Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

Document
D-91-14

KRIS :

Knowledge Representation and Inference System

- Benutzerhandbuch -

**Erich Achilles, Bernhard Hollunder,
Armin Laux, Jörg-Peter Mohren,**

September 1991

Deutsches Forschungszentrum für Künstliche Intelligenz GmbH

Postfach 20 80
D-6750 Kaiserslautern
Tel.: (+49 631) 205-3211/13
Fax: (+49 631) 205-3210

Stuhlsatzenhausweg 3
D-6600 Saarbrücken 11
Tel.: (+49 681) 302-5252
Fax: (+49 681) 302-5341

Deutsches Forschungszentrum für Künstliche Intelligenz

The German Research Center for Artificial Intelligence (Deutsches Forschungszentrum für Künstliche Intelligenz, DFKI) with sites in Kaiserslautern und Saarbrücken is a non-profit organization which was founded in 1988 by the shareholder companies ADV/Orga, AEG, IBM, Insiders, Fraunhofer Gesellschaft, GMD, Krupp-Atlas, Mannesmann-Kienzle, Philips, Siemens and Siemens-Nixdorf. Research projects conducted at the DFKI are funded by the German Ministry for Research and Technology, by the shareholder companies, or by other industrial contracts.

The DFKI conducts application-oriented basic research in the field of artificial intelligence and other related subfields of computer science. The overall goal is to construct *systems with technical knowledge and common sense* which - by using AI methods - implement a problem solution for a selected application area. Currently, there are the following research areas at the DFKI:

- Intelligent Engineering Systems
- Intelligent User Interfaces
- Intelligent Communication Networks
- Intelligent Cooperative Systems.

The DFKI strives at making its research results available to the scientific community. There exist many contacts to domestic and foreign research institutions, both in academy and industry. The DFKI hosts technology transfer workshops for shareholders and other interested groups in order to inform about the current state of research.

From its beginning, the DFKI has provided an attractive working environment for AI researchers from Germany and from all over the world. The goal is to have a staff of about 100 researchers at the end of the building-up phase.

Prof. Dr. Gerhard Barth
Director

KRIS : *K*nowledge *R*epresentation and *I*nference *S*ystem
- Benutzerhandbuch -

Erich Achilles, Bernhard Hollunder, Armin Laux, Jörg-Peter Mohren

DFKI-D-91-14

Diese Arbeit wurde finanziell unterstützt durch das Bundesministerium für Forschung und Technologie (FKZ ITW-8903 0).

KRIS : *K*nowledge *R*epresentation and *I*nference *S*ystem
– Benutzerhandbuch –

Erich Achilles Bernhard Hollunder
Armin Laux Jörg-Peter Mohren

Deutsches Forschungszentrum für Künstliche Intelligenz (DFKI)
Projektgruppe WINO
Stuhlsatzenhausweg 3
6600 Saarbrücken 11, Germany
E-mail: {beho,laux}@dfki.uni-sb.de

Zusammenfassung

Dieses Dokument gibt eine Einführung in das Arbeiten mit dem *KRIS*-System.

KRIS ist ein Wissensrepräsentationssystem der KL-ONE Familie, welches in der Projektgruppe WINO am DFKI entwickelt und implementiert worden ist. *KRIS* stellt dem Benutzer eine mächtige Konzeptsprache für die Definition von Terminologien zur Verfügung; die assertionale Sprache ist vergleichbar mit denen anderer KL-ONE Systeme. Im Unterschied zu allen anderen KL-ONE Systemen sind die typischen Inferenzen wie z.B. Subsumtion, Konsistenztest etc. durch korrekte und *vollständige* Algorithmen realisiert.

Kenntnisse der zugrundeliegenden Konzeptsprachen, der assertionalen Sprache, sowie den Inferenzmöglichkeiten werden vorausgesetzt. Eine Einführung gibt z.B. [1].

Inhaltsverzeichnis

1 Vorbemerkungen

Im folgenden wird mit <...> eine Taste dargestellt, z.B. <RETURN>. Mit

- *<Taste-1><Taste-2>*
  wird dabei das aufeinanderfolgende, und mit
- *<Taste-1> + <Taste-2>*
  das gleichzeitige Drücken von *Taste-1* und *Taste-2*

abgekürzt.

1.1 Installieren des Systems

Die $\mathcal{KRIS}$-Dateien befinden sich auf einem externen Datenträger (Diskette, Band, etc.) und werden von dort auf die Symbolics-Lisp-Maschine in ein eigenes Verzeichnis „KRIS"kopiert.[1] Die gegebene Unterverzeichnis-Struktur darf dabei nicht verändert werden.

Auf dem Datenträger befinden sich sowohl die Quelldateien (Endung `.lisp`), als auch die kompilierten Versionen (Endung `.ibin`). Um ein lauffähiges $\mathcal{KRIS}$-System zu installieren reicht es i.a. aus nur die kompilierten Dateien zu kopieren, die auf einem Mac-Ivory unter Genera 7.4 übersetzt wurden. Bei Kompatibilitätsproblemen können alternativ die Quelldateien kopiert und anschließend neu kompiliert werden. Dazu kann man die Lisp-Funktion `compile-files` aus der Datei compile-files verwenden, die automatisch alle $\mathcal{KRIS}$-Dateien übersetzt.

Das $\mathcal{KRIS}$-System wird durch Laden der Datei initialize_kris gestartet. Wird das Kommando (`load "initialize_kris"`) als Befehl in die Datei lispm-init.lisp eingetragen, kann man bei jedem Start des Lisp-Systems menügesteuert entscheiden, ob $\mathcal{KRIS}$ geladen werden soll.

1.2 Starten des Systems

Das gesamte $\mathcal{KRIS}$-System ist auf einer *Symbolics-Lisp-Maschine* implementiert. Nach dem Anmelden mit dem Login-Kommando und dem Laden der Datei initialize_kris erscheint ein Menü (siehe Abbildung 1). Dieses erlaubt dem Benutzer

- die Systemdateien zu laden,

[1] Zum Beipiel von einer „Mac"-Diskette auf die Lisp-Maschine durch den Lisp-Befehl `:copy file Host:` *Diskettenname*`:**:*.*` *Zielpfad.*

- den Pfadnamen der Systemdateien setzen[2], oder
- das Menü zu verlassen ohne die Systemdateien zu laden.

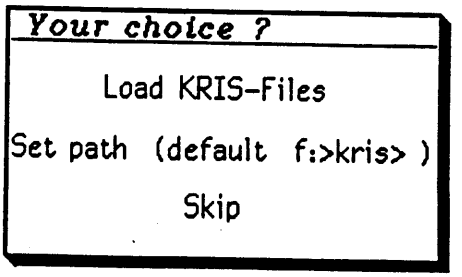


Abbildung 1: Das *KRIS*-Startmenü.

Nach Auswahl des ersten Punktes werden alle notwendigen Systemdateien geladen und anschließend wird automatisch in die *KRIS*-Oberfläche gewechselt. Mit den folgenden Select-Kommandos kann nun zwischen

- der Lisp-Oberfläche (<SELECT><L>),
- dem File-System (<SELECT><F>),
- dem Editor (<SELECT><E>),
- der *KRIS*-Oberfläche (<SELECT><K>), sowie
- einem vergrößerten Graphik-Fenster (<SELECT><G>)

gewechselt werden.

Nach Verlassen des Systems kann *KRIS* jederzeit mit <SELECT><K> wieder aktiviert werden. Außerdem besteht die Möglichkeit das System mittels der Lisp-Funktion `(start-kris)` vollständig neu zu initialisieren und zu laden.

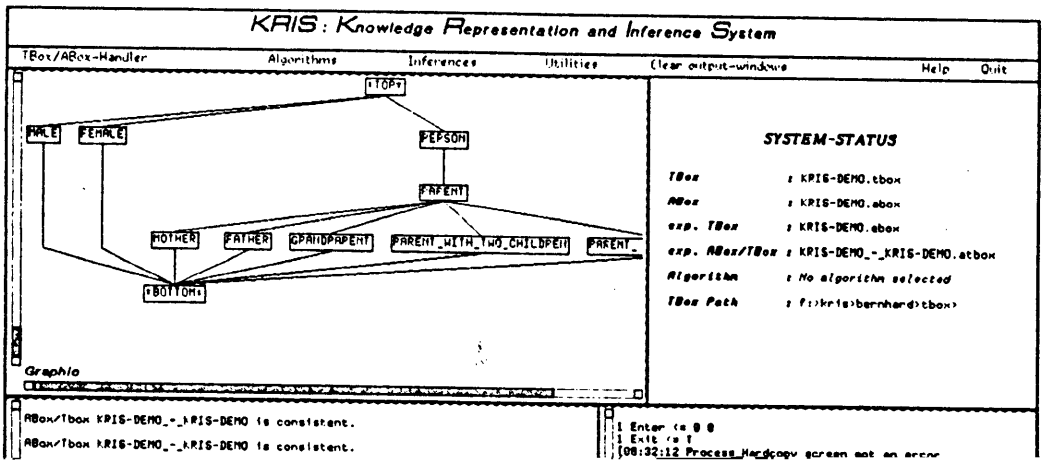
1.3 Die *KRIS*-Oberfläche

Nach dem Laden von *KRIS* wird automatisch in die *KRIS*-Oberfläche gewechselt, welche in folgende Fenster aufgeteilt ist (siehe Abbildung 2):

- eine *Kommando-Leiste* (oberes Fenster): Hier erscheinen jeweils die Kommandos, die dem Benutzer zur Verfügung stehen und *per Maus* ausgewählt werden können.

