

Atlas.ti – Einführung¹

Version 5.0

Inhaltsverzeichnis

Das Konzept von Atlas.ti	3
<i>Die Grundidee</i>	3
<i>Der allgemeine Arbeitsablauf mit Atlas.ti</i>	3
Die Oberfläche des Programms und wie man damit umgeht	4
<i>Die Symbolleiste</i>	4
<i>Das Primärdokumentfenster</i>	5
<i>Die Werkzeugleiste</i>	5
<i>Das Margin Display</i>	6
Konfiguration des Programms und seiner Komponenten	6
Anlegen einer Hermeneutischen Einheit	6
Primärtexte einbinden und organisieren	7
<i>Vorgehensweisen beim Einbinden von Dokumenten</i>	7
<i>Umbenennen von Primärdokumenten (interne und externe Namen)</i>	7
<i>Das Problem des Editierens von Primärdokumenten</i>	8
Zitieren	8
Kodieren	9
<i>Drei textunabhängige Kodiervarianten</i>	9
<i>Vier Funktionen für das sukzessive Kodieren</i>	10
Mit neu benannten Codes kodieren.....	10
In-vivo-Codes.....	10
Mit Codes aus der Liste kodieren.....	11
Mit aktivem Code kodieren.....	11
Kodieren via drag & drop.....	11
Kodes organisieren	11
<i>Was also kann man alles mit einzelnen Codes tun?</i>	12
<i>Was kann man mit der ganzen Liste tun?</i>	12
Arbeiten mit Familien	13
<i>Was sind "Familien"?</i>	13
<i>Filter und Sortierkriterien</i>	15
Filtern von Objektlisten.....	15
Relationen im Netzwerk-Editor bearbeiten	15
<i>Herstellen benennbarer Beziehungen</i>	16
<i>Der Relationen-Editor</i>	17
Textsuche und Autocoding	19
<i>Suchfunktion</i>	19
Such-Schwärme.....	20
Suche mit GREP.....	20
<i>Autocoding</i>	21
Texte schreiben in ATLAS/ti: Memos und Kommentare	22

1 von Stefanie Rühl überarbeitete und gekürzte Fassung von: Jörg Strübing: ATLAS/ti-Kurs. Einführung in des Arbeiten mit dem Programm ATLAS/ti für Windows 95 Versionen 4.0 und 4.1. In: Mitteilungen aus dem Schwerpunktbereich Methodenlehre. Heft 48, November 1997, Institut für Soziologie, FREIE UNIVERSITÄT BERLIN

Recherche in den Daten.....	23
<i>Das Query Tool.....</i>	<i>23</i>
Unterschied zwischen Query und Textsuche.....	24
Query Tool: Die Oberfläche.....	24
<i>Die Abfragesprache – gewöhnungsbedürftig, aber praktisch.....</i>	<i>25</i>
Reversed Polish Notation.....	25
<i>Die Bedienung des Query Tools.....</i>	<i>25</i>
Formulieren einer Such-Abfrage.....	25
Organisation des Such-Anfrage-Fensters.....	26
<i>Operatoren.....</i>	<i>26</i>
<i>Boolesche Operatoren.....</i>	<i>26</i>
<i>Semantische oder Thesaurus-Operatoren.....</i>	<i>27</i>
<i>Proximity-Operatoren.....</i>	<i>28</i>
<i>Einige weitere Funktionen des Query Tools.....</i>	<i>29</i>
<i>Supercodes.....</i>	<i>29</i>
Arbeiten im Team.....	30
<i>Verwaltung von Benutzern.....</i>	<i>30</i>
User Editor.....	30
Hermeneutic-Unit-Merging.....	31
Verschiedenes.....	33
<i>Object Explorer.....</i>	<i>33</i>
<i>Copy Bundle.....</i>	<i>34</i>

Das Konzept von Atlas.ti

ATLAS/ti ist ausgerichtet an der *Grounded Theory* und an der *qualitativen Inhaltsanalyse*, versteht sich aber auch als ein universelles Werkzeug für den gesamten Bereich der qualitativen Sozialforschung (bis hin zu Schnittstellen zur quantitativen Sozialforschung!).

Die Grundidee

Wenn man davon ausgeht, dass textuelle, graphische oder Audiodaten qualitativ analysieren, interpretieren, sortiert und verwaltet werden sollen und dass das Ziel dabei die Erarbeitung von analytischen Ideen und ganzen, in den Daten gründenden Theorien ist, dann bietet das Programm eine Reihe von Werkzeugen zu diesem Zweck. Es gibt die verschiedensten Funktionen zum Verwalten, Extrahieren, Analysieren, Vergleichen und Aggregieren bedeutungshaltiger Daten aus dem "Berg" gesammelter Daten. Wichtig ist dabei ein flexibler Zugang, der Raum für forschersche Kreativität einräumt, der aber zugleich ein systematisches und nachvollziehbares Arbeiten erlaubt.

ATLAS/ti setzt dem Umfang der zu analysierenden Daten praktisch keine Grenzen, ebenso wenig der Anzahl der in der Analyse erzeugten Objekte (Konzepte etc.) oder der Komplexität der gesamten analytischen Struktur.

Der allgemeine Arbeitsablauf mit Atlas.ti

1. Zunächst wird ein **Projekt kreiert**, eine sogenannte **hermeneutische Einheit** (engl: Hermeneutic Unit), die dazu da ist, die Gesamtmenge der zu einer Forschungsaufgabe anfallenden Befunde, Kodes, Memos, Strukturen und Daten unter einem Namen zu versammeln und um eine zentrale Datei herum zu organisieren.
2. In einem zweiten Schritt werden alle **Datenfiles**, die sich irgendwo auf dem Rechner oder im Netzwerk befinden können, **mit der Hermeneutic Unit verbunden**.
3. Der nächste Schritt besteht typischerweise aus dem **Lesen der Datentexte**, dem markieren bzw. **zitieren** analytisch interessanter Stellen und der **Zuordnung von Kodes** und Memos zu diesen Stellen. Dies ist in den meisten Projekten die Hauptaufgabe, zumindest was den zeitlichen Aufwand betrifft.
4. Der nächste Schritt ist gewöhnlich eine **vergleichende Analyse** der kodierten Textstellen und nötigenfalls die Einbindung weiterer Datentexte
5. Ein weiterer Arbeitsschritt (nicht notwendig erst an dieser Stelle stattfindend!) ist das **Organisieren der verschiedenen Objekttypen** (Primärdokumente, Kodes, Memos) zu Gruppen oder "Familien" (z.B. alle Beobachtungsprotokolle, alle Interviewtranskripte, alle Kodes zu Arbeitsteilungsprozessen), die dann auch selektiv oder kontrastierend analysiert werden können.
6. Erarbeiten begrifflicher, semantischer oder aussagenlogischer **Netzwerke** aus den Kodes, die wir zuvor generiert haben. Diese Netzwerke bilden die Ausgangsbasis für die entstehende **gegenstandsbezogene Theorie**.
7. Schließlich gilt es einen **Bericht** zu erzeugen, in den viele Teil der Hermeneutic Unit Eingang finden werden.

Diese Schritte und gar noch ihre **Reihenfolge** sind **nicht** festgelegt oder **zwingend**, skizzieren aber einen typischen Verlauf im Forschungshandeln.

In ATLAS/ti können alle diese **Schritte in vielfältiger Weise variiert** werden (aber natürlich kann man nicht eine gegenstandsbezogene Theorie entwickeln, ohne zunächst Daten analysiert und in irgendeiner Weise kodiert zu haben!). Insbesondere ist es möglich,

die Arbeitsschritte ineinander integriert in einem iterativ-zyklischen Prozess abzuarbeiten, was einer der Maximen qualitativen Sozialforschung entspricht.

Die Oberfläche des Programms und wie man damit umgeht











Beim ersten Start erscheint das Hauptfenster mit dem Logo und Programmangaben in der Mitte.

Oben hat man wie üblich die Menü- und Symbolleiste. Dazu gibt es eine Leiste, mittels derer man guten und schnellen Zugriff auf alle Dokumente, Memos und Codes hat (Combo-Boxen). Links gibt es eine sogenannte Werkzeugleiste, die für das jeweilige Primärdokument gilt. Öffnet man ein Primärdokument, so erscheint im rechten Fensterbereich das sogenannte Margin Display.



Die Symbolleiste

Die Funktionen der Symbolleiste von links nach rechts sind die folgenden:

-  **Netzwerk-Sichten öffnen**
-  **Anzeigen bzw. Editieren des Kommentars zur Hermeneutic Unit**
-  **Speichern der Hermeneutic Unit**
-  **Öffnen des Object Explorers**
-  **Öffnen des Query Tools, d.h. der Suchmaschine**
-  **Öffnen und Einbinden von Primärdokumenten**
-  **Object Crawler**
-  **Word Cruncher**
-  **Export für XML**
-  **Öffnen des Konfigurationsmenüs**

Hier kann man zum zuletzt **aktivierten Zitat** zurückspringen (innerhalb der

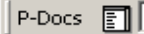
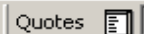
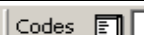
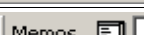


gleichen Sitzung)

Hier sollte man draufhauen, wenn man so richtig frustriert ist – geht aber nur, wenn eine Soundkarte installiert ist...

Die Combo-Boxen

Nun die Funktionen der darunter liegenden Combo-Boxen mit jeweils einem dazugehörigen Button, der den entsprechenden Manager dazu aufruft (von links nach rechts):

 P-Docs	Primärtextliste
 Quotes	Zitatliste
 Codes	Kodeliste
 Memos	Memoliste

Sowohl die **Combo-Boxen** als auch die **Extra-Listenfenster** verfügen über ein kontextbezogenes Menü, das durch einen rechten Mausklick in die jeweilige Liste aktiviert wird. In diesem Menü finden sich alle für die jeweilige Objektart relevanten Funktionen (Einbinden von Primärtexten, Ausgabe von Kodelisten, etc).













WICHTIG: Diese Kontextmenüs enthalten unterschiedliche Menüpunkte je nachdem, ob ein Objekt in der Liste aktiviert ist oder nicht: Bei aktiviertem Objekt werden nur diejenigen Funktionen angezeigt, die man in Bezug auf das Objekt ausführen kann.

Das Primärdokumentfenster

Darunter nun befindet sich das Herzstück des Hauptbildschirms, das Primärdokumentfenster. Hier wird beim Start zunächst meist nur der Eröffnungsbildschirm mit Logo gezeigt. Sobald man aber Daten eingebunden und ein Primärdokument aktiviert hat, erscheint dieses. Dann ist auch die linke Werkzeugleiste aktiviert.

Die Werkzeugleiste

Die Werkzeugleiste des Primärdokument-Fensters bietet:

-  Gehe zu **Zeilennummer** ...
-  **Suche** nach Strukturen oder Text-Strings im Primärtext
-  Erstellen eines neuen **Zitates** für die aktuelle Markierung
-  Erstellen eines neuen Kodes für die aktuelle Markierung (**Offene Kodierung**)
-  Erstellen eines Kodes unter Verwendung der Markierung als Kodennamen (**In-vivo-Kodierung**)
-  **Kodiert** die aktuelle Markierung unter Verwendung eines existierenden Kodes **aus der Liste** (DD)
-  Kodiert die aktuelle Markierung mit dem zuletzt ausgewählten Kode (**Schnell-Kodierung**)
-  Erzeugen eines **neuen Memos**
Dieses Memo wird automatisch der gerade aktiven Textstelle zugeordnet. Der Titel ist frei zu wählen, vorgeschlagen wird der zuletzt aktiv Kode.
-  **Ändern der Grenzen** der aktiven Kodierung: Zitat aktivieren, neue Zitatgrenzen markieren und zur Bestätigung diese Taste anklicken
-  **99** Zu- oder abschalten der fortlaufenden **Zeilennumerierung**
-  Zu- oder abschalten des **margin display**
-  Die aktuelle Markierung wird zum **Ausgangspunkt eines Hyperlinks** zu einer

anderen Textstelle



Die aktuelle Textstelle wird zum **Ziel einer Hyperlink-Verbindung**



Anzeigen einer **Liste der** an der derzeitigen Cursorposition existierenden **Zitate**



Zoom: Vergrößert oder verkleinert (mit STRG) die Ansicht des Primärdokuments

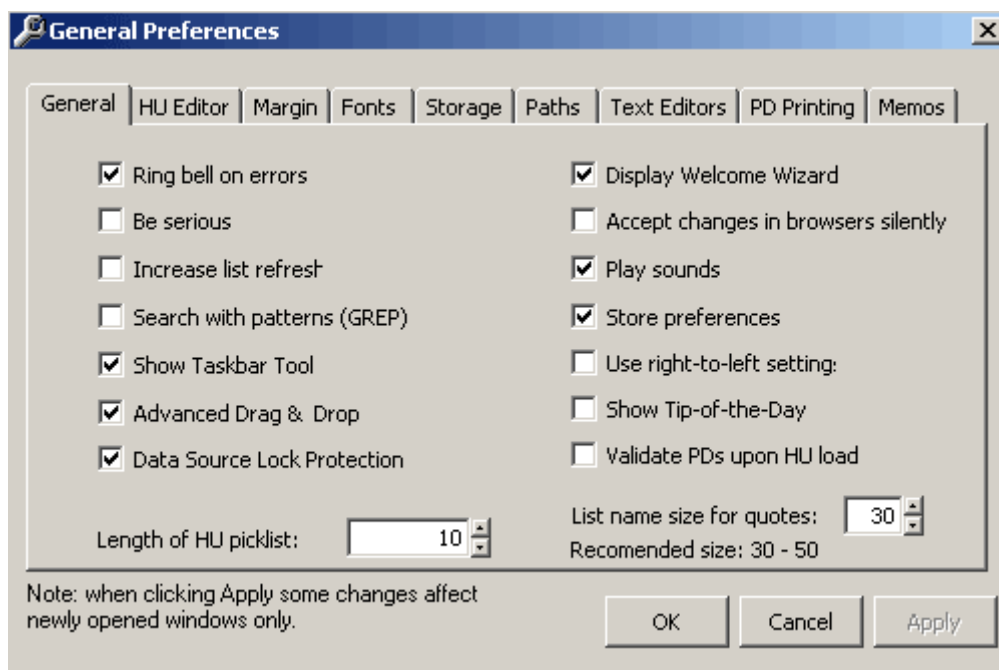
Das Margin Display

Rechts befindet sich die rechte Randleiste (margin display), die dazu dient, die im dargestellten Textausschnitt vorhandenen Zitate sowie die diesen zugeordneten Codes oder auch Memos anzuzeigen. Man sieht hier also fast wie auf dem Papier, was man sich "angestrichen" und was man dazu notiert hat.

Konfiguration des Programms und seiner Komponenten

Wie bei allen komfortableren Programmen unter Windows kann man auch ATLAS/ti in vielfältiger Weise konfigurieren, also an die eigenen Nutzungsgewohnheiten anpassen.

Das eigentliche Konfigurationsmenü befindet sich unter dem Menüpunkt "**Extras / General Preferences**". Es öffnet sich ein Set von Registerkarten, auf denen Einstellungen zu verschiedenen Bereichen vorzunehmen sind (General, Hermeneutic Unit-Editor, Fonts, Storage, Path, Text Editors).




Anlegen einer Hermeneutischen Einheit

Das Gründen eine **hermeneutische Einheit** ist der erste Schritt für die Bearbeitung eines Datenanalyse- und Theoriebildungsvorhabens in ATLAS/ti. Eine hermeneutische Einheit ist die analytische Arbeitseinheit, die einer bestimmten Auswertungs- oder Forschungsfrage gewidmet ist. So etwas wie ein Projekt also.

Eine Hermeneutic Unit ist **flexibel einsetzbar**: nur für eine Fallstudie, für eine vergleichende Auswertung verschiedener Fälle oder nur für einen bestimmten Typ Daten, die getrennt von anderen Daten ausgewertet werden sollen. Je nachdem, wie man sich entscheidet, legen man auch fest, **welche Daten** in eine bestimmte Hermeneutic Unit eingebunden werden sollen.

