Benutzerhandbuch und Dokumentation eines Experimentalsystems auf der Basis der Expertensystemschale knoX

Theo Held & Barbara Maier-Schicht

Bericht Nr. 86

Dezember 1994

Arbeiten aus dem Sonderforschungsbereich 245
"Sprache und Situation", Heidelberg/Mannheim

Psychologisches Institut
der Universität Heidelberg
Hauptstr. 47-51
69117 Heidelberg

Die Arbeit entstand innerhalb des Teilprojektes A5 (Projektleiterin Prof. Dr. M. Wintermantel) des Sonderforschungsbereichs 245 "Sprache und Situation" an der Universität Heidelberg. Wir danken der Deutschen Forschungsgemeinschaft für die Förderung unserer Arbeiten.

ISSN 0941/990X

Inhaltsverzeichnis

Vorbemerkung

In diesem Papier wird die Realisierung eines Experimentalsystems auf der Basis der Expertensystemschale knoX (Theiss, 1992) für potentielle Nutzer des Systems dokumentiert. Wir geben einen Überblick über den aktuellen Status (22. Dezember 1994) des Experimentalsystems und beschreiben seine wesentlichen Eigenschaften. Außerdem soll diese Dokumentation die Grundlage für weitere Arbeiten mit dem System (Benutzung und Weiterentwicklung des Systems oder auch die Portierung der Wissensbasis in eine andere Softwareumgebung) darstellen.

Für eine genauere Beschreibung der Expertensystemschale und deren expertensystemspezifischen Funktionen wird auf das Benutzerhandbuch von Theiss (1992) bzw. auf die Dokumentation des Prolog-Systems, unter dem knoX erstellt wurde, verwiesen.

Zur Information über die Anwendung des Experimentalsystems im Teilprojekt A5 des Sonderforschungsbereichs 245 dient das Papier von Held, Laier & Fries (1994). Hier wird der theoretische Hintergrund der beiden bisherigen Untersuchungen und der Untersuchungsgegenstand ausführlich erläutert.

Um einen Überblick über die Möglichkeiten des Systems zu bekommen und vorhandene Experimentalprogramme anwenden zu können, genügt die Lektüre der Kapitel 1 und 2. Das folgende Kapitel 3 ist dann relevant, wenn Änderungen an einem Experimentalprogramm vorgenommen werden sollen, oder wenn eine neue Untersuchung gestaltet werden soll. Die folgenden Konventionen der Textgestaltung sind zu beachten:

- Programmcode, Bildschirmausgaben und Dateinamen wie z. B. `v0main.txt` werden im `Typewriter-Font` dargestellt.
- Menüpunkte auf die im Text verwiesen wird, werden wie z. B. Bearbeiten im Sansserif-Font dargestellt.
- Codesegmente und Beispiele für Ausgabedateien werden `umrahmt` dargestellt.

1

Einführung und Überblick

Eine notwendige Voraussetzung für die Konstruktion rechnerbasierter Instruktionssysteme ist Wissen über Vorgehensweisen und Strategien derer sich Novizen oder andere zu instruierende Personen beim Informationsabruf bedienen. Zur Exploration solcher „Wege durch eine Wissensdomäne" wird im Teilprojekt (TP) A5 des Sonderforschungsbereichs 245 „Sprache und Situation" an der Universität Heidelberg die Expertensystemschale „knoX" (Theiss, 1992) eingesetzt. Mit Hilfe dieses Systems werden die für das TP relevanten Wissensbasen erstellt und verwaltet. Außerdem wurden unter knoX die für die Untersuchungen erforderlichen Benutzerschnittstellen realisiert.

Definition Expertensystem

Generell versteht man unter einem *Expertensystem*[1] ein „Programmsystem, das 'Wissen' über ein spezielles Gebiet speichert und ansammelt, aus dem Wissen Schlußfolgerungen zieht und zu konkreten Problemen des Gebietes Lösungen anbietet (künstliche Intelligenz)" (Engesser, 1988, S. 222).

Expertensystemschale

Durch die Verwendung einer *Expertensystemschale* wie knoX wird die Entwicklung eines Expertensystems erheblich erleichtert, da Inferenz- und Erläuterungsmechanismen, die dem Ziehen von Schlußfolgerungen und der Lösungsfindung dienen in einer universell einsetzbaren Form zur Verfügung gestellt werden. Die Entwicklung des Expertensystems kann sich somit auf den Aufbau der Wissensbasis beschränken (Theiss, 1992).

Zentrale Eigenschaften eines Expertensystems

Eine wesentliche Eigenschaft eines Expertensystems ist also, daß mit Hilfe gegebener Inferenzmechanismen aus dem in einer Wissensbasis enthaltenen Wissen und aus Informationen, die vom Benutzer während der Konsultation der Wissensbasis gegeben werden *Fakten* abgeleitet werden, die dem Erreichen eines vorgegebenen Zieles der Konsultation dienen. Ein solches Ziel kann z. B. sein, für einen Bankkunden die optimale Anlageform herauszufinden (siehe zu diesem Beispiel Theiss, 1992). In diesem Fall liegen allgemeine Informationen zu

[1] Eine ausführliche Darstellung der Entwicklung des Experimentalsystems mit Hilfe der Expertensystemschale knoX und der potentiellen Verwendung von typischen Expertensystem-Eigenschaften für die nächste Entwicklungsstufe des Experimentalsystems finden sich in einer Arbeit von G. Theiss und B. Maier-Schicht, die z. Zt. am SFB 245 entsteht.

den Anlagearten in der Wissensbasis vor, während kundenspezifische Informationen vom Benutzer eingegeben werden müssen. Das Expertensystem verfügt nun über Regeln, mit deren Hilfe das Ziel, nämlich die Identifikation der optimalen Anlageform, erreicht werden kann.

Abgrenzung der Verwendung von knoX zu Experimentalzwecken

Allgemein hat die Verwendung von knoX für das hier zu beschreibende Experimentalsystem mit dem eigentlichen Sinn und der Verwendung von Expertensystemen nur wenig gemein. Im Prinzip werden nur die Möglichkeiten zur Repräsentation einer Wissensbasis und des Abrufs von Elementen der Wissensbasis genutzt. Es werden zwar durchaus Basismechanismen zur Inferenz und Erläuterung genutzt, anwendungsspezifische Inferenz- und Erläuterungsmechanismen müssen zur vorliegenden Anwendung (vgl. Held et al., 1994) jedoch erst noch hinzugefügt werden.

Selbstverständlich stellt knoX nur eine von mehreren alternativen Möglichkeiten für die Organisation der Wissensbasis und die Realisierung einer Benutzerschnittstelle für den Wissensabruf dar. Prinzipiell eignen sich dafür z. B. auch die meisten handelsüblichen Hypertext- und Hypermediasysteme. Der Grund für die Verwendung von knoX zur Realisierung des Experimentalsystems liegt primär in der *potentiellen* Verwendung von Eigenschaften, die spezifisch für ein Expertensystem sind. Für den Fall, daß das System durch Funktionen zum adaptiven Instruieren ergänzt werden soll, ist der Einsatz einer Expertensystemschale natürlich gerechtfertigt.

In den folgenden Abschnitten werden wir zunächst auf die Benutzung des Systems eingehen (Kap. 2). Unter der Annahme, daß knoX vollständig auf einem Rechner installiert ist und die für die Versuchssteuerung erforderlichen Programm-Module in kompilierter Form vorliegen, wird beschrieben, wie knoX und die Versuchssteuerung gestartet werden und wie die wesentlichen Schritte der Bedienung aussehen.

Kapitel 3 enthält alle Informationen, die für eine Modifikation und Anpassung des Systems erforderlich sind. Es wird hier auch näher auf die Hardware-Voraussetzungen für den Betrieb von knoX und die Installation eingegangen.

2

Benutzung des Systems

2.1 knoX starten und beenden

Um knoX starten zu können, muß das Programm, wie in Abschnitt 3.1 beschrieben, installiert sein. Außerdem sollte der Pfad so gesetzt sein, daß alle benötigten Komponenten erreichbar sind. Weiterhin sind die Angaben zum Booten des Rechners für knoX zu beachten. Üblicherweise ist das komplette System im Verzeichnis `KNOX` installiert. knoX wird innerhalb dieses Verzeichnisses mit dem Kommando `knox` gestartet:

```
C:\KNOX> knox
```

Nach einer kurzen Wartezeit erscheint eine Oberfläche wie in Abb. 2.1 (oben) am Bildschirm.

knoX kann weitgehend mit Hilfe der Maus bedient werden. Der aktuell angewählte Menüpunkt erscheint invertiert. Durch Bewegen der Maus nach rechts oder links können die einzelnen Punkte der Menüleiste angewählt werden. Ein Klick mit der linken Maustaste bewirkt, daß ein Menü geöffnet wird. Die Unterpunkte des Menüs werden durch eine vertikale Bewegung der Maus selektiert. Die Aktivierung des Menüpunktes erfolgt wiederum mit der linken Maustaste. Alternativ kann knoX auch mit Hilfe der Tastatur bedient werden. Die Cursor-Tasten dienen dabei zur Markierung der Menüs bzw. der Menü-Unterpunkte. Mit der Return-Taste werden die markierten Stellen ausgewählt. Die Abbildung 2.1 (oben) zeigt das geöffnete Menü **Datei** mit dem angewählten Menüpunkt **Laden**. Beendet wird knoX über den Menüpunkt **Beenden** im Menü **Datei** (siehe Abb. 2.1 (oben). Vor dem Beenden erscheint eine Rückfrage, ob knoX wirklich verlassen werden soll. Wird diese Frage mit ja beantwortet, gelangt man nach dem Abbruch von knoX zurück auf die DOS-Kommandoebene. Von hier aus kann knoX neu gestartet werden.

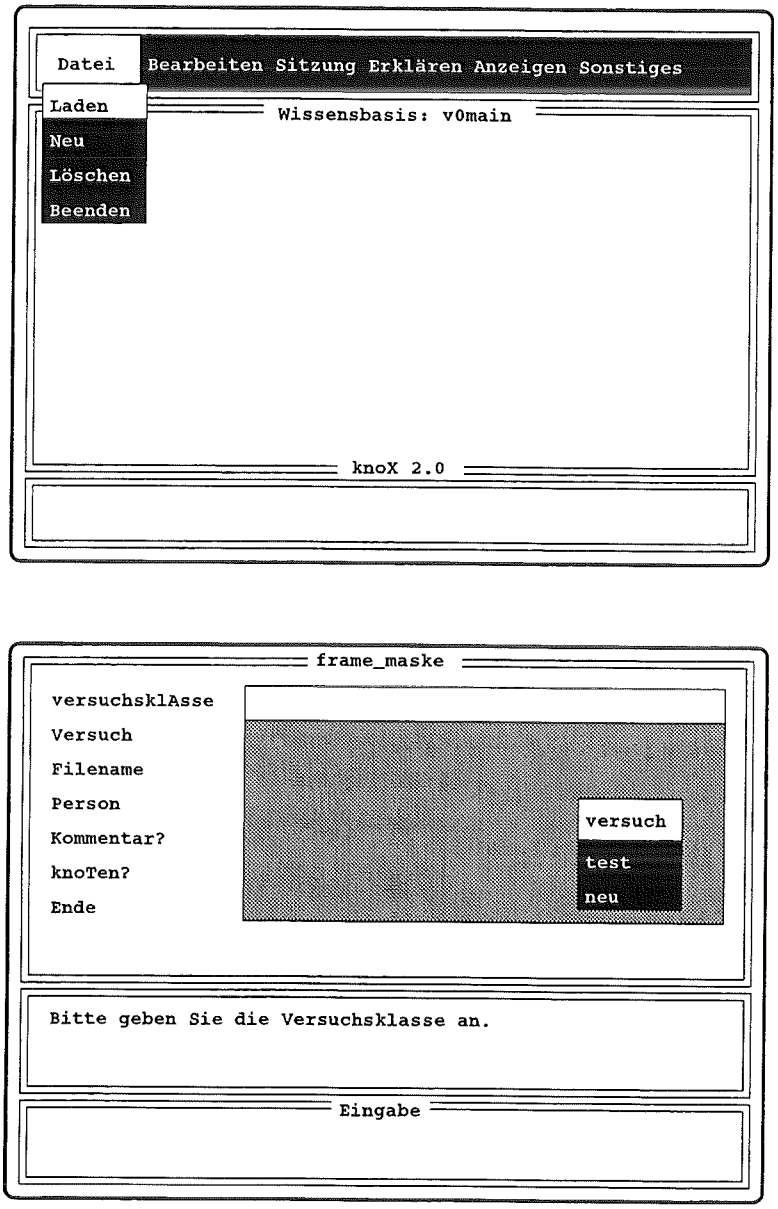
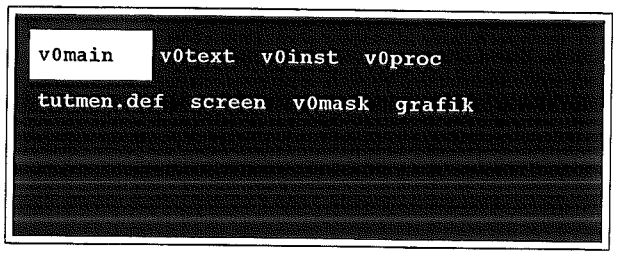


Abbildung 2.1. Start-Bildschirm von knoX (oben) und Maske für die Versuchs-Parameter (unten).

2.2 Start einer Versuchssitzung

Nachdem knoX gestartet ist, kann eine Versuchssitzung initialisiert werden. Dies geschieht über das Laden und Starten der für die Sitzung relevanten Haupt-Datei `v0main*`[1]. Über die entsprechende Main-Datei werden alle weiteren Module geladen.

Zum Aufruf von `v0main*` wird im Menü Sitzung das Item Start/Weiter gewählt. Anschließend erscheint ein Fenster mir einer Auswahl an Dateien, die in vorangegangenen Sitzungen schon einmal geladen worden sind:



Ist die gewünschte (Main-)Datei in dieser Auswahl, so muß sie nur mit der Maus markiert werden; ein Klick mit der linken Taste startet die Sitzung. Falls die Datei nicht im Auswahlfenster enthalten ist, muß sie über das Menü Datei mit dem Menüpunkt Neu geladen werden. Ist die Datei geladen, so kann die Sitzung wie oben beschrieben gestartet werden.

Nach einigen Meldungen zur internen Überprüfung der Wissensbasis erscheint ein Formular in das einige Angaben zur Versuchssitzung eingetragen werden müssen (siehe Abb. 2.1 (unten)). Diese Angaben sind maßgeblich für den Versuchsablauf; sie werden in einer Protokolldatei (s. u.) abgespeichert.

Angaben zur Versuchssteuerung

Für die `Versuchsklasse` steht eine Auswahl in Form eines Menüs zur Verfügung. Bei den Experimentalanwendungen steht die Versuchsklasse `versuch` an erster Stelle. Dieses Item muß gewählt werden. Unter dem nächsten Punkt `Versuch` wird die Kodierung des laufenden Versuchs eingetragen (nicht der Vp-Code!). Diese Kodierung wird in einer Protokolldatei abgespeichert.

Mit `Filename` gibt man den Namen der Protokolldatei an. Es wird nicht kontrolliert, ob ein bereits vorhandener Dateiname angegeben wird. Existierende Dateien werden nicht überschrieben, das neue Protokoll wird an die schon bestehende Datei angehängt. Anschließend muß unter `Person` der Vp-Code eingegeben werden. Auch diese Angabe wird in der Protokolldatei abgespeichert. Es wird nicht überprüft, ob ein Vp-Code doppelt vergeben wurde.