[2]Man beachte, daß hier ein Verzeichnisname angeben werden muß, z.B. f:>kris>.

- ein *Graphik-Fenster* (Graphic window): Hier wird der Subsumtionsgraph ausgegeben.
- ein *Status-Fenster:* Hier werden die aktuell geladene TBox bzw. ABox, der ausgewählte Algorithmus und der Pfadname angezeigt.
- ein *Ausgabe-Fenster* (Output window): Hier werden die System- und Fehlermeldungen, sowie Ergebnisse ausgegeben.
- ein *Eingabe-Fenster* (Commands window): Hier können Kommandos – alternativ zum Anklicken in der Kommando-Leiste – *per Tastatur* eingegeben werden. Außerdem erscheinen hier die Menüs, die dem Benutzer die Auswahl z.B. für das Laden, Löschen und Edieren von Dateien erlauben.



1.4 Benutzerhilfe

In fast allen Kommando-Leisten gibt es ein Help-Kommando. Nach Anklikken erscheint im Ausgabe-Fenster ein kontextbezogener Hilfstext.

1.5 Ausblick auf die folgenden Kapitel

Zunächst, im 2. Kapitel, wird im einzelnen auf die *KRIS*-Befehle in den Kommando-Leisten eingegangen. Im 3. Kapitel wird eine Beispielsitzung beschrieben. Syntax und Semantik der zugrundeliegenden Konzeptsprachen werden in Anhang A, ein Überblick über alle Systemdateien bzw. alle Befehle von *KRIS* in Anhang B und C gegegeben.

2 Die Kommando-Leisten

Das Arbeiten mit *KRIS* erfolgt hauptsächlich durch Anklicken der in der Kommando-Leiste enthaltenen Befehle oder alternativ durch die entsprechenden Eingaben im Eingabe-Fenster. Die Kommandos der obersten Ebene aktivieren i.a. neue Kommando-Leisten, welche dann an Stelle der übergeordneten Leiste erscheinen.

Achtung: Beim *ersten* Wechsel in eine neue Kommando-Leiste muß anschließend <FUNCTION><REFRESH> eingegeben werden; erst dann erscheinen die Befehle dieser Kommando-Leiste.

Das Aufrufen von Kommandos im Eingabe-Fenster erfolgt durch Eingabe des in der Kommando-Leiste stehenden Befehls. Einzige Ausnahme ist der Befehl „Expand Tbox & ABox", der durch `Expand Tbox and ABox` eingegeben wird.[3] Wird ein Befehl in dem Eingabe-Fenster eingegeben, so können diese Eingaben, genau wie im Lisp-System, durch die Leertaste komplettiert werden, wenn die eingebene Zeichenkette eindeutig zu einem *KRIS*-Befehl vervollständigt werden kann.

2.1 Die Haupt-Kommando-Leiste

Die Haupt-Kommando-Leiste enthält die folgenden Befehle (siehe auch Abbildung 3):

- *TBox/ABox-Handler:* Erlaubt das Verwalten (Erstellen, Modifizieren und Löschen) von TBox- und ABox-Dateien.
- *Algorithms:* Auswahl eines Algorithmus.

[3] Per Tastatur können Befehle aus *beliebigen* Kommando-Leisten direkt – unabhängig von der *aktuellen* Kommando-Leiste – eingegeben werden.

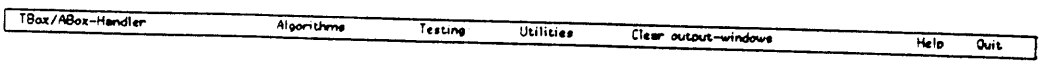


Abbildung 3: Die Haupt-Kommando-Leiste.

- *Inferences:* Auswahl und Berechnung von Inferenzen.
- *Utilities:* Laufzeitmessung von Algorithmen etc.
- *Clear Output-Windows:* Löschen der Fenster.
- *Help:* Informationen über $\mathcal{KRIS}$.
- *Quit:* Verlassen des $\mathcal{KRIS}$-Systems.

Nach Auswahl von Quit wird das System verlassen, und der Benutzer befindet sich in der Anwendung (z.B. Editor, File-System), von der aus das $\mathcal{KRIS}$-System aufgerufen wurde. Mit <SELECT><K> kann jederzeit das $\mathcal{KRIS}$-System wieder aktiviert werden.

2.2 TBox/ABox-Handler

Dem Benutzer stehen in dieser Kommando-Leiste folgende Möglichkeiten zur Verfügung:

- *Create TBox*
- *Create ABox*
- *Edit TBox*
- *Edit ABox*
- *Load File*
- *Delete File*
- *Expand TBox*
- *Expand TBox & ABox*
- *Help*
- *Quit*

Erstellen einer neuen TBox bzw. ABox

Mit *Create TBox* bzw. *Create ABox* kann der Benutzer eine neue TBox bzw. ABox erstellen. Nach Eingabe des Namens für die neu zu erstellende Datei (die Endung `.tbox` bzw. `.abox` wird automatisch angehängt), wird ein Editor-Fenster geöffnet, in dem zunächst die folgenden Einstellungen vorgenommen werden müssen:

- mit <META> + <X> `set package` <RETURN> `KRIS` <RETURN> wird die Datei dem Package KRIS zugewiesen,
- mit <META> + <X> `lisp mode` <RETURN> wird der Lisp-Modus für diese Datei eingestellt.

Nun kann die neu erzeugte Datei editiert werden.

Edieren einer TBox bzw. Abox

Nach Anklicken von *Edit TBox* bzw. *Edit ABox* erscheint ein Editor-Fenster mit der aktuell geladenen unexpandierten TBox bzw. ABox. Ist keine TBox bzw. ABox geladen, so wird dies gemeldet und es erscheint – genau wie beim Laden einer Datei – ein Menü im Eingabe-Fenster, welches die Auswahl einer TBox bzw. ABox aus dem aktuellen Pfad zum Laden in einen Editor-Buffer erlaubt. Es wird der Zmacs-Editor der Symbolics-Lisp-Maschine verwendet.

Die so erstellte TBox bzw. ABox muß anschließend mit

<CONTROL> + <X><Control> + <S>

abgespeichert werden. Nun kann mit <SELECT><K> wieder in die *KRIS*-Oberfläche gewechselt werden. Nach jedem Wechsel wird automatisch ein Syntax-Check durchgeführt, der auch einen Test auf zyklische Definitionen enthält.

Bei mehrfach benutzten Konzept-, Rollen- oder Attributenamen wird eine Warnung ausgegeben. Bei syntaktisch korrekten TBoxes bzw. ABoxes wird eine Liste der verwendeten Konstrukte ausgegeben; weiterhin wird angezeigt welche Konzepte, Rollen und Attribute in der TBox definiert worden sind.

Syntaktische Fehler in der aktuellen TBox bzw. ABox werden gemeldet, so daß der Benutzer anschließend die Möglichkeit hat, diese im Editor zu beseitigen.

Laden einer TBox bzw. ABox

Nach Anklicken von *Load File* erscheint im Eingabe-Fenster ein Menü mit allen unexpandierten und expandierten Dateien im aktuellen Verzeichnis (siehe Abbildung 4).

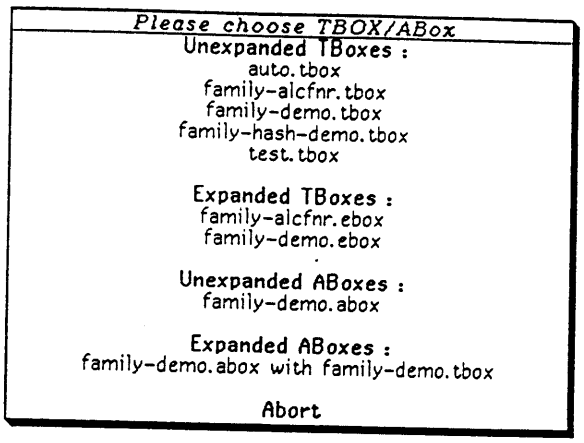


Abbildung 4: Laden eines Files.