Die Schrittfolge beim Gründen einer Hermeneutic Unit ist die folgende:

1. "New Hermeneutic Unit" aus dem Datei-Menü wählen.
2. Den **Namen für Hermeneutic Unit** eingeben. Es ist sinnvoll, einen **kurzen prägnanten Namen** zu wählen und diesen dann in einem **Kommentar genauer zu erläutern** (z.B. für ForschungskollegInnen, die mit derselben Hermeneutic Unit arbeiten sollen). Dazu genügt es den **Button**  anzuklicken, woraufhin sich ein Editor-Fenster öffnet, in das man nun einen erläuternden Kommentar über den Sinn und Zweck der Hermeneutic Unit schreiben kann. Das hilft später auch zu erkennen, wie sich unter der Hand das Erkenntnisinteresse des Projektes verschoben haben kann.
3. Speichern der neuen Hermeneutic Unit über das Menü "File/Save as"

Primärtexte einbinden und organisieren

Jetzt haben wir eine Hermeneutische Einheit, aber noch keine Daten, die man analysieren könnte. Diese müssen nun – vorausgesetzt, sie liegen als Dateien vor – in die Hermeneutic Unit eingebunden werden.

Als Daten kommen zunächst natürlich vor allem Textdaten in Betracht: Transkripte von Bandmitschnitten, Protokolle von Beobachtungen, Dokumente, die einer Inhaltsanalyse unterzogen werden sollen. Aber auch Graphiken, Bilder und Audio- und Videofiles können in ATLAS als Dateien eingebunden und analysiert werden.

Vorgehensweisen beim Einbinden von Dokumenten

a) Die Dateien werden einzeln aus dem geöffneten Windows Explorer oder Dateimanager per **drag & drop** entweder in das Textfenster des ATLAS-Hauptfensters gezogen (dann sind sie sowohl sichtbar als auch eingebunden) oder aber sie werden einzeln oder in Gruppen in das Primärtext-Listfenster (links oben) gezogen. Dann werden sie nicht sofort angezeigt, wohl aber in die Hermeneutic Unit eingebunden.

b) das gleiche kann man auch erreichen, indem man im **Menü "Documents"** den Menüpunkt **assign** wählt (oder den entsprechenden **Button in der Combo-Boxen-Leiste** anklickt) und in dem dann aufklappenden Dateimanager die entsprechenden Dateien auswählt und bestätigt.

Textdateien sollten der ANSI-Zeichensatz-Konvention entsprechen. Es nutzt also nichts, wenn man in einem komfortablen Textverarbeitungsprogramm die Texte schön formatiert.

Damit die Primärdokumente später in Atlas.ti auch editiert werden können, müssen als rft-Datei gespeichert werden.

Umbenennen von Primärdokumenten (interne und externe Namen)

Mitunter kommt es vor, dass man im Laufe der Arbeit bemerkt, dass die ursprünglichen Dateinamen ungünstig gewählt wurden.

In diesem Fall kommen zwei Möglichkeiten in Betracht:

1. Man **ändert den Dateinamen selbst (außerhalb von ATLAS/ti)**. Das hat aber den großen *Nachteil*, dass man nun die umbenannte Datei wieder neu einbinden muss, da Atlas standardmäßig nach der Datei mit dem alten Namen sucht. Besonders ärgerlich kann das werden, wenn man mit der Datei schon gearbeitet, sprich darin zitiert und kodiert hat. Dann wäre diese Arbeit verloren, weil all diese Operationen nur als Verweise in der Hermeneutic Unit-Datei verzeichnet werden, also die Kodierung nicht direkt in die Datendatei hineingeschrieben wird.
2. Man **ändert die Hermeneutic Unit-interne Bezeichnung** für die jeweilige

Primärdokument-Datei. Denn genau wie bei den Hermeneutic Units wird bei ATLAS/ti zwischen internem und externem Dateinamen unterschieden, wobei zunächst beide identisch sind. Sobald man aber über das Primärtext-Listenmenü für die aktive Textdatei den internen Namen ändert, werden beide Namen getrennt verwaltet. Auch das hat einen *Nachteil*, der nicht verschwiegen werden soll: Nach längerer Zeit bekommt man Probleme auseinander zuhalten, welche externen Dateinamen sich eigentlich hinter den internen Namen verbergen. Da es mittlerweile möglich ist, den externen Dateinamen zu ändern und die Datei dennoch weiter unter der bisherigen Primärdokument-Nr. zu verwalten, dürfte die erste Variante zu bevorzugen sein.

Die Primärdokumente werden zunächst in der Reihenfolge in die Liste eingestellt, in der sie in die Hermeneutic Unit eingebunden werden. Das ist praktisch, wenn mit der Reihenfolge gerne die Historie der Einbindung von Texten abbilden will. Oft aber hat man andere Sortierkriterien. Z.B. könnte die alphabetische Reihenfolge der Dateinamen in sich eine zeitliche Reihenfolge der Datenerhebung oder des Ereignisablaufs repräsentieren. Zu diesem Zweck kann man im Kontextmenü die Sortierkriterien ("*set sort options*") verändern und z.B. eine alphabetisch Sortierung einstellen. Damit ist man schnell und flexibel in der Darstellungsform der Liste.

Mitunter aber will man nun tatsächlich die Reihenfolge der Texte und nicht nur ihre Anzeige in der Liste verändern, also den Texten andere Primärdokument-Nummern zuordnen. Dazu kann man wiederum über das Primärtext-Listenmenü die Reihenfolge des aktiven Primärdokuments verändern (unter "*miscellaneous/change position*"), indem man die Nummer des Primärdokuments angibt hinter dem das aktive Primärdokument eingeordnet werden soll. Noch einfacher geht es, wenn man im Primärtext Extra-Listenfenster den gewünschten Text markiert und mit gedrückter linker Maustaste an die angestrebte neue Position zieht.

Man kann übrigens – über das Kontextmenü und die Funktion "*change path*" – auch den Pfad der Primärdokument-Datei ändern, für den Fall, dass man seine Dateien einmal neu organisieren will oder auf einem andern Rechner mit andere Verzeichnisstruktur arbeiten will.

Das Problem des Editierens von Primärdokumenten

Oft kommt die Frage auf, ob es möglichst, die Datentexte noch zu editieren, nachdem sie bereits als Primärdokumente in die Hermeneutic Unit eingebunden sind. Z.B. hat man Tippfehler entdeckt, stellt fest, dass die Aliasnamen nicht konsistent sind oder – peinlich, peinlich – die Namen noch gar nicht anonymisiert wurden.

Technisch ist das in ATLAS/ti kein Problem: Text ins Primärdokument-Fenster laden, Kontextmenü aktivieren, **Edit mode** wählen und das Ändern kann beginnen. Das geht aber nur, wenn die Parteidokumente nicht als .doc, sondern als .rft-Datei vorliegen.


Beim Editieren ist aber eine **wichtige Einschränkung** zu beachten, die mit dem logischen Konzept von ATLAS/ti zusammenhängt: Zitate in Primärdokumenten werden in der Hermeneutic Unit als Zeilen und Spaltennummern abgelegt und verwaltet. Wenn man nun vor der letzten im Primärdokument als Zitat markierten Passage Text einfügt oder löscht, verrutscht die Zeilen- und Spaltenreferenz, so dass die Zitatgrenzen nicht mehr stimmen.

Es ist allerdings kein Problem **innerhalb einer Zeile** Änderungen vorzunehmen, solange man darauf achtet, dass kein zusätzlicher Zeilenumbruch erfolgt. Wenn sich in der gleichen Zeile aber ein Zitat befindet, kann es allerdings passieren, dass dessen Grenzen sich verschieben.

Zitieren

Meistens findet der **Vorgang des Zitierens**, also der Identifikation und Markierung von analytisch relevanten Textpassagen, **in direktem Zusammenhang mit dem Kodieren**

dieser Passagen statt. Darauf ist ATLAS/ti auch ausgerichtet und bietet eine Reihe von Kodierfunktionen, die das Anlegen von Zitaten gleich mit beinhalten. Wir kommen dazu im nächsten Abschnitt.

Diese kombinierte Vorgehensweise ist aber **nicht zwingend**, man kann auch unkodierte Zitate anlegen. Dies geschieht, in dem man die fragliche Textpassage markiert und dann den  **Button** in der Werkzeugleiste vom Primärdokument-Fenster anklickt. Damit wird dieses Zitat unter Angabe der ersten und der letzten Zeilennummer (in Klammern hinter dem Zitat) sowie der ersten Worte **in der Zitatliste verzeichnet** (die Ziffern vor dem Zitat stehen für die Nummer des Primärtextes, die zweite für die Zitatnummer des entsprechenden Primärtextes).

Bei eingeschaltetem *Margin Display* erscheint nun dort eine viereckige Linie (*serpent*), die die Zitatstelle markiert.

(**weitere Möglichkeiten** des Zitierens: in die markierte Passage gehen, Kontextmenü öffnen und *Create Quotation* wählen)

Kodieren

Der **basale Vorgang** bei der qualitativen Datenanalyse ist die **Kodierung der Daten**. Gleichgültig, welcher Methodologie man sich verschrieben hat: Der erste Schritt, um sich das Material zu erschließen, besteht immer darin, dass man einzelne Passagen der Textdaten (oder Ausschnitte von Bilddaten) in irgendeiner Weise als sinnhafte Einheit identifiziert und dieser Einheit ein **Label zuordnet**, das vielleicht zunächst nur paraphrasierend-deskriptiv gemeint ist, später aber u.U. eine ausgearbeitete analytische Abstraktion darstellt.

Drei textunabhängige Kodiervarianten

Zuerst die rein **technischen** Möglichkeiten, wie man Codes in ATLAS/ti erzeugen kann. Zunächst unterscheidet man **drei textunabhängige Varianten**, in denen Codes erzeugt, aber noch nicht unbedingt Daten kodiert werden (man spricht dann von "**freien Codes**"):

Variante a: Neue Codes sukzessive anlegen:

⇒ Button "**Create New Item**"  im Code-Manager anklicken

⇒ Namen eingeben (nicht zu lang, sonst wird es unübersichtlich!).

Tipp: Es gibt hier auch die Möglichkeit, gleich mehrere Codes im gleichen Eingabefenster zu erzeugen. Dazu gibt man einfach hinter jedem Kodennamen einen senkrechten Strich ("|", nicht "/"!) ein und schreibt den nächsten Kodennamen direkt dahinter. Diese Möglichkeit besteht auch bei der weiter unten skizzierten Variante des Kodierens mit einem neuen Code via Button im Primärtextfenster.

Variante b: Kodeliste zu Beginn neu erzeugen:

⇒ Entweder, man geht (wie in der Variante a) über den Button "**Create a New Item**" im Code-Manager und gibt nacheinander alle Codes ein, die man vorab erzeugen will
ODER

man hat anderswo vielleicht schon eine **Liste in einer Datei**, die man gerne **übernehmen** möchte. Das geht so: erzeuge ein oder zwei Codes wie in Variante a beschrieben

⇒ wähle im Code-Manager das Menü *Output*

⇒ wähle dort die Option "*code list*" und wähle unter den Optionen "*Send output to: file and run*"

⇒ gib der Datei einen Namen und speichere sie mit der Dateierweiterung *.cod* oder *.kod*.

⇒ Dann diese Datei nach Wunsch bearbeiten (jeder Code ist eine Zeile, die mit <Enter> beendet wird. NICHT hinter das letzte Sonderzeichen gehen!)

⇒ Daten speichern

⇒ im Kontextmenü Codes über "*Miscellaneous/Import Code List*" die Liste in die aktuelle Kodeliste einfügen.

Tipp: Für den Fall, dass es **Doubletten**, gibt, werden diese mit mehreren Ausrufezeichen zu Beginn der Zeile gekennzeichnet. Wenn man Doubletten in der Kodeliste löschen will, sollte man die mit Ausrufezeichen versehenen löschen, da diese keine Zitat-Referenzen enthalten. Das Entstehen von Doubletten kann aber auch von vorne herein weitgehend vermieden werden, wenn man in der ASCII.-Datei bereits alle Codes entfernt, die identisch bereit in der Hermeneutic Unit angelegt sind.

Variante c: Kodeliste aus anderer Hermeneutic Unit übernehmen:


Funktioniert wie Variante b, nur dass man die Kodeliste aus einer anderen Hermeneutic Unit auslesen muss und diese dann auf dem beschriebenen Weg in die aktuelle Hermeneutic Unit einlesen kann. Bei Bedarf kann die ausgelesene Liste wiederum in einem Editor bearbeitet werden (ergänzen, ausdünnen, Bezeichnungen vereinheitlichen etc.)

Vier Funktionen für das sukzessive Kodieren


Der häufigste Fall in der qualitativen Datenanalyse ist sicherlich das **sukzessive Entwickeln von Codes**. In der Grounded Theory etwa beginnt die Datenanalyse in der Regel mit einer "*line by line*"-Analyse eines ersten Datensegmentes. Dabei werden einzelne Worte, Satzteile, Sätze oder auch ganze Absätze als "irgendwie bemerkenswert, interessant oder potentiell bedeutsam" wahrgenommen, markiert und zunächst mit einem vorläufigen Label versehen, das sich stark an der Sprache des untersuchten Ausschnitts der Alltagswelt orientieren sollte.

Technisch könnte man nun so vorgehen, dass man eine Stelle markiert, einen Kode erzeugt und diesen dann in einem dritten Schritt dem Zitat zuordnet. Das kann man in ATLAS/ti auch so machen, allerdings ist es etwas umständlich. Daher wurden hier diese Arbeitsabläufe bereits zu einer Aktion integriert und unter einem Button im Primärdokument-Fenster (sowie über das Kontextmenü) verfügbar gemacht:

Mit neu benannten Codes kodieren

Mit der Funktion "***create a new Code***"  im Primärdokumentfenster wird die markierte Passage als Zitat aufgenommen und zugleich ein Fenster geöffnet, in dem wir einen neuen Kodennamen eingeben können, der sogleich dem neuen Zitat zugeordnet wird. Diese Funktion ist vor allem in den ersten Arbeitsphasen des **offenen Kodierens** von Bedeutung.


In-vivo-Kodes

Der darunter befindliche Button  bietet noch eine Sonderform des Kodierens mit einem neuen Kode an, nämlich das "**In-vivo-Kodieren**". Damit ist gemeint, dass eine markierte Textstelle mit sich selbst kodiert wird. (z.B.: "hat mich traurig gemacht" wird mit dem Kode "hat mich traurig gemacht" kodiert). In der qualitativen Sozialforschung hat das "In-vivo-Kodieren" einen hohen Stellenwert, da es häufig darum geht, seine Codes und Konzepte möglichst nah an der Sprache der erforschten Alltagswelt zu bilden.


Tipp: Ein ganz spezielle Variante des *In-vivo-Kodierens* und zugleich eine interessante Möglichkeit zur gleichzeitigen Erzeugung einer Vielzahl von Codes aus einem Primärtext heraus ist die folgende: Man markiert eine beliebige Textpassage im aktiven Primärtext, zieht diese Passage mit der gedrückten linken Maustaste auf die Kodeliste und hält dann die *<alt>*-Taste gedrückt, während man die linke Maustaste loslässt. Das Resultat ist, dass nun *alle einzelnen Worte* dieser Textpassage als je ein Kode erzeugt wurden. Wenn man statt der *<alt>*-Taste die *<strg>*-Taste verwendet, wird die Textpassage – vorausgesetzt, sie ist nicht zu lang

– als *In-Vivo-Kode* erzeugt. In beiden Fällen wird die Textpassage auch mit den neu erzeugten Codes kodiert.

Mit Codes aus der Liste kodieren

Nun hat man es beim Kodieren aber nicht immer mit dem Erzeugen neuer Codes zu tun. Oft will man neue Zitate bilden und mit Codes belegen, die man schon in einer Kodeliste verzeichnet hat. Dazu dient der darunter liegenden Button . Er öffnet eine Kodeliste, aus der man dann (durch anklicken mit der linken Maustaste) beliebig viel Codes auswählen können (*code by list*). Das neue Zitat wird mit diesen Codes kodiert.

