

---

# The TETRIS Terminology Tool

**Michael Carl<sup>1</sup> — Johann Haller<sup>2</sup> — Christoph Horschmann<sup>2</sup> —  
Dieter Maas<sup>2</sup> — Jörg Schütz<sup>2</sup>**

<sup>1</sup> *Laboratoire de Recherche Appliquée en Linguistique Informatique  
Département d'informatique et de recherche opérationnelle  
Université de Montréal, Québec, Canada  
carl@iro.umontreal.ca*

<sup>2</sup> *Institut für Angewandte Informationsforschung,  
Martin-Luther-Straße 14,  
66111 Saarbrücken, Germany  
{hans ;chris ;dieter ;joerg}@iai.uni-sb.de*

---

*RÉSUMÉ. Nous présentons le projet TETRIS dédié à la spécification, à la production et à la maintenance de documents multilingues. Ce projet est réalisé en partenariat avec l'IAI (Institut de Recherche Appliquée à l'Information) et plusieurs compagnies allemandes. Nous décrivons plus particulièrement un outil capable de maintenir une base terminologique et de détecter des variantes de termes dans des textes. Nous montrons qu'une approche basée sur les exemples complète favorablement l'approche à base de règles développée jusqu'alors. Cet outil s'adresse particulièrement aux terminologues et aux auteurs techniques.*

*ABSTRACT. The TETRIS project at the Institut für Angewandte Informationsforschung is a MULTILINT follow-up project in collaboration with BMW and a number of smaller German companies. TETRIS provides linguistically intelligent components for specification, production and maintenance of multilingual documents supporting authors for technical documentation and manuals as well as terminologists. The paper presents the TETRIS terminology tool. TETRIS has a prescriptive terminology tool which detects variants of terms in documents and in the lists of terms. The paper discusses the technical realization and implementation in detail. We show that an example-based approach fruitfully complements a rule-based approach for the detection of terminological variants.*

*MOTS-CLÉS : Terminologie prescriptive et contrôlée, reconnaissance automatique des variants de termes*

*KEYWORDS: Prescriptive Terminology Management, Terminology Control, Automatic Recognition of Term Variants*

---

## 1. Introduction

Globalized industries have to produce and maintain technical documentation in many languages, adapt it for different environments and respond quickly to changing market conditions and customer expectations. With this background, the TETRIS project [TET99] provides linguistically intelligent components for specification, production and maintenance of multilingual documents supporting authors for technical documentation and manuals as well as terminologists. In the current state of the project, TETRIS integrates tools for orthographic, grammar, consistency, style, terminology and acronym checking, which are integrated into the company's internal document flow. The TETRIS application field is, thus, a linguistically intelligent Controlled Language scenario. TETRIS is a MULTILINT follow-up project [HAL 96, Mul95, cf. also [HAL 00]]. While MULTILINT was a pilot project in collaboration with a big German company (BMW), TETRIS now involves also smaller companies.

The rapid growth of technical documentation on repair, maintenance and service literature in big companies requires linguistically intelligent tools for checking their form, content and consistency. Consistent and coherent texts are easier to understand for humans but also facilitate and optimize machine translation of these documents into increasingly many target languages. Terminology tools provide an environment for simplified and controlled vocabulary with well-defined meanings, a thesaurus of frequent terms, their synonyms and so-called negative terms which are to be avoided.

In the past years, big companies have recognized the importance for creating and maintaining corporate terminology, also with the aim to achieve a distinguished corporate identity. Terminology tools which are integrated into commercial translation memories such as Trados MultiTerm or Star's TermStar are sometimes used to support the creation and maintenance of the terminology. Even though these tools offer easy possibilities to creating and extending the terminology, they provide little help in detecting variants or synonyms of terms in documents.

Since terminology is added to the databases as it occurs in the documents, such an approach could be described as *descriptive*. In fact, as these tools work in the context of translation memories, full-forms of terms and variants are memorized together with their translations without keeping track which form is to be preferred and which forms are legal or prohibited variants.

This approach to conventional terminology management appears justified from a user's point of view. The main objective of these tools is the retrieval of as many forms as possible from a source language document in order to have appropriate translations at hand during the translation process. The descriptive approach in terminology management and maintenance has emerged from an uncontrolled authoring and translation environment and is rooted in the way current translation memory technology works. Many translators (i.e non-specialists) need to deal with texts of which they have no prior knowledge. Such translators need to practice ad hoc terminology management for concepts which they are unfamiliar with. As Wright [WRI 97] :148 points out this can lead to disastrous translations.

In a controlled language scenario — such as in TETRIS — this approach is inappropriate and misplaced. TETRIS follows a conceptual approach where each concept is represented by one and only one authorized full form<sup>1</sup>. Any entry contained in the TETRIS terminology is assumed to be an authorized form. Based on these authorized forms, the TETRIS terminology tool (henceforth TTTT) automatically detects a number of variants and notifies authors and/or terminologists of this findings. Therefore, the TETRIS approach can be described as *prescriptive* as its purpose is to standardize and normalize the use of terms according to the authorized terminology (cf. [WRI 97] :329). As every terminology, even a standardized one, undergoes modifications and extensions, TTTT also provides an environment to dynamically elaborate, update and modify a prescriptive terminology.