Unterschieden werden die Dateien durch die Endungen

.tbox für unexpandierte TBoxes

.ebox für expandierte TBoxes

.abox für unexpandierte ABoxes

.atbox für expandierte ABoxes

Enthält das Menü mehr Dateien als dargestellt werden können, gibt es die Möglichkeit mit der Maus zu „scrollen“: Bewegt man die Maus auf den linken Menü-Rand, so erscheint dort ein schmaler Scroll-Balken. Wie üblich kann man nun mittels der Maustasten die Menü-Einträge verschieben.

Beim Laden einer expandierten TBox (Endung .ebox) wird automatisch die entsprechende unexpandierte TBox geladen; beim Laden einer expandierten ABox (Endung .atbox) die entsprechende unexpandierte TBox, die expandierte TBox sowie die unexpandierte Abox. Beim Laden von Dateien

werden die unexpandierten TBoxes bzw. ABoxes automatisch auf syntaktische Korrektheit getestet (siehe oben).

Löschen einer TBox bzw. ABox

Nach Anklicken von *Delete File* erscheint ebenfalls ein Menü mit allen im aktuellen Verzeichnis enthaltenen Dateien. Mit der Maus kann eine Datei zum Löschen ausgewählt werden.

Expandieren einer TBox

Das Expandieren der aktuellen TBox erfolgt durch den Befehl *Expand TBox*. Eine Datei *dateiname*.tbox wird dabei expandiert und das Ergebnis in die Datei *dateiname*.ebox geschrieben. Falls bereits eine expandierte Version der aktuellen TBox expandiert vorhanden ist, wird nicht expandiert; im Ausgabe-Fenster erscheint eine entsprechende Meldung.

Expandieren einer ABox bzgl. einer TBox

Das Expandieren der aktuellen ABox bzgl. der aktuellen TBox erfolgt durch den Befehl *Expand TBox & ABox*. Hierbei wird zu einer ABox *ABox-name*.abox und einer TBox *TBox-name*.tbox eine Datei *ABox-name_-_TBox-name*.atbox erzeugt.

Setzen des Pfadnamens

Der Befehl *Set Path* erlaubt das Setzen des aktuellen Verzeichnisses.

2.3 Die Algorithmen-Auswahl

Das $\mathcal{KRIS}$-System stellt zur Zeit Algorithmen für die Sprachen

- $\mathcal{ALC}$
- $\mathcal{ALCN}$
- $\mathcal{ALCFN}$
- $\mathcal{ALCFNR}$

zur Verfügung. Den ersten drei Algorithmen liegt die „Trace"-Methode, beschrieben in [3,2], zugrunde. Der Algorithmus für die Sprache $\mathcal{ALCFNR}$ ist regelbasiert implementiert (siehe [2]).

Nachdem ein Algorithmus ausgewählt wurde wird getestet, ob die aktuelle TBox bzw. ABox mit diesem Algorithmus korrekt bearbeitet werden kann, d.h. ob mit diesem alle in der Eingabe vorkommenden Sprachkonstrukte behandelt werden können. Ist dies nicht der Fall, so wird eine Warnung ausgegeben.

2.4 Inferenzen

Die Inferenzmöglichkeiten, die von $\mathcal{KRIS}$ zur Verfügung gestellt werden, sind in Abbildung 5 dargestellt.

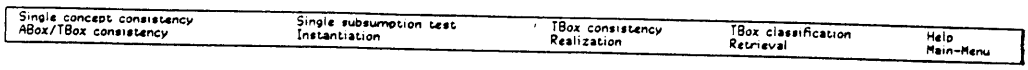


Abbildung 5: Die Kommando-Leiste für die *KRIS*-Inferenzen.

- *Single concept consistency:* Im Eingabe-Fenster erscheint ein Menü mit allen Konzepten der aktuellen TBox (siehe Abbildung 6). Aus diesen kann der Benutzer ein Konzept auswählen, das dann auf Konsistenz getestet wird. Die Ausgabe erfolgt, wie auch bei den folgenden Befehlen, im Ausgabe-Fenster.

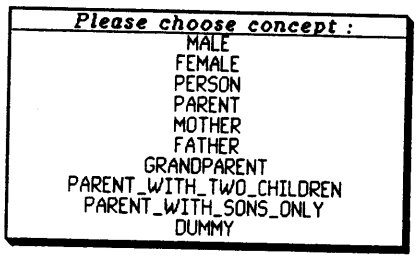


Abbildung 6: Auswahl eines Konzeptes.

- *Single subsumption test:* Hier erscheint ebenfalls ein Menü mit allen Konzepten der aktuellen TBox, aus dem nacheinander zwei Konzepte ausgewählt werden. Es wird getestet, ob das erste Konzept das zweite Konzept subsumiert.

- *TBox consistency:* Die aktuelle TBox wird auf Konsistenz getestet, d.h. es wird geprüft, ob jedes darin enthaltene Konzept konsistent ist.

- *TBox classification:* Hier wird die Subsumtionshierarchie der aktuellen TBox berechnet. Ausgegeben wird diese Hierarchie in dem Graphik-Fenster (siehe Abbildung 7). Mit <SELECT><G> kann auf ein grösseres Ausgabe-Fenster umgeschaltet werden.

- *ABox/TBox consistency:* Die aktuelle ATBox wird auf Konsistenz getestet, d.h. es wird geprüft, ob die aktuelle ABox bzgl. der aktuellen TBox konsistent ist.

- *Instantiation:* Im Eingabe-Fenster muß ein Individuum und eine Konzeptbeschreibung eingegeben werden.[4] Alternativ zu der Konzeptbeschreibung kann auch ein Konzeptname der aktuellen TBox eingegeben

[4]Beachte, daß weder das Individuum, noch die Konzeptbeschreibung in der aktuellen ATBox enthalten sein müssen.

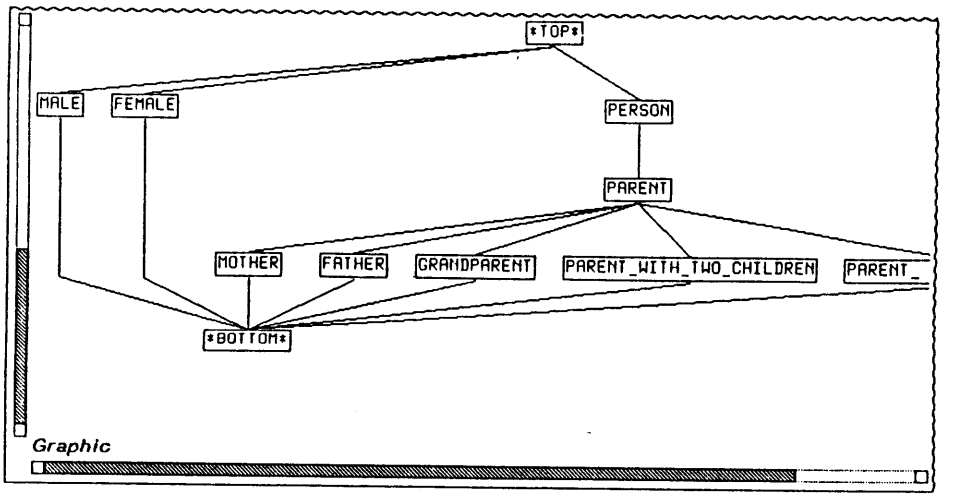


Abbildung 7: Ausgabe eines Subsumtionsgraphen.

werden. Folgt aus der aktuellen ATBox, daß das Individuum Instanz der Konzeptbeschreibung ist, so liefert dieser Test *true*, ansonsten *false*.

- *Realization:* Berechnet für *ein* oder alternativ für *alle* Individuen der ABox die spezifischsten Konzepte der Tbox, die das Individuum bzw. die Individuen enthalten.
  Um diese Inferenz durchführen zu können, muß die Subsumtionshierarchie bereits berechnet worden sein. Mit der „Bottom-up"- bzw. „Top-down"-Strategie werden die spezifischsten Konzepte in dieser Hierarchie von unten nach oben, bzw. von oben nach unten gesucht.

