem Ziel "ist_Mutter_von(X,theresa)" instantiiert werden. Dieses Fakt erhält daher den besagten Deckel über seinen zugehörigen Knoten.

Kann das Ziel dagegen erfüllt werden, so werden im Detail– und im Unifikationsmodus die Instantiierungen angezeigt und *nach oben* weitergereicht. Ist damit der Anfrageterm erfüllt, erscheint eine Meldung, ob eine weitere Suche fortgesetzt werden soll (→ siehe Abb. 4.8).

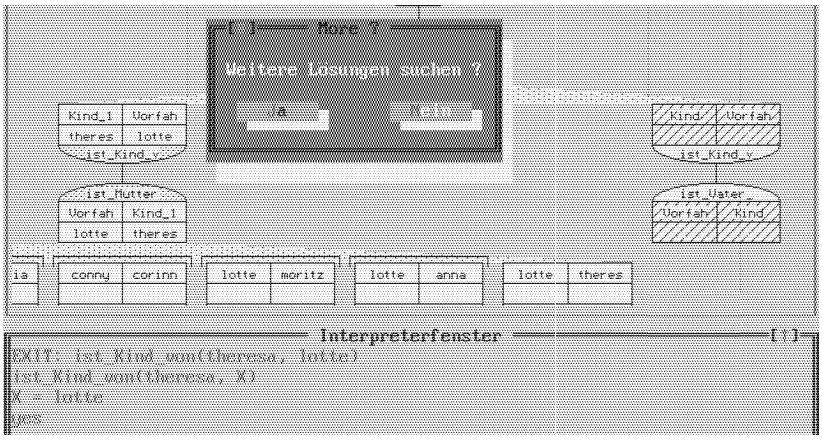

Abbildung 4.8: Erste Lösung des Anfrageterms "?-ist_Kind_von(theresa,X)"

Unser Beispiel erhält als Lösung auf die Anfrage "ist_Kind_von(theresa,X)"

```
X=lotte
```

Wenn Sie die Suche mit F7 fortsetzen, sehen Sie, wie der Interpreter eine Re–Erfüllung im linken Teilbaum versucht, aus dem er die erste Erfüllung "X = lotte" erhielt. Die Knoten auf dem aktiven Pfad werden dabei zur Kennzeichnung eines *REDO* mit einem Schatten versehen (→ siehe Abb. 4.9).

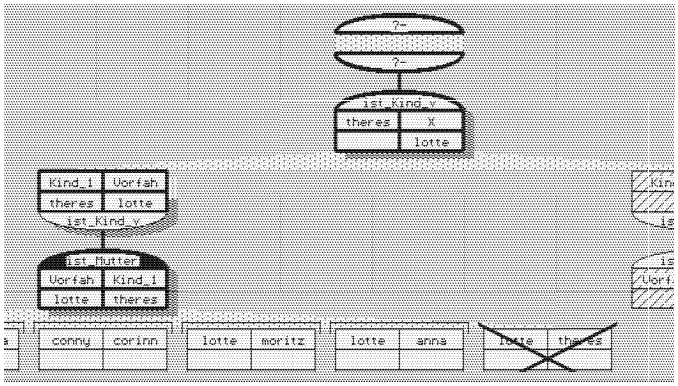

Abbildung 4.9: Versuch der Re–Erfüllung "?-ist_Kind_von(theresa,X)"

Da dies, wie wir bereits wissen (→ Kap. 4.1, S. 28), fehlschlagen muß, wird der Knoten, der das Fakt "ist_Mutter_von(lotte,theresa)." repräsen-

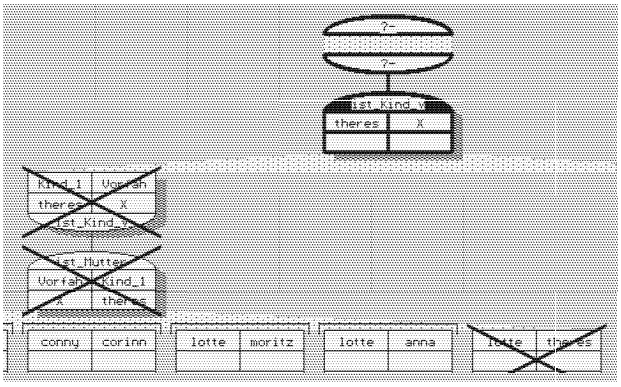

Abbildung 4.10: Fehlschlag der Re–Erfüllung "?-ist_Kind_von(theresa,X)"

tiert, "durchgestrichen" (→ siehe Abb. 4.10).