In this paper, we first outline the aims of TTTT, we give examples of variants detected in TTTT, show how this is helpful in elaborating a prescriptive terminology and discuss alternative approaches. In section 3, we describe processing strategies of TTTT. We show how TTTT integrates a rule-based and an example-based component. We argue that for certain tasks the example-based approach is computationally cheaper while for other tasks the rule-based component is more suitable. While the rule-based approach provides a means to easily model fine-grained linguistic knowledge, the example-based component is used to restrict the search space. Section 4 describes in detail the software components of TTTT : a rule-based partial parser KURD and an example-based partial parser EDGAR. Both systems have different advantages but yield equivalent representations to complement each other. Section 5 describes in detail how variants of terms are represented and detected in TTTT. We show what kind of information is needed in the rule-based and in the example-based component and how both components interact. In particular we show that using abstracted examples increases coverage at a low computational cost while the rule-based component ensures precision.

## 2. Aims of TTTT

A tool for checking terms and their variants in documents is at best as good as the terminology it uses. Ideally, a terminology contains one term for each concept in the domain and each relevant concept is represented by one authorized term. In addition, a terminology may contain definitions and/or collocations of terms, a thesaurus, synonymous writings or negative forms of the terms which are to be avoided.

As terminologies grow it becomes increasingly difficult to ensure consistency of terms and a 1-to-1 mapping of linguistic forms and concepts. Conventional terminology tools offer little help in this respect, as they contain essentially collections of forms with little automatic support for detecting inconsistencies, variants and non-obvious redundancies.

---

1. A possible relation between full forms and other occurrences like abbreviations, acronyms etc. is only partially implemented in the TETRIS authoring environment.

The aim of TTTT is to overcome this shortcoming by detecting variants in lists of terms and documents. TTTT currently recognizes a number of variants where at least one component differs from the authorized form. TTTT recognizes :

- writing variants : typographical variants such as use of upper/lower case letters or hyphenation ;
- derivational variants : use of different word classes such as adjectives, adverbs or nouns ;
- permutation variants : different syntactic order of extensions and head words ;
- variation by omission : deletion of components in a complex term ;
- synonyms : use of a different but semantically related word.

In this section we give examples of variants detected in a list of 16,039 BMW terms ; the same variants could also have been detected in documents. In this list, 1896 clusters of terms were found containing two or more forms. 1309 of these clusters contain synonyms, writing and derivational variants and 587 clusters contain syntactic variants, permutations and omissions. Ideally, each cluster should contain only one authorized form. By generating these clusters, TTTT suggests the terminologist to revise the entries. In case a cluster contains two (or more) forms which the terminologist actually considers not to be variants of each other, an appropriate flag can be set. Both forms are henceforth considered as distinctive forms denoting different concepts. In this way, TTTT helps a terminologist to conceptually stabilize a terminology, avoid “changes in aspects of his science” (cf. [POL 95]) and establish conventions in the domain.

### 2.1. *Writing and derivational Variants*

In the table below, the variants in cluster 1 differ in the use of lower case letters and upper case letters. Cluster 2 is writing variants where both forms on the left and right side differ with respect to a hyphen and an upper case letter. Cluster 3 differs in the use of upper case/lower case letters and the adjective *halbharter* (semi-rigid) left while the adverb derivation *Halbhart* is used on other variant<sup>2</sup>.

1	<i>integrierte Universal-Fernbedienung</i> (integrated universal remote control)	<i>Integrierte Universal-Fernbedienung</i> (Integrated universal remote control)
2	<i>Getriebeseriennummer</i> (driving-gear serial number)	<i>Getriebe-Seriennummer</i> (driving-gear serial number)
3	<i>halbharter Schaumstoff</i> (semi-rigid foam)	<i>Halbhart-Schaumstoff</i> (semi-rigid foam)

2. In this paper, German terms will be written in *italics* while their English translations will be given (sans serif).

## 2.2. Synonyms

The TETRIS term tool detects synonyms if the term is made up of two or more lexemes. Experience has shown that admitting synonyms for single lexeme terms generates too much noise. Similar to Hamond and Nazarenko [HAM 01, cf. section 2.5], TTTT allows synonyms in head and expansion of the compound. Variants in cluster 4 differ in their expansion *lagern* (to stock), *halten* (to hold), *befestigen* (to fix), *aufhängen* (to attach) while the head *gummi* (rubber) is identical in all variants. The variants in cluster 5 differ in their head-noun *Einrichtung* (set up, installation, equipment, facility) vs. *Anlage* (plant, system, unit, establishment, installation, etc.) while the expansion *Abgaskontrolle* (exhaust gas control) is identical. Cluster 6 contains variants which differ in both, the head *prüfen*, *messen*, *geben*, *anzeigen* (to check; to test, to measure, to give, to display) and the expansion *Vorrichtung*, *Einrichtung*, *Anlage*, *Einheit* (apparatus, device, unit, machine, installation, fixture, etc.).

4	<i>Lagergummi</i> (bearing rubber) <i>Aufhängegummi</i> (suspension rubber)	<i>Haltegummi</i> (retaining rubber)	<i>Befestigungsgummi</i> (mount rubber)
5	<i>Abgaskontrolleinrichtung</i> (exhaust control device)	<i>Abgaskontrollanlage</i> (exhaust emission control system)	
6	<i>Prüfvorrichtung</i> (checking device) <i>Gebereinheit</i> (input unit)	<i>Prüfeinrichtung</i> (testing equipment) <i>Anzeigeneinheit</i> (display unit)	<i>Messanlage</i> (measuring machine)

## 2.3. Permutation Variation

Variation by permutation is a kind of syntactic variation. Permutation variation may occur in two directions : a noun phrase may be generated from a compound noun or a compound noun may be generated from a noun phrase. In the examples below, a compound noun is shown on the right side while its paraphrased noun phrase is on the left side. In the cluster 7, the head nouns *Minuspol* (negative pole) is percolated from its end position in the compound noun to the front position in the syntactic construction. In cluster 8 the expansion *Güte* (goodness, quality, rating) on the left side is in the same time replaced by a synonym *Sicherung* (stabilization, fuse, safety, security, assurance).