Mit aktivem Kode kodieren

Schließlich hat man es gerade in der Phase des **axialen und des selektiven Kodierens** häufig mit dem Fall zu tun, dass man nacheinander einer größeren Zahl von Zitaten den gleichen Kode zuweisen will. Dies geschieht mit dem untersten dieser vier Kodierbuttons . Man sieht bei dieser Aktion nicht viel passieren, aber: in der *margin area* wird der Kode hinzugefügt und in der Kodeliste wird bei diesem Kode der Zitatstellen-Zähler um 1 hoch gesetzt.

Alle diese Funktionen sind **auch über** das **Kontextmenü** einer markierten Textstelle im Primärdokument-Fenster erreichbar. Auch die Kontextmenüs der Zitatliste und der Kodeliste bieten einen Zugang zu diesen Kodierfunktionen. Es ist ein Frage der aktuellen Arbeitssituation und der persönlichen Präferenz, was man nutzt.

Kodieren via drag & drop

Schließlich bietet ATLAS/ti noch eine weitere Variante des Kodierens, nämlich das **drag & drop-Kodieren**: Dabei wird eine Textstelle markiert und dann ein Zitat in der Kodeliste aktiviert und mit gedrückter linker Maustaste auf die markierte Textstelle gezogen. Nach dem Loslassen ist die Textstelle zum Zitat geworden und der Kode logisch mit dem Zitat verbunden.

Eine ganz andere Form des Kodierens, nämlich die über den graphischen Netzwerk-Editor. Diese Funktion wird aber erst im Zusammenhang mit der Einführung in den Netzwerk-Editor vorgestellt.

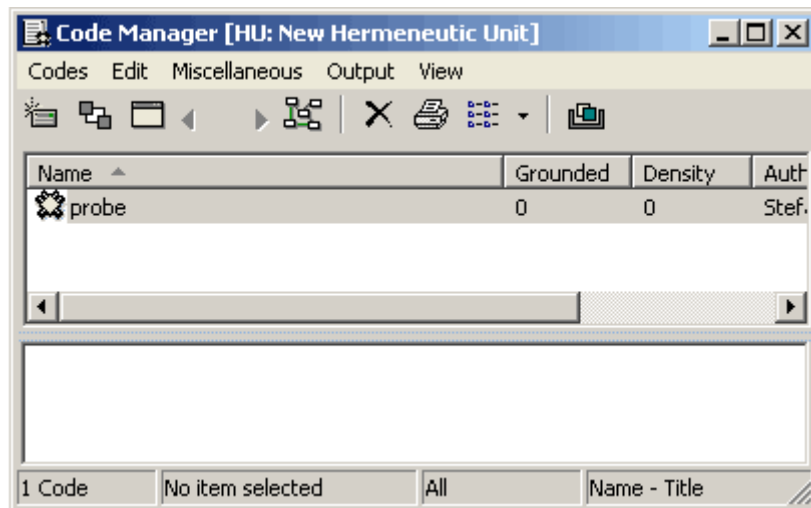
Kodes organisieren

Durch **einmaliges Anklicken** eines Codes ist dieser aktiviert. Alle nachfolgenden Aktionen beziehen sich zunächst auf diesen aktiven Kode.

Doppelklicken des Codes bewirkt, dass entweder, wenn nur ein Zitat zugeordnet ist, dieses im Primärdokument-Fenster aufgerufen wird ODER, wenn dem Kode mehrere Zitate zugeordnet sind, eine Liste dieser Zitatstellen aufklappt, aus der man mit der Maus eine Stelle auswählen kann, die dann angezeigt wird.

Über den **Code-Manager** und sein **Kontext-Menü** sind eine Vielzahl von Funktionen zum Organisieren verfügbar.

Dabei ist immer sorgfältig zu unterscheiden, ob man sich auf die gesamte Liste bezieht oder auf einen markierten und damit aktivierten einzelnen Kode. Den Unterschied kann man daran erkennen, dass bei neu geöffneter Hermeneutic Unit und ohne dass ein Kode oder ein Zitat aktiviert ist, im einheitlichen Kontextmenü immer diejenigen Stellen nur grau dargestellt sind, die z.Zt. nicht verfügbar sind. Es würde ja keinen Sinn machen, mit "rename" gleichzeitig alle Kode-Labels umzubenennen.



Tipp: Wenn man in der geöffneten Liste einmal einen Codes aktiviert hat, bleibt er dies solange, bis ein anderer Code ausgewählt wird. Wie aber kommt man in den Zustand zurück, in dem kein Code aktiv ist? Auf den ersten Blick scheint die Lösung zu lauten: Man deaktiviert die Markierung des aktiven Codes. Dieser erscheint dann nicht mehr invertiert. Allerdings funktioniert das nicht so ganz: Der zuletzt aktive Code wird trotzdem immer so behandelt, als sei er noch aktiv!

Was also kann man alles mit einzelnen Codes tun?

- Umbenennen
- Kommentieren
- zur Kodierung benutzen
- ein um diesen Code fokussiertes Netzwerk öffnen (s.u.)
- den Code mit anderen Objekten (anderen Codes, Memos, Zitaten) verbinden (das Verbinden mit Zitaten ist natürlich identisch mit dem Vorgang des Kodierens, nur das hier vorausgesetzt wird, dass beide Objekte, Zitat und Code schon definiert sind!)
- Autokoding-Funktion öffnen
- Löschen
- mit anderen Codes verschmelzen ("miscellaneous/merge codes")

Man kann auch Codes von Zitaten löschen. Dazu stehen unter "miscellaneous" die Funktionen "unlink quotation" und "unlink all quotations" zur Verfügung. Bei "**unlink quotation**" wird eine Liste der mit dem markierten Code verbundenen Zitate angezeigt, aus der sich dann diejenigen herausuchen lassen, die vom Code getrennt werden sollen. Die Zitate bleiben erhalten!

Tipp: Die Funktion "**unlink all quotations**" hingegen ist mit einiger Vorsicht zu gebrauchen: Hier werden nämlich überraschenderweise nicht alle Zitate des aktiven Codes sondern alle Zitate der gesamten Hermeneutic Unit von allen Codes gelöst! Ein Vorgang, den man nur höchst selten einmal durchführen wollen. Allerdings gibt es noch eine Nachfrage, bei der man erst bestätigen muss, dass man tatsächlich alle Codes von ihren Zitaten löschen will.

Was kann man mit der ganzen Liste tun?

- Familien-Editor öffnen und bearbeiten (s.u.)
- Sortierkriterien ändern (s.u.)

- Listenfilter ändern (ist ein Unterschied!; s.u.)
- Autokoding-Funktion öffnen (s.u.)
- diverse Arten von Kodelisten ausgeben
Es stehen zur Verfügung: die einfache Liste aller Codes; die Kode-Hierarchie; Codes und ihre "Nachbarn", Tabelle der Kodierungen pro Primärtext und eine Liste der Codes mit einer Auflistung der Fundstellen in den verschiedenen Primärtexten nach Zeilennummern.
- Ausgabe über SPSS-Schnittstelle (wird hier nicht weiter behandelt)

Arbeiten mit Familien

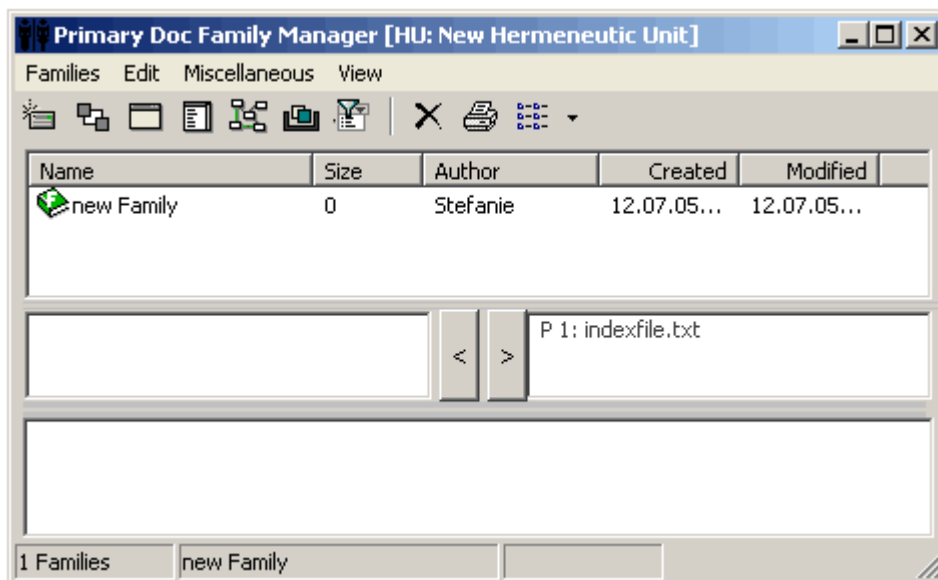
Was sind "Familien"?

In ATLAS/ti wird ein Strukturierungsmedium angeboten, das als "Familien-Editor" firmiert. Solche Editoren werden angeboten für die drei Objektarten Primärdokument, Kode und Memo.

Der Begriff ist allerdings etwas missverständlich, wenn man dabei an den alltagsweltlichen Familien-Begriff denkt. Ursprünglich geht dieses Konzept aber auch auf eine andere Vorstellung zurück: Barney Glaser, der Mit-Erfinder der Methodologie der Grounded Theory (Glaser/Strauss 1967), hat 1978 in einer Erweiterung dieses Ansatzes die Orientierung der konzeptuellen Kodierarbeit an sogenannten "**Kodierfamilien**" vorgeschlagen. Darunter kann man sich so etwas wie Gruppierungen von auf einer gewissen analytischen Ebenen gebräuchlichen Konzepten ("theoretischen Codes") vorstellen. Ein Beispiel für Kodierfamilien ist z.B. die "Zweck-Mittel-Familie", bestehend aus Codes wie: Ziel, Zweck, antizipierte Konsequenz, Ergebnis usw (Glaser 1978: S. 77).

Der Unterschied zwischen Familien im alltagssprachlichen Gebrauch und Familien im Glaserschen Sinn ist der, dass in "richtigen Familien" bereits ein formal definiertes Verhältnis der einzelnen Personen (oder Rollen) zueinander besteht, eben der Verwandtschaftsgrad. Die Kodierfamilien bei Glaser stellen aber nichts anderes dar als eine **lose Ansammlung von konzeptuell auf einer Ebene zu verortenden Begriffen**, ohne dass damit irgend etwas über eine interne Beziehung der Begriffe untereinander ausgesagt wäre.

ATLAS/ti bedient sich dieses Konzeptes in den drei Objektarten Primärdokument, Kode und Memo. Für jede dieser Objektarten lässt sich ein separater "**Familien-Editor**" öffnen (z.B. über da Kontextmenü der Liste).



Diese **Editoren** bestehen im Prinzip aus vier Fenstern: Oben ist das Familien-Listenfenster, in dem sich alle bislang kreierte Familien befinden und über dessen Kontextmenü sich eine Reihe von Familien-bezogenen Funktionen abrufen lassen (Neu anlegen, Löschen, Umbenennen, ...). Ganz unten findet sich das uns aus den Extra-Listen bereits vertraute gelbe **Kommentar-Fenster**, in dem ein Kommentar zu jeder Familie angelegt werden kann. In der Mitte liegen zwei Fenster nebeneinander, die in der Mitte durch zwei Operatoren-Buttons verbunden sind. Sobald man eine Familie gegründet hat bzw. eine existierende Familie aktiviert hat, erscheinen im rechten der beiden mittleren Fenster in roter Schrift diejenigen Objekte der Objektart, die nicht-Mitglieder der aktiven Familie sind, während links die Liste der dieser Familie zugeordneten Objekte steht. Durch Auswahl von Objekten in den Listen und Betätigung eines der beiden Operatoren lassen sich nun **Objekte wahlweise der Familie zuordnen** (von rechts nach links bzw. aus ihr entfernen (von links nach rechts). Dabei können alle Objekte Mitglieder beliebig vieler Familien sein – es gibt also **keine Ausschließlichkeit der Zuordnung** wie etwa in einer harten kategorialen Klassifikation.

Ein Beispiel: Im Bereich Primärtexte haben wir die unterschiedlichsten Datentypen, die zugleich zu den verschiedensten Fällen gehören und quer dazu jeweils verbindende oder trennende Merkmale aufweisen. So gibt es etwa Interviews mit unterschiedlich geschlechtlichen Akteuren, von denen einige zu Fall A, andere aber zu Fall B gehören. Zugleich haben wir vielleicht Dokumente, Expertengespräche und Transkripte von Beobachtungen sowie ein Forschungstagebuch mit Notizen. Mit dem Familien-Editor können wir nun etwas Ordnung in diese Vielfalt des Materials bringen, indem man Familien gründet wie z.B. "Interviews", "Beobachtungen", "Akten", "männliche Befragte", "weibliche Befragte", "Fall A", "Fall B" usw. und jedes Primärdokument allen Familien zuordnet, zu denen es sinnvollerweise gehört. Dann wäre ein Interview mit einer Managerin im Unternehmen A zugleich in den Familien "weibliche Befragte", "Interview" und "Fall A".

Was ist nun der **Vorteil** einer solchen Strukturierung? Auf der Ebene der Primärtexte gewinnt man damit ein **wertvolles Sortier-Kriterium**, dessen man sich an verschiedenen Stellen der analytischen Arbeit wieder bedienen kann. Zunächst kann man damit **filtern**, welche Primärdokumente in der Primärdokument-Liste angezeigt werden sollen. Beim **"Autokodieren"** kann man die Suche auf Datenfiles beschränken, die einer bestimmten Primärdokument-Familie angehören.

Wie gesagt: Familien lassen sich auch für Memos (zur Gliederung der Sorten von Memo-Texten) und für Codes anlegen.

Eine in Bezug auf Kode-Familien oft gestellte **Frage** betrifft den prinzipiellen **Unterschied** zwischen dem Strukturierungsmittel der **Familien** und Strukturierung des Codesystems

über **definierte Beziehungen**.

Die Antwort ist einfach: Die **Kode-Kode-Relationen** (die wir im Detail nach kennen lernen werden) bilden in ihrer Gesamtheit ein **semantisches Netzwerk**, das den Kern unserer zu entwickelnden Theorie bilden soll und das später auch (zu Prüfung von ad hoc-Hypothesen z.B.) sehr differenziert recherchiert werden kann. Die **Familien** sind dagegen ein sehr **effizientes Sortierhilfsmittel**, das aber nicht unmittelbar Eingang in die Theoriebildung findet. Es ist also ein **konzeptueller Unterschied**, ob ich eine Unterbegriff-Oberbegriff-Beziehung als Familie anlege (Familie als Oberbegriff, zugeordnete Codes als Unterbegriffe) oder ob ich dies in einem Netzwerk definierter Kode-Kode-Beziehungen vergegenständliche.

Filter und Sortierkriterien

Alle in ATLAS/ti angebotenen Listen von Objekten, also die vier standardmäßig in Combo-Boxen dargestellten Listen für Primärdokumente, Zitate, Codes und Memos, verfügen über zwei Funktionen zur Erhöhung der Transparenz in der Listendarstellung, die Sortierkriterien und das Filtern. Beide sind über das einschlägige Menü bzw. das jeweilige Kontextmenü zugänglich.

Die Option "*set sort options*" bietet erwartungsgemäß eine Reihe von Auswahlmöglichkeiten für die Sortierreihenfolge der Listeneinträge. So kann etwa für die Liste der Codes festgelegt werden, ob die Reihung alphabetisch (das ist die Standardeinstellung bei Codes), nach Erzeugungs- oder Benutzungsdatum, nach der Anzahl der Zitatbezüge ("*quotations*") oder nach der Zahl der Bezüge zu anderen Codes ("*references*") erfolgen soll. Welcher Filter jeweils gewählt ist, wird bei den Extra-Listenfenstern im rechten Feld der Statuszeile angezeigt.