Auf dem Weg zurück zur Wurzel werden dabei alle Instantiierungen und Unifikationen aufgehoben. Der Interpreter versucht nun eine Erfüllung des Ziels mit dem rechten Nachfolger der Wurzel. Schalten Sie nun einmal in den Übersichtsmodus. Dazu öffnen Sie über den Menüpunkt *Visualisierung/Übersichtsbaum* ein weiteres Fenster und vergrößern dies mit F5 oder wählen einen anderen Darstellungsmodus für das Hauptfenster.

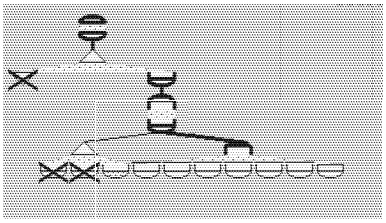

Abbildung 4.11: Übersichtsmodus des Beweisbaums zu "?-ist_Kind_von(theresa,X)"

Sie erhalten nun den vollständigen Beweisbaum in Ihrem Fenster. Wenn Sie den Programmlauf mit F7 fortsetzen, können Sie beobachten wie der Beweisbaum ständig auf– und wieder abgebaut wird, bei dem Versuch eine weitere Lösung zu finden (→ siehe Abb. 4.11).

Schließlich findet der Interpreter die erwartete Lösung

```
X=ulrich
```

und eine weitere schlägt bekanntlich fehl.

Wenn Sie nun in den oberen Teil des Interpreterfensters wechseln, können Sie sich das Trace–Protokoll betrachten:

```
*** Trace von ZWERG-PROLOG ***
CALL: ist_Kind_von(theresa, X)
CALL: ist_Mutter_von(X, theresa)
EXIT: ist_Mutter_von(lotte, theresa)
EXIT: ist_Kind_von(theresa, lotte)
ist_Kind_von(theresa, X)
X = lotte
yes
REDO: ist_Kind_von(theresa, lotte)
query aborted!
no
?- ist_Kind_von(theresa,X).
CALL: ist_Kind_von(theresa, X)
CALL: ist_Mutter_von(X, theresa)
EXIT: ist_Mutter_von(lotte, theresa)
EXIT: ist_Kind_von(theresa, lotte)
ist_Kind_von(theresa, X)
X = lotte
yes
REDO: ist_Kind_von(theresa, lotte)
REDO: ist_Mutter_von(lotte, theresa)
FAIL: ist_Mutter_von(X, theresa)
CALL: ist_Vater_von(X, theresa)
CALL: ehemann(X, Mutter_1)
EXIT: ehemann(peter, kathi)
CALL: ist_Mutter_von(kathi, theresa)
FAIL: ist_Mutter_von(kathi, theresa)
REDO: ehemann(peter, kathi)
EXIT: ehemann(claus, conny)
CALL: ist_Mutter_von(conny, theresa)
FAIL: ist_Mutter_von(conny, theresa)
REDO: ehemann(claus, conny)
EXIT: ehemann(ulrich, lotte)
CALL: ist_Mutter_von(lotte, theresa)
EXIT: ist_Mutter_von(lotte, theresa)
EXIT: ist_Vater_von(ulrich, theresa)
EXIT: ist_Kind_von(theresa, ulrich)
ist_Kind_von(theresa, X)
X = ulrich
```

```
yes
REDO: ist_Kind_von(theresa, ulrich)
REDO: ist_Vater_von(ulrich, theresa)
REDO: ist_Mutter_von(lotte, theresa)
FAIL: ist_Mutter_von(lotte, theresa)
REDO: ehemann(ulrich, lotte)
FAIL: ehemann(X, Mutter_1)
FAIL: ist_Vater_von(X, theresa)
FAIL: ist_Kind_von(theresa, X)
no
```

Lassen Sie sich dieses Protokoll ruhig einmal ausdrucken und wiederholen Sie die Suche nach Theresa's Eltern. Sie werden anhand der Visualisierung dieses Protokoll sicherlich leicht nachvollziehen können. Stellen Sie ruhig weitere Anfragen an unsere Miniwelt, z. B. nach unserem frisch vermählten Ehepaar Natascha und Jan. Ob sie bereits Kinder haben?