[1] Die Namen der meisten Programmteile, die für das Experimentalsystem relevant sind, setzen sich stets aus drei Teilen zusammen. Der erste Teil ist in allen Fällen „`v0`“ (steht für Version 0), der zweite Teil ist einer der Modulnamen „`main`“, „`text`“, „`proc`“, „`inst`“ oder „`mask`“. Der dritte Teil spezifiziert das experimentelle Setting innerhalb dessen das Modul verwendet wird. Dieser Teil wird im Text nur als „`*`“ dargestellt, wenn die Angaben für die Module aller Settings gelten. Ein vollständiger Dateiname ist z. B. „`v0mainah`“ für die Hauptdatei des Settings „Aufbau“ mit „Handlung“ als erstem Menüpunkt im Startmenü.

Die drei folgenden Felder des Formulars werden automatisch ausgefüllt, sofern die Versuchsklasse `versuch` gewählt worden ist. Die Felder `Kommentar` und `knoTen` sind jeweils mit der Zeichenkette `nein` belegt. Dies bedeutet, daß während einer Versuchssitzung keine Kommentare[2] zu den einzelnen Informationstexten eingegeben werden können und daß die internen Kodierungen[3] der Informationstexte (siehe Abschnitte 3.3 und 3.4) nicht am Bildschirm angezeigt werden. Das Feld `Ende` ist mit dem Eintrag `normal` belegt. Dies bedeutet, daß die Versuchssitzung nicht automatisch nach einer vorgegebenen Zeitspanne abgebrochen wird. Alternativ können in dieses Feld die Werte `15`, `30` und `45` (jeweils für einen Zeitraum in Minuten) eingegeben werden.

Sind alle Felder ausgefüllt, fragt das Programm nach, ob alle Angaben korrekt sind. Wird diese Frage verneint, folgt die Frage, welcher der Einträge geändert werden soll. Das zu bearbeitende Feld wird mit dem Großbuchstaben im Feldnamen angegeben, d.h. daß z.B. zur Änderung der `versuchsklAsse` ein „`A`“ eingegeben werden muß. Wenn der Buchstabe eingegeben ist, wird der jeweilige Feldinhalt gelöscht und eine neue Angabe kann eingetragen bzw. aus einem Menü ausgewählt werden. Wenn alle Werte korrekt sind und die entsprechende Frage bejaht worden ist, wird das Formular verlassen. Anschließend erscheint der Start-Bildschirm des Experimentalprogramms. Die folgenden Punkte fassen die Vorgehensweise zum Starten des Experimentalprogramms zusammen:

1. knoX starten: `C:\KNOX> knox`
2. Main-Datei laden (z.B. `v0mainah`)
3. Versuchsklasse `versuch` im Formular auswählen
4. Namen der Protokolldatei in das Formular eingeben
5. Versuchspersonen-Code in das Formular eingeben
6. Bei Bedarf Einträge im Formular ändern (durch Eingabe der *Großbuchstaben* in den einzelnen Feldbezeichnungen)
7. Konsultation der Wissensbasis beginnen

[2]Die Kommentare sind als Unterstützung bei der Entwicklung der Wissensbasis gedacht. Wenn das Feld `Kommentar` mit `ja` ausgefüllt wurde, erscheint ein Eingabefeld für Kommentare am unteren Bildschirmrand. Im Laufe der Systemtests können z.B. Änderungsvorschläge zu Informationstexten eingeben werden. Die Kommentare werden in der Protokolldatei abgespeichert und können nach Beendigung einer Sitzung ausgewertet werden.

[3]Wenn das Feld `knoTen` mit `ja` ausgefüllt wurde, erscheinen die Kodierungen der Knoten zusammen mit den Textnummern der Informationstexte über den Texten am Bildschirm. Dies ist zur Überprüfung der korrekten Auswahl des nächsten Knotens während des Systemtests gedacht.

2.3 Bedienung des Experimentalprogramms

Die Wissensbasis ist dem Untersuchungsgegenstand (vgl. Held et al., 1994) entsprechend hierarchisch organisiert und in drei *Haupthierarchien* unterteilt: (1) die Theoriehierarchie, (2) die Apparaturhierarchie und (3) die Handlungshierarchie. Diese Hierarchien werden von *hierarchischen Verknüpfungen* zwischen den einzelnen Informationseinheiten gebildet. Solche Verknüpfungen sind durch „präziser"-, „allgemeiner"- und „weiter"-Beziehungen zwischen Informationseinheiten gegeben. Untereinander sind die Hierarchien durch *assoziative Verknüpfungen*[4] verbunden, die es z. B. ermöglichen, theoretische Informationen abzurufen, die in Bezug zu einer bestimmten Handlung oder zu einem Apparaturteil stehen. In Abb. 2.2 ist ein Ausschnitt aus der Wissensbasis in graphischer Form dargestellt. Durchgezogene Linien stehen für hierarchische Verbindungen zwi-

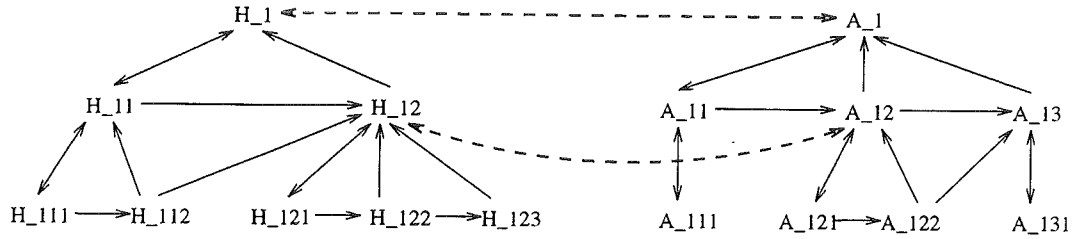


Abbildung 2.2. Beispiel für die hierarchisch-assoziative Organisation der Wissensbasis

schen den einzelnen Informationseinheiten. Gestrichelte Linien zeigen assoziative Verbindungen. Die Pfeilspitzen an den Verbindungslinien geben die möglichen Richtungen der Navigation durch die Wissensbasis an. Im gezeigten Beispiel sind zwei Teilhierarchien aus den Haupthierarchien „Handlung" (H-Knoten) und „Apparatur" (A-Knoten) zu sehen. Die Versuchspersonen haben in diesem Beispiel die Möglichkeit von der Informationseinheit H_12 aus folgende Informationseinheiten zu erreichen:

- die *allgemeinere* Einheit H_1
- die *präzisere* Einheit H_121 (H_122 und H_123 können nur über H_121 erreicht werden, jedoch nicht direkt von H_12 aus)
- die *apparaturspezifische* Einheit A_12, deren Information für die in H_12 beschriebene Handlung relevant ist
- die jeweils *zurückliegende* Einheit (also die Einheit, die zeitlich vor H_12 betrachtet worden ist)

[4]Da die Wissensbasis durch ihre Strukturierung im Prinzip ein Hypertext ist, werden für die Charakterisierung der Verknüpfungen der Informationseinheiten Termini aus diesem Bereich verwendet. Für eine genaue Begriffsdefinition sei auf Kuhlen (1991) verwiesen.

Außerdem besteht an jedem Punkt der Hierarchie die Möglichkeit, zum Anfangsstatus der Konsultation zurückzukehren. Abb. 2.3 zeigt den Bildschirminhalt der Anfangskonfiguration. Durch eine Bewegung der Maus kann jeder der drei

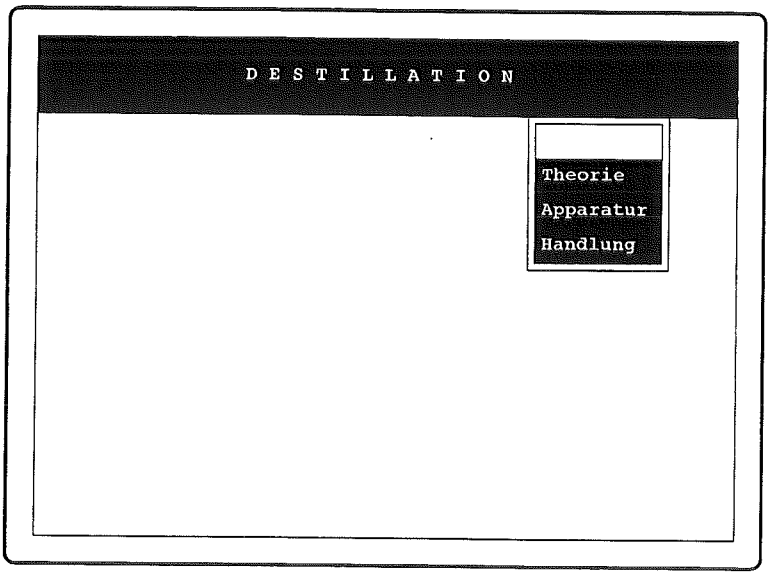


Abbildung 2.3. Bildschirminhalt zu Beginn eines Versuchsdurchgangs

Punkte Theorie, Apparatur und Handlung im Menü angewählt werden. Sobald ein Menüpunkt angewählt worden ist, wird der entsprechende Text invers dargestellt. Vor der Auswahl durch die Versuchsperson ist stets die Auswahl eines Leerfeldes (in Abb. 2.3 über dem Menüpunkt Theorie zu sehen) voreingestellt. Die Vp hat also stets explizit eine Auswahl zu treffen — dies soll verhindern, daß die Auswahl eines bereits markierten Menüpunktes begünstigt wird. Der Informationstext, der einem Menüpunkt zugeordnet ist, wird nach Betätigung der linken Maustaste angezeigt. In Abb. 2.4 ist der Bildschirminhalt zu sehen, der nach Anwahl des Menüpunktes Theorie im Anfangsbildschirm (Abb. 2.3) ausgegeben wird. Da dieser Teil des theoriebezogenen Abschnitts der Wissensbasis nur aus einer Sequenz von nebengeordneten Texten besteht, die mit „weiter"-Verknüpfungen verbunden sind, enthält das Menü keine Optionen zur Anwahl allgemeinerer oder präziserer Information. Wird nun in dem in Abb. 2.4 gezeigten Menü der Punkt Handlung gewählt, so gelangt man zu dem in Abb. 2.5 gezeigten Bildschirm. Von dieser Informationseinheit aus kann man nun durch Anklicken des Menüpunktes präziser zu hierarchisch untergeordneter detaillierterer Information gelangen. Im gezeigten Fall würde ein Bildschirm mit folgendem Text ausgegeben:

```
Installieren Sie das Verdampferteil.
```

Dies ist die erste Teil-Handlung, die zum Aufbau der Apparatur erforderlich ist. Eine Vp, die bereits weiß, wie das Verdampferteil installiert wird, kann zur

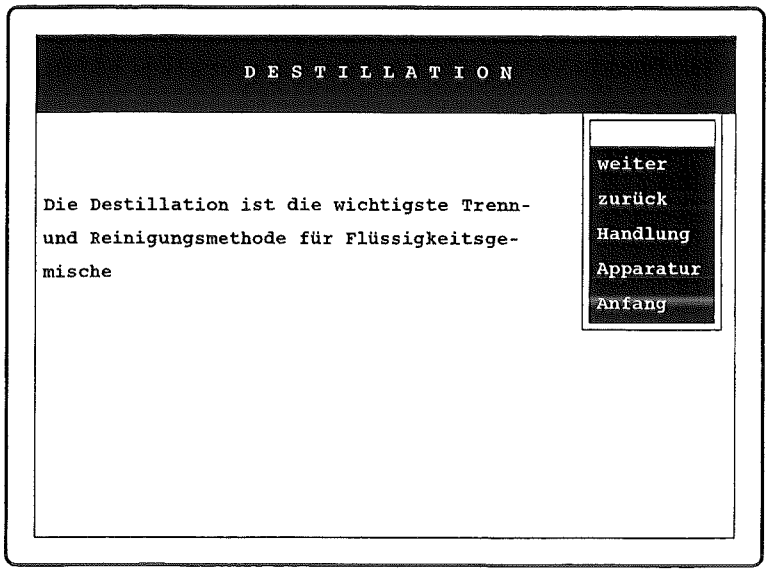


Abbildung 2.4. Der erste Bildschirminhalt zum Menüpunkt „Theorie"

nächsten Teil-Handlung weitergehen. Anderenfalls können präzisere Informationen zu dieser Tätigkeit abgerufen werden.

Der Punkt „nächs.Handl."[5] (siehe Abb. 2.5) führt zum nächsten globalen Handlungsschritt, der auf den Aufbau der Apparatur folgt: in diesem Fall zum Betreiben der Destillationsanlage.

Da der in Abb. 2.5 gezeigte Text in der allgemeinsten Ebene der Wissensbasis liegt, führt die Auswahl des Menüpunktes allgemeiner zur Ausgabe folgender Meldung:

```
Durch die ausgewählte Option 'allgemeiner' gibt es
hier keine Fortsetzung.
Bitte wählen Sie eine andere Option aus.
```

Dieser Typ von Meldung wird immer dann ausgegeben, wenn eine Informationseinheit nicht über eine angewählte Verbindung verfügt. Betrachtet man die in Abb. 2.2 gezeigte Struktur, so gibt es z. B. von der Informationseinheit H_1 aus keine allgemeiner-Verbindung, von H_12 keine weiter-Verbindung und von H_121 keine präziser-Verbindung.

Mit Hilfe des Menüpunktes zurück gelangt man zur vorhergehenden Informationseinheit (siehe im o. g. Fall Abb. 2.4). Durch die Punkte Theorie und Apparatur wird mit dieser Information zusammenhängende apparaturspezifische oder theoretische Information angewählt. Der Menüpunkt Anfang führt stets zu dem in Abb. 2.3 gezeigten Bildschirm zurück. Diese Option dient dazu, ein „Verirren" in der Wissensbasis zu vermeiden. Außerdem sind Informationen aus

[5]Die Abkürzung des Menütextes ist erforderlich, da sonst neben dem Menü zu wenig Raum für die Informationstexte auf dem Bildschirm verbleiben würde.

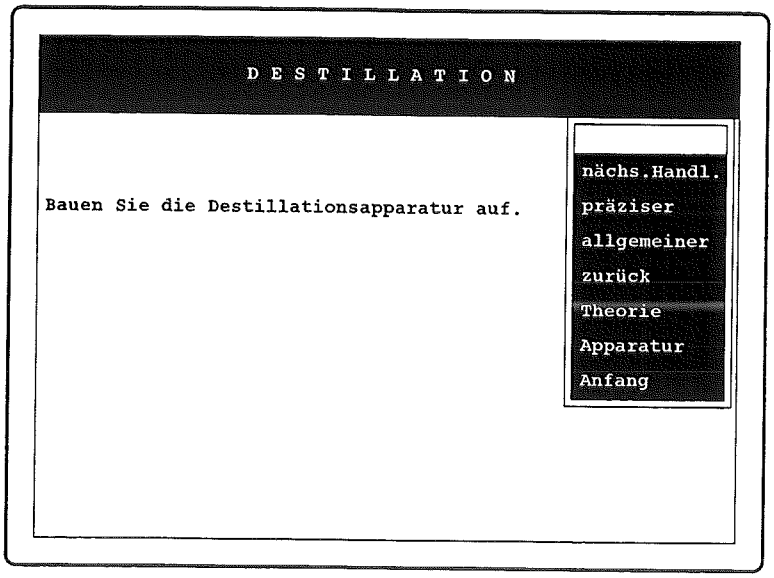


Abbildung 2.5. Bildschirminhalt: Wechsel von „Theorie“ zu „Handlung“

anderen Teilhierarchien über einen solchen Sprung zum Anfang schneller zu erreichen.