7	<i>Minuspol der Batterie</i> (negative pole of battery)	<i>Batterie-Minuspol</i> (battery negative pole)
8	<i>Sicherung der Güte</i> (assurance of quality)	<i>Qualitätssicherung</i> (quality control)

#### 2.4. Variation by Omission

The term tool also checks variants by omission for entries which have three or more lexemes. Cluster 9 has a three lexeme term and a two lexeme term where the 2 lexeme term is missing the middle component *Licht* (light). In addition the lexemes in the reduced variant are separated by a hyphen. Cluster 10 consists of a 4 lexeme term and a 3 lexeme term where the latter one is missing the component *Reihe* (row,series,line). Cluster 11 contains three variants. The head noun *Untersuchung* (investigation, examination, check-up) is replaced by the synonym *Kontrolle* (control,inspection,monitoring) and in cluster 12 both omission and synonym substitution occurs in parallel.

9	<i>Abblendlichtrelais</i> (low-beam relay)	<i>Abblend-Relais</i> (dimmer relay)	
10	<i>4-Zylinder-Reihenmotor</i> (4-cylinder in line engine)	<i>4-Zylinder-Motor</i> (4-cylinder engine)	
11	<i>Abgassonderuntersuchung</i> (special exhaust inspection)	<i>Abgasuntersuchung</i> (exhaust inspection)	<i>Abgaskontrolle</i> (exhaust gas control)
12	<i>Luft-Saugventil</i> (air intake valve)	<i>Belüftungsdüse</i> (ventilation nozzle)	

#### 2.5. Previous Research on Terminology Variation

A number of researchers have investigated terminological variation for different languages and with different means and goals. Royouté [ROY 96, ROY 99] and Jacquemin [JAC 96, JAC 01] investigate a number of term variations for French :

– (variation by) inflection

this subsumes derivational variation, such as *acoustic test / acoustic testing* and variation by number, such as *deficiency / deficiencies*.

– (variation by) insertion

the inserted element modifies the head of the term as e.g. in *thin film vs. thin gold film* or the inserted element becomes the head of the term as in *quantum well structure vs. quantum structure*.

– (variation by) permutation

this type of transformation concerns noun-noun structures such as *electron diffraction* which appears as a permuted form in *diffraction of fast electrons*

– (variation by) coordination

adjectives (and nouns) can be coordinated as in *electron diffraction vs. electron and photoelectron diffraction*

Jacquemin finds that “clustering of terms related through coordinations yields classes of conceptually close terms while graphs resulting from insertion denote generic/specific relations.” [JAC 96, 425]

The research of Royauté is based on the notion of variation and stability of terminological noun phrases. The absence of variation can be interpreted as a sign of the conceptual stabilization of the term and provides a measure for “changes in aspects of sciences” (cf. [POL 95]) :

While the stability of a term is a sign that the notion which it represents is becoming ordinary, its variation, on the contrary, often reveals a conceptual instability of the notion, underscoring the activity of an emerging or growing sector.

In another piece of work, Hamond and Nazarenko [HAM 01, HAM 98] seek to detect synonyms in order to help structuring of terminologies. The authors use three rules to detect synonymy relations between terms. An identical mechanism is also implemented in TETRIS. Two compound candidate terms are synonyms if :

- the heads are identical and the expansions are synonymous ;
- the heads are synonymous and the expansions are identical ;
- the heads are synonymous and the expansions are synonymous ;

Through interaction with a terminologist, the authors find that recall is a more useful property than precision : errors (or ‘noise’) that come along with higher recall are suggestive and may be interesting for terminologists.

### 3. Functions and Strategies of TTTT

TTTT works—in contrast to commercial terminology tools—on lexemes and lemmas and represents morpho—syntactic information of terms and words. While lexemes are the abstracted skeleton which is shared between the authorized form of a term and its variant, differences in features such as type of derivation, typographical information or the lemma determines the type of the variant.

The heart of TTTT consists of three tools. The first tool, TermLint, is designed for a terminologist to evaluate the entries in a list of terms. TermLint checks the consistency and uniqueness of these terms and returns a list of clusters to the terminologist. The functional components of this tool are listed in the upper part in figure 1. Via a graphical interface, the terminologist can sort, link, modify or annotate these terms.

Another tool is designed for authors. This tool checks documents for the consistent use of terms. The author’s tool recognizes variants of terms in texts, highlights them in a document production environment and triggers a message showing the authorized form of the term. The functional components of this tool are listed in the lower part in figure 1. Via a mailing system, authors can communicate with the terminologist in order to dynamically update the terminology.

The third tool generates from a terminology a set of variants to be recognized in the author’s and the terminologist’s tools. The components of this tool are listed in the

middle part in figure 1. This database of variants contains the variants of the terms and is at the very core of TTTT.

TTTT presupposes an initial list of terms. These terms might have been proposed by authors or engineers or they might have been automatically extracted from existing corpora or during the authoring process. TETRIS comprises a term mining tool [HON 01]. Similar to the approach of Jacquemin [JAC 96, JAC 01], this tool follows the premiss of “updating rather than acquiring” and builds upon existing terminology. Word forms which are neither authorized terms nor unauthorized variants and which comply with certain term formation criteria are detected in the background and are a possible source for terminology extension.