4.5.2 Visualisierung einzelner Terme

Nun interessiert Sie vielleicht nicht so sehr der Programmlauf selbst, sondern die Strukturen in PROLOG. Hierfür ist die Termvisualisierung vorgesehen:

- Bei der statischen Termvisualisierung geben Sie einen beliebigen Term ein und im Graphikfenster wird der zugehörige Termbaum ausgegeben.
- Mit der dynamischen Termvisualisierung können Sie die Instantiierungen im Termbaum während eines Programmlaufs verfolgen.

4.5.2.1 Statische Termvisualisierung

Zur statischen Termvisualisierung geben Sie die Tastenkombination Alt V, T ein und es wird ein Menü zur Termeingabe geöffnet (→ siehe Abb. 4.12). In die Eingabezeile können Sie z. B. folgenden Term eingeben:

```
ist_Vater_von(Vater,ist_Kind_von(Kind,Vorfahr)).
```

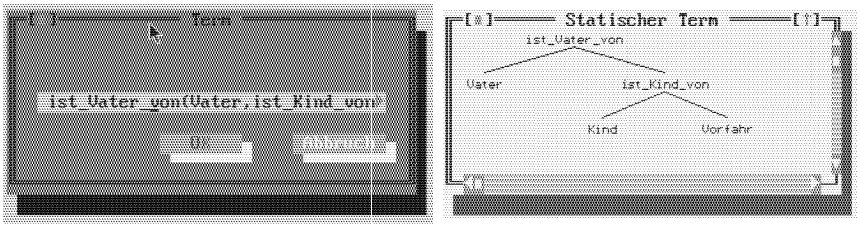

Abbildung 4.12: Termeingabe und die zugehörige Baumdarstellung

Sie erhalten dann eine Darstellung wie in Abbildung 4.12.

Die statische Termdarstellung können Sie jederzeit wählen.

4.5.2.2 Dynamische Termvisualisierung

Die dynamische Termdarstellung erhalten Sie nur während eines Programmlaufs.

Hierzu wechseln Sie in das Graphikfenster. Sie können dann mit der Maus einen Knoten oder dessen Argumente selektieren. Durch Anklicken mit der Maustaste erscheint die Termdarstellung in einem eigenen Fenster.

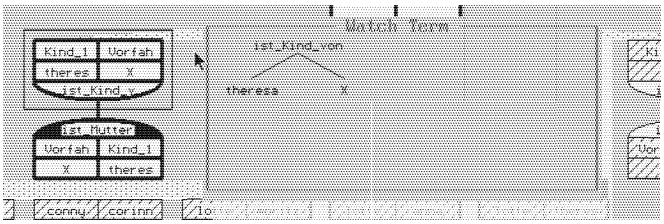

Abbildung 4.13: Termeingabe für dynamische Termdarstellung

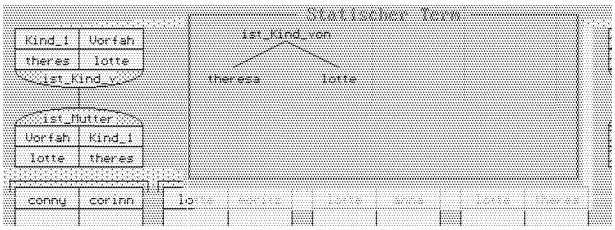

Abbildung 4.14: Dynamische Termdarstellung mit Instantiierung

Über die Tastatur erreichen Sie dies, indem Sie mit gedrückter (Shift)–Taste und eine der Cursortasten einen Knoten auswählen. Um den selektierten Knoten erscheint als Markierung ein eckiger Rahmen (→ siehe Abb. 4.13). Wenn Sie nun (Return) drücken, öffnet sich ebenfalls das Termfenster mit der gewünschten Darstellung. Um ein Argument auszuwählen, drücken Sie solange (Tab) bzw. (Shift) (Tab) bis das gewünschte Argument invertiert dargestellt wird. Wenn Sie den Programmlauf fortsetzen, werden Variablen instantiiert, sofern es eine Lösung gibt.