Auf diese Weise kann man sich z.B. schnell einen Überblick über das Maß an "Groundedness" verschaffen, also darüber, wie stark ein Kode im Datenmaterial verankert ist, oder über die theoretische Dichte ("Density"), also die Intensität der Einbindung in ein semantisches Netz. Vor allem aber dienen die Sortierkriterien dazu, sich die für den jeweiligen Arbeitsschritt günstigste Reihung der Objekte zu organisieren (beim selektiven Kodieren im Sinne der Grounded Theory ist es z.B. zweckmäßig, die besonders häufig benutzten Codes zuerst anzuzeigen, da in dieser Phase vorwiegend mit ihnen gearbeitet wird.).

Filtern von Objektlisten

Man kann aber die Listeninhalte nicht nur unterschiedlich sortieren, man kann auch auswählen ob alle oder nur ein nach bestimmten Kriterien zu bestimmender Teil der Objekte einer Liste angezeigt werden sollen. Diese Funktion wird als "**Filtern**" bezeichnet. Als **Kriterien** je nach Art der Liste unterschiedliche Faktoren zur Auswahl, die hier nicht alle aufgelistet werden können. Für die Kodeliste ist besonders zu erwähnen, dass man hier auch die Anzeige bzw. nicht-Anzeige derjenigen Codes wählen kann, die von **KoautorInnen** erzeugt wurden – eine wichtige Funktion für die Forschungsarbeit im Team. Das jeweilige Filterkriterium wird im mittleren Feld der Statuszeile der jeweiligen Extra-Liste angezeigt.

Relationen im Netzwerk-Editor bearbeiten

Zwar ist es ohne weiteres möglich, die Beziehungen zwischen Objekten im entstehenden semantischen Netzwerk über die "Link"-Funktionen der verschiedenen Listenfenster herzustellen. Doch würde uns damit nach und nach der Überblick über die Gesamtstruktur des Netzwerkes verloren gehen. Daher stellt ATLAS/ti einen graphischen Netzwerk-Editor zur Verfügung, mit dessen Hilfe es möglich wird, die gesamte Gestaltung des semantischen Netzwerkes graphisch zu bewerkstelligen.

Am einfachsten öffnet man den Netzwerk-Editor durch einen Mausklick auf das Netzwerksymbol  in der Button-Leiste des jeweiligen Listenfensters. Dadurch wird ein neues Fenster geöffnet, in dem das in der jeweiligen Liste gerade aktive Objekt als graphisches Objekt dargestellt wird. Falls dieses Objekt bereits mit anderen Objekten verknüpft wurde, werden diese ebenfalls im Netzwerk-Editor angezeigt. Da allerdings häufig sehr viel Zitate an mit einzelnen Codes verknüpft sind, werden beim Öffnen eines fokussierten Netzwerkes auf diesem Code die Zitat-Objekte nicht mit angezeigt (können aber nach Bedarf hinzu geholt werden).²

Die Beziehung zwischen verschiedenen im Netzwerk-Editor repräsentierten Objekten wird durch eine Linie symbolisiert, die sich wie ein "Gummiband" verhält: sobald man ein Objekt auf der Oberfläche verschiebt, dehnt oder zieht sich die Linie entsprechend. Damit können kann man die Objekt nach Belieben so anordnen, wie es visuell eingängig erscheint, ohne dass man Gefahr läuft, die Beziehung der Objekt zueinander nicht mehr nachvollziehen zu können. In der Regel wird man den Netzwerk-Editor dazu nutzen wollen, die Beziehungen der dargestellten Objekte zu weiten Objekten her- und darstellen zu wollen. Der erste Schritt dazu ist das **Importieren weiterer Objekte**. Der einfachste Weg dazu ist eine drag & drop Operation: Das Objekt, z.B. ein Code, wird in der entsprechenden Liste angeklickt, mit gedrückter linker Maustaste in den Netzwerk-Editor gezogen und dort "fallen gelassen" (Maustaste loslassen).

Man kann auch über das Menü bzw Kontextmenü die **Option "import neighbours"** wählen. Dann werden alle verknüpften Objekte, soweit sie nicht schon im Netzwerk-Editor präsent sind, importiert. Bei Codes mit einer großen Zahl von Objekten kann das ziemlich unangenehm werden: Schnell hat man hunderte von Zitatobjekten im Fenster. Daher gibt es zwei Sicherungen: Erstens enthält das Nodes-Menü (nicht aber das Kontextmenü der Objekte!) die Funktion "**undo import neighbours**", mit der man den letzten (und nur den letzten!) Import rückgängig machen kann. Zweitens kann man schon beim Import selbst Vorsorge treffen, dass nicht die falschen Objekte importiert werden. Dazu hält man bei Wahl der Funktion die <ctrl>-Taste gedrückt, was dazuführt, dass a) bei Codeobjekten der Import von *quotations* (Zitaten) unterdrückt wird, und b) bei Zitatobjekten nur *quotations* importiert werden (praktisch für hyperlink-networks).

Tipp: Jedes Objekt im Netzwerk-Editor hat ein eigenes, der Objektart entsprechendes Kontextmenü, das sich mit einem rechten Mausklick auf das Objekt öffnet. Hier können Aktionen wie das Importieren weiterer Nachbar-Objekte, das Entfernen des Objektes aus der Netzwerkansicht, das Löschen des Objektes aus der , das Öffnen einer weiteren, um das fragliche Objekt fokussierten Netzwerkansicht u.ä. angestoßen werden. Bei Code-Objekten lässt sich auch eine Liste der verbundenen Zitate anzeigen, aus der ein Zitat ausgewählt werden kann, das dann sogleich im Primärdokument-Fenster im Kontext angezeigt wird.

Herstellen benennbarer Beziehungen

Auch die **Beziehung** zwischen einem Objekt und einem anderen lässt sich einfach **per drag & drop herstellen**. Dazu wird das erste Objekt mit einem rechten Mausklick aktiviert, dann die <shift>-Taste sowie die linke Maustaste gedrückt gehalten und die Maus auf das zweite Objekt gezogen. Es erscheint ein "Gummifaden", der zwischen den beiden Objekten gespannt wird. Bei Code-Code oder Zitat-Zitat-Beziehungen öffnet sich zugleich ein **Optionsmenü**, aus dem nun auszuwählen ist, welche **Art von Beziehung** zwischen den beiden Objekten gelten soll (bei diesen beiden Arten von Beziehungen kann man benennbare und mit unterschiedlichen Eigenschaften ausgestattet Beziehungen

² Codes werden im Netzwerk-Editor mit unterschiedlichen graphischen Symbolen angezeigt. Dies geschieht nach folgenden Regeln: *gelbes Quadrat*: alle Codes für die nicht eine der folgenden Bedingungen zutrifft; *gelbes Quadrat mit rotem Punkt*: Code heute kreiert; *gelbes Quadrat mit dahinter liegender Seite*: Code mit Kommentar; *gelbes Quadrat mit Kreuz*: Code hat mindestens 5 Code-Nachbarn; *Baum*: Code mit mindestens 3 Zitatreferenzen; *grüner Ball*: Keine Code-Nachbarn; *rotes Quadrat*: Supercode.

definieren, zu andern Objekten sind lediglich unbenannte Beziehungen herstellbar). Ein Reihe dieser **Beziehungen** ist bereits vordefiniert, andere können **selbst definiert werden** (zum Relationen-Editor siehe weiter unten).

Tipp: Bei asymmetrischen und transitiven Beziehungen ist es von Interesse zu wissen, **welches** der beiden zu verbindenden **Objekte** der **Ausgangspunkt** und **welches** das **Ziel der unidirektionalen Beziehung** ist. Im Netzwerk-Editor ist alles ganz intuitiv: Für die Beziehung "a ist Teil von b" ("is part of") wählt man zuerst a und dann b, also erst das Teil und dann die Gesamtheit oder den Oberbegriff aus. Genau umgekehrt aber verhält es sich beim Herstellen von Beziehung via Kontextmenü der Kodeliste. Dies Verkehrtung ist zwar gewöhnungsbedürftig, aber gewollt, denn: Bei der Arbeit mit dem Kontextmenü macht es Sinn, etwas für die "is a"- oder die "is part of"-Beziehung zunächst den Oberbegriff zu aktivieren und dann aus der Kode-Auswahlliste die zuzuordnenden Unterbegriffe auszuwählen. Im Netzwerk-Editor ist die Arbeitsweise umgekehrt: Dort kann man in einem Arbeitsgang gleich eine ganze Reihe von Objekten markieren und alle zusammen mit einem andern Objekt verbinden (nicht aber gleichzeitig mit mehreren). Hier also wählt man immer erst die Unterbegriffe und dann die Oberbegriffe.

ACHTUNG: Alle Operationen, die im Netzwerk-Editor durchgeführt werden, wirken sich **tatsächlich** auf die aus. Es handelt sich also beim Knüpfen von Verbindungen, beim Löschen von Objekten oder beim Erzeugen neuer z.B. Codes **nicht um rein graphische Operationen**, sondern um anders visualisierte logische Operationen in der . Lediglich die Funktion "*Remove from view*" hat ausschließlich auf der graphischen Ebene Bedeutung.

Wer den Netzwerk-Editor schließen will, wird gefragt, ob er/sie die Netzwerkansicht speichern will. Dabei geht es aber nicht darum, die im Zuge der graphischen Bearbeitung des Netzwerkes vorgenommenen Änderungen in der Beziehungsstruktur der Objekte zueinander zu sichern – dies geschieht ausschließlich mit dem Sichern der als Ganzes. Vielmehr erhält man hier die Gelegenheit, den jeweils dargestellten Ausschnitt aus dem gesamten in der existierenden semantischen Netzwerk inklusive des u.U. mühsam hergestellten Layouts zu speichern. Wer also mit der jeweiligen Netzwerkansicht noch einmal arbeiten will und dazu nicht jedesmal von neuem die verschiedenen Objekte importieren und die günstigste visuelle Anordnung wiederherstellen will, sollte auf die Speichern-Abfrage mit ja antworten. Wem es lediglich um das Bearbeiten der logischen Bezüge der Objekte untereinander in einer graphischen Oberfläche geht, kann getrost auf das Speichern verzichten.

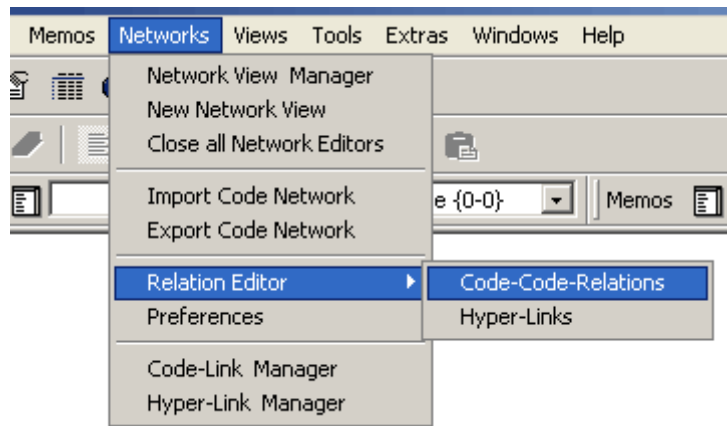
Der Netzwerk-Editor hat eine Vielzahl von Funktionen, die hier nicht alle im einzelnen vorgestellt werden können. Hervorzuheben ist, dass es für die wichtigsten Bearbeitungsfunktionen eine frei beweglich Toolbar gibt, die auf oder neben dem Netzwerk-Editor angeordnet werden kann.

Der Relationen-Editor

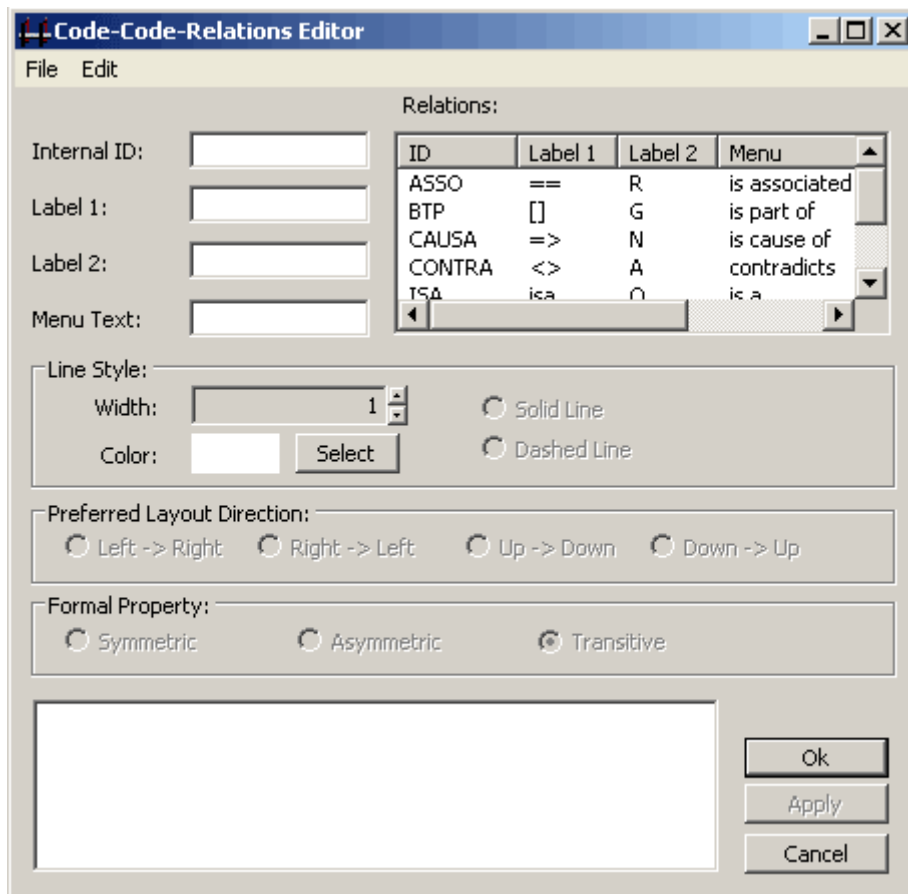
Obwohl eine Reihe von typischen Kode-Kode-Relationen in ATLAS/ti bereits vordefiniert angeboten werden, gibt es – wie erwähnt – die Möglichkeit, sich im Laufe der analytischen Arbeit eigene zusätzliche Beziehungen zu definieren. Die ist auf zweierlei Arten möglich:

a) Wie oben beschrieben kann man im Prozess der Verbindung zweier Kode-Objekte anstatt der angebotenen vordefinierten Beziehungsmuster die Option "user defined relation" wählen. In diesem Fall wird man nacheinander um die Angabe von internem Namen (nur für die Verwaltung der Beziehung in der), Zeichen für die visuelle Darstellung im Netzwerk-Editor, Menütext, Strichstärke der Verbindungslinie sowie logischer Status der Beziehung (transitiv, asymmetrisch, symmetrisch) gebeten.

b) Eine zweite, etwas komfortablere Variante besteht darin, im Netzwerk-Editor im Menü "Links" die Option "**relation editor**" zu wählen.



Daraufhin öffnet sich ein Fenster mit umfangreichen Konfigurationsmöglichkeiten für Kode-Kode-Beziehungen. Dort lassen sich zunächst alle Parameter festlegen, die auch in der ersten Variante benannt werden. Darüber hinaus jedoch lassen sich hier jedoch noch weitere Festlegungen treffen: Farbe und Art der Verbindungslinie können ebenso definiert werden, wie die Richtung, in der die Beziehung im Netzwerk-Editor standardmäßig dargestellt werden soll (z.B. "Teil-von"-Beziehungen von unten nach oben ordnen). Außerdem kann hier ein Kommentar ein gegeben werden.