- *Retrieval:* Bestimmt für eine Konzeptbeschreibung oder alternativ für einen Konzeptnamen der aktuellen TBox alle Individuen der ABox, die Instanz dieses Konzeptes sind.
  Um diese Inferenz durchführen zu können, muß die Realization für *alle* Individuen der ABox schon berechnet worden sein.

Bemerkung: Zur Durchführung der ersten vier Befehle muß eine EBox geladen sein, für die letzten vier eine ATBox.

2.5 Die Utilities-Kommando-Leiste

Hier hat der Benutzer die Möglichkeit zur Zeitmessung der verschiedenen Inferenzalgorithmen. Zusätzlich können Konzepte in expandierter bzw. simplifizierter Form angezeigt werden, sowie die Algorithmen anhand von benutzerdefinierten Beispielen auf Korrektheit getestet werden. Die Utilities-Kommando-Leiste beinhaltet die folgenden Befehle:

- Show expanded concept
- Show simplified concept
- Run KRIS with time-protocol
- Check TBoxes
- Check ATBoxes

Bei den ersten drei Punkten muß eine unexpandierte TBox geladen sein, für die beiden letzten eine EBox bzw. eine ATBox.

Show expanded concept

Hier kann sich der Benutzer die expandierte Form eines Konzeptes anzeigen lassen.

Show simplified concept

Hier kann sich der Benutzer die simplifizierte Form, d.h. die Negations-Normalform des Konzeptes, anzeigen lassen.

Run KRIS with time-protocol

Hier werden die einzelnen Zeiten für Expandieren, Simplifizieren und Konsistenztest aller Konzepte der aktuellen TBox aufgelistet.[5]

Check TBoxes

Zum Testen der Algorithmen kann man zu einer TBox *dateiname*.tbox eine Ergebnis-Datei *dateiname*.ergtbox erstellen, in der zu allen (oder einigen) Konzepten der TBox ein Eintrag der folgenden Art steht:

`(konzeptname t)` oder `(konzeptname nil)`

Bei Testen eines Algorithmus werden die berechneten Ergebnisse des Konsistenztestes mit den benutzerspezifizierten Werten verglichen. Stimmt ein Eintrag in der Ergebnis-Datei mit dem entsprechenden Ergebnis des Konsistenztestes *nicht* überein, so erscheint eine Fehlermeldung.

[5] Bedingt durch die Zeitmessung der Symbolics-Lisp-Maschine wird bei sehr geringen Laufzeiten (< 0.008 sec.) der Wert 0.0 ausgegeben.

Check ATBoxes

Hier kann ähnlich wie oben zu einer Datei *dateiname*.atbox eine Ergebnis-Datei *dateiname*.ergatbox erstellt werden. Die Einträge sind von der Form:

```
(konzeptname (individuum-1 ... individuum-n))
```

Wird eine benutzerspezifizierte Instanz „individuum-i in konzeptname" von dem zu testenden Algorithmus *nicht* berechnet, so wird dies ausgegeben.

2.6 Das Status-Fenster

Hier werden die folgenden Systeminformationen angezeigt (Abbildung 8):

- Name der aktuellen TBox
- Name der aktuellen ABox
- Name der aktuellen EBox
- Name der aktuellen ATBox
- Name des aktuellen Algorithmus
- der aktuelle Pfadname

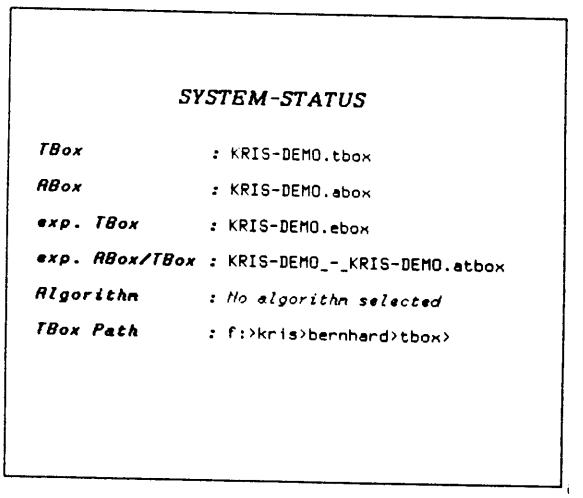


Abbildung 8: Das Status Fenster.

Nach jeder Änderung (z.B. Auswahl eines anderen Algorithmus) wird das Status-Fenster aktualisiert.

2.7 Die Clear-Windows-Kommando-Leiste

Hier hat der Benutzer die Möglichkeit, die Anzeige eines bestimmten oder aller Ausgabe-Fenster zu löschen:

- *Graphics-Window* : löscht den Inhalt des Graphik-Fensters
- *Output-Window* : löscht den Inhalt des Ausgabe-Fensters
- *Commands-Window* : löscht den Inhalt des Eingabe-Fensters
- *All output windows* : löscht den Inhalt aller Fenster

3 Beispielsitzung

Im folgenden wird anhand eines Ablaufprotokolls eine Rechner-Sitzung mit $\mathcal{KRIS}$ exemplarisch beschrieben. Es sollen die folgenden Aktionen ausgeführt werden:

- Erstellen einer TBox
- Expandieren der TBox
- Berechnung von Inferenzen auf der TBox
- Erstellen einer ABox
- Berechnung von Inferenzen auf der ABox

Angenommen die Systemdateien sind bereits geladen und wir befinden uns in der $\mathcal{KRIS}$-Oberfläche.

Zunächst wollen wir eine TBox erstellen. Dazu wählen wir in der Haupt-Kommando-Leiste den Befehl *TBox/ABox Handler* aus. Wie schon oben erwähnt, erscheint die neue Kommando-Leiste erst nach der Eingabe von <FUNCTION><REFRESH>. Nun wählen wir den Befehl *Create TBox* und im Eingabe-Fenster erscheint:

```
Create TBox:  (Enter new Filename:)
```

Hier wird nun der TBox-Name `demo` (ohne Endung `.tbox`) eingegeben. So kommen wir in den Editor und erstellen dort die TBox aus Abbildung 9.

Nach dem Abspeichern der TBox wechseln wir mit <SELECT><K> wieder in die $\mathcal{KRIS}$-Oberfläche. Wie nach jedem Wechsel vom Editor zurück zu $\mathcal{KRIS}$ ist dies mit <RETURN> zu bestätigen.

Angenommen, es haben sich beim Eintippen folgende Fehler eingeschlichen:

```
(defprimattribute sex)
(defprimconcept male)
(defprimconcept female (not male))
(defprimconcept person (some sex (or male female)))
(defprimrole child)
(defconcept parent (and person (some child person)))
(defconcept mother (and parent (some sex female)))
(defconcept father (and parent (not mother)))
(defconcept grandparent (and parent (some child parent)))
(defconcept parent_with_two_children (and parent (atleast 2 child)))
(defconcept parent_with_sons_only (and parent
                                       (all child (some sex male))))
```

Abbildung 9: Unsere Beispiel-TBox.

1. in der 1. Zeile steht ein nichterlaubtes Symbol, z.B. ein „*“.
2. bei der Definition des Konzepts `parent` wurde in `defconcept` das erste `c` vergessen.
3. bei der Definition von `mother` wurde `ans` statt `and` eingegeben.

Dann gibt der Syntax-Checker die folgenden Fehlermeldungen aus:

```
———— TBOX - ERRORS ————
Error in line 1:
unexpected Symbol *
Error in line 7:
unexpected Definition-Symbol <DEFONCEPT>
Error in definiton of Concept <MOTHER> in line 8:
Unknown operator.  <ANS> is none of <AND, OR, ...>
```

Auf die Frage

```
Quit Editor (Yes or No ?):
```

antworten wir mit No, um die Syntaxfehler [illegible]

```
AND OR ALL NOT ATLEAST SOME
```

Introduced concepts:
`male female person parent mother father grandparent parent-with-two-children parent-with-sons-only`

Introduced roles:
`child`

Introduced features:
`sex`

Die so erstellte TBox wird jetzt mit dem Befehl *Expand TBox* aus der Kommando-Leiste expandiert. Die expandierte Version wird automatisch abgespeichert.

Um Inferenzen ausführen zu können, geben wir im Eingabe-Fenster den Befehl `ALCFN` ein. Damit haben wir einen passenden Algorithmus ausgewählt (ohne zuerst in die Haupt-Kommando-Leiste und dann in das Algorithmen-Menü wechseln zu müssen).

Um ein Konzept aus der erstellten TBox auf Konsistenz zu testen, geben wir den Befehl *Single concept consistency* ein. Aus dem nun im Eingabe-Fenster erscheinenden Menü wählen wir z.B. das Konzept `mother` aus; im Ausgabe-Fenster erhalten wir die Meldung

```
Concept MOTHER in consistent.
```

Das Klassifizieren der TBox, d.h. das Berechnen der Subsumtionshierarchie, erreichen wir durch Eingabe von *TBox classification.* Im Graphik-Fenster erscheint die Ausgabe aus Abbildung 10.