For a recent overview of other tools for term mining and automatic term extraction see [CAB 01, JAC 01]. The question of where and under what conditions terms arise and from which point in time they have the status of a term, authorized or not and how they are defined, will not be elaborated here. A number of international standards (e.g. ISO 704, ISO 1087, ISO 10241) have been designed to aid standardizers in meeting this need cf. [WRI 97]. Answering this question would require two considerations : What concepts are to be denominated and which authorized forms should be chosen. In the remainder of this section we look, instead, closer to the three core tools of TTTT as plotted in figure 1 and describe their basic functions.

### **3.1. Architecture of TTTT**

TermLint (in the upper part in figure 1) checks the consistency and uniqueness of terms. TermLint is designed for and used by terminologists who decide in collaboration with authors which terms are to be included into the company’s term pool, which are their authorized forms and which are prohibited variants. TermLint contains modules for orthographic, grammatical and term formation checking, which will not be elaborated here.

The lower part in figure 1 represents the author’s tool which is used ‘online’ when checking a document for consistent use of terms. It provides authors with useful hints for using terms in their authorized forms.

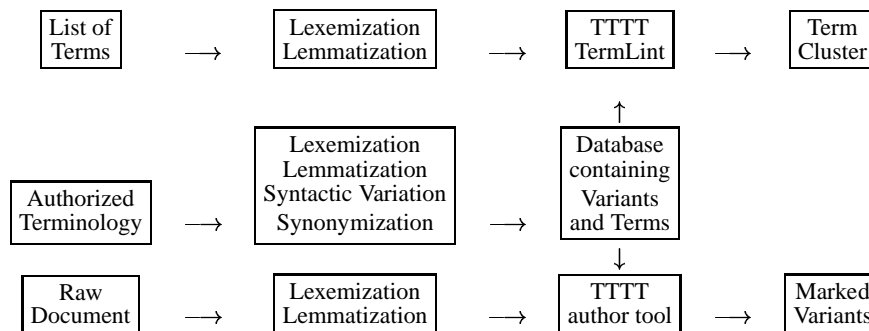
Both tools first perform morphological analysis, lexemization and lemmatization of the linguistic expressions in a document (or a list of terms). The stream of linguistically enriched tokens is then matched against a database containing variants of authorized terms. In case a sequence of words in the document matches an entry in the database, an appropriate marker is set pointing to the authorized form.

In the case of TermLint (upper stream), it is checked whether a term in the list of terms is possibly a variant of a term in the terminology. TermLint produces term clusters containing two or more entries which it classifies as variants of each other as shown in section 2. These term clusters are a valuable help for the terminologist to revise the entries in the terminology and provide a means of deciding the consistency

of a terminology. In case TermLint clusters two entries which actually represent two different concepts, an appropriate authorization flag can be set such that the entries will henceforth be considered as different terms.

In the author's tool, detected variants of terms are marked in a document production environment where a message shows the authorized form of the term together with a pre-defined comment

TTTT integrates a rule-based approach and an example-based approach. In its current implementation, the rule-based approach is used to generate variants from authorized terms which are stored in a database. Rules are also used to consolidate the findings of the matching process. A previous implementation of TTTT applied the rule-based component at run-time to generate variants of terms in the text.



**Figure 1.** Main components of TTTT. up : Checking Consistency of a List of Terms with TermLint. middle : Compiling a Terminology into a database of variants. down : Checking Consistency of the Terms in Documents

### 3.2. A Database Approach

There are basically two ways of matching a candidate variant in a document (or in a list of terms) onto the list of authorized terms ; a database approach and a run-time approach :

1) in the run-time approach, a candidate sequence of words in the document undergoes a number of transformations which map it onto an authorized term. The original sequence in the document is then marked as a variant of the authorized retrieved term.

2) in the database approach a limited number of possible variants are generated for each authorized term. The variants are stored in a database with a link to their authorized terms. A matched sequence of words in the document is marked as a variant of the term from which the database entry was generated.

The run-time solution requires a small database of authorized terms while, in a previous implementation of our system, the time to transform and map a candidate sequence of words in a document onto its authorized form increases linearly in time with every possible transformation that this sequence undergoes. Jacquemin's *FASTR* [JAC 01] implements the run-time approach. Using a set of metarules, Jacquemin remains below this linear limit.

TTTT implements the database approach. The database approach has the drawback that all possible variants which the tool is supposed to recognize are generated and stored in a database. However, storing underspecified variants, increases the size of the base only marginally compared to the gain of coverage. The outstanding advantage of the database approach is, however, *log*-time retrieval.

The terminology for one of the TETRIS-partners (BMW) contains 16,039 entries at the time of writing this article. From this terminology, more than 72,000 synonymous and syntactic variants were generated<sup>3</sup> such that roughly 88,000 entries are stored in the database. Note that these 88,000 entries represent abstracted forms, i.e. sequences of lexemes. As each entry represents a number of surface forms, on average probably roughly in the order of 10, more than 700,000 surface form variants are recognized from the 16,039 authorized terms !

Entries are indexed and matched based on the values of the features they encode. As term variation adds redundancies to the entries, the number of different indexes grows more slowly than the number of entries. Thus, 8563 indexes were required to store and retrieve the 16,039 terms, while 11,007 indexes were generated for all 88,000 variants. Theoretically, the database approach needs at most  $m^2 * \log(i) * f$  operations where  $m$  is the length of the sentence,  $i$  is the number of indexes in an EDGAR database and  $f$  is the number of features matched. Practically, retrieving a variant of a term in the bigger database increases search time by a factor of approximately 1.3 compared to retrieving a term in the list of 16,039 authorized terms while coverage increases by a factor of about 5.