Wenn Relationen geändert oder neu angelegt werden sollen, dann ist es wichtig zu wissen, ob diese Änderungen global für alle Hermeneutic Units am jeweiligen Rechner oder nur lokal für die in Bearbeitung gelten. Dazu muss man wissen, dass ATLAS/ti einer objektorientierten Logik folgt und die Eigenschaften einer jeweiligen Hermeneutic Unit zunächst von den Standardeinstellungen des Programms abgeleitet werden. Änderungen wirken sich entsprechend auch nur auf die Hermeneutic Unit aus. Ungefähr so verhält es sich dann auch bei den Relationen (und übrigens auch bei der Definition von Such-

Schwärmen in der Textsuche): Änderungen der Relationen von Kode–Kode– oder Hyperlink–Beziehungen werden zunächst nur in der Hermeneutic Unit gespeichert und sind fest mit ihr verbunden. Wird die Hermeneutic Unit auf einen anderen Rechner verlagert, wandern die Eigenschaften der Relationen natürlich mit. Andere Hermeneutic Units aber können diese geänderten Relationen nur unter zwei Bedingungen vererbt bekommen:


1. Wenn die Hermeneutic Unit mit den neu oder verändert definierten Relationen bei der Gründung einer weiteren Hermeneutic Unit gerade geöffnet ist, übernimmt die neue Hermeneutic Unit neben den Standards von ATLAS/ti auch die Modifikationen der anderen Hermeneutic Unit.
2. Wenn man die veränderten Relationen (oder eben Such-Schwärme) ausdrücklich in der Standard–Relationen–Datei "default.rel" für Kode–Kode–Relationen oder "hyph.rel" für Hyperlinks (bzw. "srchbib.skt" für Such-Schwärme) abspeichert, also damit den Standard von ATLAS/ti auf dem jeweiligen Rechner verändert. Um bei der Weiterarbeit auf einem zweiten Rechner über die gleichen modifizierten Standards zu verfügen, muss man dann aber entweder die betreffenden Standard–Dateien (aus dem Programm–Verzeichnis von ATLAS/ti) kopieren oder aber aus der Hermeneutic Unit mit den veränderten Relationen heraus auf dem zweiten Rechner die dortigen Standard–Dateien noch einmal überschreiben.

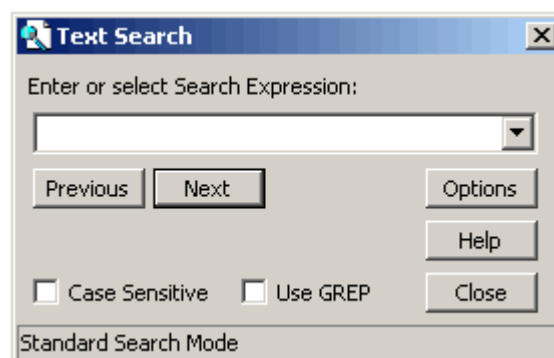
Textsuche und Autocoding

Es gibt zwei ähnlich aufgebaute Funktionen in ATLAS/ti, die dazu dienen, dass man Kodieroperationen teilautomatisieren bzw. die Suche nach Textstellen beschleunigen kann.

WICHTIG: Die Arbeit mit diesen Funktion setzt immer voraus, dass man in der Lage ist, einen "*string*" im Text zu definieren, nach dem zu suchen ist. Es muss ein formaler, eindeutiger Indikator auffindbar sein. Allerdings ist es nicht erforderlich, dass man (wie beim *autocoding* etwa) eine harte kategoriale Festlegung trifft, dass ein bestimmtes Konzept allein durch bestimmte formale Indikatoren angezeigt wird. Die Suche nach solchen *strings* ist vielmehr ein **heuristisches Mittel**, mit dem man auf einschlägige Textstellen aufmerksam werden kann, ohne zu behaupten, damit alle einschlägigen Stellen identifiziert zu haben. Das Lesen des Textes erspart man sich damit also nicht, wohl aber beschleunigt man für einen Teil der Fundstellen die Arbeit des Auffindens und Kodierens.

Suchfunktion

Die einfachere Variante ist ein schlichte **Suchfunktion für *strings***, also für eine beliebige Zeichenfolge. Der Button dafür befindet sich auf der Werkzeugleiste des Primärdokument-Fensters .



Hier gilt es genau zu überlegen, ob man genau ein Wort einträgt oder mehrere Worte oder einen Teil eines Wortes (z.B. den invariablen Wortstamm, um alle Beugungsformen und Wortvarianten gleichzeitig zu sichten, z.B.: "laun" findet "übellaunig", "launisch", "Laune", "Launen"). Wählt man den Stamm zu kurz, ist die Gefahr irrelevanter Ergebnisse höher ("enster" findet alle Arten von Fenstern, aber auch "Gespenster"), wählt man zulange

strings, insbesondere solche mit mehreren Worten, ist die Gefahr hoch, dass man nicht alle diese Ausdrücke auffindet, weil teilweise ein Zeilenumbruch dazwischen sitzt.

Wichtig ist auch, dass man zum Durchsuchen eines kompletten Textes immer zuerst den Cursor an den Beginn desselben setzen sollte, da die Suche immer dort beginnt, wo der Cursor aktuell steht. Standardmäßig durchsucht ATLAS/ti mit dieser Funktion den aktuellen Text. Setzt man die Suche in Gang, wird man jedoch gefragt, ob man darüber hinaus alle Primärtexte durchsuchen lassen will.

Hat man mittels der Suchfunktion eine Textstelle aufgefunden, in der der Suchbegriff vorkommt, so kann man sehr einfach mit der Maus in das Primärdokument-Fenster wechseln, die gewünschten Zitatgrenzen markieren und die Passage kodieren, um danach wieder in die Suchfunktion zu wechseln.

Ähnlich wie in der Suchfunktion von komfortablen Textverarbeitungen kann man auch in diesem Suchwerkzeug entscheiden, ob man die Groß- und Kleinschreibung beachten möchte oder nicht. Dies wird hier mit der Klickbox "**case sensitive**" entschieden.

Such-Schwärme

Oft kennt man nicht nur einen, sondern eine ganze Reihe formaler Indikatoren, die auf ein bestimmtes Konzept hinweisen können. Es wäre also praktisch, man könnte gleich in einem Durchgang nach allen diesen Begriffen suchen.

In ATLAS/ti ist es zu diesem Zweck möglich, Such-Schwärme zu definieren, die aus ganzen Sammlungen von Suchbegriffen bestehen. Diese Suchbegriffe können wiederum durch Einsatz von Asteriden (*) als generalisierten Platzhaltern auf Wortstämme verkürzt werden. Im *Autocoding*-Fenster finden sich englischsprachige Beispiele für diese Strategie (die gleichen Beispiel-Such-Schwärme finden sich auch in der aufklappbaren Liste im Suche-Menü).

Such-Schwärme bestehen aus zwei Teilen:

- a) ein Name für den Such-Schwarm (einzugeben in Großbuchstaben) und
- b) die Liste der zu suchenden Begriffe.

Die Trenner zwischen *strings* innerhalb des Such-Schwarms sind senkrechte Striche ("|"; ASCII-Zeichen 124). Ein Beispiel:

```
"ARBEIT=: $arbeitsteilung/*arbeit|Job/*beschäftigung|Arbeits*"
```

Der nach dem Titel als erstes eingefügte Suchbegriff, dem ein Dollar-Zeichen vorangestellt ist, repräsentiert einen anderen Such-Schwarm, der hier gewissermaßen als Teil des neuen Such-Schwarms ARBEIT integriert wird. Das **Dollarzeichen** ist der Indikator dafür. Begriffe, die ein Sternchen vorne oder hinten haben, werden sowohl in der Form ohne Stern und in Groß- oder Kleinschrift gefunden (also: "arbeit" "Arbeit") als Wortteile größere, zusammengesetzter Worte ("Nachtarbeit").

Such-Schwärme können über den Button "options" abgespeichert werden (standardmäßig als "*srchbib.skt*" im Programm-Verzeichnis von ATLAS/ti; andere Dateien können angelegt werden). Umgekehrt kann man vor Beginn der Suche auf diesem Weg auch eine benutzerdefinierte Datei mit Such-Schwärmen laden. Standardmäßig wird der Inhalt der Datei "*srchbib.skt*" geladen, die man auch modifizieren kann (vgl. dazu und zur Frage der Übertragung modifizierter Such-Schwärme auf andere Hermeneutische Einheiten den analogen Hinweis im Abschnitt zum Relationen-Editor).

Suche mit GREP

Eine elaborierte Variante der Textsuche ist die Suche unter Verwendung der sogenannten GREP-Ausdrücken. Diese Variante der Suchfunktion bietet einen Teil der in der "Regular Expressions"-Sprache verwendeten Ausdrücke als zur Suche im Text an. Im Prinzip geht es dabei um die Integration von Kontroll-Elementen, mit deren Hilfe sich über den einfachen *string* hinausgehende formale Definitionsmerkmale der gesuchten Textstellen ausdrücken

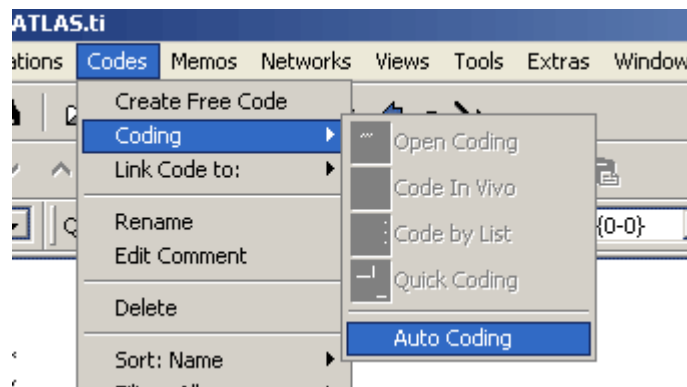
lassen. Mit GREP kann man z.B. alle Textstellen in Klammern finden oder festlegen, dass der gesuchte *string* nur dann gefunden werden soll, wenn er sich am Anfang einer Zeile befindet.

Die in ATLAS/ti angebotenen GREP-Ausdrücke beinhalten folgende Zeichen:

- ^ Bindet den nachfolgenden *string* an den Zeilenanfang
- \$ Bindet den vorangehenden *string* an das Zeilenende
- . findet jeden beliebigen einzelnen Buchstaben (universaler, mengengenauer Platzhalter; "ein.." findet "einen", "einem", "einer", aber auch "meine"" oder "einfach")
- * Findet jede den vorangehenden Ausdruck oder den um ein Zeichen gekürzten Ausdruck ("so*" findet "s" und "so" nicht aber "o")
- + Findet wenigstens ein Vorkommen des vorangehenden Ausdrucks ("o+" findet "o" ebenso wie "oo" oder "ooo")
- ? Findet einen vorangestellten Suchausdruck oder einen um einen Buchstaben verkürzten Teil davon ("eine?" findet "ein", "mein", "meine", "eine" nicht aber "eigentlich")
- [] Findet eine Liste oder ein Reihe von Zeichen: [a-z] oder [0-9] oder [aeiou]
- [^] Invertierung des vorhergehenden Ausdrucks: Findet alle nicht mit der Liste übereinstimmenden Ausdrücke: [^0-9] = alle nicht-numerischen Zeichen im Text
- :d Findet eine beliebige Zahl (engl. "digit"): ":d:d" findet zweistellige Zahlen oder zwei aufeinander folgende Ziffern einer mehrstelligen Zahl, nicht aber einzelne Ziffern.

Autocoding

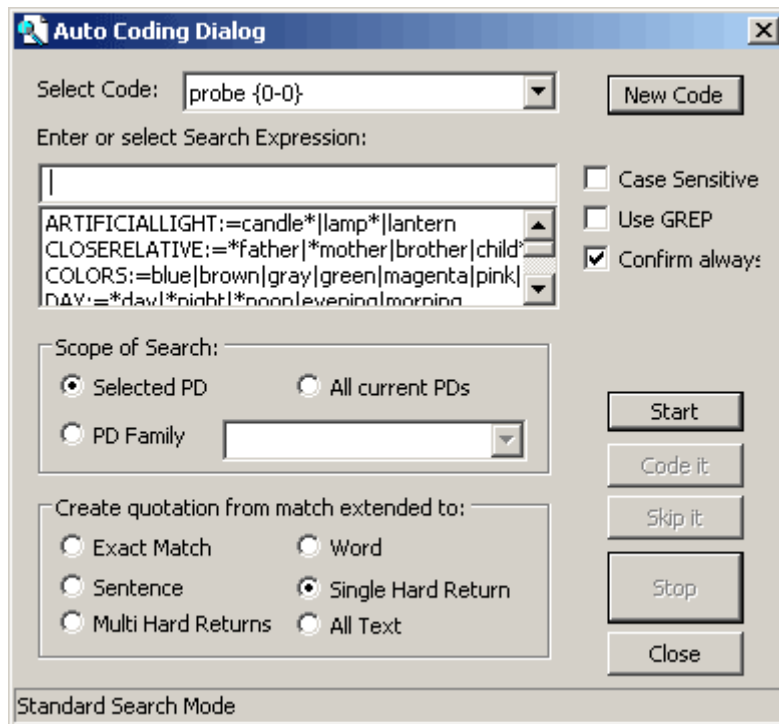
Die komplexere Variante der Suchfunktion ist die **Autocoding-Funktion**



Hier kann man nach *strings* suchen und dafür sorgen, dass diese *strings* oder eine definierte Umgebung dieser *strings* mit einem vorher bestimmten Code kodiert wird. Als Code wird der vorher aktive Code gesetzt. Die Auswahl kann aber über das in die autocoding-Funktion integrierte Code-Listenfenster jederzeit geändert werden.

Dabei kann man optional festlegen, ob

- a) neben dem *string* selbst der Satz oder der Absatz mit kodiert werden soll und
- b) die Suche sich auf den aktiven Primärtext, die Primärtext-Familie oder alle Primärdokumente beziehen soll.



Tipp: ATLAS/ti versteht einen Satz von Punkt zu Punkt, d.h. wenn man innerhalb des Satzes einen Punkt verwendet (bei Abkürzungen etwa), dann wird dies als Satzende oder -anfang interpretiert. Absätze werden als durch von zwei Absatzmarken voneinander getrennt verstanden. Ist nur eine Absatzmarke vorhanden, wird der folgende bzw. vorangehende Absatz etc. bis zur nächsten Leerzeile mit markiert oder kodiert!

Ebenso wird auch die Option angeboten, nicht sofort alle Fundstellen automatisch zu kodieren, sondern **jede Stelle einzeln zu bestätigen** bzw. zu verwerfen. Auch hier ist die Verwendung von GREP, die "case sensitive"-Suche und das Arbeiten mit Such-Schwärmen möglich.

Texte schreiben in ATLAS/ti: Memos und Kommentare

Das A und O sozialwissenschaftlicher Analysen ist der Text – nicht nur der, den wir analysieren, sondern vor allem der, den wir aus unsrer analytischen Arbeit heraus produzieren.

ATLAS/ti stellt zu diesem Zweck fast in jedem Arbeitsbereich einer Hermeneutic Unit eine oder mehrere Editor-Funktionen zur Verfügung und bietet darüber hinaus noch eine separate Verwaltung der Memos über die Memoliste an.