Nun wollen wir die ABox aus Abbildung 11 erstellen. Dazu wählen wir den Befehl *Create ABox.*

Nach dem Abspeichern erscheint (bei korrekter Eingabe) die folgende Meldung:

```
———— ABox has correct syntax ————
```

Um nun die erstellte ABox zusammen mit der schon verhandenen TBox zu kombinieren, geben wir den Befehl *Expand Tbox and ABox* ein. Danach haben wir dann die Möglichkeit, Inferenzen auf beiden Wissenskomponenten durchzuführen.

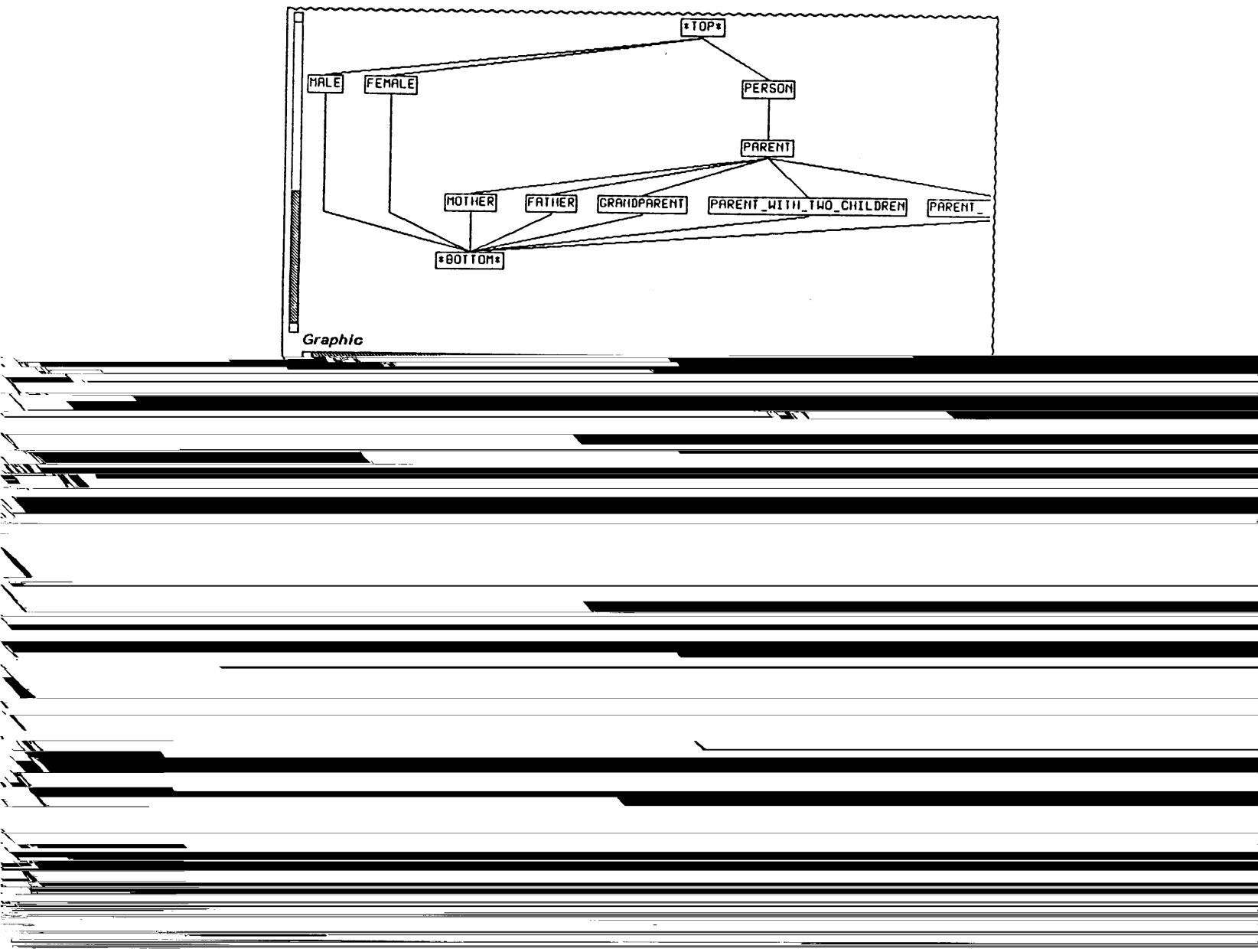


Abbildung 10: Die Subsumtionshierarchie der Beispiel-TBox.

```
(assert-ind Tom father)
(assert-ind Tom Peter child)
(assert-ind Tom Harry child)
(assert-ind Mary parent_with_sons_only)
(assert-ind Mary Tom child)
(assert-ind Mary Chris child)
```

Abbildung 11: Unsere Beispiel-ABox.

Mit *ABox/TBox consistency* lassen wir uns zunächst bestätigen, daß unsere ABox bzgl. der TBox konsistent ist.

Jetzt interessieren wir uns dafür, welches die spezifischsten Konzepte für das Individuum `Mary` aus unserer ABox sind, und wählen daher den Befehl *Realization*. Um später eine *Retrieval*-Operation ausführen zu können (die eine vollständig berechnete Realization voraussetzt), klicken wir in dem nun erscheinenden Menü zunächst `Every individual` an, so daß die spezifischsten Konzepte aus der TBox für *jedes* Individuum der ABox bestimmt

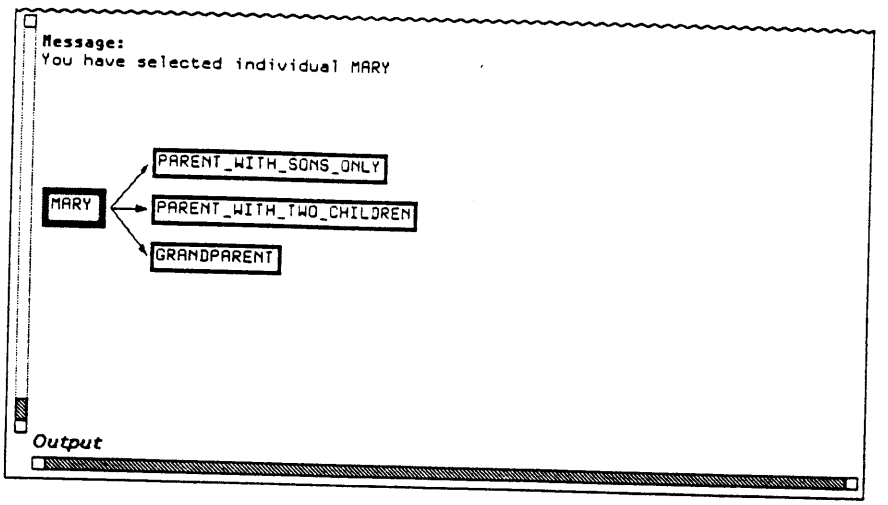


Abbildung 12: Das Ergebnis des Realization-Befehls.

Konzept durch Anklicken aus. Der Retrieval-Algorithmus stellt fest, daß in der aktuellen ABox **Mary** das einzige Individuum ist, daß in **Grandparent** enthalten ist, und wir erhalten die Ausgabe wie in Abbildung 13 angegeben.

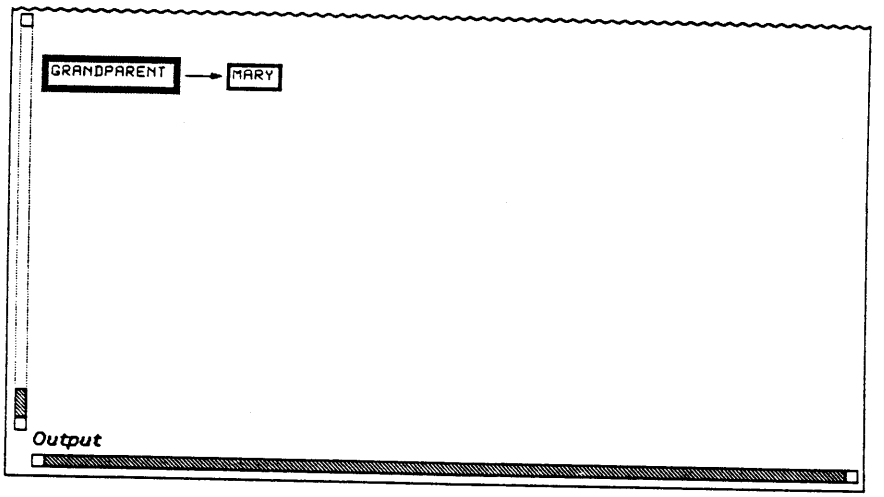


Abbildung 13: Das Ergebnis unserer Retrieval-Anfrage.