### 3.3. A Rule-Based Approach

TTTT integrates a rule-based approach and an example-based approach. In its current implementation, the rule-based approach is used to generate variants of authorized terms which are stored in a database. A previous implementation of TTTT [SCH 98] followed a run-time approach, where the rule-based component was applied at run-time to guess the authorized forms of variants in the text. These guessed forms were then validated in the database. This approach is computationally much more expensive because many forms were produced which actually don't match an entry in the database. However, the number of required operations increases linearly with the number of variants to be recognized.

---

3. Here we do not count the writing and derivation variants which would have to be added to this number.



**Figure 2.** *The run-time Approach to recognizing Terms*

The components of the previous implementation are shown in figure 2. In the run-time approach as documented in [SCH 98], only synonymous variants were generated since computation of syntactic variants is computationally too expensive. For instance, to figure out whether the sequence *Sicherung der Güte* (assurance of quality) is a variant of the term *Qualitätssicherung* (quality control) as in cluster 8 in section 2, a number of transformations in the sentence would have to be executed where each transformation may or may not map onto an authorized form. However, it is likely that only very few generated forms actually match an authorized term. As processing would become too expensive, syntactic variants were not covered in the previous implementation of TTTT.

#### 4. Software Components of the TETRIS Terminology Tool

In TTTT—and in MULTILINT—lists of terms, documents and texts are morphologically analyzed, lemmatized and lexemized by means of MPRO (cf. [MAA 95, MAA 96]). The output of MPRO is a sequence of sets of feature bundles which encodes a number of linguistic properties of the words. Variants of terms are automatically generated based on the term’s morpho-syntactic properties and the sequences of lemmas and lexemes. The variants are generated by means of a partial rule-based parser KURD [CAR 97]. Terms and their generated variants are stored in an example-based machine-translation system EDGAR [CAR 99] which is applied in TTTT for recognizing terms and their variants in texts and documents.

EDGAR maps terms and their variants onto the morphologically analyzed text and generates for each matched variant a partial, shallow derivation tree. While the mother node of this derivation tree contains head information of the authorized term, the leaves describe the variant as it occurred in the document. This structure is passed to KURD to compare the head information of the authorized term in the mother node and the information contained in the daughters. Where there are differences, the occurrences are marked with an appropriate status flag indicating the type of variant together with its correct, authorized form.

In this section, we describe a recent version of KURD and EDGAR. Contrasted with those described in [CAR 97] and [CAR 99], the recent versions make use of more complex data structures and are enhanced with a number of further operators. We show that KURD and EDGAR produce equivalent data structures which can be mutually refined and consolidated. TTTT takes advantage of these properties.

**OBJ** :: **FB** ( ‘, ‘ **FB** ) \* ‘ . ‘  
**FB** :: **AVM** ( ‘; ‘ **AVM** ) \*  
**AVM** :: ‘{ ‘ ( **Att op VAL** ( ‘, ‘ **Att op VAL** ) \* ) ‘ } ‘  
**Att** :: alpha numerical string of length > 0 and < 80  
**op** :: ‘=‘ | ‘~=‘  
**VAL** :: **FB** | **VAR** | atom<sup>x</sup>  
**VAR** :: ‘\_‘ | ‘\_‘ followed by a string of upper-case letters or numbers

Brackets ‘(‘ and ‘)’ signify optionality and grouping ; the asterisk ‘\*‘ zero or more elements.

<sup>x</sup> atoms must not start with an underscore ‘\_‘ and may not contain a coma ‘,’ a semicolon ‘;‘ or a curly bracket ‘{‘ or ‘}‘. Blanks, tabs and newlines are discarded from the atom.

**Figure 3.** *Representation in TETRIS*

#### 4.1. *Representation in TETRIS*

Representation in TETRIS is largely influenced by EUROTRA, a European project on machine translation in the time span 1982 - 1993. The basic data structure is a linguistic object **OBJ**, typically a sentence. As shown in figure 3, an **OBJ** consists of a sequence of feature bundles, **FB**s and is terminated by a final dot. Each **FB** is separated by a comma. A **FB** is a disjunction of matrices, **AVM**s, which are enclosed in curly brackets ‘{‘ and ‘}‘ and which are separated by a semicolon. An **AVM**, in turn, is a conjunction of attribute-values, **Att op VAL**, where each **Att** may occur only once. The value of an attribute may be complex i.e. it may be a **FB**, it may be atomic i.e. a string, or it may be a variable, **VAR**. In EDGAR, variables may only be universal variables i.e. ‘\_‘, while KURD-rules allow for any kind of variables (see below).

The morphological analyzer, lexemizer and lemmatizer MPRO yields the analysis of the German phrase *die weichen Lagergummis* (the soft bearing rubber) which is represented as **OBJ**<sub>1</sub> as shown in figure 4.

The features *c* and *sc* encode the part-of-speech (POS) and a sub-category, *s* provides semantic information, *lu* encodes the lemma and *ls* the lexeme(s) of the word. The feature *ehead* encodes the nominal agreement information (i.e. number *nb*, case *case*, gender *g*). Note that *Gummi* (rubber) in *Lagergummi* can be masculine or neuter which is taken account by the feature *nb=m*; *n*. There are further features which depend on the POS as for instance *deg* for adjectives (*c=adj*) and *vtyp* for verbs (*c=verb*) or which depend on the language *as*, for instance, information on inflection and derivation.