Beenden der Versuchssitzung

Da die Versuchsperson keine Möglichkeit haben soll, eine Versuchssitzung selbstständig abzubrechen, wurde kein Menüpunkt zum Beenden der Sitzung vorgesehen. Die Sitzung kann jedoch jederzeit vom Versuchsleiter mit der Tastenkombination

`Strg-Alt-f`

abgebrochen werden, d. h. die Tasten `Strg`, `Alt` und `f` müssen gleichzeitig gedrückt werden.

2.4 Konvertieren der Protokolldatei

Das Experimentalprogramm legt eine Protokolldatei an, mit deren Hilfe der Ablauf der Konsultation rekonstruiert werden kann. Außer den Angaben zum Versuch und zur Versuchsperson, die vor Beginn der Untersuchung in das Start-Formular (siehe Abb. 2.1 (unten)) eingetragen wurden, ist im Protokoll die komplette Sequenz der Informationsabrufe, alle bearbeiteten Knoten und die dazugehörigen Zeitpunkte des Aufrufs von Informationen enthalten. Der Anfang einer Protokolldatei kann z. B. folgendermaßen aussehen:

```
File_name: test\vp07aa.pro
Versuchsklasse: versuch
Versuchsperson: vp07
Versuch: destill-02a
Kommentar möglich? nein
```

```
Knoten ausgeben? nein
Die Sitzung wird beendet :normal
ANFANG DER ERKLAERUNG
Wahl: handlung
AKTUELL: hh0 Zeit: 16:38:49
Textnr.: hh0
ZL: hh0
Wahl: praezis
AKTUELL: hh1 Zeit: 16:38:56
Textnr.: hh1
ZL: hh1
Wahl: praezis
AKTUELL: hh11 Zeit: 16:38:58
Textnr.: hh11
ZL: hh11
Wahl: praezis
AKTUELL: hh111 Zeit: 16:39:5
Textnr.: hh111
ZL: hh111
Wahl: zurueck
AKTUELL: hh11 Zeit: 16:39:13
Textnr.: hh11
Wahl: inter_apparatur
AKTUELL: aa12 Zeit: 16:39:19
Textnr.: aa12
ZL: aa12
Wahl: praezis
AKTUELL: aa121 Zeit: 16:39:37
Textnr.: aa121
ZL: aa121
Wahl: praezis
AKTUELL: aa1211 Zeit: 16:39:46
Textnr.: aa1211
ZL: aa1211
        .
        .
        .
```

Im Kopf der Datei finden sich die allgemeinen Angaben zur Untersuchung. Nach dem Eintrag `ANFANG DER ERKLAERUNG` beginnt das Protokoll des Versuchsablaufs. Der Eintrag `Wahl` gibt an, welchen Menüpunkt die Vp angewählt hat. Im obigen Beispiel hat die Vp also zu Beginn des Versuchsdurchgangs „Handlung“ gewählt. Der aktuell dargebotene Knoten besitzt den Code „hh0“ (`AKTUELL:`). Die Wahl wurde um 16 Uhr 38 und 49 Sekunden getroffen (`Zeit:`). Der dem Knoten zugeordnete Text ist ebenfalls mit „hh0“ kodiert (`Textnr.:`).[6]

[6]In diesem Zusammenhang ist darauf hinzuweisen, daß ein und demselben Knoten unterschiedliche Texte zugeordnet sein können. Dies ist dann der Fall, wenn sogenannte „Zwischentexte“ verwendet werden. Diese Knoten dienen als „Übergangshilfe“ zwischen den Haupthierarchien, indem sie beim Hierarchiewechsel zwischengeschaltet werden. Dies führt zu der Situation, daß bei einem Wechsel von der Handlungs- zur Theoriehierarchie der aktuelle Knoten

Außerdem wird die Knotenbezeichnung „hh0“ in die „Zurückliste“ aufgenommen (ZL:). Die Zurückliste wird abgearbeitet, wenn der Benutzer den zurück-Menüpunkt anwählt. Die nächste Wahl der Vp ist präzis. Damit wird der Knoten hh1 angesteuert, etc.

Da diese Form der Protokollierung nur bedingt als Input für Auswertungs-Software geeignet ist, wurde ein Filter geschrieben, der das Protokoll in ein „handlicheres“ Format bringt und bereits einige Auswertungen durchführt. Das Filter ist ein awk-Skript (siehe Aho, Kernigan & Weinberger, 1988). Um es verwenden zu können, muß das awk-Programm (z. B. im MKS-Toolkit enthalten, auch PD/Shareware-Versionen verfügbar) zur Verfügung stehen. Der Quell-Code des Filters („ausw.awk“) findet sich im Anhang (S. 35). Als weitere Voraussetzung muß eine Datei existieren, in der die Kodierungen aller in der Untersuchung wählbaren Knoten enthalten sind (ein Code pro Zeile). Wenn diese Datei den Namen „v0insta.nod“ hat und die Protokolldatei „vp07aa.pro“ konvertiert und ausgewertet werden soll, dann sieht der Aufruf des Filters folgendermaßen aus:

```
awk -f ausw.awk v0insta.nod vp07aa.pro > vp07aa.erg
```

Die resultierende Datei vp07aa.erg enthält sieben Spalten. Im Einzelnen stehen in diesen Spalten die Wahl der Vp, der dazugehörige Knoten und Text (jeweils die Kodierungen), die laufende Nummer der Wahl, das gewählte Gebiet (1 := Theorie, 2 := Handlung, 3 := Apparatur), der Level innerhalb der Hierarchie (von 1 bis 5) und die für die Bearbeitung des Knotens verwendete Zeit in Sekunden:

Wahl	Knoten	Text	NR	GEB	LEV	Zeit
handlung	hh0	hh0	1	2	1	7
praezis	hh1	hh1	2	2	1	2
praezis	hh11	hh11	3	2	2	7
praezis	hh111	hh111	4	2	3	8
zurueck	hh11	hh11	5	2	2	6
inter_apparatur	aa12	aa12	6	3	2	18
praezis	aa121	aa121	7	3	3	9
praezis	aa1211	aa1211	8	3	4	15

Am Ende der Datei befindet sich eine Aufstellung einiger wichtiger Kennwerte für die Versuchssitzung:

zwar ein Handlungsknoten ist, der gezeigte Informationstext (Zwischentexte) jedoch zum Bereich Theorie gehört. Im graphentheoretischen Sinn sind die Zwischentexte keine Knoten des Graphen der Wissensbasis; sie sind lediglich einem Knoten *beigeordnet*.

```
Anzahl der Wahlen:                          334
Durchschnittliche Bearbeitungszeit:         7.988024 sec.
Gesamtbearbeitungszeit:                     44.466667 Minuten

Anzahl aufgerufener Theorieknoten:    11 (3.293413% der aufgerufenen
                                                              Knoten)
Anzahl aufgerufener Handlungsknoten: 188 (56.287425% der aufgerufenen
                                                              Knoten)
Anzahl aufgerufener Apparateknoten:  135 (40.419162% der aufgerufenen
                                                              Knoten)

Anzahl unbearbeiteter Knoten:        162 (51.265823% aller Knoten),
davon 151 Theorie    (93.209877% aller Theorieknoten)
       4 Handlung    (4.761905% aller Handlungsknoten)
       7 Apparatur   (10.000000% aller Apparaturknoten)
```

3

Modifikation und Anpassung des Systems

3.1 Installation, vorhandene Dateien und Systemvoraussetzungen

Das Experimentalsystem wird auf zwei Installationsdisketten zur Verfügung gestellt.[1] Auf der Diskette 1 sind alle Dateien enthalten, die zur Installation von knoX und dem Experimentalsystem auf einem geeigneten Rechner (Spezifikationen s. u.) erforderlich sind. Diskette 2 enthält Dokumentationsdateien und eine einige nützliche Werkzeuge für die Nutzung des Experimentalsystems. Die Installation des Systems besteht aus drei Phasen. In der ersten Phase wird ein Installationsverzeichnis für knoX angelegt und die Dateien auf den Installationsdisketten werden in dieses Verzeichnis übertragen.[2] In Phase zwei müssen die DOS-Systemdateien `autoexec.bat` und `config.sys` für den Betrieb von knoX modifiziert werden. In der dritten Phase werden die Dateien des Experimentalsystems kompiliert (dieser Vorgang wird in Abschnitt 3.2 beschrieben). Die erste Phase erfordert die Durchführung folgender Schritte:

Installationsanleitung

1. Legen Sie auf der Festplatte Ihres Rechners ein Installationsverzeichnis für knoX an. Sollen alle Dateien des Systems auf der Festplatte `C:` unter dem Verzeichnis `knox` installiert werden, so geben Sie bitte folgendes ein:
   `mkdir C:\knox <Return>`
   Die Betätigung der Return-Taste (↵) wird mit `<Return>` angegeben. Beachten Sie, daß Sie zur Installation und zum Betrieb des Systems minde-

[1] InteressentInnen wenden sich bitte an das Psychologische Institut der Universität Heidelberg, Sonderforschungsbereich 245, z. Hd. Frau Dipl. Math. Bärbel Maier-Schicht, Hauptstraße 47-51, 69117 Heidelberg.

[2] Da nicht zu erwarten ist, daß das System häufig von Personen installiert werden muß, die wenig oder keine Erfahrung im Umgang mit Rechnern haben, wurde auf die Automatisierung des Installationsvorganges verzichtet.

stens 5 MB Speicherplatz auf Ihrer Festplatte zur Verfügung haben sollten.

2. Legen Sie die Installationsdiskette 1 in das 3,5"Diskettenlaufwerk Ihres Rechners ein. Im folgendenden wird dieses Laufwerk als Laufwerk `A:` bezeichnet. Der Installationsvorgang funktioniert jedoch auch mit jeder anderen Bezeichnung des Diskettenlaufwerks.

3. Wechseln sie nun in das Installationsverzeichnis, das Sie in ersten Schritt (s. o.) angelegt haben:
   `cd C:\knox <Return>`

4. Geben Sie nun zum Entpacken der Dateien folgenden Befehl ein:
   `A:tar xvzf A:knoxdist.tar`
   Sie können nun am Bildschirm den Verlauf des Entpackens der Dateien verfolgen. Wenn alle Dateien in das Installationsverzeichnis übertragen sind und keine Fehler beim Entpacken aufgetreten sind, werden Sie nach kurzer Zeit wieder die Eingabeaufforderung (Prompt) am Bildschirm sehen.

5. Entfernen Sie die Installationsdiskette 1 aus dem Laufwerk und legen die Diskette 2 ein.

6. Geben Sie zum Entpacken der Dateien wiederum den Befehl
   `A:tar xvzf A:knoxdist.tar`
   ein. Wenn alle Dateien entpackt sind, ist die erste Phase der Installation abgeschlossen.

Die folgende Tabelle gibt einen vollständigen Überblick über die in der Distribution enthaltenen Dateien, die sich nun im Installationsverzeichnis Ihrer Festplatte befinden sollten.

Datei	**Kurzbeschreibung**
	Systemdateien für knoX/mprolog
BMP.EXE	Programm zum Anzeigen von Graphiken in knoX
CFIG386.EXE	Prolog-Systemdatei
EDIT.COM	Standardeditor für knoX
EDITOR.BAT	Skript zum Aufruf des Editors von knoX aus
GRAFIK.TXT	Modul zur Darstellung von Graphiken in knoX (Text)
GRAFIK.BIN	Modul zur Darstellung von Graphiken in knoX (binär)
KNOX.BAT	Skript zum Aufruf von mprolog/knoX
KNOX.DEF	knoX-Maustreiber/-steuerung
KNOX.MNU	knoX-Maustreiber/-steuerung
KNOX.PBP	knoX-exec Datei
KNOX_DIR.TXT	Liste mit Dateien, die beim Start einer Sitzung angeboten werden
KNOX_OUT.TXT	Hier werden Erklärungen ausgegeben

Fortsetzung nächste Seite ...

Datei	Kurzbeschreibung
KNOXMEN.DEF	Definitionen der Menü- und Fensterstrukturen
LIST.EXE	Programm zur Auflistung von knoX-Dateien am Bildschirm
MODEEG.EXE	Prolog-Systemdatei
MPRO.EXE	mprolog-Programm (Laufzeitumgebung)
STANDARD.PBM	Prolog-Systemdatei
STANDINF.PBM	Prolog-Systemdatei
SYS386.EXE	Systemdatei
Dateien des Experimentalsystems	
SCREEN.BIN	Definitionen zur Bildschirmausgabe (binär)
SCREEN.TXT	Definitionen zur Bildschirmausgabe (Text)
TUTMEN.DEF	Definitionen der Benutzerschnittstelle
TUTMENA.DEF	Definitionen der Benutzerschnittstelle/Menüvariante Apparatur
TUTMENH.DEF	Definitionen der Benutzerschnittstelle/Menüvariante Handlung
TUTMENT.DEF	Definitionen der Benutzerschnittstelle/Menüvariante Theorie
VOINST.TXT	Instanzendatei Gesamtstruktur (Text)
VOINSTA.TXT	Instanzendatei Aufbau-Version (Text)
VOINSTB.TXT	Instanzendatei Betreiben-Version (Text)
VOINSTF.TXT	Instanzendatei Fahrradschlauch-Flicken (Text)
VOMAIN.TXT	Main-Datei Gesamtstruktur (Text)
VOMAINAA.TXT	Main-Datei Aufbau-Version/Menüvariante Apparatur (Text)
VOMAINAH.TXT	Main-Datei Aufbau-Version/Menüvariante Handlung (Text)
VOMAINAT.TXT	Main-Datei Aufbau-Version/Menüvariante Theorie (Text)
VOMAINBA.TXT	Main-Datei Betreiben-Version/Menüvariante Apparatur (Text)
VOMAINBH.TXT	Main-Datei Betreiben-Version/Menüvariante Handlung (Text)
VOMAINBT.TXT	Main-Datei Betreiben-Version/Menüvariante Theorie (Text)
VOMAINF.TXT	Main-Datei Fahrradschlauch-Flicken (Text)
VOMASK.BIN	Definitionen zur Bildschirmmaske (binär)
VOMASK.TXT	Definitionen zur Bildschirmmaske (Text)
VOPROC.TXT	Prozeduren-Datei (Text)
VOPROCA.TXT	Prozeduren-Datei/Menüvariante Apparatur (Text)
VOPROCF.TXT	Prozeduren-Datei Fahrradschlauch-Flicken (Text)
VOPROCH.TXT	Prozeduren-Datei/Menüvariante Handlung (Text)
VOPROCT.TXT	Prozeduren-Datei/Menüvariante Theorie (Text)
VOTEXT.TXT	Textbasis Destillation (Text)
VOTEXTF.TXT	Textbasis Fahrradschlauch-Flicken (Text)
Utility-Dateien (in <Installationsverzeichnis> \utils)	
AWK.EXE	AWK-Programm für DOS
AUSW.AWK	AWK-Skript zur Konvertierung der Ergebnisdateien
DOT.EXE	DOT-Programm für DOS
DOT.LIN	DOT-Programm für Linux
MKDOT.AWK	AWK-Skript zur Erzeugung von Graphen der Wissensbasis
XNODE.AWK	AWK-Skript zur Extraktion aller Knoten einer Instanzendatei
Dokumentations-Dateien (in <Installationsverzeichnis> \doc)	
DOTDOC.PS	Benutzerhandbuch zu dot (PostScript-Format)
KDOC.PS	Die vorliegende Dokumentation (PostScript-Format)

Zur Laufzeit von knoX werden Dateien mit Namen CACHE00[1235].BIN angelegt. Diese Dateien werden zum Start von knoX nicht benötigt. Sie werden erzeugt, um mit Hilfe eines Reset-Befehls von knoX aus wieder auf bereits abgearbeitete Instruktionen zurückgreifen zu können.