Wir beenden damit unsere Beispielsitzung und können das $\mathcal{KRIS}$-System jetzt entweder mittels *Quit* verlassen, oder einfach mit <SELECT><...> eine andere Anwendung auswählen.

4 Anhang A: Syntax und Semantik

Im folgenden werden die Syntax und Semantik der Konzeptsprachen, TBoxes und Aboxes beschrieben.

Syntax der Konzeptsprachen

Ausgehend von drei disjunkten Mengen von Symbolen, den *Konzept-*, *Rollen-* und *Attribut*-Namen, werden die Mengen der

- Konzept-Terme,
- Rollen-Terme und
- Attribut-Terme

wie folgt induktiv definiert: Jeder Konzept-Name ist ein Konzept-Term, jeder Rollen-Name ist ein Rollen-Term und jeder Attribut-Name ist ein Attribut-Term. Die beiden ausgezeichneten Konzept-Namen *top* und *bottom* sind spezielle Konzept-Terme (das *Top-Konzept* bzw. das *Bottom-Konzept*). Seien nun C, $C_1, \ldots, C_k$ Konzept-Terme, R, $R_1, \ldots, R_l$ Rollen-Terme, sowie f, g, $f_1, \ldots, f_m$ Attribut-Terme, und sei n eine nicht-negative ganze Zahl. Dann sind

(and $C_1 \ldots C_k$),	(conjunction)
(or $C_1 \ldots C_k$),	(disjunction)
(not C),	(negation)
(all R C), (all f C),	(value restriction)
(some R C), (some f C),	(exists restriction)
(atleast n R)	(number restrictions)
(atmost n R)	
(equal f g),	(agreement)
(not-equal f g)	(disagreement)

Konzept-Terme,

(and $R_1 \ldots R_l$),	(role conjunction)

ist ein Rollen-Term und

(and $f_1 \ldots f_m$),	(attribute conjunction)
(compose $f_1 \ldots f_m$)	(composition)

sind Attribut-Terme.

Syntax der TBoxes

Namen für Konzept-, Rollen- und Attribut-Terme werden durch *terminologische Axiome* eingeführt. Sei A ein Konzept-Name, P ein Rollen-Name, f ein Attribut-Name und sei C ein Konzept-Term, R ein Rollen-Term und g ein Attribut-Term. Dann sind

(defprimconcept A)	(defprimrole P)	(defprimattribute f)
(defprimconcept A C)	(defprimrole P R)	(defprimattribute f g)
(defconcept A C)	(defrole P R)	(defattribute f g)

terminologische Axiome.

Eine TBox T ist eine endliche Menge von terminologischen Axiomen, die folgenden Bedingungen genügen müssen:

1. jeder Konzept-, Rollen- und Attribut-Name darf in T höchtens einmal als erstes Argument eines terminologischen Axioms erscheinen
2. T darf keine zyklischen Definitionen enthalten.

Syntax der ABoxes

Aufbauend auf einer weiteren Menge von Symbolen, den *Individuen-Namen*, werden nun *assertionale Axiome* definiert. Seien dazu a und b Individuen-Namen, C ein Konzept-Term, R ein Rollen-Term und g ein Attribut-Term. Dann sind

(assert-ind a C) (assert-ind a b R) (assert-ind a b g)

assertionale Axiome.

Eine ABox besteht aus einer endlichen Menge assertionaler Axiome.

Semantik der Konzeptsprachen

Eine *Interpretation* $\mathcal{I}$ einer Konzeptsprache besteht aus einer Menge $\Delta^{\mathcal{I}}$ (dem Definitionsbereich von $\mathcal{I}$) und einer Funktion $\cdot^{\mathcal{I}}$ (der *Interpretationsfunktion* von $\mathcal{I}$). Die Interpretationsfunktion bildet jeden Konzept-Namen A auf eine Teilmenge $A^{\mathcal{I}}$ von $\Delta^{\mathcal{I}}$, jeden Rollen-Namen P auf eine Teilmenge $P^{\mathcal{I}}$ von $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, und jeden Attribut-Namen f auf eine partielle Funktion $f^{\mathcal{I}}$ von $\Delta^{\mathcal{I}}$ nach $\Delta^{\mathcal{I}}$ ab. Die Interpretationsfunktion wird wie folgt auf Konzept-, Rollen- und Attribut-Terme fortgesetzt: Seien $C, C_1, \ldots, C_k$ Konzept-Terme, $R, R_1, \ldots, R_l$ Rollen-Terme und $f, g, f_1, \ldots, f_m$ Attribut-Terme, die bereits

interpretiert sind. Dann ist

$$
\begin{aligned}
(*\mathsf{top}*)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \\
(*\mathsf{bottom}*)^{\mathcal{I}} &:= \emptyset \\
(\mathsf{and}\ C_1 \ldots C_k)^{\mathcal{I}} &:= C_1^{\mathcal{I}} \cap \ldots \cap C_k^{\mathcal{I}} \\
(\mathsf{or}\ C_1 \ldots C_k)^{\mathcal{I}} &:= C_1^{\mathcal{I}} \cup \ldots \cup C_k^{\mathcal{I}} \\
(\mathsf{not}\ C)^{\mathcal{I}} &:= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(\mathsf{all}\ R\ C)^{\mathcal{I}} &:= \{\, a \in \Delta^{\mathcal{I}} \mid \forall b : (a,b) \in R^{\mathcal{I}} \Rightarrow b \in C^{\mathcal{I}} \} \\
(\mathsf{all}\ f\ C)^{\mathcal{I}} &:= \{\, a \in \Delta^{\mathcal{I}} \mid a \in dom\, f^{\mathcal{I}} \Rightarrow f^{\mathcal{I}}(a) \in C^{\mathcal{I}} \} \\
(\mathsf{some}\ R\ C)^{\mathcal{I}} &:= \{\, a \in \Delta^{\mathcal{I}} \mid \exists b : (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \} \\
(\mathsf{some}\ f\ C)^{\mathcal{I}} &:= \{\, a \in dom\, f^{\mathcal{I}} \mid f^{\mathcal{I}}(a) \in C^{\mathcal{I}} \} \\
(\mathsf{atleast}\ n\ R)^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}| \geq n\} \\
(\mathsf{atmost}\ n\ R)^{\mathcal{I}} &:= \{a \in \Delta^{\mathcal{I}} \mid |\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}| \leq n\} \\
(\mathsf{equal}\ f\ g)^{\mathcal{I}} &:= \{\, a \in dom\, f^{\mathcal{I}} \cap dom\, g^{\mathcal{I}} \mid f^{\mathcal{I}}(a) = g^{\mathcal{I}}(a) \} \\
(\mathsf{not\text{-}equal}\ f\ g)^{\mathcal{I}} &:= \{\, a \in dom\, f^{\mathcal{I}} \cap dom\, g^{\mathcal{I}} \mid f^{\mathcal{I}}(a) \neq g^{\mathcal{I}}(a) \} \\
(\mathsf{and}\ R_1 \ldots R_l)^{\mathcal{I}} &:= R_1^{\mathcal{I}} \cap \ldots \cap R_l^{\mathcal{I}} \\
(\mathsf{restr}\ R\ C)^{\mathcal{I}} &:= \{\, (a,b) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}} \} \\
(\mathsf{and}\ f_1 \ldots f_m)^{\mathcal{I}} &:= f_1^{\mathcal{I}} \cap \ldots \cap f_m^{\mathcal{I}} \\
(\mathsf{compose}\ f_1 \ldots f_m)^{\mathcal{I}} &:= f_1^{\mathcal{I}} \circ \ldots \circ f_m^{\mathcal{I}},
\end{aligned}
$$

wobei $|X|$ für die Kardinalität der Menge X und $\circ$ für die Komposition von Funktionen stehen. Man beachte, daß $f_1^{\mathcal{I}} \cap \ldots \cap f_m^{\mathcal{I}}$ und $f_1^{\mathcal{I}} \circ \ldots \circ f_m^{\mathcal{I}}$ partielle Funktionen sind, falls $f_1^{\mathcal{I}}, \ldots, f_m^{\mathcal{I}}$ partielle Funktionen sind.