The German word *die* has two readings, as an article *sc=art* or as a relative pronoun *sc=rel*. The word *weichen* has five different readings : as a noun (points ;switch), an adjective (soft ;tender) a finite and an infinite verb (leave,soak), or their nominalizations. While most of the compound's *Lagergummi* morpho-syntactic features depend on the head *gummi* some features also provide information on the expansion. Thus,

```

{lu=d_art,c=w,sc=art,ehead={nb=sg,case=acc;nom,g=f,infl=weak};
                        {nb=plu,case=acc;nom,infl=weak}}};
{lu=d_rel,c=w,sc=rel,ehead={nb=sg,case=acc;nom,g=f};
                        {nb=plu,case=acc;nom}},
{lu=weiche,ls=weiche,c=noun,ehead={nb=plu,g=f}}};
{lu=weich,ls=weich,c=adj,deg=base,
 ehead={nb=sg,infl=weak;strong,case=dat;gen,g=f;m;n};
        {nb=sg,infl=weak;strong,null,case=acc,g=m};
        {nb=sg,infl=null,case=gen,g=m;n};
        {nb=plu,infl=null,case=dat,g=f;m;n};
        {nb=plu,infl=weak;strong,case=acc;dat;gen;nom,g=f;m;n}}};
{lu=weichen,ls=weichen,c=verb,vtyp=inf};
{lu=weichen,ls=weichen,c=verb,vtyp=fiv,per=1;3,tns=pres};
{lu=weichen,ls=weichen,c=noun,ehead={nb=sg,g=n}},
{lu=lagergummi,ls=lager#gummi,c=noun,
 ehead={nb=plu,case=acc;dat;nom,g=m;n},ss=loc&ag#mat}.

```

**Figure 4.** *The MPRO object OBJ<sub>1</sub> for die weichen Lagergummis (the soft bearing rubber)*

```

[1 -> 2 conf:0.0000000 tree:NULL
  {lu=d_art,c=w,sc=art,ehead={nb=sg,case=acc;nom,g=f,infl=weak};
                        {nb=plu,case=acc;nom,infl=weak}}};
  {lu=d_rel,c=w,sc=rel,ehead={nb=sg,case=acc;nom,g=f};
                        {nb=plu,case=acc;nom}},
[2 -> 3 conf:0.0000000 tree:NULL
  {lu=weiche,ls=weiche,c=noun,ehead={nb=plu,g=f}}};
  {lu=weich,ls=weich,c=adj,deg=base,
   ehead={nb=sg,infl=weak;strong,case=dat;gen,g=f;m;n};
         {nb=sg,infl=weak;strong,null,case=acc,g=m};
         {nb=sg,infl=null,case=gen,g=m;n};
         {nb=plu,infl=null,case=dat,g=f;m;n};
         {nb=plu,infl=weak;strong,case=acc;dat;gen;nom,g=f;m;n}}};
  {lu=weichen,ls=weichen,c=verb,vtyp=inf};
  {lu=weichen,ls=weichen,c=verb,vtyp=fiv,per=1;3,tns=pres};
  {lu=weichen,ls=weichen,c=noun,ehead={nb=sg,g=n}}
[3 -> 4 conf:0.0000000 tree:NULL
  {lu=lagergummi,ls=lager,c=noun,
   ehead={nb=plu,case=acc;dat;nom,g=m;n},ss=loc&ag}
[4 -> 5 conf:0.0000000 tree:NULL
  {lu=lagergummi,ls=gummi,c=noun,
   ehead={nb=plu,case=acc;dat;nom,g=m;n},ss=mat}
]]]].

```

**Figure 5.** *The OBJ<sub>1</sub> converted into GRAPH<sub>1</sub>*

the feature *ls* enumerates the lexemes in compounds separated by a hash # as in *ls=lager#gummi*. The feature *ss* enumerates the semantic properties of the lexemes.

While the morphological analyzer MPRO yields objects which encode the morpho-syntactic properties of a sequence of words (e.g. a sentence), EDGAR and KURD convert an object into a non-cyclic **GRAPH**. A **GRAPH** is a set of connected nodes, each of which consists of a **HEADER**, a **FB** an inner recursion i.e. a **GRAPH** inside square brackets, and an outer recursion i.e. a **GRAPH** outside square brackets. Each **HEADER** has a label (i.e. an identification number), a pointer to the label of the successor **GRAPH** a confidence value and a tree value. The inner recursion represents a list of nodes, i.e. a path through the **GRAPH**, while the outer recursion(s) represent(s) (an) alternative path through the **GRAPH**.

```
GRAPH    ::  '[' HEADER FB (GRAPH)*  ']' (GRAPH)*
HEADER  ::  LABEL CONF TREE
LABEL   ::  number '->' number
```

When converting an MPRO object into a **GRAPH**, the lexemes of compound words can be decomposed in order to be better accessible by rules. The **OBJ**<sub>1</sub> has the representation as a **GRAPH**<sub>1</sub> shown in figure 5. The compound noun *Lagergummi* in **GRAPH**<sub>1</sub> is divided into the two nodes 3 and 4. The confidence and tree information in the **HEADER** is used for statistical purpose, which is not treated in this article.

#### 4.2. Rule-Based partial parsing with KURD

KURD modifies a **GRAPH** according to a list of rules which is defined in one or more rule-files. A KURD rule works like a finite-state automaton and describes a pattern in the **GRAPH**. A pattern consists of a number of sequences of node(s). Single nodes in this pattern may be modified added or deleted. KURD works in a non-monotonous way, it does not back-track or generate multiple solutions. Rules apply in the order given in the rules file; changing this order may also change the output of KURD. However, a number of powerful KURD operators are implemented such that also slightly context sensitive patterns can be recognized and expressed in a generalized way.