Modifikationen von `autoexec.bat` und `config.sys`

Außer der Installation der systemspezifischen Dateien müssen einige Modifikationen in den Dateien `autoexec.bat` und `config.sys` des jeweiligen Systems vorgenommen werden. Da diese Modifikationen zumindest teilweise nur dann sinnvoll sind, wenn knoX geladen werden soll, wird empfohlen eine separate Boot-Option für knoX auf dem Rechner einzurichten. Dies kann entweder mit Hilfe eines PD/Shareware Boot-Managers oder mit der seit MSDOS Version 6.0 verfügbaren Möglichkeit zur Definition unterschiedlicher Boot-Profile realisiert werden. Die folgenden Beispiele zeigen die Boot-Profile in `autoexec.bat` und `config.sys` für die Boot-Option „knox“ auf einem Rechner der unter MS-DOS 6.2 läuft. Alle für knoX und das Experimentalsystem relevanten Dateien sind auf diesem Rechner im Verzeichnis `C:\KNOX` installiert. Zunächst der Ausschnitt aus `autoexec.bat`:

```
:knox
   .
   .
   .
C:\DOS\SUBST M: C:\KNOX
PATH=M:\;C:\;C:\dos
SET MPMSWAP=M:\MPROLOG.SWP
SET LMOUSE=C:\MOUSE
C:\MOUSE\MOUSE
C:\MOUSE\LOGIMENU m:knox
C:
CD KNOX
CALL KNOX
CLS
   .
   .
   .
```

Dieser Ausschnitt enthält nur die Zeilen, die sich unmittelbar auf das Funktionieren von knoX beziehen. Weitere Anweisungen, wie z. B. die Auswahl eines Tastaturtreibers oder der länderspezifischen `CODEPAGE` müssen noch ergänzt werden. Die Angaben zum Maustreiber beziehen sich auf eine serielle Logitech-Maus; die Treiberdateien sind im Verzeichnis `C:\MOUSE` installiert. Die DOS-Systemdateien befinden sich im Verzeichnis `C:\DOS`. Der für knoX relevante Ausschnitt aus `config.sys` sieht folgendermaßen aus:

```
[knox]
device=c:\dos\ansi.sys /x
device=c:\dos\setver.exe
device=c:\dos\display.sys con=(ega,850,1)
stacks 0,0
COUNTRY=049,850,C:\DOS\COUNTRY.SYS
SHELL=C:COMMAND.COM c:\dos\ /E:2048 /P
lastdrive=n
```

```
break=on
files=40
buffers=25
```

Auch in diesem Fall müssen die Pfadangaben eventuell an das lokale System angepaßt werden. Im Gegensatz zum oben gezeigten Ausschnitt der Datei `autoexec.bat`, wird hier ein komplettes Beispiel für einen Abschnitt aus `config.sys` gegeben. Entscheidend ist, daß der Treiber `HIMEM.SYS` für den Betrieb von knoX *nicht* geladen sein darf, da sonst Konflikte mit der Speicherverwaltung von knoX auftreten und bereits ein Start von knoX nicht mehr möglich ist. In der folgenden Liste werden die *Systemvoraussetzungen* für knoX spezifiziert:

	Minimum	*Empfehlung*
Rechner:	80386 SX	ab 80486 DX 25 Mhz
Hauptspeicher:	2 MB	mindestens 8 MB
Betriebssystem:	ab DOS 5.0	
Plattenspeicher:	5 MB	
Graphik:	EGA-Farbgraphik	VGA-Farbgraphik
Maus:	Seriell oder Bus	

Im folgenden geben wir eine Übersicht über die für das Experimentalsystem relevanten Dateien. Auf die Funktion und den Inhalt der Dateien wird dann in den kommenden Abschnitten genauer eingegangen. Jede dieser Dateien liegt im ASCII-Format vor und kann somit mit jedem Standard-Texteditor modifiziert werden. Nach der Modifikation einer oder mehrerer dieser Dateien muß das Experimentalsystem neu in ein internes Binärformat umgewandelt werden (siehe dazu den folgenden Abschnitt 3.2).

`v0main*.txt`	Es handelt sich um die „Master-Datei" innerhalb derer zentrale Frames und Instanzen definiert werden. Außerdem werden im „Include-Teil" dieser Datei einige der folgenden Module eingebunden.
`v0proc*.txt`	Dieses Modul enthält die Regeln, Funktionen und Behaviours des Experimentalsystems.
`v0inst*.txt`	Hier wird die Struktur der Wissensbasis in Form von Instanzen festgelegt (siehe Abschnitt 3.3). Außerdem werden den Knoten der Struktur Texte zugeordnet.
`v0text*.txt`	In diesem Modul sind die Informationstexte enthalten, die den einzelnen Knoten der Wissensbasis zugeordnet sind (siehe Abschnitt 3.4).
`v0mask.txt`	Hier werden Funktionen definiert, die für das Ausfüllen von Bildschirmmasken benötigt werden. Im Falle des Experimentalsystems sind die Funktionen für das Eingeben der Versuchs-Parameter (siehe Abb. 2.1) relevant.
`tutmen.def`	In diesem Modul wird die Benutzeroberfläche des Systems festgelegt. Neben der Gestaltung der Menüs werden die Dimensionen der Bildschirmfenster und die Farbgebung der Oberfläche determiniert.
`screen.txt`	Einige Funktionen zur Bildschirmausgabe werden hier definiert.

3.2 Ändern und Kompilieren von Input-Dateien

Modifikationen der in Abschnitt 3.1 aufgelisteten Dateien können innerhalb von knoX mit Hilfe der eingebauten Editierfunktion vorgenommen werden. Wenn eine Datei geladen ist (siehe Abschnitt 2.2) kann sie über das Menü Bearbeiten mit dem Unterpunkt Editieren in einen Editor geladen werden (siehe dazu Abb. 3.1). Hinweise zur Bedienung des Editors können mit der `F1`-Taste abgerufen werden. Eine Datei wird z. B. mit der Tastenkombination `F3-S` abgespeichert, der Editor wird mit `F3-Q` verlassen. knoX bietet jedoch die Möglichkeit, auch einen anderen Editor zu verwenden. Dazu muß in der Batch-Datei `editor.bat`, die sich im Installationsverzeichnis von knoX befindet, der Aufruf des Editors modifiziert werden, d. h. anstatt der Zeile `@call m:edit +1 %1` muß `@call <Aufruf des anderen Editors>` eingefügt werden.

Nach dem Editieren und Ändern einer Datei müssen die Änderungen in das interne binäre Format von knoX umgesetzt werden. Dies geschieht über den Unterpunkt Übersetzen im Menü Bearbeiten (siehe Abb. 3.1). Für das korrekte Übersetzen des Experimentalsystems muß wegen Abhängigkeiten zwischen den Programm-Modulen eine feste Reihenfolge eingehalten werden. Die Module `v0text*`, `v0proc*`, `v0inst*`, `v0mask` und `screen` sind Include-Dateien des Hauptmoduls `v0main`. Wurde nun z. B. die Datei `v0text*.txt` geändert,

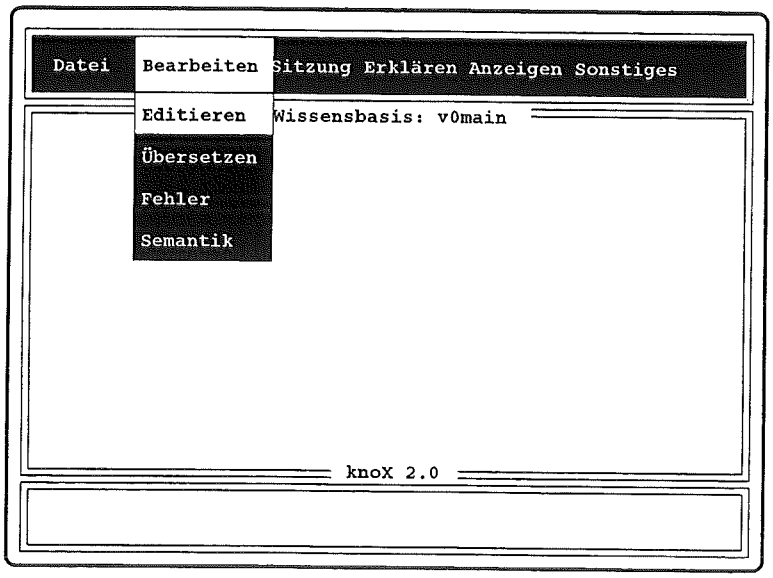


Abbildung 3.1. Editieren einer Datei. Im gezeigten Fall wird die Datei `v0main.txt` in den Editor geladen.

muß zunächst diese Datei neu übersetzt werden. Um die Änderungen innerhalb des Experimentalsystems wirksam werden zu lassen, muß anschließend auch `v0main*.txt` neu übersetzt werden, da `v0text*` über eine Include-Anweisung von `v0main*` eingelesen wird. Grundsätzlich sind Include-Dateien vor den Dateien, die diese Dateien (per Include-Befehl) einlesen, zu übersetzen.

Die Datei `tutmen.def` wird nicht direkt übersetzt. Die in ihr enthaltenen Definitionen werden vom Modul `v0proc*` zur Laufzeit eingelesen. Eine Modifikation von `tutmen.def` macht das Übersetzen von `v0proc*.txt` jedoch nicht erforderlich (außerdem muß auch `v0main*.txt` nicht neu kompiliert werden).

Wegen weiterer Abhängigkeiten zwischen den Modulen sollte die Übersetzung grundsätzlich in folgender Reihenfolge durchgeführt werden:

(1) `v0text*.txt`, `v0mask.txt` und `screen.txt`
(2) `v0inst*.txt`
(3) `v0proc*.txt`
(4) `v0main*.txt`

Sofern knoX während der Übersetzung syntaktisch oder semantisch falschen Code entdeckt, wird eine Fehlermeldung ausgegeben. Eine genauere Lokalisation der Fehler ist über den Unterpunkt Fehler im Menü Bearbeiten möglich (siehe Abb. 3.1).

Übersetzen der im Installationsverzeichnis enthaltenen Anwendungen

Im Installationsverzeichnis sind die „Quelldateien“ für verschiedene Anwendungen des Experimentalsystems enthalten. Diese Anwendungen machen Gebrauch von einer Wissensbasis zum Thema Äthanoldestillation (siehe Held et al., 1994). Es sind folgende Versionen zu unterscheiden:

- **Gesamt-Version:** Diese Version verwendet die komplette Wissensbasis. Es wird sowohl das Aufbauen, als auch das Betreiben der Destillationsapparatur beschrieben.
- **Aufbau-Version:** Es wird nur das Wissen zum Aufbau der Destillationsapparatur vermittelt.
- **Betreiben-Version:** Es wird nur das Wissen zum Betreiben der Destillationsapparatur vermittelt.

Die Aufbau- und Betreiben-Versionen wurden in den von Held et al. (1994) beschriebenen Untersuchungen verwendet. Da in den Untersuchungen die Reihenfolge der Menüpunkte des Anfangs-Menüs (siehe Abb. 2.3) variiert wurde, gibt es für jede dieser Versionen drei Varianten mit unterschiedlichen Menükonfigurationen. Jeweils eine dieser Konfigurationen beginnt mit dem Menüpunkt Theorie, Handlung oder Apparatur.

Die folgende Tabelle gibt einen Überblick über die Dateien, die zu den einzelnen Versionen der Anwendungen gehören:

Main-Datei	Instanzen-Datei	Text-Datei	Proc-Datei	TUTMEN-Datei
Gesamt-Version				
v0main.txt	v0inst.txt	v0text.txt	v0proc.txt	tutmen.def
Aufbau-Version				
v0mainat.txt	v0insta.txt	v0text.txt	v0proct.txt	tutment.def
v0mainah.txt	v0insta.txt	v0text.txt	v0proch.txt	tutmenh.def
v0mainaa.txt	v0insta.txt	v0text.txt	v0proca.txt	tutmena.def
Betreiben-Version				
v0mainbt.txt	v0instb.txt	v0text.txt	v0proct.txt	tutment.def
v0mainbh.txt	v0instb.txt	v0text.txt	v0proch.txt	tutmenh.def
v0mainba.txt	v0instb.txt	v0text.txt	v0proca.txt	tutmena.def

Im Installationsverzeichnis liegen zunächst nur die Dateien `screen.bin` und `mask.bin` in kompilierter Form vor. Um nun eine Variante einer Anwendung des Experimentalsystems lauffähig zu machen, müssen die restlichen zu dieser Variante gehörenden Dateien noch übersetzt werden. Um z. B. die Betreiben-Version in der Menüvariante mit Theorie als erstem Menüpunkt lauffähig zu machen, müssen folgende Dateien in der gezeigten Reihenfolge übersetzt werden:

(1) `v0text.txt`
(2) `v0instb.txt`
(3) `v0proct.txt`
(4) `v0mainbt.txt`

Die Anwendung kann dann unter dem Namen `v0mainbt` gestartet werden.

3.3 Die Instanzen-Datei `v0inst*.txt`

In der Datei `v0inst*.txt` wird die (hierarchisch-assoziative) Struktur der Wissensbasis des Experimentalsystems festgelegt. Ein Beispiel für diese Organisation der Wissensbasis findet sich in Abb. 2.2 auf Seite 9. Ohne genauer auf die allgemeinen Eigenschaften und Funktionen von Instanzen innerhalb knoX einzugehen, wollen wir in diesem Abschnitt beispielhaft erläutern, wie die Struktur der Wissensbasis über die Instanzen-Datei `v0inst*.txt` gestaltet und modifiziert werden kann. Wir betrachten zunächst einen Auszug aus der Datei `v0inst.txt`[3]:

```
instance aa1 of apparatur
with text = "aa1"              /* Zuweisung des Textes */
     handlung = hh1            /* Knoten bei Sprung zu Handlung */
     inter_handlung = "ah1" /* Zwischentext bei Sprung zu Handlung */
     inter_theorie = "at1"  /* Zwischentext bei Sprung zu Theorie */
     praezis = aa11.           /* Knoten fuer Praezis-Sprung */

     instance aa11 of apparatur
     with text = "aa11"
          handlung =  hh11
          inter_theorie = "at11"
          inter_handlung = "ah11"
          allgemein = aa1     /* Knoten fuer Allgemein-Sprung */
          weiter = aa12       /* Knoten fuer Weiter-Sprung */
          praezis = aa111.
              .
              .
              .
instance hh1 of handlung
with text = "hh1"
     inter_theorie = "ht1"
     inter_apparatur = "ha1"
     apparatur = aa1
     weiter = hh2
     praezis = hh11.

     instance hh11 of handlung
     with text = "hh11"
          inter_theorie = "ht11"
          inter_apparatur = "ha11"
          apparatur = aa12
          allgemein = hh1
          weiter = hh12
          praezis = hh111.
```

Wie bereits erwähnt besteht die Struktur der Wissensbasis aus drei Teilstrukturen für die Bereiche Theorie, Handlung und Apparatur. Obiges Beispiel

[3]Es handelt sich hierbei um die Grundform der Instanzen-Dateien, die nicht für eine spezielle experimentelle Bedingung modifiziert wurde. Aus diesem Grund enthält der Dateiname kein weiteres Label.