Anhang A. Glossar

aktiver Pfad:	Der aktive Pfad besteht aus allen Knoten und Kanten, die zwischen der Wurzel des Baums (der Anfrage) und dem aktuellen Knoten (s.d.) liegen.
aktiver, aktueller Knoten:	Der Knoten, der laut PROLOG-Fahrplan als nächstes eine Änderung erfährt (z.B. Expandierung).
Backtracking:	Zurückgehen zu einem Ziel im Beweisbaum, um dieses neu zu erfüllen. Tritt ein, wenn ein Ziel fehlgeschlagen ist.
Breakpoint:	Ein Breakpoint ist eine Situation, in der der PROLOG-Interpreter stoppt, wenn eine angegebene Bedingung erfüllt ist. Bei PROLOG-Interpretern werden Breakpoints durch PROLOG-Terme angegeben; es wird gestoppt, wenn dieser Term als Ziel auftaucht.
Disjunktion:	Ist eine Bedingung erfüllt, wenn mindestens eine Teilbedingung erfüllt ist, spricht man von Disjunktion. Diese entsprechen im Visualisierungskonzept den ODER-Knoten.
Expandieren eines Knotens:	Wenn ein ODER-Knoten expandiert wird, wird das gesamte Prädikat auf einmal angezeigt (im Gegensatz zu konventionellen Trace-Protokollen, die klauselorientiert arbeiten). UND-Knoten können in diesem Sinn nicht expandiert werden.
Fakt, Faktum:	Ein Eintrag in die PROLOG-Wissensbasis, der kein ": –" enthält. Matcht ein Faktum, ist das aktuell bearbeitete Teilziel sofort erfüllt.
Instantiierung:	Während der Durchsuchung der Wissensbasis durch den PROLOG-Interpreter werden Variablen durch Terme ersetzt. Dieser Prozeß wird Instantiierung genannt.
Klausel:	Klausel ist der Oberbegriff für Fakten und Regeln. Die Einträge in die Wissensbasis sind Klauseln.
Klauselkopf:	Bezeichnet das erste Argument des Systemprädikats ": –".
Klauselrumpf/–körper:	Bezeichnet das zweite Argument des Systemprädikats ": –".
Konjunktion:	Eine Konjunktion ist erfüllt, wenn alle Teilbedingungen erfüllt sind. Diese entspricht einem UND-Knoten.
Matchen:	Ein PROLOG-Ziel "matched" mit einer Klausel, wenn der Klauselkopf mit dem Ziel unifizierbar ist.

ODER-Knoten:	Im Zusammenhang mit der Visualisierung: eine unten flache Halbellipse. (vgl. Disjunktion)
Prädikat:	Ein Prädikat besteht aus der Menge der Klauseln mit gleichem Funktor und gleicher Stelligkeit.
Regel:	Ein Ziel, das mit einer Regel matcht, ist nur erfüllt, wenn alle Teilziele erfüllbar sind, die im Rumpf der Regel aufgeführt sind.
Struktur:	Eine PROLOG-Struktur besteht aus einem Funktor und Argumenten. Ist die Stelligkeit null, hat die Struktur keine Argumente und ist somit ein Atom.
Term:	Oberbegriff von Struktur, Variable und Integer.
UND-Knoten:	Im Zusammenhang mit der Visualisierung: eine oben flache Halbellipse. (vgl. Konjunktion)
Unifikation:	Prozeß der "Gleichmachung" von Termen, u.U. durch Instantiierung von Variablen.
Wissensbasis:	Mit diesen Begriffen wird die interne Daten-/Programm-Repräsentation des Interpreters bezeichnet, die aus Klauseln besteht, auf die während des Inferenzmechanismus zur Ableitung der Anfrage zugegriffen wird.

Literaturverzeichnis

[BRA87] Bratko, I .: PROLOG, Addison-Wesley, 1. Aufl. 1987

[CLO87] Clocksin, W. J. und Mellish, C. S. : Programming in PROLOG, Springer, 3. Aufl 1987

[KLE86] Kleine, Büning, Schmitgen: PROLOG, Teubner 1986

[NIL81] Nillson, N. J. : Principles of Artificial Intelligence, Tioga 1981

[ROE92] Röhner, G. : PROLOG, Lehrerweiterbildung Informatik, HIBS, HILF

[SHA88] Shapiro, E. und Sterling, L. : The Art of PROLOG, MIT-Press