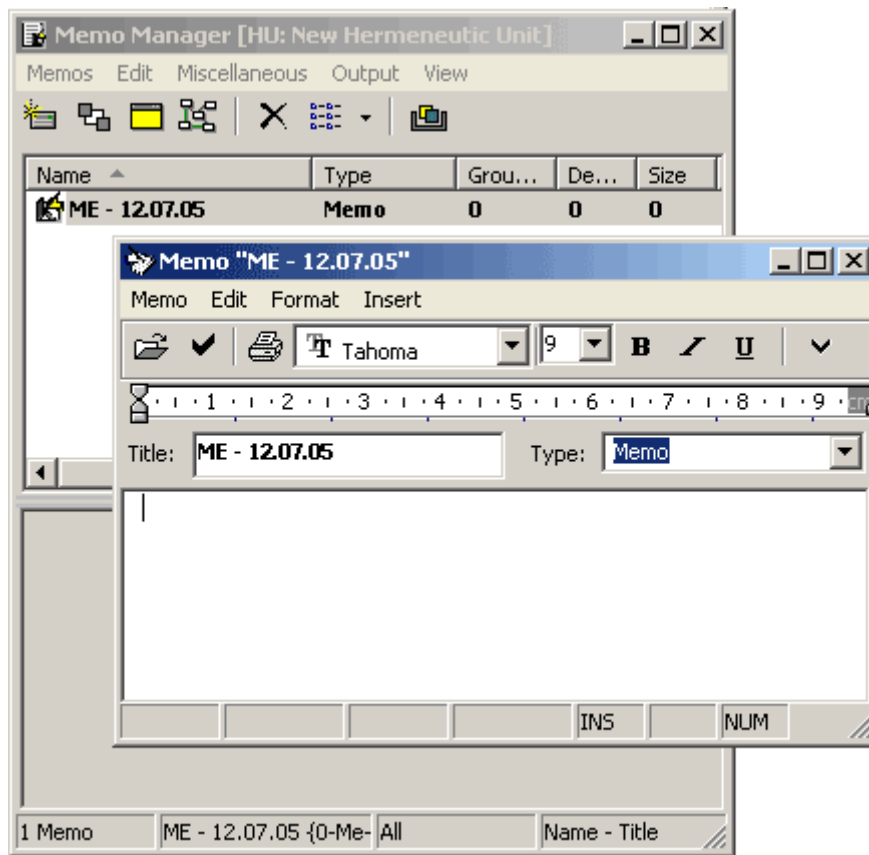
Zu unterscheiden ist zwischen **Kommentaren** und **Memos**:

Kommentare sind (programmlogisch) keine eigenständigen Objekte, die man "verwalten" kann, sondern sie sind textuelle "Anhängsel" anderer Objekte. So können wir etwa zu Codes, Zitaten oder auch zu Familien jeweils erläuternde Kommentare anlegen, die wir bei jeder Aktivierung des entsprechenden Objektes in einem Kommentar-Fenster angezeigt bekommen können.

Memos hingegen sind eigenständige Objekte, die ausdrücklich erzeugt werden müssen, die in einer eigenen Liste verwaltet werden und die man schon bei der Erzeugung oder erst im späteren Arbeitsprozess mit anderen Objekten in (allerdings nicht qualitativ definierbare) Beziehungen setzen kann.


Zum **Erzeugen von Memos** stehen uns verschiedenen Zugänge zur Verfügung, die auch davon abhängen, zu welchem Objekt das Memo in Verbindung stehen soll. Die einfachste

Variante ist das Erzeugen eines "freien", also völlig unverbundenen Memos.



Nach der Eingabe eines Namen (Standard ist "ME" + das aktuelle Datum) ein Editor-Fenster zur Eingabe der Memo zur Verfügung.

Wenn man beabsichtigt, zu einer bestimmten Textstelle, die zitiert und vielleicht auch kodiert ist, ein **Memo anzulegen**, dann gibt es dazu eine besonders einfache Prozedur:

Man klickt in der Werkzeugleiste den Button  unterhalb der Kodier-Funktionen an. Daraufhin erscheint ein Memo-Fenster zur Eingabe des Memo-Namens und der Memo.

Alle Memos werden in der Hermeneutic Unit abgelegt. Es sind also im Normalfall keine eigenständigen Dateien. Wenn die Texte lang und länger werden, wird also auch die Hermeneutic-Unit-Datei größer und größer, d.h. auch: langsamer und langsamer. Um hier Abhilfe zu schaffen, gibt es die Möglichkeit, einzelne oder alle Memos aus der Hermeneutic Unit auszulagern und **separat als einzelne Dateien zu speichern**. Diese Funktion befindet sich im Memo-Manager unter "outout". Dort kann man wählen, ob man alle oder nur einige Memos auslagern möchte. Zu empfehlen ist hier sicherlich, sich **beim Auslagern auf die längerer Memos zu beschränken**.

WICHTIG: Die Dateinamen für die ausgelagerten Memos werden von ATLAS/ti selbst verwaltet, denn das Programm muss diese ja in der weiteren Arbeit auf der Festplatte wiederfinden und identifizieren. Daher darf man **keinesfalls** die (etwas kryptischen) **Namen dieser Daten** von außen **verändern**. Wohl aber darf man die Memos nun in separaten Editoren bearbeiten – solange man darauf achtet, dass das Dateiformat (.txt oder .rtf) nicht verändert wird.

Recherche in den Daten

Das Query Tool

Es ist erfreulich, dass Programme wie ATLAS/ti uns das Kodieren großer Mengen

qualitativer Daten sowie die Erarbeitung semantischer Netzwerke so sehr erleichtern – allerdings erhalten wir damit analytische Gebilde von einer Größe und Komplexität, die wir durch einfache visuelle Inspektion kaum noch überschauen, geschweige denn zuverlässig auf ihre Konsistenz und Verlässlichkeit prüfen können. Dazu bedarf es eines zusätzlichen Typs von Werkzeug, das häufig unter dem Namen "Retrieval-Funktion" firmiert, weil es dazu dient, uns eine Selektion der angelegten Zitatstellen abhängig von zu benennenden Indikatoren (also den Codes) "zurückzubringen".

Unterschied zwischen Query und Textsuche

Der **Unterschied zur einfachen Textsuche** besteht darin, dass hier nicht nach formalen Indikatoren in den Daten selbst gesucht wird, sondern dass alle Textstellen heraus gesucht werden, denen wir einen *externen* formalen Indikator, eben den Code, zugewiesen haben. In ATLAS/ti wird dieses Werkzeug als "**Query Tool**", also als Abfrage-Werkzeug bezeichnet. Es ist das wahrscheinlich komplexeste Werkzeug im ganzen Programm.

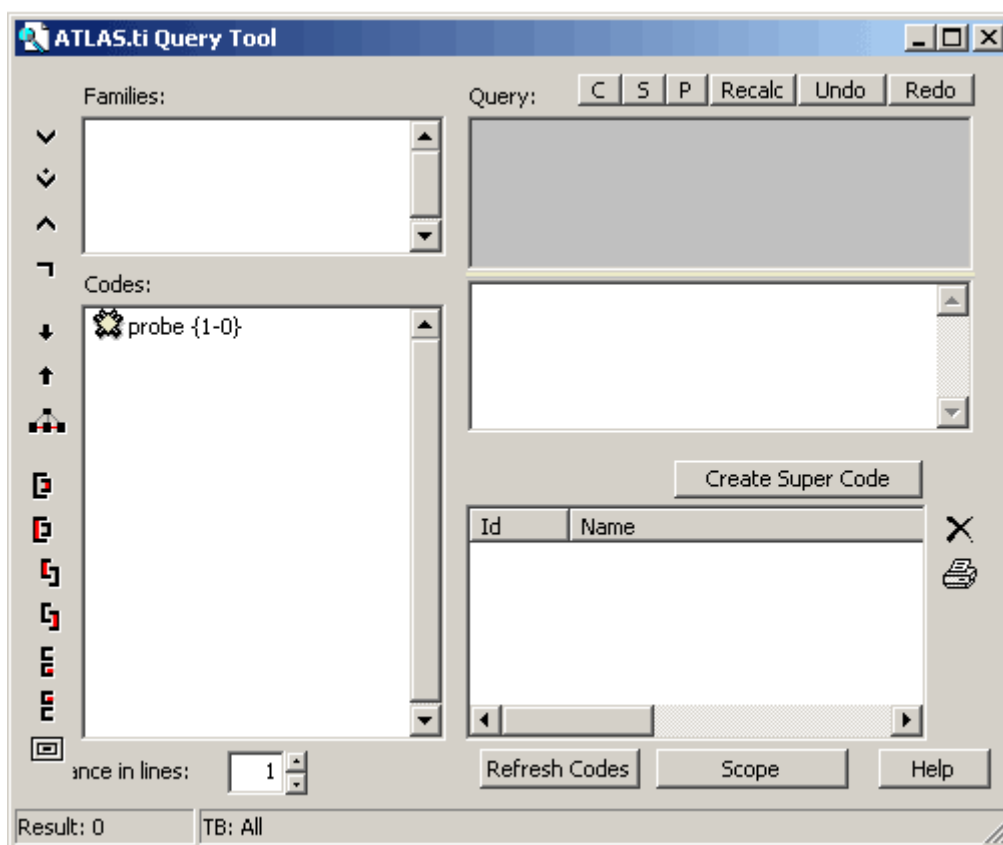
Eine Abfrage ("*query*") besteht aus einer mehr oder weniger komplexen Kombination von Operanden und Operatoren, die die Bedingungen spezifizieren, die Zitate erfüllen müssen, um heraus gesucht zu werden.

Bereits außerhalb des *Query Tools* werden in ATLAS/ti laufend *Retrieval*-Prozeduren ausgeführt: Jeder Doppelklick auf einen Code in der Kodeliste führt zu einer Suche nach allen Zitatstellen, die mit diesem Code belegt wurden. Als Ergebnis wird bei nur einem Treffer sofort das Zitat im Kontext angezeigt oder aber – bei mehr als einem Treffer – es wird eine Liste der Zitate zur Auswahl angeboten. Das *Query Tool* dient den komplexeren Suchen, bei denen nach mehr als einem Code und nach bestimmten, z.B. logisch definierbaren Kombinationen von Codes gesucht werden soll. Werfen wir einen Blick auf die Oberfläche dieses Werkzeuges, das sich mit einem Klick auf den entsprechenden Button



in der Symbolleiste öffnen lässt:

Query Tool: Die Oberfläche



Wir sehen ein in fünf Bereiche geteiltes Fenster vor uns: Oben links wird eine Liste der Kode-Familien angezeigt, unten links eine Liste der Codes. Diese beiden Objektarten sind die basalen **Operanden** für Such-Abfragen. Rechts oben befinden sich zwei durch eine verschiebbare Linie getrennte Fenster, in denen die Such-Abfrage als Ausdruck dargestellt wird. Unten rechts befindet sich das Ergebnis-Fenster, in dem die gefundenen Zitate zu jedem Zeitpunkt der Suche (und der sukzessiven Konstruktion des Suchbegriffs) dargestellt werden.

"Garniert" wird diese Anordnung durch einige Ensembles von Toolbars und kleineren Gruppen von Buttons. Insbesondere fällt an der linken Seite die vertikal angeordnete Toolbar mit gruppierten Operatoren auf. Oben rechts befindet sich eine Gruppe von sechs Buttons, die bei der Konstruktion von Such-Ausdrücken benötigt werden. Zwischen den beiden Suchausdruck-Fenstern und dem Ergebnis-Fenster sind Schalter für die – weiter unten erläuterte – *Supercode*-Funktion und für einen Wechsel des Anzeigemodus der unteren Suchausdruckfensters (*Infix*- vs. *Prefix*-Darstellung) angeordnet. Rechts vom Ergebnisfenster gibt es die zwei schon aus anderen Zusammenhängen geläufigen *Icons* für das Löschen und das Drucken von Ergebnissen, während sich unter diesem Fenster neben dem Hilfe-Button noch die Vorauswahl der zu recherchierenden Texte ("*Textbase selection*") aufrufen und die Fensterdarstellung aktualisieren läßt.

Die Abfragesprache – gewöhnungsbedürftig, aber praktisch

Reversed Polish Notation

Bevor wir zum Procedere der Konstruktion von Such-Abfragen kommen, noch ein Wort zur in ATLAS/ti verwendeten Abfragesprache. Alle Abfragen werden unter Verwendung der sogenannten "**Reversed Polish Notation**" (RPN) gebildet. Das klingt komplizierter als es ist.

Der wichtigste Unterschied zwischen dieser Notation und der uns aus verschiedenen Datenbank-Abfragesystemen sowie von Taschenrechnern geläufigen "Infix"-Notation besteht darin, dass man bei der RPN keine in Klammern verschachtelten Ausdrücke konstruiert. Statt dessen werden die Abfragen "inkremental", also Schritt für Schritt entwickelt, indem zuerst die Operanden (in der Regel zwei, in einigen Ausnahmefällen – NOT, UP, DOWDN – nur einer) und dann der auf sie anzuwendende Operator eingegeben werden. Das Interessante daran ist, dass **jeder Klick** oder Doppelklick auf einen Operanden oder eine Operator bereits ein **Ergebnis erzeugt**, das unverzüglich im Ergebnis-Fenster angezeigt wird. Zugleich ist es **praktisch unmöglich**, eine **syntaktisch falsche Abfrage zu produzieren** (allerdings kann die Abfrage ohne weiteres inhaltlich unsinnig sein), wie es beim Verschachteln von Klammerausdrücken leicht passieren konnte.

Im Grunde sind wir durch die Benutzung von mausgesteuerten Programm-Oberflächen längst an die Logik dieser Abfragesprache gewöhnt, denn auch dort gilt: Erst wird ein Gegenstand ausgewählt, dann die mit ihm durchzuführende Operation (z.B. Kopieren) ausgewählt.

Die Bedienung des Query Tools

Formulieren einer Such-Abfrage

Jede Abfrage beginnt also damit, dass wir – wahlweise aus der Kode- oder aus der Kode-Familienliste – einen, in der Regel aber **zwei Operanden auswählen**. Jeder der Operanden wird durch das Anklicken sofort im oberen rechten Fenster angezeigt. Zugleich erscheint im Ergebnis-Fenster die Liste der ihm zugeordneten Zitate. Sobald man dann aus der umfangreichen Liste der Operatoren (s.u.) denjenigen ausgewählt hat, der auf die beiden Operanden angewandt werden soll, erscheint auch er im Fenster oben rechts (und

zwar nun schon als Teil des kompletten Such-Ausdrucks mit den gewählten Operanden in einer Zeile), während im Ergebnis-Fenster das Resultat der Operation angezeigt wird (bei einer Booleschen UND-Operation etwa die Schnittmenge der Zitatreferenzen der beiden gewählten Codes). Die **Anfrage-Ergebnisse** können sofort nicht nur in der Liste inspiziert, sondern – durch eine Doppelklick auf einzelne Zitatreferenzen – auch im Kontext des Primärtext-Fensters angezeigt und bearbeitet werden.

Organisation des Such-Anfrage-Fensters

Das obere **Such-Anfrage-Fenster** ist so organisiert, dass jeder einzelne Ausdruck als ein Element in einem Stapel erscheint, wobei die zuletzt gewählten Ausdrücke oben abgelegt werden. So werden z.B. zwei angeklickte Codes dort oben nacheinander aufgeschichtet. Sobald aber durch hinzufügen eines passenden Operators eine gültige Suchanfrage formuliert wurde, wird dieser ganze Ausdruck (Kode plus Operator als eine Schicht in diesem Stapel dargestellt. Daran wird auch deutlich, dass man von nun an diesen Ausdruck ebenso als Operanden benutzen kann (jedenfalls mit den meisten Operatoren) wie einfache Codes.

Die **Suche lässt** sich nun **Schritt für Schritt erweitern** und tentativ in die richtige Richtung weiterentwickeln. Das bisherige Ergebnis stellt für die Fortsetzung der Such-Abfrage nun den ersten Operanden dar. Durch Hinzufügen eines zweiten Operanden und eines auf die beiden anzuwendenden Operators erhalten wir ein neues Ergebnis. Auf diese Weise kann die Such-Abfrage beliebig lange fortgesetzt und spezifiziert werden.

Für den Fall, dass wir bei einer Operation ein unbefriedigendes Ergebnis erhalten haben und gerne beim vorhergehenden Schritt wieder einsetzen würden, steht uns rechts oben ein "**undo**"-Button zur Verfügung, der die letzte Operation rückgängig macht, den Rest des Such-Ausdrucks aber stehen lässt. Falls wir in diesem Moment bemerken, dass die letzte Operation so falsch gar nicht war, können wir sie mit dem "**redo**"-Button wiederherstellen.