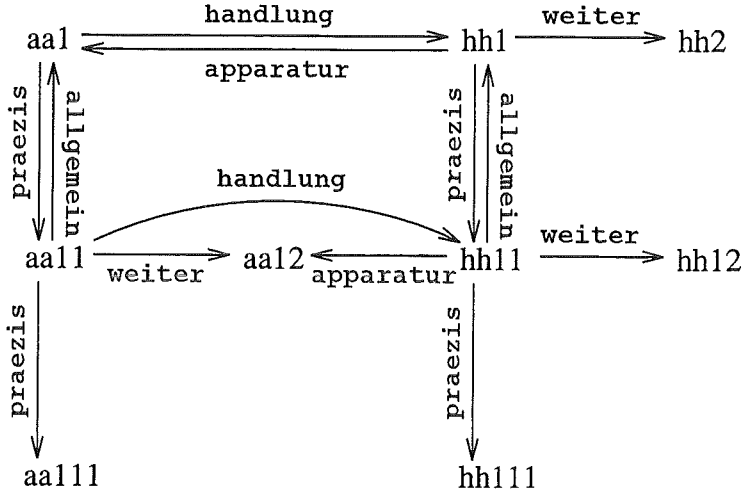
zeigt jeweils zwei Instanzen aus den Bereichen Apparatur und Handlung. Jede Instanz kann eine Reihe von Angaben zur Verknüpfung der Instanz mit anderen Instanzen oder Texteinheiten enthalten. Diese Angaben sind in die sogenannten „Slots" der Instanz eingetragen. Die Instanz `aa1` verfügt über die fünf Slots `text`, `handlung`, `inter_handlung`, `inter_theorie` und `praezis`. Jeder dieser Slots ist mit einem Wert ausgefüllt, der entweder auf eine andere Instanz (Werte ohne Hochkommata, z. B. `aa11`) oder auf eine Texteinheit (Werte mit Hochkommata, z. B. `"ah1"`) zeigt.

Verweise auf andere Instanzen erfolgen über die Slots `theorie`, `handlung`, `apparatur`, `weiter`, `allgemein` und `praezis`. Diese Slots entsprechen den Unterpunkten der innerhalb des Experimentalsystems dargebotenen Menüs (siehe hierzu z. B. Abb. 2.4 und 2.5)[4]; durch sie wird die Struktur der Wissensbasis festgelegt. Wenn z. B. im Experimentalsystem der Menüpunkt präziser gewählt wird, so wird zu der Instanz übergegangen, deren Name innerhalb des Slots `praezis` der aktuellen Instanz angeben ist.

Die Slots `text`, `inter_theorie`, `inter_handlung` und `inter_apparatur` steuern die Ausgabe der mit den Instanzen korrespondierenden Informationstexte. Der Instanz `aa1` der Teilstruktur `apparatur` wird durch das Statement `text = "aa1"` ein Text mit dem Label `aa1` zugeordnet (zur Organisation der Texte siehe Abschnitt 3.4). Einen besonderen Fall stellen die sogenannten „Zwischentexte" dar (siehe dazu auch Fußnote 6 auf Seite 6). Diese Texte werden mit den Slots `inter_theorie`, `inter_handlung` und `inter_apparatur` referenziert. Ihre Funktion besteht darin, den Übergang zwischen den Teilstrukturen der Wissensbasis in semantischer Hinsicht zu unterstützen und zu erleichtern. Wird z. B. von der Instanz `aa1` der Apparatur-Struktur zur Handlungs-Struktur (Instanz `hh1`) gewechselt, so wird der im Slot `inter_handlung` angegebene Zwischentext mit dem Label `ah1` ausgegeben und nicht der Text `hh1`, der mit der Instanz `hh1` verbunden ist. Der Zwischentext stellt explizit den Bezug zwischen der Information zur Apparatur und der damit korrespondierenden Information zur Handlung her. In der Anwendung des Experimentalsystems zum Bereich Destillation wird nur von Zwischentexten Gebrauch gemacht, wenn ein Sprung von der Handlungs- oder Apparaturhierarchie zum Bereich Theorie gemacht wird. Dies liegt daran, daß die Information, die innerhalb der Theoriehierarchie geboten wird (insgesamt nur acht Knoten) nicht ausreichend gewesen wäre.

Die strukturelle Beziehung die durch die oben gezeigten Instanzen etabliert wird, ist in folgender Abbildung dargestellt (Zwischentexte sind nicht berücksichtigt):

[4]Eine Ausnahme stellt der Menüpunkt Anfang dar, dessen Auswahl stets zum Ausgangspunkt der Konsultation des Systems führt. Außerdem wird die „Weiter"-Anweisung im Falle der Handlungs-Struktur mit nächs. Handl. und im Falle der Apparatur-Struktur mit nächs. Bauteil umschrieben.



Die Graphik zeigt die Beziehungen zwischen Instanzen. Die Modifikation einer solchen Struktur erfolgt über die Slots der jeweiligen Instanzen. Soll z. B. die „Präziser"-Information, die mit der Instanz `aa1` verbunden ist, nicht mehr über die Instanz `aa11` abgerufen werden, sondern über die Instanz `aa12`, so ist im Slot `praezis` der Instanz `aa1` lediglich der Wert `aa11` gegen den Wert `aa12` auszutauschen. Anschließend sind Instanzen-Datei und Main-Datei natürlich neu zu übersetzen.

Die Struktur der Wissensbasis kann mit Hilfe des Programms „`dot`" (Koutsofios, 1993) und des AWK-Filters „`mkdot.awk`" (siehe Anhang) anschaulich als gerichteter Graph dargestellt werden. Das Programm `dot` wurde speziell zur optimalen (hierarchischen) Darstellung gerichteter Graphen entwickelt. Versionen für MS-DOS und Linux wurden uns von AT&T Bell Laboratories für Forschungszwecke kostenlos zur Verfügung gestellt.[5] Die MS-DOS Version `dot.exe` ist nur für die Berechnung relativ kleiner Graphen geeignet. Zur Darstellung komplexerer Strukturen muß auf die Linux-Version (bzw. auf Versionen für andere Unix-Plattformen) zurückgegriffen werden. Beide Programmversionen befinden sich auf den Installationsdisketten bzw. im Verzeichnis

```
<Installationsverzeichnis>\utils
```

Im Anhang findet sich ein Beispiel für einen mit `dot` erzeugten Graphen. Der Graph zeigt die Handlungs-Hierarchie der Betreiben-Version der Wissensbasis. Generell soll die graphische Darstellung der Struktur von Wissensbasen die Fehlersuche bei der Entwicklung von Wissensbasen unterstützen.

Das Filter `mkdot.awk` wandelt Instanzen-Dateien wie `v0inst*.txt` in Input-Dateien für `dot` um. Je nach gewünschter Darstellung kann eine manuelle Modifikation des Outputs von `mkdot.awk` erforderlich sein. Das Filter wird wie folgt aufgerufen:

[5] Der Ansprechpartner für Fragen zu `dot` ist Mr. Stephen C. North, e-mail: `north@att.research.com`. Der Source-Code für `dot` ist nicht verfügbar; die Programmversionen können als uuencodierte Binärdateien bezogen werden.

`awk -f mkdot.awk <Instanzendatei> > <Input-Datei f³r dot>`

Der Aufruf von `dot` lautet:

`dot -Tps <Input-Datei f³r dot> -o <Postscript-Ausgabedatei>`

3.4 Die Textbasis `v0text*.txt`

Die Menge aller abrufbaren Informationstexte ist in der separaten Datei `v0text*.txt` enthalten. In dieser Datei sind auch die Labels zu finden, über die die Texte innerhalb der Instanzen referenziert werden. Folgender Ausschnitt aus `v0text.txt` vermittelt einen Eindruck von der Struktur dieser Datei:

```
text_nr(aa1,
["Die Destillationsapparatur besteht aus mehreren",nl,
"Glas- und Metallbauteilen.",nl,
"Sie befindet sich auf einem Tisch unter einem Abzug."]).

text_nr(aa11,
["Bestandteile der Destillationsapparatur sind",nl,
"zwei Halteeinrichtungen.",nl,
"Sie bestehen aus mehreren Bauteilen und haben",nl,
"ein vergleichsweise hohes Gewicht."]).

text_nr(aa111,
["Die erste Halteeinrichtung gehört zum Verdampferteil.",nl,
"Sie besteht aus der Muffe und der Stativklammer für",nl,
"die Destillationsblase, der Muffe und der Stativklammer",nl,
"für die Kolonne sowie einem Stativ."]).

text_nr(aa12,
["Ein Bestandteil der Destillationsapparatur ist",nl,
"das Verdampferteil.",nl,
"Es hat eine vertikale Anordnung und befindet sich",nl,
"neben dem Kühlerteil. Es besteht aus der",nl,
"Destillationsblase, der Heizvorrichtung und der Kolonne."]).

text_nr(hh1,
"Bauen Sie die Destillationsapparatur auf.").
text_nr(hh11,
"Installieren Sie das Verdampferteil.").
text_nr(hh111,
["Richten Sie die Halterung für die Destilla-",nl,
"tionsblase und die Kolonne ein."]).
text_nr(hh2,
"Betreiben Sie die Destillationsapparatur.").
text_nr(hh12,
"Installieren Sie das Kühlerteil.").
```

Die Grundform eines Texteintrages ist `text_nr(<Textlabel>,["<Text>"])`. Dem Text „*Installieren Sie das Verdampferteil*“ ist z. B. das Label `hh11` zugeordnet. Für alle Texte, die keine Zwischentexte sind, gilt die Konvention, daß das Label dem Namen der Instanz entspricht innerhalb derer der Text abgerufen wird. Das Label der Zwischentexte enthält die Nummer der aufrufenden Instanz und eine Markierung, die die Art des Überganges zwischen Teilstrukturen symbolisiert. Ein Zwischentext, der von der Instanz der Handlungs-Struktur `hh1` aus bei einem Wechsel zur Theorie-Struktur aufgerufen wird, besitzt das Label `ht1` (siehe auch Abschnitt 3.3).

Die Texte werden in Hochkommata eingegeben. Zeilenumbrüche müssen mit der Anweisung „`nl`“ veranlaßt werden. Bei der Eingabe der Texte ist insbesondere zu beachten, daß der dargestellte Textblock die Ausmaße des Bildschirmfensters nicht überschreitet und daß keine Überschneidung mit den Menüs auftreten.[6]

3.5 Graphikimport

knoX bietet die Möglichkeit, graphisches Material in Wissensbasen einzubinden und bei Bedarf am Bildschirm auszugeben. Innerhalb des Experimentalsystems soll diese Möglichkeit für die Darstellung von Teilen der Destillationsapparatur neben den entsprechenden Informationstexten genutzt werden.

Eine Untersuchung mit Graphik-Unterstützung ist geplant, es existiert jedoch bislang keine vollständige Wissensbasis mit graphischen Elementen. Die im folgenden beschriebenen Modifikationen können aber leicht auf beliebige bestehende Wissensbasen angewandt werden.

3.5.1 Voraussetzungen

Neben einiger Modifikationen in `v0main*.txt`, `v0inst*.txt`, `v0proc*.txt` und `v0text*.txt` müssen das Tool `grafik.txt`, das Programm `bmp.exe` und bei Bedarf ein Graphik-Konverter, der eine Umwandlung gängiger Graphikformate in das `bmp`-Format ermöglicht, verfügbar sein.

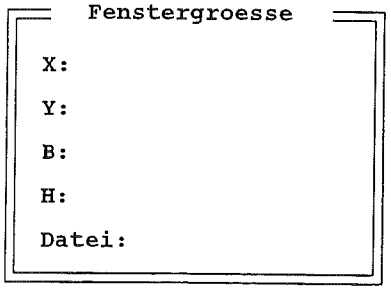
[6] Als Ergänzung des Systems wäre eine komfortablere Form der Texteingabe und -formatierung wünschenswert. Prinzipiell ist es möglich, die Bildschirmausgabe der Textblöcke auf einen Bereich zu begrenzen, innerhalb dessen es keine Überschneidungen mit anderen Bildelementen wie z. B. Menüs kommen kann (dies wurde in einer Testversion des Systems bereits realisiert). Noch zu bearbeiten ist das Problem der (automatischen) Formatierung der Texte. Es wäre wünschenswert, daß die Zeilenumbrüche nicht von Hand eingegeben werden müssen und daß die Umbrüche nach Textänderungen automatisch aktualisiert werden.

3.5.2 Graphiken erzeugen

Graphiken, die in eine Wissensbasis eingebunden werden sollen, müssen im `bmp`-Format vorliegen. Bilder, die mit Hilfe eines Scanners erzeugt werden, können i. a. direkt in diesem Format abgespeichert werden. Die gängigen Programme zur Erzeugung und Bearbeitung von Bitmaps, wie z. B. `paint` und `coreltrace` unter DOS/WINDOWS oder `xpaint` und `xv` unter UNIX, unterstützen dieses Format.

Da die Darstellung eines `bmp`-Bildes unter knoX relativ zeitaufwendig ist, sollten die Bilder in das interne Prolog-Format umgewandelt werden. Dafür steht das Tool `graphik` zur Verfügung. Dieses Tool läuft unter knoX und wird wie eine Wissensbasis geladen und übersetzt. Gestartet wird unter dem Menüpunkt Sitzung mit Start/Weiter. Nach dem Start erscheint das Menü `Grafik`:

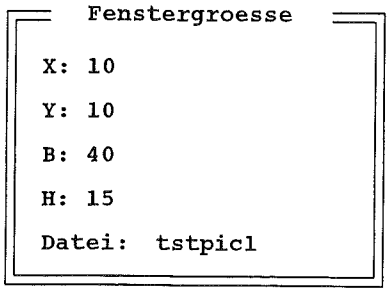
Der Menüpunkt anzeigen führt zur Dialogbox `Fenstergroesse`, in die Position, Größe und Name des anzuzeigenden Bildes (im `bmp-Format`) einzutragen sind:



In die Felder `X` und `Y` sind zur Plazierung der Graphik auf dem Bildschirm die Koordinaten der linken oberen Ecke des Graphik-Fensters einzutragen. Die Koordinaten werden als Zeichenposition angegeben, deshalb hat die `X`-Koordinate zwischen 0 und 80, die `Y`-Koordinate zwischen 0 und 25 zu liegen.

Die Felder `B` und `H` werden mit der gewünschten Breite und Höhe des darzustellenden Graphik-Fensters ausgefüllt. Für die möglichen Werte gelten die obigen Angaben zur Fensterpositionierung. Werden Koordinaten oder Dimensionen angegeben, die außerhalb des darstellbaren Bereichs liegen, wird ohne Fehlermeldung zum Menü `Grafik` zurückgesprungen.

In das Feld `Datei` ist schließlich der Name der `bmp`-Datei *ohne die Extension* `.bmp` einzugeben. Es empfiehlt sich, alle Werte zu notieren, da diese für den Aufruf der Graphik innerhalb der Wissensbasis wieder benötigt werden. Folgende Abbildung gibt ein Beispiel für die ausgefüllte Dialogbox:



Sind alle Angaben korrekt, so wird nach Beendigung der Eingaben mit RETURN die gewünschte Graphik ausgegeben. In der aktuellen Fassung ist das Graphik-Fenster umrahmt und grau unterlegt. Die Einstellungen zur Präsentation der Graphik können im Behaviour `zeige_bild(BILD_ID)` (s. u.) in `v0proc*.txt` vorgenommen werden.

Als nächster Schritt kann die Graphik in das interne Prolog-Format umgewandelt werden. Dies geschieht über den Menüpunkt **speichern**. Nach der Anwahl dieses Punktes wird die Dialogbox `Datei:` am unteren Bildschirmrand geöffnet. Hier ist ein Dateiname *ohne Extension* einzugeben. Sinnvollerweise wählt man denselben Namen wie für die `bmp`-Datei. Die neue Datei wird nach Eingabe von RETURN automatisch mit der Extension `.pic` abgespeichert.