Each rule in a rule-files consists of a name a description part and an action part. The description part, is mapped onto the **GRAPH** in order to find a longest matching pattern. A description consists of one or more conditions each of which describes a sequence in the pattern. In case the description matches a pattern in the **GRAPH**, marked nodes are modified by the action part. The description is applied either from left to right or from right to left, dependent on a direction flag.

When matching the description on the **GRAPH**, one or more nodes can be marked in the pattern. These marked nodes are modified in the action by means of a number

of KURD-operators. The following example illustrates how a KURD rule works. A more detailed description of the operators is given below.

#### 4.2.1. A simple KURD Rule

The KURD rule `noun_phrase` describes the pattern of a simple noun phrase. This pattern consists of a sequence of two obligatory segments an article, `{c=w,sc=art}` and one to up to three noun(s) `{c=noun}`. In addition, the pattern may contain a sequence of zero or more adjectives `{c=adj}` and an optional node following the noun phrase. Each of the first two segments is described by means of a condition in the rule and the noun sequence is described by three conditions. The sequences must also have compatible agreement features, i.e. the values of their `ehead` features must be unifiable.

```
noun_phrase =
  Aa{c=w,sc=art,ehead=_E},
  *Be{c=adj,ehead=_E},
  Ce{ls=_L1,c=noun,ehead=_E},
  ^Ce{ls=_L2,c=noun,ehead=_E},
  ^Ce{ls=_L3,c=noun,ehead=_E},
  ^De{}
: Ao{c=dp}c{ls=$_LU1+_LU2+_LU3}->D,
  Au{sc=art,ehead=_E},
  Bu{c=adj,ehead=_E},
  Cu{c=noun,ehead=_E}.
```

The **FBs** in the segments of the matched pattern are marked by the marker A, B, C and D respectively. These marked **FBs** are modified in the action part as will be shown below. Further, each condition has one of four different **SCOPES**. A **SCOPE** determines the minimal and/or maximal length of the segment in the path :

- + matches a sequence of one or more nodes
- \* matches a sequence of zero or more nodes
- ^ matches an optional node
- matches exactly one node this is the default **SCOPE** ; can be omitted

KURD requires either of two quantifier ‘e’ or ‘a’ in each condition. The meaning of the quantifier ‘e’ is that at least one **AVM** of the **FB** in the pattern is unifiable with the **AVM** of the condition, while the quantifier ‘a’ means that all **AVMs** of the **FB** are unifiable with the **AVM** of the condition.

For instance, given the two readings of the word *die*, the test `a{sc=art}` fails as there is also a reading as a relative pronoun `sc=rel`. In contrast, the test `e{sc=art}` returns true as it is compatible with the article reading `sc=art`.

The **GRAPH** is modified according to the actions in the action part. There are four actions in the action part of the rule. The action `Ao{c=dp}->D` in rule `noun_phrase` generates the mother node `{c=dp}` which spans the sequences of nodes from A to D. The marker A points to the article of the noun phrase where the mother node is

attached. The marker D points to the **FB** following the noun phrase, such that the sequence from A to D spans the whole phrase.

The action  $c\{lu=\$_LU1+_LU2+_LU3\}$  concatenates the instantiations of the variables  $_LU1$ ,  $_LU2$  and  $_LU3$  and replaces the concatenated string in the  $lu$ -feature of the new mother. The unification operator  $u$  unifies the **AVM** of the action with the marked **FB(s)**. Note that actions on A apply to the new mother node if they occur between the  $o$  operator and the target marker D. The second action in the rule,  $Au\{sc=art\}$ , thus, applies to the daughter node with the **FB**  $\{c=w, sc=art\}$  where it discards the relative pronoun reading.

The action  $Bu\{c=adj\}$  unifies  $\{c=adj\}$  with all interpretations of the marked **FBs** thereby disambiguating the **FBs** in the **GRAPH**. After applying the rule `noun_phrase` onto the above **OBJ**, the modified **GRAPH** has the following features :

```
[1 -> 2 conf:0.000000 tree:NULL
  {lu=d_art,ls=d_art,c=w,sc=art,
   ehead={nb=plu,infl=weak,case=acc;nom,g=m;n}}
[2 -> 3 conf:0.000000 tree:NULL
  {lu=weich,c=weich,adj,deg=base,
   ehead={nb=plu,infl=weak,case=acc;nom,g=m;n}}
[3 -> 4 conf:0.000000 tree:NULL
  {lu=lagergummi,ls=lager,c=noun,ss=loc&ag,ehead=
   ehead={nb=plu,infl=weak,case=acc;nom,g=m;n}}
[4 -> 5 conf:0.000000 tree:NULL
  {lu=lagergummi,ls=gummi,c=noun,ss=mat,
   ehead={nb=plu,infl=weak,case=acc;nom,g=m;n}}
]]][1 -> 5 conf:0.000000 tree:NULL
  {c=dp,ls=lagergummi}
].
```

The above **GRAPH** has the following graphical representation :

```
      {c=dp,ls=lagergummi}
     /                   \
    die weichen Lagergummi
```

#### 4.2.2. Operators in Kurd

Marked **FB** are modified or nodes are inserted or killed. **KURD** provides a number of operators, which all apply an **AVM** to marked **FB(s)** in the **GRAPH**. The following operators are currently implemented, some of which will be described more closely in this section :

**k** kills the nodes from the **GRAPH**.

**u** unifies **AVM** with **FB(s)**.

**r** replaces features in **FB(s)** by those specified in **AVM**.

**d** deletes features in **FB(s)** which are specified in **AVM**.

**o** inserts a mother node spanning from **FB** to target **MARK**.

- i inserts a node before **FB**(s).
- a inserts a node behind **FB**(s).
- c implements a couple of concatenation operations, see below.
- g generalizes **FB**(s) with the **AVM**.
- s subtracts the **AVM** from **FB**(s).
- t conditionally modifies **AVMs** and assigns default values.