Operatoren

Es werden drei verschiedene Typen von Operatoren angeboten. Im Query Tool finden sie sich in der linken, vertikalen Toolbar. Von oben nach unten sind dort angeordnet:

- 4 **Boolesche Operatoren**, die die mathematisch-logische Kombination von Suchworten erlauben.
- 3 **Semantische Operatoren**, mit deren Hilfe sich die von uns entwickelte Netzwerkstruktur der wechselseitigen Beziehung zwischen den Codes untersuchen erkunden lässt. Man kann dies Operatoren auch als Thesaurus Operatoren bezeichnen.
- 6 "**Proximity**"-Operatoren dienen dazu, die räumliche Nähe zwischen Text-Segmenten zu eruieren. Gesucht werden kann nach Einbettungen (*embeddedness*), Überlappungen (*overlapping*) und gleichzeitigem Auftreten (*co-occurrence*) von Text-Segmenten, die uns interessieren.

Es gibt eine "**tooltip**"-Hilfe, die für jeden der Operatoren Auskunft über die genaue Verwendung gibt. Insbesondere bei den Proximity-Operatoren taucht immer wieder die Frage auf, in welcher Reihenfolge die Operanden in die Operatoren-Funktion eingefügt werden.

Hier ein kurze Überblick über die Operatoren im einzelnen:

Boolesche Operatoren

Folgende vier Boolesche Operatoren sind verfügbar: OR, XOR, AND, und NOT. Die ersten drei benötigen als binäre Operatoren jeweils zwei Operanden. Einzig die NICHT-Operation kommt mit nur einem Operanden aus. Allerdings können die Operanden selbst bereits in sich sehr komplex strukturiert sein (z.B. wenn es sich um Kode-Familien, um "Supercodes"

(s.u) oder um vorab bereits komponierte Abfrage-Ergebnisse handelt.

OR Die einfache ODER-Operation liefert alle Zitate, die mindestens mit einem der Operanden kodiert sind, aber auch diejenigen, die mit mehr als einem der fraglichen Operanden verbunden sind. Diese Operation hat also typischerweise einen hohen Output, ist allerdings auch wenig präzise.

XOR Während die OR-Operation also im Grunde mit "wenigstens eines von" zu übersetzen ist, kommt die in Abfragesystemen weniger gebräuchliche XOR-Funktion der alltäglichen Verwendung des Wortes "oder" viel näher. Mit XOR werden alle Fundstellen heraus gesucht, die entweder mit dem einen oder mit dem anderen Operanden kodiert sind. Fundstellen, die beide Codes aufweisen, werden nicht angezeigt. Es geht also um ein "Entweder-Oder".

AND Die Operation UND findet all jene Zitate, die alle mit den Operanden spezifizierten Bedingungen erfüllt. Diese Funktion ist also hoch selektiv und produziert in der Regel ein sehr geringes Output.

NOT Mit dem NOT-Operator wird die Abwesenheit einer Bedingung geprüft. Als Ergebnis liefert er alle Zitate, die nicht mit dem fraglichen Code kodiert sind.

Übrigens kann man sich jede Kode-Familie, wenn man sie im Query Tool verwendet, als eine große OR-Operation mit allen zugeordneten Codes vorstellen. Da die Kode-Familien nicht wechselseitig ausschließend sind, jeder Code also zugleich in verschiedenen Kode-Familien beinhaltet sein kann, besteht bei Verwendung von Kode-Familien in komplexen *Retrieval*-Prozeduren immer auch die Gefahr, semantischen Nonsens zu produzieren. Z.B. ist es schon logisch unsinnig, eine XOR-Operation mit zwei Kode-Familien als Operanden durchzuführen, sofern auch nur ein Code in beiden Familien präsent ist.

Semantische oder Thesaurus-Operatoren

Die Operatoren dieser Gruppe machen sich die (teilweise) hierarchisierte Struktur des semantischen Netzwerkes zunutze, das wir in der vorherigen tentativen Theoriebildungs-Arbeit aufgebaut haben.

DOWN Der DOWN-Operator durchsucht das Netzwerk von konzeptuell höherrangigen Codes zu den basaleren und trägt alle Zitate zusammen, die dabei auftauchen. Bei diesem Operator werden nur **transitive Beziehungen** (wie "is a" oder "is part of") zwischen Codes verwendet, alle anderen Beziehungen bleiben außer Betracht. Mit dem DOWN-Operator bekommen wir in der Regel relativ viele Treffer. Da hinter der Beziehung der hier verbundenen Codes aber eine an den Daten entwickelte Theorie steht, ist das Ergebnis dennoch in der Regel "präziser", als das des oben erwähnten OR-Operators.

UP Der UP-Operator findet alle Zitate, die entweder dem selektierten Code oder dem ihm direkt übergeordneten Code zugeordnete sind. Übrigens: Im Unterschied zu einer klassischen unidimensionalen Thesaurus-Struktur idealtypischer Klassifikationssysteme kann es in semantischen Netzwerken vorkommen, dass ein Code mehrere Oberbegriffe hat (z.B. je nach der Dimension auf die sich die Zuordnung bezieht, also: "Arbeitseinkommen" kann Bestandteil von "Arbeitsmotivation" sein, aber auch von "Einkommensarten").

SIBlings der SIBlings- oder Geschwister-Operator sucht alle Zitate, die mit dem gewählten Code kodiert sind oder mit einem Code, der ebenfalls Unterbegriff eines Codes ist, von dem der gewählte Code selbst ein Unterbegriff ist. Wenn wir im Beispiel der Arbeitseinkommen bleiben: Mit der SIBlings-Funktion würden alle Zitate zu den verschiedenen Einkommensarten und zu den verschiedenen Arten von Arbeitsmotivation zusammengetragen werden (es gibt sicherlich semantisch sinnvollere Abfragen, aber es ist ja nur ein Beispiel für das Prinzip).

WICHTIG: Für diese semantischen Operationen dürfen (verständlicherweise) nur Codes, nicht aber Kode-Familien, Supercodes oder vorgängige Suchergebnisse als Operanden

verwendet werden, denn sonst wären die Beziehungen nicht mehr eindeutig bestimmbar.

Proximity-Operatoren

Die Operatoren dieser Gruppe beziehen sich alle auf die relative Nähe oder Ferne zweier Operanden zueinander, daher benötigen sie auch jeweils genau zwei Operanden für eine gültige Such-Abfrage. Da Proximity-Operatoren nicht-kommutativ (nicht vertauschbar) sind, spielt bei ihnen die Reihenfolge der Eingabe der Operanden eine große Rolle. Denn während "A oder B" gleichbedeutend mit "B oder A" ist, ist "A folgt B" deutlich etwas anderes als "B folgt A".

Als Faustregel für die Eingabe-Prozedur bei diesen Operatoren kann man sich merken, dass sie immer in der Reihenfolge der natürlich sprachlichen Darstellung einzugeben sind: "A folgt B" erfordert also, dass erst A und dann B angeklickt werden, bevor schließlich der Operator angehängt wird.

Aufgrund der Nicht-Kommutativität gibt es die Proximity-Operatoren jeweils in zwei Versionen (A zu B und B zu A).

Einbettungs-Operatoren suchen nach Zitaten, deren eines das andere enthält oder umschließt und die mit den spezifizierten Codes kodiert sind.

WITHIN findet alle Zitate, die mit A kodiert und von einem mit B kodierten Zitat umschlossen sind (A WITHIN B).

ENCLOSES findet dagegen alle Zitate mit dem Code A, die ein Zitat mit dem Code B umschließen (A ENCLOSES B).

Die **Überlappungs-Operatoren** suchen nach Zitaten, die einander überlappen.

Dabei sucht

OVERLAPPED_BY alle diejenigen Stellen, an denen der zuerst zu benennende Operand (A) durch den als zweites benannten Operanden (B) überlappt wird (A OVERLAPPED_BY B).

OVERLAPS hingegen sucht das Gegenteil davon, also diejenigen Passagen, in denen der zuerst benannte Operand (bzw. die mit ihm kodierten Zitate) den zweiten Operanden überlappt (A OVERLAPS B).

Distanz-Operatoren (besser: Sequenz-Operatoren) suchen nach zwei Kriterien: Erstens nach einer **bestimmten Abfolge**, in der die jeweils mit Operand A bzw. mit Operand B kodierten Zitatstellen aufeinander folgen und zweitens unter der Bedingung einer bestimmten **maximalen "Entfernung"** der Zitate voneinander (ausgedrückt in einer spezifizierbaren Anzahl von Zeilen).

FOLLOWS findet dabei jene mit A kodierten Zitate, die einer mit B kodierten Passage folgen (A FOLLOWS B).

PRECEDES findet umgekehrt alle mit A kodierten Zitate, die mit B kodierten Zitaten vorangehen (A PRECEDES B).

Wichtig: Die Anzahl von Zeilen als Entfernungsindikator ist natürlich ein Behelf. Weder ist es methodologisch begründbar, warum eine bestimmte Entfernung noch signifikant ist, während eine andere es schon nicht mehr sein soll, noch ist die Bedeutung des Zeilenmaßes über die verschiedenen Textsorten und Formatierungen hinweg stabil. Es handelt sich also wiederum lediglich um ein heuristisches Hilfsmittel mit einiger Unschärfe, für das wir selbst aus unserer analytischen Arbeit heraus bestimmen müssen, wie es sinnvoll eingesetzt werden kann.

CO-OCCURS schließlich ist der einzige Operator dieser Gruppe, der nur in einer Version existiert, da es hier – weniger spezifisch also – nur darum geht festzustellen, ob Zitate zu zwei angegebenen Codes in beliebiger Reihenfolge überlappen oder einander wechselseitig

einschließen "A CO-OCCURRING WITH B" ist also semantisch dasselbe wie "B CO-OCCURRING WITH A". Dieser Operator findet also die Summe der Treffer der vier "overlap"- und "encloses"-Operatoren.

Einige weitere Funktionen des Query Tools

Neben den vorgestellten Operatoren bietet das *Query Tool* noch eine Reihe weiterer Funktionen. Ein Teil davon ist ähnlich wie in einem Taschenrechner zu bedienen und dient dem Komfort bei der Konstruktion von Suchanfragen:

- C** **Löscht** den gesamten **Satz von Ausdrücken** (im rechten oberen Such-Anfrage-Fenster)
- S** **Vertauscht** die beiden im Stapel oben liegenden **Elemente** (sehr praktisch, wenn man sich bei der Eingabe-Reihenfolge vertan hat.)
- P** **kopiert** den **obersten Ausdruck** des Stapels noch einmal und legt ihn darüber ab – nützlich vor allem, wenn man einmal einen komplexen Ausdruck reproduzieren muss und ihn nicht "Klick für Klick" noch einmal eingeben möchte.

Recalc Falls zu einem im geöffneten *Query Tool* gerade zur Suche benutzten Code parallel noch neue Zitate produziert wurden, erlaubt diese Funktion die **Neuberechnung des Suchergebnisses**, ohne dass die Anfrage als ganz neu erstellt werden muss.

Undo Entfernt den im Stapel oben liegenden Ausdruck aus der Suchanfrage.

Redo neutralisiert die Undo-Funktion, indem der gerade entfernte Ausdruck wieder auf dem Stapel abgelegt wird.

Darüber hinaus finden sich folgende Funktionen in Form von Buttons im *Query Tool*-Fenster:

Supercode Erlaubt die Bildung eines sogenannte Supercodes (wird weiter unten näher vorgestellt)

Prefix Display Der untere Teil des zweigeteilten Suchanfrage-Fensters stellt den Suchausdruck entweder in **Prefix** oder in **Infix-Schreibweise** dar. Letzteres ist die Standardeinstellung. Mit dem Button kann man zwischen den beiden Darstellungsarten wechseln.

Distance in lines Hier wird die Anzahl von Zeilen eingestellt, die die Operatoren FOLLOWS und PRECEDES bei ihrer Suche als Entfernungskriterium benutzen (Standardeinstellung: 5 Zeilen).

Refresh Mit dieser Funktion kann man die **Kode- und die Kode-Familien-Liste** im *Query Tool* **aktualisieren**, falls man unterdessen Änderungen in der Hermeneutic Unit vorgenommen hat.

Der Button "**Textbase Selection**" schließlich öffnet ein weiteres ähnlich strukturiertes Fenster, in dem man **einstellen** kann, **welche Primärtexte** oder Primärtext-Familien überhaupt **in die Suche einbezogen werden** sollen. Wenn wir etwa die verschiedenen Typen von Daten (Interviews, Beobachtungsprotokolle, Dokumente o.ä.) in unterschiedliche Primärdokument-Familien sortiert haben, dann können wir hier bestimmen, dass wir uns nur auf eine oder einen Teil der insgesamt verfügbaren Datentypen beziehen wollen (z.B. nur Interview und Beobachtungsprotokolle, aber keine Dokumente). Oder wir können auf diesem Weg die Daten eines bestimmten Falles für die Suche auswählen.

Supercodes

Was aber steckt hinter der schon mehrfach erwähnten "Supercode"-Funktion? Dahinter verbirgt sich die Möglichkeit, eine **komplette Suchanfrage als dynamischen Kode**

abzuspeichern. Was bedeutet das? Nun, auf diesem Weg wird nicht etwa ein bestimmtes Anfrage-*Ergebnis*, also die Treffer-Liste zur späteren Verwendung gespeichert, sondern die komplette Such-Operation selbst wird zu einem Kode, der sich in der weiteren analytischen Arbeit seine "Treffer" selbst sucht. Das funktioniert so:

Wenn man eine Suchanfrage konstruiert hat, mit der man ganz besonders zufrieden ist und von der man meint, dass sie einen bestimmten Aspekt der entstehenden Theorie gut abbildet (Thomas Muhr spricht hier von einer "**frozen hypothesis**"), dann kann man durch Anklicken des Supercode-Buttons und die Eingabe eines Supercode-Namens eine Art neuen Kode erzeugen, der **auch in der Kodeliste verzeichnet** und **im Netzwerk-Editor "operabel"** ist. Der hinter dem Supercode stehende Suchausdruck wird als Kode-Kommentar abgelegt und ist somit jederzeit leicht einsehbar. Jedesmal, wenn man in der weiteren analytischen Arbeit einen der in diesen Supercode als Operand eingebundenen Kodes benutzt, also mit ihm kodiert oder ihn neu mit anderen Kodes in Beziehung setzt, wird dadurch auch das "Ergebnis" verändert, das der Supercode erbringt – und zwar nicht erst bei erneutem Öffnen des *Query Tools*, sondern schon beim einfachen Aktivieren des Supercodes in der Liste – denn bei jeder Aktivierung des Supercodes wird die dahinter liegende Suche noch einmal ausgeführt.

Ein bisschen Aufpassen muss man bei der Verwendung von Supercodes allerdings schon, da man mit dieser Funktion auch **Schleifen** produzieren kann, z.B. Supercodes, die Kode-Familien beinhalten, die wiederum den gleichen Supercode beinhalten. Und: Natürlich macht es wenig Sinn, mit Supercodes direkt die Daten zu kodieren. Deshalb ist die Möglichkeit des direkten Kodierens mit Supercodes unterbunden.

Arbeiten im Team

ATLAS/ti stellt eine Reihe von Hilfsmitteln zur Arbeit im Team zur Verfügung. Zwar ist es auch mit der Netzwerk-Version nicht möglich, zeitgleich von mehreren Rechnern aus an der gleichen Hermeneutic Unit zu arbeiten (das ginge nur, wenn die Hermeneutic Unit keine handliche ASCII Datei mehr wäre, sondern eine Datenbank), doch gibt es andere Methoden, die für die Kooperation im Team nützlich sind.

Als erstes wäre da die Verwaltung unterschiedlicher Benutzer, mit unterschiedlichen Rechten und einschließlich der Verwaltung der von ihnen im Arbeitszusammenhang verursachten Veränderungen. Zum zweiten kann man die von unterschiedlichen Forschern erzeugten Hermeneutic Units zu einem Thema miteinander verschmelzen um so zu einem gemeinsamen Ergebnis zu kommen. Schließlich kann man die Ergebnisse der Arbeit in unterschiedlicher Weise so nach außen zugänglich machen, dass andere darauf zugreifen können. In allen diesen Bereichen wartet ATLAS/ti mit interessanten Funktionen auf.