Mit dem Menüpunkt **laden** kann nun die in das Prolog-Format konvertierte Datei angezeigt werden. Auch in diesem Fall öffnet sich die Dialogbox `Fenstergroesse`. Wichtig ist nun, daß genau dieselben Werte für `X`, `Y`, `B` und `H` eingegeben werden wie beim Abspeichern der Datei. Stimmen die Werte nicht überein, ist die Darstellung des Bildes fehlerhaft.

3.5.3 Einbindung in die Wissensbasis

Die hier vorgestellten Modifikationen des Programm-Codes bewirken, daß zu jedem aufgerufenen Text-Knoten eine entsprechende Graphik ausgegeben wird. Denkbar und sinnvoll wäre die Möglichkeit, daß Graphiken durch die Benutzerin des Systems über einen Menüpunkt abgerufen werden können. Auch diese Variante ist auf der Basis der folgenden Erläuterungen leicht realisierbar.

Die folgenden Abschnitte geben einen Überblick über Modifikationen im Quellcode des Experimentalsystems, die zur Einbindung von Graphiken erforderlich sind.

v0proc*

In der Datei v0proc* muß das Behaviour ausgeben(Knoten) modifiziert werden. Mit dem Zusatz get_value(Knoten,bild,Bild) wird die Graphikdatei, die einem Knoten zugeordnet ist, ermittelt. Durch start(zeige_bild(Bild)) wird die Ausgabe der Graphik veranlaßt. Die betreffenden Stellen sind im folgenden Code-Segment durch „/**/“ markiert:

```
behaviour ausgeben(Knoten)
is get_value(Knoten,text,Text) and
   get_value(Knoten,bild,Bild) and                          /**/
   start(bildschirm_ausgeben(Knoten,Text)) and
   start(zeige_bild(Bild)) and                              /**/
   start(protokollieren("AKTUELL: ",Knoten,Text)) and
   start(neu_eintragen(Knoten,Text)) and
   put_value(akt_zustand,knoten,Knoten) and
   get_value(akt_zustand,gebiet,Gebiet) and
   menue_angeben(Gebiet,Menue) and
   (if comparison(kommentar(strategie),=,ja)
    then dialog_box("Kommentar?",Input,[rectangle= (0,22,78,1),
                                        name="Kommentar"]) and
         file_schreiben(["Kommentar: ",Input])
   ) and
   put_value(akt_zustand,anbieten,Menue).
```

Das Behaviour zeige_bild(BILD_ID) ist zentral für die Graphikdarstellung. Innerhalb dieses Behaviours werden die Parameter einer Graphik eingelesen, eine evtl. bereits am Bildschirm ausgegebene Graphik wird gelöscht und eine neue Graphik wird geladen:

```
behaviour zeige_bild(BILD_ID) is
    bild_nr(BILD_ID,X,Y,B,H,DATEINAME) and
    remove_window(bild_win) and
    create_window(bild_win,[rectangle=(X,Y,B,H),paper=7,
                            pen=15,type=graphic]) and
    open_window(bild_win) and
    set_graphics(bild_win) and
    concatenate(DATEINAME,".pic",FN) and
    load_picture(FN).
```

v0main*

In v0main* müssen lediglich die Frames apparatur, handlung und theorie um den Slot bild ergänzt werden. Wir zeigen hier als Beispiel den Frame handlung:

```
frame handlung
slots text :        string(7);
      bild :        string(7);                          /**/
      apparatur :   instance(apparatur);
      allgemein :   instance(handlung);
      praezis :     multivalued,instance(handlung);
      weiter :      instance(handlung);
      inter_theorie : string(7);
      inter_apparatur : string(7).
```

v0inst*

Innerhalb der Instanzendatei ist lediglich für jeden Knoten der Kurzname der aufzurufenden Graphik einzutragen. Das folgende Beispiel zeigt die Instanz des Knotens aa1.

```
instance aa1 of apparatur
with text = "aa1"
     bild = "pic1"                  /**/
     handlung = hh1
     inter_theorie = "at1"
     inter_handlung = "ah1"
     praezis = aa11.
```

Der Kurzname der Graphik, die dem Knoten aa1 zugeordnet ist, lautet pic1. Die Referenz zwischen dem Kurznamen und dem realen Dateinamen der Graphik wird in der Datei v0text* hergestellt.

v0text*

Ebenso, wie in v0text* die Instruktionstexte zu den einzelnen Kurznamen der Knoten aufgelistet sind, werden für die Einbindung von Graphiken im prolog_part dieser Datei die Namen der Graphikdateien (im internen Format) und die Parameter der Graphiken (Position und Größe) angegeben.[7] Der folgende Ausschnitt demonstriert dies für drei Graphiken:

```
prolog_part.

/************** BILDER ********************/
bild_nr(pic1,20,5,50,15,tstabb21).
bild_nr(pic2,20,5,50,15,tstabb22).
bild_nr(pic3,20,5,50,15,tstabb23).
```

[7] Man kann natürlich auch eine separate Datei mit Bilddefinitionen anlegen (z. B. v0bild*) und getrennt von v0text* übersetzen und einlesen.

Die Parameter der Anweisung `bild_nr(...)` sind definiert im Behaviour `zeige_bild(BILD_ID)` in `vOproc*` (s. o.). Die Graphik, auf die mit dem Kurznamen `pic` verwiesen wird, existiert physikalisch als Datei `tstabb21.pic`. Sie wird 20 Spalten vom linken und 5 Zeilen vom oberen Bildschirmrand entfernt positioniert. Ihre Größe beträgt 50 Bildschirmspalten in der Breite und 15 Zeilen in der Höhe. Die Graphik ist so lange am Bildschirm zu sehen, bis eine neue Informationseinheit angewählt wird.

3.6 Modifikation der Benutzeroberfläche

Die wesentlichen Einstellungen zur Benutzeroberfläche befinden sich in den Dateien `vOproc*.txt` (siehe dort das Behaviour „oberflaeche") und `tutmen.def` (wie bereits erwähnt, wird `tutmen.def` von `vOproc*.txt` eingelesen). Folgender Ausschnitt aus `tutmen.def` ist ein Beispiel für die Definition der Menüs des Experimentalsystems:

```
                    /* MENU: ANFANG                                    */

menu(anfang,"",[blank,theorie,apparatur,handlung],
               [rectangle=(63,1,15,8),escape =[8448]]).
           /*  [rectangle=(0,1,78,1),escape =[8448]]).     quer  */
           /*  [rectangle=(63,1,15,8),escape =[8448]]).    längs */

menu_item(theorie,"Theorie").
menu_item(apparatur,"Apparatur").
menu_item(handlung,"Handlung").
```

Es handelt sich hier um die Definition des Anfangsmenüs des Experimentalsystems (siehe Abb. 2.3). Dieses Menü `anfang` enthält die Unterpunkte `blank`, `theorie`, `apparatur` und `handlung`. Das Format des Menüs wird durch die `rectangle`-Anweisung bestimmt. Die Anweisung

```
rectangle=(63,1,15,8)
```

definiert eine Menü-Box, deren linke obere Ecke die (Bildschirm-) Koordinaten 63 und 1 besitzt; die Breite des Menüs beträgt 15 Zeichen, die Höhe 8 Zeilen. Die `escape`-Anweisung gibt eine Liste von Zeichencodes an, die den Tasten entsprechen, mit denen man ein Menü verlassen kann.[8] Mit der Funktion `menu_item` wird den definierten Menüpunkten ein Text zugewiesen, der dann auch am Bildschirm ausgegeben wird.

[8] Auch ohne `escape`-Anweisung kann ein Menü stets mit der ESCAPE-Taste verlassen werden.

Literaturverzeichnis

Aho, A. V., Kernigan, B. W. & Weinberger, P. J. (1988). *The AWK programming language.* Reading, MA: Addison-Wesley.

Engesser, H. (ed.) (1988). *Duden Informatik.* Mannheim: Dudenverlag.

Held, T., Laier, R. & Fries, S. (1994). *Erwerb von Wissen zur Destillation: Zwei empirische Untersuchungen*, Arbeitsbericht aus dem Teilprojekt A5 des Sonderforschungsbereichs 245 „Sprache und Situation", Universität Heidelberg.

Koutsofios, E. (1993). *Drawing graphs with dot*, dot User's Manual, AT&T Bell Laboratories.

Kuhlen, R. (1991). *Hypertext: ein nichtlineares Medium zwischen Buch und Wissensbank.* Berlin, Heidelberg: Springer.

Sugiyama, K., Tagawa, S. & Toda, M. (1981). Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man, and Cybernetics*, **11**(2), 109–125.

Theiss, G. (1992). *Benutzerhandbuch der hybriden Expertensystemschale knoX*, Technical report.

Anhang

Das Filter ausw.awk

```
#
# ausw.awk
#
# Ein Filter zur Konvertierung und Auswertung der Protokolldateien
# der Destillationsuntersuchungen.
#
# Autor: T. Held, Univ. Heidelberg
#
# Version vom 01.06.1994
#
# Aufruf:  awk -f ausw.awk <NODE_FILE> <PROTOKOLL_FILE>
#
# Das <NODE_FILE> enth"alt die Kodes aller Knoten der untersuchten
# Wissensbasis.
#
# Das <PROTOKOLL_FILE> wurde w"ahrend eines Versuchsdurchganges erzeugt.
#
#
BEGIN{printf("     Wahl      Knoten     Text    NR  GEB    LEV    Zeit\n\n")
switch = 1
wahl = ""
timeused = 1000
}
{
#
# Array mit vorhandenen Knoten fuellen (aus v0insta.nod)
#
if(NF<2)  {
            nodearr[$1]++
            nodecnt++
            gebn = substr($1,0,2)
            if(gebn=="ht" || gebn=="at" || gebn=="tt") zntheo++
            if(gebn=="hh") znhand++
            if(gebn=="aa") znappa++
}

#
# Starte Auwertung
#
if($1=="Wahl:")  { if(switch==1) {owahl = wahl}
                   wahl = $2
                   switch = 0
                   switch2 = 1
                 }


if($1=="Textnr.:") {
                    oldtnr = $2
                   }
if($1=="AKTUELL:" && switch2==1) {switch = 1
                    oaktuell = aktuell
                    aktuell = $2
                    if(timeused!=0 && timeused<600) {
                        printf("     %3d\n",timeused)
                        tottime = tottime + timeused
                                                  }
                    if(owahl!="") {
                        wahlzahl++
                        geb = substr(oldtnr,0,2)
                        if(geb=="ht" || geb=="at" || geb=="tt") { gebnr = 1
                                                                 ztheo++ }
                        if(geb=="hh") { gebnr = 2
                                        zhand++ }
                        if(geb=="aa") { gebnr = 3
                                        zappa++ }
#
# Bestimme die Position in der Hierarchie
#
```

```
                        hier = length(substr(oldtnr,2,6)) - 1
#
# Registriere, dass der Knoten aufgerufen wurde
#
                        readarr[oldtnr]++
#
# Ausgabe
#

    printf("%17s    %9s    %9s    %3s    %d    %d",owahl,oaktuell,oldtnr,wahlzahl,gebnr,hier)
                      }
    oldtime = acttime
    split($4,tarr,":")
    acttime = (tarr[1]*3600)+(tarr[2]*60)+tarr[3]
    timeused = acttime - oldtime
    switch2 = 0
                    }
if($1=="Zeit:")     {
                     oaktuell = aktuell
                     printf("     %3d\n",timeused)
                     tottime = tottime + timeused
                     wahlzahl++
                          geb = substr(oldtnr,0,2)
                          if(geb=="ht" || geb=="at" || geb=="tt") { gebnr = 1
                                                                   ztheo++ }
                          if(geb=="hh") { gebnr = 2
                                          zhand++ }
                          if(geb=="aa") { gebnr = 3
                                          zappa++}

#
# Bestimme die Position in der Hierarchie
#
                        hier = length(substr(oldtnr,2,6)) - 1
#
# Registriere, dass der Knoten aufgerufen wurde
#
                        readarr[oldtnr]++
#
# Ausgabe
#

    if(owahl!="") printf("%17s    %9s    %9s    %3s    %d    %d",owahl,oaktuell,oldtnr,wahlzahl,
                                                                  gebnr,hier)
    oldtime = acttime
    split($2,tarr,":")
    acttime = (tarr[1]*3600)+(tarr[2]*60)+tarr[3]
    timeused = acttime - oldtime
    printf("     %3d\n",timeused)
    tottime = tottime + timeused
                  }
}
END{
    meanread = tottime/wahlzahl
    prozt = (ztheo/wahlzahl) * 100
    prozh = (zhand/wahlzahl) * 100
    proza = (zappa/wahlzahl) * 100

    for(i in nodearr) {
                      if(readarr[i] < 1) { unreadcnt++
                                           unreadarr[i]++
                         gebu = substr(i,0,2)
                         if(gebu=="ht" || gebu=="at" || gebu=="tt") zutheo++
                         if(gebu=="hh") zuhand++
                         if(gebu=="aa") zuappa++
                                        }
                      }
    prozunr = (unreadcnt/nodecnt) * 100
    prozunrt = (zutheo/zntheo) * 100
    prozunrh = (zuhand/znhand) * 100
    prozunra = (zuappa/znappa) * 100

 printf("\nT: %s  H: %s  A: %s",zntheo,znhand,znappa)
 printf("\n\nAnzahl der Wahlen: \t\t\t%d\n",wahlzahl)
 printf("Durchschnittliche Bearbeitungszeit: \t%f sec.\n",meanread)
```

```
printf("Gesamtbearbeitungszeit: \t\t%f Minuten\n\n",tottime/60)
printf("Anzahl aufgerufener Theorieknoten:   %3d (%f%% der aufgerufenen Knoten)\n",ztheo,
                                                                        prozt)
printf("Anzahl aufgerufener Handlungsknoten: %3d (%f%% der aufgerufenen Knoten)\n",zhand,
                                                                        prozh)
printf("Anzahl aufgerufener Apparateknoten:  %3d (%f%% der aufgerufenen Knoten)\n\n",zappa,
                                                                          proza)
printf("Anzahl unbearbeiteter Knoten:        %3d (%f%% aller Knoten),\n",unreadcnt,prozunr)
printf("davon %3d Theorie    (%f%% aller Theorieknoten)\n",zutheo,prozunrt)
printf("      %3d Handlung   (%f%% aller Handlungsknoten)\n",zuhand,prozunrh)
printf("      %3d Apparatur  (%f%% aller Apparaturknoten)\n",zuappa,prozunra)
  }
```

Das Filter `mkdot.awk`

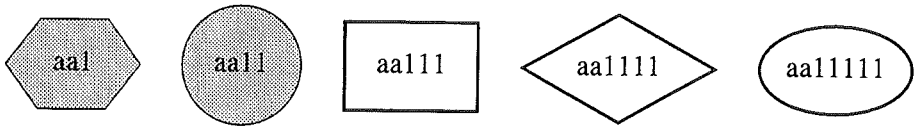
Dieses Filter dient der graphischen Darstellung der Struktur einer Wissensbasis. Diese Struktur wird innerhalb des Experimentalsystems durch die Instanzen-Dateien festgelegt. Das Filter liest eine Instanzen-Datei ein und gibt eine Datei mit Kommandos für das Graphen-Zeichenprogramm `dot` aus. Für eine genaue Beschreibung von `dot` wird auf die Dokumentation von Koutsofios (1993) verwiesen. Die Dokumentation liegt als PostScript-Datei im Verzeichnis

```
<Installationsverzeichnis>\doc
```

vor. Je nachdem, ob ein Graph aller Knoten der Wissensbasis oder nur eine Teilhierarchie ausgegeben werden soll, muß der Quelltext des Filters modifiziert werden. Es wurde darauf verzichtet für dieses Filter eine komfortablere Benutzerschnittstelle zu gestalten, da seine Verwendung nicht zu den Routine-Aufgaben der Nutzerin des Experimentalsystems gehört. Graphen der Wissensbasis werden üblicherweise nur während der Entwicklungsphase benötigt. Zur Modifikation des Filters sind zumindest Grundkenntnisse der Skript-Sprache `awk` erforderlich. Eine ausgezeichnete Dokumentation bietet Aho et al. (1988). Eine unter DOS lauffähige Version befindet sich im Verzeichnis

```
<Installationsverzeichnis>\utils
```

Die Knoten werden je nach Position in der hierarchischen Struktur in unterschiedlicher Form ausgegeben. In der aktuellen Version der Wissensbasis des Experimentalsystems treten fünf hierarchische Niveaus auf (`aa?` bis `aa?????` bzw. `hh?` bis `hh?????`). Die folgende Graphik gibt ein Beispiel:



Die Kanten sind dem Verbindungstyp entsprechend mit Labeln versehen („w“ für weiter, „p“ für präzis, „th“ für Theorie, „ap“ für Apparatur und „ha“ für Handlung). Außerdem sind „weiter“-Kanten mit durchgezogenem Strich gezeichnet (solid), „präzis“-Kanten gepunktet (dotted) und alle Sprünge zwischen Teilhierarchien („Theorie“-, „Apparatur“- und „Handlung“-Kanten) gestrichelt (dashed).