Semantik der TBoxes

Die Semantik der terminologischen Axiome ist wie folgt definiert. Eine Interpretation $\mathcal{I}$ *erfüllt* ein terminologisches Axiom

$$
\begin{array}{lll}
(\mathsf{defprimconcept}\ A\ C) & \text{gdw} & A^{\mathcal{I}} \subseteq C^{\mathcal{I}}, \\
(\mathsf{defconcept}\ A\ C) & \text{gdw} & A^{\mathcal{I}} = C^{\mathcal{I}}, \\
(\mathsf{defprimrole}\ P\ R) & \text{gdw} & P^{\mathcal{I}} \subseteq R^{\mathcal{I}}, \\
(\mathsf{defrole}\ P\ R) & \text{gdw} & P^{\mathcal{I}} = R^{\mathcal{I}}, \\
(\mathsf{defprimattribute}\ f\ g) & \text{gdw} & f^{\mathcal{I}} \subseteq g^{\mathcal{I}}, \\
(\mathsf{defattribute}\ f\ g) & \text{gdw} & f^{\mathcal{I}} = g^{\mathcal{I}},
\end{array}
$$

wobei A ein Konzept-Name, P ein Rollen-Name, f ein Attribut-Name, C ein Konzept-Term, R ein Rollen-Term und g ein Attribut-Term ist. Man beachte, daß die terminologischen Axiome (defprimconcept A), (defprimrole P), und (defprimattribute f) von jeder Interpretation erfüllt werden.

Eine Interpretation $\mathcal{I}$ ist eine *Modell* einer TBox $\mathcal{T}$ genau dann, wenn $\mathcal{I}$ alle terminologischen Axiome in $\mathcal{T}$ erfüllt.

Semantik der ABoxes

Abschließend wird die Semantik der Individuen-Namen und der assertionalen Axiome definiert. Dazu wird die Interpretationsfunktion $\cdot^{\mathcal{I}}$ einer Interpretation $\mathcal{I}$ auf Individuen-Namen fortgesetzt, indem Individuen auf Elemente des Definitionsbereiches abgebildet werden. Dabei wird gefordert, daß $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ für $a \neq b$ (unique name assumption), d.h. Individuen mit verschiedenen Namen stellen auch *verschiedene* Objekte dar.

Sind a, b Individuen-Namen, C ein Konzept-Term, R ein Rollen-Term und g ein Attribut-Term, so *erfüllt* eine Interpretation $\mathcal{I}$ das assertionale Axiom

(assert-ind a C)	gdw	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
(assert-ind a b R)	gdw	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
(assert-ind a b f)	gdw	$f^{\mathcal{I}}(a^{\mathcal{I}}) = b^{\mathcal{I}}$

Eine Interpretation $\mathcal{I}$ ist ein *Modell* für eine ABox $\mathcal{A}$ bzgl. einer TBox $\mathcal{T}$, genau dann, wenn $\mathcal{I}$ alle assertionalen Axiome in $\mathcal{A}$ und alle terminologischen Axiome in $\mathcal{T}$ erfüllt.

5 Anhang B: $\mathcal{KRIS}$-Dateien

Die folgenden Verzeichnisse und System-Dateien werden von $\mathcal{KRIS}$ benötigt:

1. *Rechnername*>KRIS>Algorithmen
   Dieses Verzeichnis hat folgende Unterverzeichnisse:
   (a) ALC
   (b) ALCN
   (c) ALCFN
   (d) ALCFNR
   (e) Hilfsfunktionen
   Funktionen, die die Konsistenz-Algorithmen verwenden:
   i. input-syntax.lisp: Makros für die Eingabe-Syntax der TBoxes und ABoxes (defrole, defconcept, assert-ind etc.)
   ii. basicfunctions.lisp: Funktionen, die von mehreren Moduln benötigt werden

2. *Rechnername*>KRIS>Hilfstexte
   Textdateien, die als Hilfstexte ausgegeben werden, und Funktionen zu ihrer Ausgabe

3. *Rechnername*>KRIS>Laufzeitmessung

4. *Rechnername*>KRIS>Oberfläche
   Das Verzeichnis Oberfläche enthält Dateien zum Aufbau der $\mathcal{KRIS}$-Oberfläche und zur Auswahl der Inferenzen (Graphik-Ausgabe, Menü-Steuerung etc.)
   (a) abox-expansion.lisp
   (b) abox-alcfnr-help-functions.lisp
   (c) abox-help-functions.lisp
   (d) abox-testing.lisp: Schnittstellenfunktionen für ABox Inferenzen
   (e) alcfnr-abox-consistency.lisp Funktionen zum Durchführen der ABox Inferenzen für $\mathcal{ALCFNR}$
   (f) alcn-abox-consistency.lisp Funktionen zum Durchführen der ABox Inferenzen für $\mathcal{ALCN}$
   (g) realization.lisp
   (h) retrieval.lisp

(i) algorithm-handling.lisp Schnittstellenfunktionen für TBox Inferenzen

(j) expansion-and-simpli.lisp: Funktionen zur Expansion und Simplifizierung von TBoxen

(k) graphic-display.lisp: Funktionen zur Ausgabe der Subsumtionshierarchie

(l) special-graphic-file.lisp: Definition eines vergrößerten Bildschirm für die Graphik-Ausgabe

(m) menu-functions.lisp: Schnittstellenfunktionen für Kommando-Leisten und Menüs zur Auswahl von Konzepten, Dateien etc.

(n) message-functions.lisp: Funktionen zur Ausgabe von System- und Fehlermeldungen

(o) status-display.lisp

(p) system-access.lisp: Funktionen zum Zugriff auf Dateien und Verzeichnisse

(q) tbox-handling.lisp: Funktionen zum Laden, Edieren Speichern, Löschen und Neuerstellen von TBoxen und ABoxen

(r) tbox-testing.lisp: Funktionen zum Durchführen der TBox Inferenzen

(s) tbox-utilities.lisp Schnittstellenfunktionen für die Zeitmessung

(t) check-algorithms.lisp: Funktionen zum Testen der Korrektheit der Algorithmen

5. *Rechnername*>KRIS>Syntax-Checking

(a) Check-Main.lisp: Funktionen zum eigentlichen Syntax-Check

(b) Syntax-Check.lisp: Funktionen zum zeichenweisen Einlesen von Dateien

6 Anhang C: Die Menüstruktur von *KRIS*

Im folgenden werden alle Befehle der Kommando-Leisten von *KRIS* zusammengefaßt.

- TBox/ABox-Handler
  - Create TBox
  - Create ABox
  - Edit TBox
  - Edit ABox
  - Load file
  - Delete File
  - Expand TBox
  - Expand TBox & ABox
  - Help
  - Main-Menu
- Algorithms
  - ALC
  - ALCN
  - ALCFN
  - ALCFNR (rule-based)
  - Help
  - Main-Menu
- Inferences
  - Single consistency check
  - Single subsumption test
  - TBox consistency test
  - TBox classification
  - TBox/ABox consistency
  - Instantiation
  - Realization

  - Retrieval
  - Help
  - Main-Menu

- Utilities
  - Show expanded concept
  - Show simplified concept
  - Run KRIS with time-protocol
  - Check TBoxes
  - Check ABoxes
  - Help
  - Main-Menu

- Clear Output Windows
  - Graphics-Window
  - Output-Window
  - Commands-Window
  - All output-Windows
  - Main-Menu

- Help
- Quit

Literatur

[1] F. Baader, B. Hollunder. *KRIS: Knowledge Representation and Inference System—System Description.* DFKI Technical Memo TM-90-03, DFKI, Postfach 2080, D-6750 Kaiserslautern, Germany.

[2] B. Hollunder, W. Nutt. *Subsumption Algorithms for Concept Description Languages.* DFKI Research Report RR-90-04, DFKI, Postfach 2080, D-6750 Kaiserslautern, Germany.

[3] M. Schmidt-Schauß, G. Smolka. *Attributive concept descriptions with complements.* Artificial Intelligence, 47, 1991.

Deutsches
Forschungszentrum
für Künstliche
Intelligenz GmbH

DFKI
-Bibliothek-
PF 2080
6750 Kaiserslautern
FRG

DFKI Publikationen

Die folgenden DFKI Veröffentlichungen oder die aktuelle Liste von erhältlichen Publikationen können bezogen werden von der oben angegebenen Adresse.
Die Berichte werden, wenn nicht anders gekennzeichnet, kostenlos abgegeben.

DFKI Publications

The following DFKI publications or the list of currently available publications can be ordered from the above address.
The reports are distributed free of charge except if otherwise indicated.