Verwaltung von Benutzern

Die Verwaltung von Benutzern spielt sich auf zwei Ebenen ab: Zunächst werden Benutzer und ihrer Rechte für das Programm als Ganzes definiert und über eine *login*-Prozedur abgefragt. Sodann kann man für einzelne Hermeneutic Units die Benutzungsrechte differenziert vergeben.

User Editor

Zur Verwaltung der berechtigten Benutzer des Programms gibt es eine **Eingabemaske**, die man über das "Extras"-Menü erreicht. Dort kann mit "**New user**" ein Dialog abgerufen werden, in dessen Verlauf man vier Informationen eingeben muss: Den internen Namen oder "*account*" (er wird zur Bezeichnung der Autorenschaft in den Hermeneutic Units verwendet, das Passwort, den Vornamen und den Nachnamen des jeweiligen Benutzers. Zugleich werden dort auch die **Benutzer-Rechte** des neuen "*users*" festgelegt (einfacher Benutzung bis zu Administrationsrechten). Sinnvoll ist es natürlich, wenn nur eine Person für die Verwaltung des Programms verantwortlich ist und die Administrationsrechte besitzt.

Standardmäßig wird das Programm mit einer Einstellung installiert, bei der ein sogenannter "Super User" automatisch eingeloggt wird. D.h. wenn man das Programm nach der Installation aufruft, hat man automatisch Administrator-Rechte. Man wird aber unter dem recht irreführenden Namen "Super" geführt. Die Änderung dieser Grundeinstellung erfolgt nicht über die Konfigurations-Registerkarten im Menü *Extras*, sondern durch das Editieren der "user.ini"-Datei im Textbank-Verzeichnis (Aber Vorsicht: Unbedingt zuerst eine Sicherungskopie anfertigen, denn man kann hier viel zerstören. Am Ende dieser Datei gibt es eine Option "AutoLogin", die zunächst auf "enabled" steht und bei Bedarf in "disabled" geändert werden kann. Darunter steht der Username und das Passwort der automatisch einzuloggenden user. Hier kann man auch seinen eigenen account eintragen, wenn man nicht gerne "Super" sein will.

Tipp: Die Benutzung des *User Editors* ist etwas gewöhnungsbedürftig und teilweise nicht eben intuitiv zu bedienen. Insbesondere muss man nach jeder Änderung über das Menü des *User Editors* die Option Speichern wählen, damit die Einträge in die Datei "hermendc.hdb" im Verzeichnis "...atlasti/Programm/" gespeichert werden – das Betätigen des OK-Buttons genügt nicht. Auch hat es keinen Zweck, in die vier oberen Felder mit den Daten des jeweils aus der Liste ausgewählten Users irgendwelche Einträge zu machen – diese werden nicht angenommen. Auch kann man nicht wirklich mit einem Klick zwischen den in der Mitte links angezeigten beiden Varianten von Benutzerrechten ("standard" und "administrator") wählen, sondern muss über das Menü "Edit" die Option "Change Access Rights" wählen. Dort stehen einem dann die drei Varianten "low", "high" und "administrator" zur Verfügung.

Mit diesen Daten kann man sich **beim Start der Programmes einloggen** (wenn in der user.ini der entsprechende Eintrag gemacht ist, kann das Einloggen auch automatisch erfolgen) Selbst dort, wo nur ein einzelner Benutzer am Rechner arbeitet, macht das Einloggen mit den korrekten Benutzer-Daten Sinn, weil damit eben auch in allen Arbeitsschritten die Autorenschaft bezeichnet wird. Sollte man später seine Ergebnisse mit denen anderer Kollegen von anderen Rechnern zusammenführen wollen, so sind diese "Stempel" der Autorenschaft eine wichtige Informationsquelle.

Sobald nun eine **neue** Hermeneutic Unit angelegt wird, wird sie automatisch als "**Eigentum**" des zur diesem Zeitpunkt **eingeloggtten Benutzers** verwaltet. Diese Person kann dann auch darüber bestimmen, ob und welche weiteren der in der Benutzer-Verwaltung verzeichneten Benutzer welche Rechte in Bezug auf die Hermeneutic Unit bekommen sollen. Zur Wahl stehen: keine, Nur-lese-Status und Lese-und-Schreib-Status. Damit kann man sehr genau regulieren, wer wie viel Unfug in den eigenen Daten anrichten darf.

Tipp: Wenn man noch in der Grund-Konfiguration eine erste Hermeneutic Unit erzeugt, sich erst danach selbst einen Benutzer-Namen und Rechte definiert und sich als solche(r) einloggt, dann ist zu beachten, dass man von ATLAS nicht mehr als Eigentümer seiner zuvor angelegten Hermeneutic Unit erkannt wird ("Eigentümer" ist statt dessen "Super"). Das Problem ist schnell gelöst: Entweder man legt eine neue Hermeneutic Unit erst nach dem Einloggen mit dem eigenen *account* an oder – wenn "das Kind schon in den Brunnen gefallen ist" – man loggt sich noch einmal als "Super" ein und registriert ändert bei geöffneter Hermeneutic Unit über das Menü "*Extras/-Change Access Rights*" sich selbst, d.h. den neu angelegten *account* als "*Coauthor*" der Hermeneutic Unit. Dann kann man sich wieder mit diesem neuen *account* einloggen und seine Hermeneutic Unit benutzen. Wichtig: Zum Einloggen braucht man nicht das Programm zu verlassen, sondern kann dies auch über den Menüpunkt "*Extras/-Login*" erledigen.

Hermeneutic-Unit-Merging

Bei der Funktion Hermeneutic-Unit-Merging geht es darum, dass zwei separat erzeugte

und bearbeitete hermeneutische Einheiten zum gleichen Projekt zu einem späteren Zeitpunkt zusammengeführt und homogenisiert werden können. Damit lassen sich z.B. Analyse-Arbeiten, die eine Zeit lang arbeitsteilig im Team durchgeführt wurden, am Ende zusammenführen. So kann man die unterschiedlichen analytischen Ansätze, die theoretischen Konzepte etc. gut vergleichen und mit ein wenig Nachbearbeitung auch integrieren.

Wichtig ist dabei, dass nun nicht etwa nur alles zusammengeworfen wird, sondern dass erstens identifizierbar bleibt, welcher Bearbeiter was gemacht hat und dass zweitens unterschiedliche Strategien des Zusammenführens definiert werden können. Beides wird in diesem Werkzeug von ATLAS/ti geleistet. Die Aufgabe ist nicht trivial, denn es muss z.B. geklärt werden, was veranlasst werden soll, wenn in der einen Hermeneutic Unit etwa der Kode A als Oberbegriff von B konzeptualisiert ist, während in der anderen Hermeneutic Unit ein gleich bezeichneter Kode A als Teil von eines Kodes B konzipiert wird. Es muss also entweder eine Dominanz-Regel definiert oder ein Weg gefunden werden, um beide Strukturen zunächst nebeneinander zu übernehmen, um dann später zu entscheiden, welche der Varianten denn zutreffend ist.

Die Funktion findet sich im Extras-Menü unter dem Punkt "Merge with Hermeneutic Unit". Ruft man sie auf, öffnet sich ein "Wizzard"-Fenster und fordert dazu auf, die Quellen-Hermeneutic-Unit und die Ziel-Hermeneutic-Unit zu benennen bzw. per *browse* auszuwählen. Quelle ("*source*") ist dabei diejenige Hermeneutic Unit, die in die z.Zt. gerade aktive ("*target*") Hermeneutic Unit integriert werden soll. Die Unterscheidung ist wichtig, denn es werden hier nicht zwei gleichrangige Entitäten zu einer neuen zusammengeführt, sondern es wird eine aktive Hermeneutic Unit um eine zweite Hermeneutic Unit ergänzt. Wenn es also darum geht, dass z.B. bei einer bestimmten Objektart Doubletten nicht additiv hinzugefügt, sondern gelöscht werden, dann werden die Doppelgänger in der "*source*"-Hermeneutic Unit gelöscht, während die aus der "*target*"-Hermeneutic Unit komplett übernommen werden. Mit "*Next*" wird diese Auswahl bestätigt. Daraufhin öffnet sich ein Fenster mit einem tabellenförmigen Auswahlmenü, in dem die Details der Zusammenführung, also die Merging-Strategie, festgelegt werden. Dabei ist zu entscheiden, welche Elemente überhaupt übernommen werden sollen und wie mit redundanten Elementen zu verfahren ist: Ein in beiden Hermeneutic Units vorkommender Code beispielsweise kann in dem einen Fall sinnvoll einfach "addiert" werden, während es in einem andern Fall angezeigt sein mag, die beiden identischen Codes in der neuen Hermeneutic Unit in zwei Codes zu differenzieren (z.B. weil der Kode in der eine Hermeneutic Unit für andere Bedeutungen steht als in der anderen Hermeneutic Unit). Besonders kritisch ist der Fall, in dem man sich zwar entschieden hat die beiden Codes als identisch zu behandeln, damit aber in der neue Hermeneutic Unit überlappende Zitationen produziert (weil in der einen alten Hermeneutic Unit z.B. ein ganze Absatz damit belegt wurde, in der anderen aber nur ein Teil des Absatzes).

Das Auswahl-Fenster des Hermeneutic Unit-Wizzards offeriert **Optionen auf zwei Ebenen:**

Zunächst kann mit den vier Items auf der linken Seite **zwischen verschiedenen Strategien gewählt** werden. Diese hängen in der Regel davon ab, um welche Art von Zusammenführung es sich handeln soll: Es können zwei Kopien der gleichen Hermeneutic Unit sein, mit denen nur unterschiedliche Dokument bearbeitet wurden; es kann sich um wiederum identische Kopien handeln, mit denen in der Zwischenzeit verschiedene Projekt-Mitarbeiter an gleichen Dokumenten gearbeitet haben; es können aber auch heterogene Hermeneutic Units miteinander integriert werden, also solche, bei denen zu keinem Zeitpunkt eine Identität bestand. Jede der verschiedenen Strategien ist unten links kurz in einem Satz erläutert. Hat man sich für eine Strategie entschieden, so erscheinen die damit verbundenen Vorgaben in Bezug auf die einzelnen Objektarten auf der rechten Seite in Form von angekreuzten Feldern in einer **Matrix von Objektarten und Verfahrensweisen**. Hier können nachträglich noch Veränderungen, also eine Art

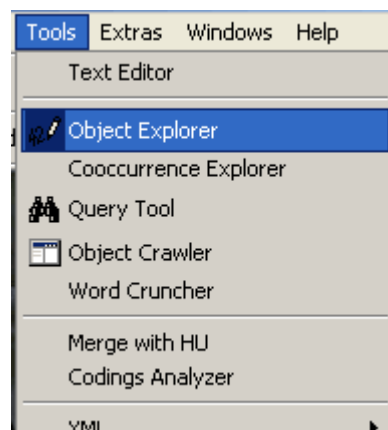
Feinabstimmung der Zusammenführungsstrategie, vorgenommen werden. Diese Optionen sollten unbedingt regelmäßig geprüft werden, denn die vordefinierten Strategien können nur grobe Abstraktionen der typischen Verfahrensweisen sein und decken sich häufig nicht genau mit den spezifischen Gegebenheiten und Anforderungen der Nutzer. Hat man seine Auswahl getroffen, klickt man "*finish*" und der Vorgang wird durchgeführt.

Wichtig: Der Vorgang des Zusammenführens ist eine recht heikle Angelegenheit, bei der man vieles falsch machen kann. Da hier nicht eine neue Hermeneutic Unit erzeugt, sondern eine der beiden Hermeneutic Units um Elemente der anderen erweitert wird, ist der Vorgang für die "*target*"-Hermeneutic Unit unumkehrbar. Deshalb ist **unbedingt** anzuraten, diesen **Vorgang** grundsätzlich **nur mit Kopien der Original-Hermeneutic Units durchzuführen**. Die Originale sollte man erst löschen, wenn sich im praktischen Gebrauch erwiesen hat, dass die Zusammenführung erfolgreich verlaufen ist!

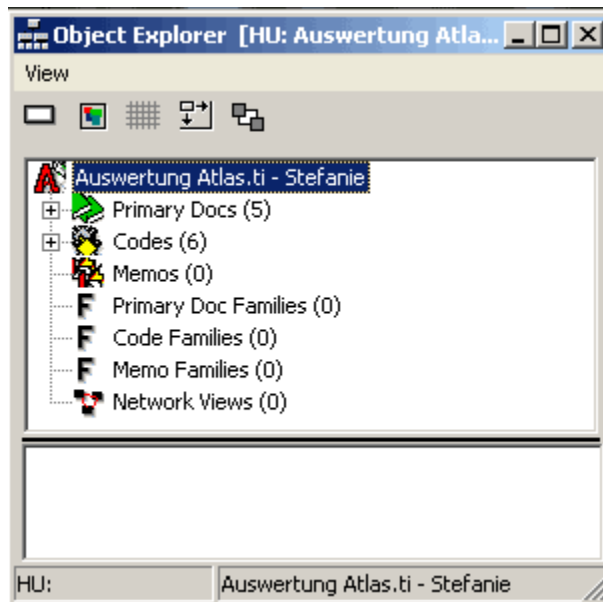
Verschiedenes

Object Explorer

Ein zusätzliches Hilfsmittel, um die Transparenz der Bezüge zwischen den verschiedenen Objekten wieder herzustellen, ist der sogenannte "Explorer". Dieses Werkzeug wird zugänglich, indem auf den entsprechenden Menüpunkt im *Tools*-Menü klickt.



Es erscheint ein neues Fenster mit einer **Baumstruktur**, auf der (fast) alle Objekttypen mit Icon und Bezeichnung einmal auf der obersten Ebene repräsentiert sind: Primärdokumente, Codes, Memos, Primärdokument-Familien, Kode-Familien, Memo-Familien und Netzwerk-Ansichten. Zitate werden auf dieser Ebene nicht repräsentiert, erscheinen aber als Bestandteil der Primärtexte spätestens auf der zweiten Ebene.



Ein **Klick** auf die Plus-Zeichen vor Objekten bzw. ein Doppelklick auf die Objekte selbst **öffnet die nächste Ebene**: z.B. sind unterhalb der Zitate die mit ihnen jeweils verbundenen Zitate angeordnet, darunter die mit diesen Codes verbundenen weiteren Codes oder Memos. Hier kann man sich also schnell einen Überblick über die Zusammenhänge zwischen den verschiedenen Objekten verschaffen. Das Interessante an diesem Werkzeug ist, dass man über die objektspezifischen Kontextmenüs jederzeit die wichtigsten Operationen mit den Objekten durchführen kann, z.B. Umbenennen, Editieren, fokussierte Netzwerke öffnen oder die Zuordnung zu Familien verändern.

Der untere Teil des *Object Explorers* besteht aus einem mit einer verschiebbaren roten Trennleiste abgetrennten Editor-Fenster, in dem man z.B. das jeweils angewählte Memo editieren oder den oben gerade aktivierten Code kommentieren kann.

Copy Bundle

Falls man an mehr als einem Rechner an verschiedenen Arbeitsplätzen arbeitet, ist es gut zu wissen, dass ATLAS/ti eine Funktion bereithält, mit der man a) alle Dateien einer jeweiligen hermeneutischen Einheit in ein Verzeichnis (z.B. eine Diskette) kopieren und sie b) über ein "*Batchfile*" (eine Stapelverarbeitungsdatei) auf einem anderen Rechner wieder in die originären Verzeichnisse überspielen kann.

Die Funktion findet sich im *Tools*-Menü. Wenn man sie aufruft, wird man lediglich aufgefordert, ein Verzeichnis anzugeben, in dem die Dateien gespeichert werden sollen. Nachdem das erfolgt ist, wird eine Liste der Dateien angezeigt, die das Programm nun zu kopieren beabsichtigt.

Dabei wird auch ein *batch file*, also ein kleines Programm produziert, das in der Lage ist, die Dateien auf jedem beliebige gleich bezeichneten Datenträger wieder in die Original-Verzeichnisse zurück zu speichern.