Das Programm `dot` bietet in beschränktem Maße die Möglichkeit, auf die Anordnung von Knoten und Kanten Einfluß zu nehmen (z. B. das `rank=same` Kommando). Da die Graphen der Wissensbasen zumindest auf den ersten Blick unübersichtlich wirken, liegt es nahe, den Versuch zu unternehmen, die Knoten bzw. Kanten übersichtlicher anzuordnen. Ein Teil dieser Arbeit wird bereits von `mkdot.awk` übernommen, indem zumindest die ersten beiden Hierarchiestufen jeweils in einer Ebene angeordnet werden. Ausführliche Versuche, das Layout des Graphen „von Hand“ zu optimieren ergaben jedoch, daß die automatische

Anordnung von `dot` in den meisten Fällen zum besten Ergebnis führen dürfte. Dies liegt daran, daß die automatische Anordnung in Hinblick auf minimale Kreuzungshäufigkeit und bestmögliche Übersichtlichkeit des Graphen optimiert ist (siehe dazu auch Sugiyama, Tagawa & Toda, 1981).

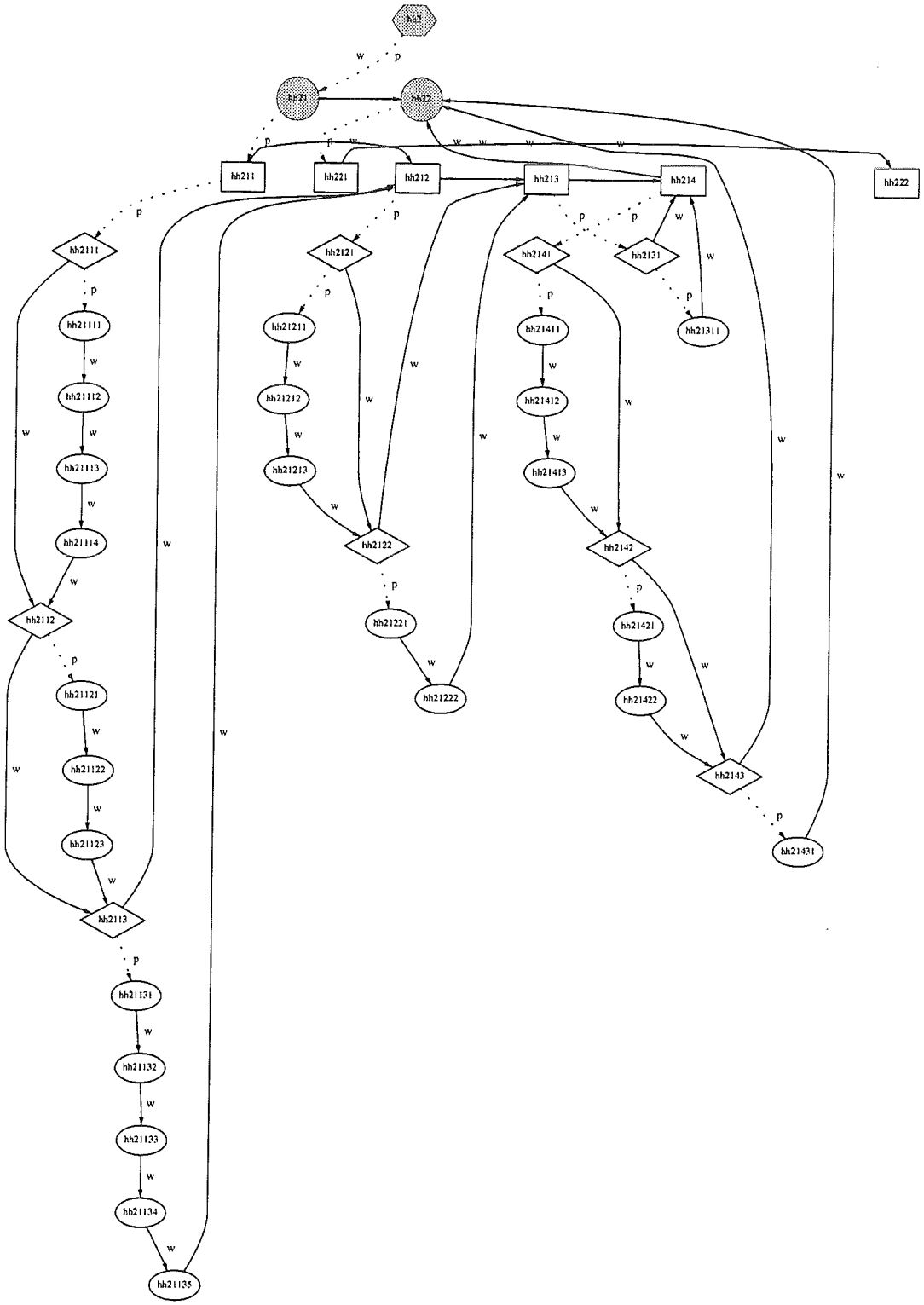
```
#
# mkdot.awk
#
# Ein Filter zur Umwandlung von Instanzen-Dateien der Destillationsuntersuchun-
# gen in Input-Dateien fuer das Graphen-Zeichenprogramm "dot".
#
# Autor: T. Held, Univ. Heidelberg
#
# Version vom 28.11.1994
#
# Aufruf:  awk -f mkdot.awk <INSTANZEN-DATEI> > <DOT-INPUT-DATEI>
#
# Die <INSTANZEN-DATEI> muss in der nicht kompilierten Form verwendet
# werden (also z.B. v0insta.txt).
#
# Die <DOT-INPUT-DATEI> dient als Eingabe fuer das Programm dot. Ein Graph
# wird mit folgendem Kommado erzeugt:
#
# dot -Tps <DOT-INPUT-DATEI> -o <POSTSCRIPT-OUTPUT-DATEI>
#
#
BEGIN{
we=""
pr=""
al=""
inst=""
ap=""
ha=""
th=""
#
# Hier beginnt der Prolog der dot-Datei. Falls andere Grundeinstellungen
# gewuenscht werden, muessen hier Aenderungen vorgenommen werden (z.B.
# Fontparameter, Skalierung, etc.
#
print "digraph G {"
print "page=\"8.2,11.5\";"
print "ratio=auto;"
print "fontname=Helvetica;"
print "node[fontsize=12, shape=box];"
# print "ranksep=1;"
#
# Ende des Prologs
#
}
{
#

#
if($1=="instance") { ausgabe=0
# Je nach gewuenschter (Teil-)Hierarchie ist eines der folgenden vier
# if-Statements auszukommentieren. Alle anderen Statements muessen
# kommentiert sein. Im gezeigten Fall wird die Gesamthierarchie ausgegeben.
#     if($4=="apparatur") {  ausgabe=1    # Fuer Apparaturhierarchie
#     if($4=="handlung") {   ausgabe=1    # Fuer Handlungshierarchie
#     if($4=="theorie") {    ausgabe=1    # Fuer Theoriehierarchie
     if($1=="instance") {   ausgabe=1     # Fuer Gesamthierarchie
#
#
inst=$2
#
# Hier wird das Aussehen der Knoten der einzelnen Hierarchieebenen festgelegt
#
if(length(inst)==3) printf("%s [shape=polygon,sides=6,peripheries=2,style=filled,
                                                    color=lightgrey];\n",inst)
if(length(inst)==4) printf("%s [shape=circle,style=filled,color=lightgrey];\n",inst)
if(length(inst)==5) printf("%s [shape=box];\n",inst)
if(length(inst)==6) printf("%s [shape=diamond];\n",inst)
if(length(inst)==7) printf("%s [shape=ellipse];\n",inst)
#
# Ende der Knotendefinition
#
we=""
pr=""
```

```
al=""
ha=""
ap=""
th=""
it=""
ih=""
ia=""
                        }
                    }
#
# Hier beginnen die Ausgabe-Anweisungen fuer den Body der dot-Datei.
# Je nach gewuenschter Ausgabe (von Kanten) muessen die folgenden
# Anweisungs-Bereiche kommentiert oder auskommentiert werden.
# Standardeinstellung ist die Ausgabe von 'praezis' und 'weiter'
# Kanten.
#
# Ausgabe von 'weiter'-Kanten
#
if($1=="weiter" && ausgabe==1) {
                  if($3~/\./) {le=length($3)
                               $3=substr($3,1,le-1)}
                  we=$3
                  print inst " -> " we " [label=\"w\"];"
                          if(length(inst)==length(we)) {
                                        arr[length(inst),inst] = inst
                                        arr[length(inst),we] = we
                                                    }
}
#
# Ausgabe von 'praezis'-Kanten
#
if($1=="praezis" && ausgabe==1) {
                 if($3~/\./) {le=length($3)
                              $3=substr($3,1,le-1)}
                 pr=$3
                 print inst " -> " pr " [style=dotted,label=\"p\"];"
}
#
# Ausgabe von 'allgemein'-Kanten
#
#if($1=="allgemein" && ausgabe==1) {
#                  if($3~/\./) {le=length($3)
#                               $3=substr($3,1,le-1)}
#                  al=$3
#                  print inst " -> " al " [label \"a\"];"
#}
#
# Ausgabe von Kanten in Richtung Handlungshierarchie
#
if($1=="handlung" && ausgabe==1) { ha=$3
                print inst " -> " ha " [style=dashed,label=\"ha\"];"
}
#
# Ausgabe von Kanten in Richtung Apparaturhierarchie
#
if($1=="apparatur" && ausgabe==1) {
                   if($3~/kuehlerteil/) $3="kuehlert."
                   if($3~/destillationsblase/) $3="destill.bl."
                   if($3~/heizvorrichtung/) $3="heizvor."
                   if($3~/destillationsapparatur/) $3="destill.app."
                   if($3~/verdampferteil/) $3="verd.teil"
                   if($3~/halter/) $3="halterung\_1"
                 ap=$3
                 print inst " -> " ap " [style=dashed,label=\"ap\"];"
}
#
# Ausgabe von Kanten in Richtung Theoriehierarchie
#
if($1=="theorie" && ausgabe==1) { th=$3
                print inst " -> " th " [style=dashed,label=\"th\"];"
}
#
# Ausgabe von Kanten zu Theorie-Zwischentexten
#
#if($1=="inter_theorie" && ausgabe==1) {
```

```
#                if($3~/\"/) {le=length($3)
#                            $3=substr($3,2,le-2)}
#                it=$3
#                print inst " -> " it " [style=dashed,label=\"it\"];"
#}
#
# Ausgabe von Kanten zu Handlungs-Zwischentexten
#
#if($1=="inter_handlung" && ausgabe==1) {
#                if($3~/\"/) {le=length($3)
#                            $3=substr($3,2,le-2)}
#                ih=$3
#                print inst " -> " ih " [style=dashed,label=\"ih\"];"
#}
#
# Ausgabe von Kanten zu Apparatur-Zwischentexten
#
#if($1=="inter_apparatur" && ausgabe==1) {
#                if($3~/\"/) {le=length($3)
#                            $3=substr($3,2,le-2)}
#                ia=$3
#                print inst " -> " ia " [style=dashed,label=\"ia\"];"
#}
}
END{
# Zum Abschluss werden die Knoten festgelegt, die in derselben Hierarchie-
# ebene ausgegben werden sollen (samerank). Als Standard werden jeweils
# die Knoten mit einer Labellaenge von 4 Zeichen und 5 Zeichen in einer
# Ebene ausgegeben. Dies kann jedoch im folgenden for-Statement geaendert
# werden (durch Variation der in der for-Schleife gesetzten maximalen
# Laenge der beruecksichtigten Knotennamen).
# Die Zusammenfassung der naechst tieferen Hierarchie-Stufe in einer Ebene
# konnte von der verfuegbaren Version von dot nicht bewerkstelligt werden
# (Linux-Version 92c (06-16-94)). Die komplexen Gesamthierarchien muessen ohne
# Gruppierung von Knoten gezeichnet werden
#
for(i=4;i<=5;i++) {
printf("{ rank = same; ")
for(j in arr) {
          if(substr(j,1,1)==i) printf("\"%s\"; ",substr(j,3,i))
            }
printf("}\n")
                 }
print "}"
}
```

Ein Beispiel für die graphische Darstellung eines Teiles der Wissensbasis



Verzeichnis der Arbeiten aus dem Sonderforschungsbereich 245

Heidelberg/Mannheim

Nr. 1 Schwarz, S., Wagner, F. & Kruse, L.: Soziale Repräsentation und Sprache: Gruppenspezifische Wissensbestände und ihre Wirkung bei der sprachlichen Konstruktion und Rekonstruktion geschlechtstypischer Episoden. Februar 1989.

Nr. 2 Wintermantel, M., Laux, H. & Fehr, U.: Anweisung zum Handeln: Bilder oder Wörter. März 1989.

Nr. 3 Herrmann, Th., Dittrich, S., Hornung-Linkenheil, A., Graf, R. & Egel, H.: Sprecherziele und Lokalisationssequenzen: Über die antizipatorische Aktivierung von Wie-Schemata. April 1989.

Nr. 4 Schwarz, S., Weniger, G. & Kruse, L. (unter Mitarbeit von R. Kohl): Soziale Repräsentation und Sprache: Männertypen: Überindividuelle Wissensbestände und individuelle Kognitionen. Juni 1989.

Nr. 5 Wagner, F., Theobald, H., Heß, K., Schwarz, S. & Kruse, L.: Soziale Repräsentation zum Mann: Gruppenspezifische Salienz und Strukturierung von Männertypen. Juni 1989.

Nr. 6 Schwarz, S. & Kruse, L.: Soziale Repräsentation und Sprache: Gruppenspezifische Unterschiede bei der sprachlichen Realisierung geschlechtstypischer Episoden. Juni 1989.

Nr. 7 Dorn-Mahler, H., Grabowski-Gellert, J., Funk-Müldner, K. & Winterhoff-Spurk, P.: Intonation bei Aufforderungen. Teil I: Theoretische Grundlagen. Juni 1989.

Nr. 8 Dorn-Mahler, H., Grabowski-Gellert, J., Funk-Müldner, K. & Winterhoff-Spurk, P.: Intonation bei Aufforderungen. Teil II: Eine experimentelle Untersuchung. Dezember 1989.

Nr. 9 Sommer, C. M. & Graumann, C. F.: Perspektivität und Sprache: Zur Rolle von habituellen Perspektiven. August 1989.