DFKI Research Reports

RR-90-01
Franz Baader: Terminological Cycles in KL-ONE-based Knowledge Representation Languages
33 pages

RR-90-02
Hans-Jürgen Bürckert: A Resolution Principle for Clauses with Constraints
25 pages

RR-90-03
Andreas Dengel, Nelson M. Mattos: Integration of Document Representation, Processing and Management
18 pages

RR-90-04
Bernhard Hollunder, Werner Nutt: Subsumption Algorithms for Concept Languages
34 pages

RR-90-05
Franz Baader: A Formal Definition for the Expressive Power of Knowledge Representation Languages
22 pages

RR-90-06
Bernhard Hollunder: Hybrid Inferences in KL-ONE-based Knowledge Representation Systems
21 pages

RR-90-07
Elisabeth André, Thomas Rist: Wissensbasierte Informationspräsentation:
Zwei Beiträge zum Fachgespräch Graphik und KI:
1. Ein planbasierter Ansatz zur Synthese illustrierter Dokumente
2. Wissensbasierte Perspektivenwahl für die automatische Erzeugung von 3D-Objektdarstellungen

24 pages

RR-90-08
Andreas Dengel: A Step Towards Understanding Paper Documents
25 pages

RR-90-09
Susanne Biundo: Plan Generation Using a Method of Deductive Program Synthesis
17 pages

RR-90-10
Franz Baader, Hans-Jürgen Bürckert, Bernhard Hollunder, Werner Nutt, Jörg H. Siekmann: Concept Logics
26 pages

RR-90-11
Elisabeth André, Thomas Rist: Towards a Plan-Based Synthesis of Illustrated Documents
14 pages

RR-90-12
Harold Boley: Declarative Operations on Nets
43 pages

RR-90-13
Franz Baader: Augmenting Concept Languages by Transitive Closure of Roles: An Alternative to Terminological Cycles
40 pages

RR-90-14
Franz Schmalhofer, Otto Kühn, Gabriele Schmidt: Integrated Knowledge Acquisition from Text, Previously Solved Cases, and Expert Memories
20 pages

RR-90-15
Harald Trost: The Application of Two-level Morphology to Non-concatenative German Morphology
13 pages

RR-90-16
Franz Baader, Werner Nutt: Adding Homomorphisms to Commutative/Monoidal Theories, or: How Algebra Can Help in Equational Unification
25 pages

RR-90-17
Stephan Busemann: Generalisierte Phasenstrukturgrammatiken und ihre Verwendung zur maschinellen Sprachverarbeitung
114 Seiten

RR-91-01
Franz Baader, Hans-Jürgen Bürckert, Bernhard Nebel, Werner Nutt, Gert Smolka: On the Expressivity of Feature Logics with Negation, Functional Uncertainty, and Sort Equations
20 pages

RR-91-02
Francesco Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, Werner Nutt: The Complexity of Existential Quantification in Concept Languages
22 pages

RR-91-03
B.Hollunder, Franz Baader: Qualifying Number Restrictions in Concept Languages
34 pages

RR-91-04
Harald Trost: X2MORF: A Morphological Component Based on Augmented Two-Level Morphology
19 pages

RR-91-05
Wolfgang Wahlster, Elisabeth André, Winfried Graf, Thomas Rist: Designing Illustrated Texts: How Language Production is Influenced by Graphics Generation.
17 pages

RR-91-06
Elisabeth André, Thomas Rist: Synthesizing Illustrated Documents A Plan-Based Approach
11 pages

RR-91-07
Günter Neumann, Wolfgang Finkler: A Head-Driven Approach to Incremental and Parallel Generation of Syntactic Structures
13 pages

RR-91-08
Wolfgang Wahlster, Elisabeth André, Som Bandyopadhyay, Winfried Graf, Thomas Rist: WIP: The Coordinated Generation of Multimodal Presentations from a Common Representation
23 pages

RR-91-09
Hans-Jürgen Bürckert, Jürgen Müller, Achim Schupeta: RATMAN and its Relation to Other Multi-Agent Testbeds
31 pages

RR-91-10
Franz Baader, Philipp Hanschke: A Scheme for Integrating Concrete Domains into Concept Languages
31 pages

RR-91-11
Bernhard Nebel: Belief Revision and Default Reasoning: Syntax-Based Approaches
37 pages

RR-91-12
J.Mark Gawron, John Nerbonne, Stanley Peters: The Absorption Principle and E-Type Anaphora
33 pages

RR-91-13
Gert Smolka: Residuation and Guarded Rules for Constraint Logic Programming
17 pages

RR-91-14
Peter Breuer, Jürgen Müller: A Two Level Representation for Spatial Relations, Part I
27 pages

RR-91-15
Bernhard Nebel, Gert Smolka: Attributive Description Formalisms ... and the Rest of the World
20 pages

RR-91-16
Stephan Busemann: Using Pattern-Action Rules for the Generation of GPSG Structures from Separate Semantic Representations
18 pages

RR-91-17
Andreas Dengel & Nelson M. Mattos: The Use of Abstraction Concepts for Representing and Structuring Documents
17 pages

RR-91-19
Munindar P. Singh: On the Commitments and Precommitments of Limited Agents
15 pages

RR-91-20
Christoph Klauck, Ansgar Bernardi, Ralf Legleitner FEAT-Rep: Representing Features in CAD/CAM
48 pages

RR-91-22
Andreas Dengel: Self-Adapting Structuring and Representation of Space
27 pages

RR-91-23
Michael Richter, Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Akquisition und Repräsentation von technischem Wissen für

TM-90-03
Franz Baader, Bernhard Hollunder: KRIS: Knowledge Representation and Inference System -System Description-
15 pages

TM-90-04
Franz Baader, Hans-Jürgen Bürckert, Jochen

TM-91-11
Peter Wazinski: Generating Spatial Descriptions for Cross-modal References
21 pages

DFKI Documents

D-89-01
Michael H. Malburg, Rainer Bleisinger:
HYPERBIS: ein betriebliches Hypermedia-Informationssystem
43 Seiten

D-90-01
DFKI Wissenschaftlich-Technischer Jahresbericht 1989
45 pages

D-90-02
Georg Seul: Logisches Programmieren mit Feature -Typen
107 Seiten

D-90-03
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Abschlußbericht des Arbeitspaketes PROD
36 Seiten

D-90-04
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: STEP: Überblick über eine zukünftige Schnittstelle zum Produktdatenaustausch
69 Seiten

D-90-05
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner: Formalismus zur Repräsentation von Geo-metrie- und Technologieinformationen als Teil eines Wissensbasierten Produktmodells
66 Seiten

D-90-06
Andreas Becker: The Window Tool Kit
66 Seiten

D-91-01
Werner Stein , Michael Sintek: Relfun/X - An Experimental Prolog Implementation of Relfun
48 pages

D-91-03
Harold Boley, Klaus Elsbernd, Hans-Günther Hein, Thomas Krause: RFM Manual: Compiling RELFUN into the Relational/Functional Machine
43 pages

D-91-04
DFKI Wissenschaftlich-Technischer Jahresbericht 1990
93 Seiten

D-91-06
Gerd Kamp: Entwurf, vergleichende Beschreibung und Integration eines Arbeitsplanerstellungssystems für Drehteile
130 Seiten

D-91-07
Ansgar Bernardi, Christoph Klauck, Ralf Legleitner
TEC-REP: Repräsentation von Geometrie- und Technologieinformationen
70 Seiten

D-91-08
Thomas Krause: Globale Datenflußanalyse und horizontale Compilation der relational-funktionalen Sprache RELFUN
137 pages

D-91-09
David Powers and Lary Reeker (Eds):
Proceedings MLNLO'91 - Machine Learning of Natural Language and Ontology
211 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-10
Donald R. Steiner, Jürgen Müller (Eds.)
MAAMAW'91: Pre-Proceedings of the 3rd European Workshop on „Modeling Autonomous Agents and Multi-Agent Worlds"
246 pages
Note: This document is available only for a nominal charge of 25 DM (or 15 US-$).

D-91-11
*Thilo C. Horstmann:*Distributed Truth Maintenance
61 pages

D-91-12
Bernd Bachmann:
$H^{iera}C_{on}$ - a Knowledge Representation System with Typed Hierarchies and Constraints
75 pages

D-91-13
International Workshop on Terminological Logics
Organizers: Bernhard Nebel, Christof Peltason, Kai von Luck
131 pages

D-91-14
Erich Achilles, Bernhard Hollunder, Armin Laux, Jörg-Peter Mohren: KRIS: Knowledge Representation and Inference System
- Benutzerhandbuch -
28 Seiten

KRIS : *K*nowledge *R*epresentation and *I*nference *S*ystem
- Benutzerhandbuch -

Erich Achilles, Bernhard Hollunder, Armin Laux, Jörg-Peter Mohren

D-91-14
Document