Nr. 10 Grabowski-Gellert, J. & Winterhoff-Spurk, P.: Schreiben ist Silber, Reden ist Gold. August 1989.

Nr. 11 Graf, R. & Herrmann, Th.: Zur sekundären Raumreferenz: Gegenüberobjekte bei nicht-kanonischer Betrachterposition. Dezember 1989.

Nr. 12 Grosser, Ch. & Mangold-Allwinn, R.: Objektbenennung in Serie: Zur partnerorientierten Ausführlichkeit von Erst- und Folgebenennungen. Dezember 1989.

Nr. 13 Grosser, Ch. & Mangold-Allwinn, R.: Zur Variabilität von Objektbenennungen in Abhängigkeit von Sprecherzielen und kognitiver Kompetenz des Partners. Dezember 1989.

Nr. 14 Gutfleisch-Rieck, I., Klein, W., Speck, A. & Spranz-Fogasy, Th.: Transkriptionsvereinbarungen für den Sonderforschungsbereich 245 „Sprechen und Sprachverstehen im sozialen Kontext". Dezember 1989.

Nr. 15 Herrmann, Th.: Vor, hinter, rechts und links: das 6H-Modell. Psychologische Studien zum sprachlichen Lokalisieren. Dezember 1989.

Nr. 16 Dittrich, S. & Herrmann, Th.: „Der Dom steht hinter dem Fahrrad." – Intendiertes Objekt oder Relatum? März 1990.

Nr. 17 Kilian, E., Herrmann, Th., Dittrich, S. & Dreyer, P.: Was- und Wie-Schemata beim Erzählen. Mai 1990.

Nr. 18 Herrmann, Th. & Graf, R.: Ein dualer Rechts-links-Effekt. Kognitiver Aufwand und Rotationswinkel bei intrinsischer Rechts-links-Lokalisation. August 1990.

Nr. 19 Wintermantel, M.: Dialogue between expert and novice: On differences in knowledge and means to reduce them. August 1990.

Nr. 20 Graumann, C. F.: Perspectivity in Language and Language Use. September 1990.

Nr. 21 Graumann, C. F.: Perspectival Structure and Dynamics in Dialogues. September 1990.

Nr. 22 Hofer, M., Pikowsky, B., Spranz-Fogasy, Th. & Fleischmann, Th.: Mannheimer Argumentations-Kategoriensystem (MAKS). Mannheimer Kategoriensystem für die Auswertung von Argumentationen in Gesprächen zwischen Müttern und jugendlichen Töchtern. Oktober 1990.

Nr. 23 Wagner, F., Huerkamp, M., Jockisch, H. & Graumann, C. F.: Sprachlich realisierte soziale Diskriminierungen: empirische Überprüfung eines Modells expliziter Diskriminierung. Oktober 1990.

Nr. 24 Rettig, H., Kiefer, L., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Wenn Versuchspersonen ihre Bezugsskalen selbst konstruieren. November 1990.

Nr. 25 Kiefer, L., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Klassische Urteilseffekte bei individueller Skalenkonstruktion. November 1990.

Nr. 26 Hofer, M., Pikowsky, B., Fleischmann, Th. & Spranz-Fogasy, Th.: Argumentationssequenzen in Konfliktgesprächen zwischen Müttern und Töchtern. November 1990.

Nr. 27 Funk-Müldner, K., Dorn-Mahler, H. & Winterhoff-Spurk, P.: Kategoriensystem zur Situationsabhängigkeit von Aufforderungen im betrieblichen Kontext. Dezember 1990.

Nr. 28 Groeben, N., Schreier, M. & Christmann, U.: Argumentationsintegrität (I): Herleitung, Explikation und Binnenstrukturierung des Konstrukts. Dezember 1990.

Nr. 29 Blickle, G. & Groeben, N.: Argumentationsintegrität (II): Zur psychologischen Realität des subjektiven Wertkonzepts – ein experimenteller Überprüfungsansatz am Beispiel ausgewählter Standards. Dezember 1990.

Nr. 30 Schreier, M. & Groeben, N.: Argumentationsintegrität (III): Rhetorische Strategien und Integritätsstandards. Dezember 1990.

Nr. 31 Sachtleber, S. & Schreier, M.: Argumentationsintegrität (IV): Sprachliche Manifestationen argumentativer Unintegrität – ein pragmalinguistisches Beschreibungsmodell und seine Anwendung. Dezember 1990.

Nr. 32 Dietrich, R., Egel, H., Maier-Schicht, B. & Neubauer, M.: ORACLE und die Analyse des Äußerungsaufbaus. Februar 1991.

Nr. 33 Nüse, R., Groeben, N. & Gauler, E.: Argumentationsintegrität (V): Diagnose argumentativer Unintegrität – (Wechsel-)wirkungen von Komponenten subjektiver Werturteile über argumentative Sprechhandlungen. März 1991.

Nr. 34 Christmann, U. & Groeben, N.: Argumentationsintegrität (VI): Subjektive Theorien über Argumentieren und Argumentationsintegrität – Erhebungsverfahren, inhaltsanalytische und heuristische Ergebnisse. März 1991.

Nr. 35 Graf, R., Dittrich, S., Kilian, E. & Herrmann, Th.: Lokalisationssequenzen: Sprecherziele, Partnermerkmale und Objektkonstellationen (Teil II). Drei Erkundungsexperimente. März 1991.

Nr. 36 Hofer, M., Pikowsky, B., &. Fleischmann, Th.: Jugendliche unterschiedlichen Alters im argumentativen Konfliktgespräch mit ihrer Mutter. März 1991.

Nr. 37 Herrmann, Th., Graf, R. & Helmecke, E.: „Rechts" und „Links" unter variablen Betrachtungswinkeln: Nicht-Shepardsche Rotationen. April 1991.

Nr. 38 Herrmann, Th. & Grabowski, J.: Mündlichkeit, Schriftlichkeit und die nicht-terminalen Prozeßstufen der Sprachproduktion. Februar 1992.

Nr. 39 Thimm, C. & Kruse, L.: Dominanz, Macht und Status als Elemente sprachlicher Interaktion. Mai 1991.

Nr. 40 Thimm, C. & Kruse, L.: Sprachliche Effekte von Partnerhypothesen in dyadischen Situationen. September 1993.

Nr. 41 Thimm, C., Maier, S. & Kruse, L.: Statusrelationen in dyadischen Kommunikationssituationen: Zur Rolle von Partnerhypothesen. April 1994.

Nr. 42 Funk-Müldner, K., Dorn-Mahler, H. & Winterhoff-Spurk, P.: Nonverbales Verhalten beim Auffordern – ein Rollenspielexperiment. Dezember 1991.

Nr. 43 Dorn-Mahler, H., Funk-Müldner, K. & Winterhoff-Spurk, P.: AUFF_{KO} – Ein inhaltsanalytisches Kodiersystem zur Analyse von komplexen Aufforderungen. Oktober 1991.

Nr. 44 Herrmann, Th.: Sprachproduktion und erschwerte Wortfindung. Mai 1992.

Nr. 45 Grabowski, J., Herrmann, Th. & Weiß, P.: Wenn „vor" gleich „hinter" ist – zur multiplen Determination des Verstehens von Richtungspräpositionen. Juni 1992.

Nr. 46 Barattelli, St., Koelbing, H.G. & Kohlmann, U.: Ein Klassifikationssystem für komplexe Objektreferenzen. September 1992.

Nr. 47 Haury, Ch., Engelbert, H. M., Graf, R. & Herrmann, Th.: Lokalisationssequenzen auf der Basis von Karten– und Straßenwissen: Erste Erprobung einer Experimentalanordnung. August 1992.

Nr. 48 Schreier, M. & Czemmel, J.: Argumentationsintegrität (VII): Wie stabil sind die Standards der Argumentationsintegrität ? August 1992.

Nr. 49 Engelbert, H. M., Herrmann, Th. & Haury, Ch.: Ankereffekte bei der sprachlichen Linearisierung. Oktober 1992.

Nr. 50 Spranz-Fogasy, Th.: Bezugspunkte der Kontextualisierung sprachlicher Ausdrücke in Interaktionen. Ein Konzept zur analytischen Konstitution von Schlüsselwörtern. November 1992.

Nr. 51 Kiefer, M., Barattelli, St. & Mangold-Allwinn, R.: Kognition und Kommunikation: Ein integrativer Ansatz zur multiplen Determination der lexikalischen Spezifität der Objektklassenbezeichnung. Februar 1993.

Nr. 52 Spranz-Fogasy, Th.: Beteiligungsrollen und interaktive Bedeutungskonstitution. Februar 1993.

Nr. 53 Schreier, M. & Groeben, N.: Argumentationsintegrität (VIII): Zur psychologischen Realität des subjektiven Wertkonzepts. Eine experimentelle Überprüfung für die 11 Standards integeren Argumentierens. Dezember 1992.

Nr. 54 Sommer, C. M., Freitag, B. & Graumann, C. F.: Aggressive Interaction in Perspectival Discourse. März 1993.

Nr. 55 Huerkamp, M., Jockisch, H., Wagner, F. & Graumann, C. F.: Facetten expliziter sprachlicher Diskriminierung: Untersuchungen von Ausländer-Diskriminierungen anhand einer deutschen und einer ausländischen Stichprobe. Februar 1993.

Nr. 56 Rummer, R., Grabowski, J., Hauschildt, A. & Vorwerg, C.: Reden über Ereignisse: Der Einfluß von Sprecherzielen, sozialer Nähe und Institutionalisiertheitsgrad auf Sprachproduktionsprozesse. April 1993.

Nr. 57 Blickle, G.: Argumentationsintegrität (IX): Personale Antezedensbedingungen der Diagnose argumentativer Unintegrität. Juli 1993.

Nr. 58 Herrmann, Th., Buhl, H.M., Schweizer, K. & Janzen, G.: Zur repräsentationalen Basis des Ankereffekts. Kognitionspsychologische Untersuchungen zur sprachlichen Linearisierung. September 1993.

Nr. 59 Carroll, M.: Keeping spatial concepts on track in text production. A comparative analysis of the use of the concept path in descriptions and instructions in German. Oktober 1993.

Nr. 60 Speck, A.: Instruieren im Dialog. Oktober 1993.

Nr. 61 Herrmann, Th. & Grabowski, J.: Das Merkmalsproblem und das Identitätsproblem in der Theorie dualer, multimodaler und flexibler Repräsentationen von Konzepten und Wörtern (DMF-Theorie). November 1993.

Nr. 62 Rummer, R., Grabowski, J. & Vorwerg, C.: Zur situationsspezifischen Flexibilität zentraler Voreinstellungen bei ereignisbezogenen Sprachproduktionsprozessen. November 1993.

Nr. 63 Christmann, U. & Groeben, N.: Argumentationsintegrität (X): Realisierung argumentativer Redlichkeit und Reaktionen auf Unredlichkeit. November 1993.

Nr. 64 Christmann, U. & Groeben, N.: Argumentationsintegrität (XI): Retrognostische Überprüfung der Handlungsleitung subjektiver Theorien über Argumentationsintegrität bei Kommunalpolitikern/innen. November 1993.

Nr. 65 Schreier, M.: Argumentationsintegrität (XII): Sprachliche Manifestationsformen argumentativer Unintegrität in Konfliktgesprächen. Dezember 1993.

Nr. 66 Christmann, U., Groeben, N. & Küppers, A.: Argumentationsintegrität (XIII): Subjektive Theorien über Erkennen und Ansprechen von Unintegritäten im Argumentationsverlauf. Dezember 1993.

Nr. 67 Christmann, U. & Groeben, N.: Argumentationsintegrität (XIV): Der Einfluß von Valenz und Sequenzstruktur argumentativer Unintegrität auf kognitive und emotionale Komponenten von Diagnose- und Bewertungsreaktionen. Dezember 1993.

Nr. 68 Schreier, M., Groeben, N. & Mlynski, G.: Argumentationsintegrität (XV): Der Einfluß von Bewußtheitsindikatoren und (Un-)Höflichkeit auf die Rezeption argumentativer Unintegrität. Februar 1994.

Nr. 69 Thimm, C., Rademacher, U. & Augenstein, S.: "Power-Related Talk (PRT)": Ein Auswertungsmodell. Januar 1994.

Nr. 70 Kiefer, L., Rettig, H., Sommer, C. M. & Graumann, C. F.: Perspektivität und soziales Urteil: Vier Sichtweisen zum Thema "Ausländerstop". Januar 1994.

Nr. 71 Graumann, C. F.: Discriminatory Discourse. Conceptual and methodological problems. 1994.

Nr. 72 Huerkamp, M.: SAS-Makros zur Analyse und Darstellung mehrdimensionaler Punktekonfigurationen. 1994.

Nr. 73 Galliker, M., Huerkamp, M., Wagner, F. & Graumann, C. F.: Funktionen expliziter sprachlicher Diskriminierung: Validierung der Kernfacetten des Modells sprachlicher Diskriminierung. 1994.

Nr. 74 Buhl, H.M., Schweizer, K. & Herrmann, Th.: Weitere Untersuchungen zum Ankereffekt. April 1994.

Nr. 75 Herrmann, Th.: Psychologie ohne 'Bedeutung'? Zur Wort-Konzept-Relation in der Psychologie. Mai 1994.

Nr. 76 Neubauer, M., Hub, I. & Thimm, C.: Transkribieren mit LaTeX: Transkriptionsregeln, Eingabeverfahren und Auswertungsmöglichkeiten. Mai 1994.

Nr. 77 Thimm, C. & Augenstein, S.: Sprachliche Effekte in hypothesengeleiteter Interaktion: Durchsetzungsstrategien in Aushandlungsgesprächen. Mai 1994.

Nr. 78 Sommer, C. M., Rettig, H., Kiefer, L. & Frankenhauser, D.: "Germany will be one single concrete block ...". Point of View and Reference to Topic Aspects in Adversial Discussions on Immigration. September 1994.

Nr. 79 Maier, S. & Kruse, L.: Ein Design zur Erfassung einer dialogischen Kommunikationssituation: Das Experiment "Terminabsprache". November 1994.

Nr. 80 Grabowski, J.: Schreiben als Systemregulation – Ansätze einer psychologischen Theorie der schriftlichen Sprachproduktion. Oktober 1994.

Nr. 81 Hermanns, F.: Schlüssel-, Schlag- und Fahnenwörter. Zu Begrifflichkeit und Theorie der lexikalischen ≪politischen Semantik≫. Dezember 1994.

Nr. 82 Kiefer, L., Rettig, H., Frankenhauser, D., Sommer, C. M. & Graumann, C. F.: Perspektivität und Persuasion: Effektivität perspektivenrelevanter Persuasionsstrategien. Dezember 1994.

Nr. 83 Liebert, W.-A.: Das analytische Konzept "Schlüsselwort" in der linguistischen Tradition. Dezember 1994.

Nr. 84 Buhl, H. M., Schweizer, K. & Herrmann, Th.: Der Einfluß von Räumlichkeit und Reizmodalität auf den Ankereffekt. Dezember 1994.

Nr. 85 Koelbing, H.G., Mangold-Allwinn, R., Barattelli, St., Kohlmann, U. & Stutterheim, C. v.: Welchen Einfluß hat der Ausführende auf den Instruierenden ? Dezember 1994.

Nr. 86 Held, Th. & Maier-Schicht, B.: Benutzerhandbuch und Dokumentation eines Experimentalsystems auf der Basis der Expertensystemschale knoX. Dezember 1994.

Nr. 87 Maier-Schicht, B., Theiss, G. & Held, Th.: Ein Expertensystem als Experimentalsystem. Februar 1995.