#### 4.2.3. Operators *u*, *g* and *s*

The operators *u*, *g* and *s* compute the unification, generalization and subtraction of the features in the marked **FB**(s) and the **AVM** in the action. The result of the operation is assigned to the marked nodes. Unification computes the intersection of the sets and is analogous to logical AND. Generalization and subtraction can be expressed by unification. For any **FB** *A*, *B* and *C* we have the following equivalences :

$$\begin{aligned}
 u(A, B) &\equiv u(B, A) \\
 g(A, B) &\equiv \sim u(\sim A, \sim B) \\
 s(A, B) &\equiv u(A, \sim B)
 \end{aligned}$$

Unification of an **AVM** *C* and a disjunction of **AVMs** [*A*; *B*] is equivalent to transforming a conjunctive form into a disjunctive form according to the distributive law while negation of complex features is computed according to DeMorgan's law :

$$\begin{aligned}
 u([A; B], C) &\equiv u(A, C) ; u(B, C) \\
 \sim u(A, B) &\equiv \sim A ; \sim B \\
 \sim [A; B] &\equiv u(\sim A, \sim B)
 \end{aligned}$$

Unification takes also place in the description part of a KURD rule when using variables. Negated variables perform the subtraction operation. Unification is thus the basic operation in KURD<sup>4</sup>.

#### 4.2.4. Operator *c*

The operator *c* performs a number of operations for string concatenation. The values in the *c* operator must start with a dollar sign \$. Some of the values in the *c* operator have special meanings. The + operator is binary, all other operators are unary. With the exception of \$inc and \$dec, all operators may appear in complex expressions.

---

4. A more comprehensive description of the KURD operators is provided at <http://www.iai.uni-sb.de/~carl/kurd.html>.

+ : performs string concatenation.  
 . : denotes the value(s) of the attributes in the marked **FB**.  
 \$A : denotes the number of the current **ACTION**.  
 \$O : denotes the length of current **OBJ**.  
 \$R : denotes the number of the current **RULE**.  
 \$N : denotes the number of the current **OBJ**.  
 \$inc : adds 1 to the value of attribute  
 \$dec : subtracts 1 from the value of attribute

#### 4.2.5. Operators *i*, *a* and *k*

The *i* and *a* operators insert a new node before and after the marked nodes respectively, while the *k* operator kills the marked node(s). By means of these operators it is possible to generate a new word order in an **OBJ**, to insert arbitrary nodes and **FB**(s) or to delete parts in the **GRAPH**.

These operators are used to generate syntactic variants from a term. The rule `permute`, for instance, generates the syntactic variant *Anlage zum Heizen* (Installation for Heating) from the authorized term *Heizanlage* (heating installation) :

```

permute =
  Ae{c=noun} ,
  Be{c=noun,ls=_LS}
  : Ai{ls=_LS,c=noun}i{ls=zu,c=w,sc=p},
  Bk{ }.

```

The rule inverts the word order of the head noun and the expansion and inserts the preposition *zum* (for;to) between both components. Notice that two insertion operations and a kill operation are required. The insertion of multiple nodes before the marker respects to order of insertion. In the action part, first the head *Anlage* (installation) is inserted before the expansion *heizen* (heating) and in a second insertion the preposition is inserted before the expansion.

### 4.3. Example-Based partial parsing with EDGAR

EDGAR is a prototype implementation of an example based MT system as described in [CAR 99]. In the context of TETRIS, EDGAR is used for recognizing terms and their variants. Similar to KURD, EDGAR matches and modifies pattern in a **GRAPH**. However, EDGAR (re)uses entries from a database for this matter while KURD uses a set of rules. Once an entry in an EDGAR database matches a pattern in the **GRAPH**, a mother node is generated. The mother **FB** is enriched with head information of the matched entry. The generated structure is then consolidated by KURD rules.

Entries in an EDGAR database are less complex than KURD-rules. The values to be matched in a pattern must be spelled out in the entry, no marker and scopes are allowed. While EDGAR matches the pattern contained in the database in parallel, KURD follows an iterative approach by applying successively each rule on every node in the **GRAPH**.

EDGAR thus supports a data-based approach while KURD allows for more fine-grained rule-based linguistic modeling. However, we will show that KURD and EDGAR produce equivalent representations. It thus becomes a matter of system design whether to use a more rule-based or a more data-based approach.

This section first discusses how an EDGAR database is compiled from a set of objects **OBJ**. Then it is shown how entries are matched onto a **GRAPH**. The section also shows how negated features can be indexed and retrieved.

#### 4.3.1. *Compiling a Database*

An EDGAR database is compiled from a set of objects **OBJ**. Objects in a database are referred to as entries. Each entry is given a unique EDGAR-internal number. For each entry, a subset of features is indexed. The database consists of four files : a data-file containing the head information of the indexed entry, an index file containing the indexed features, a file which maps the indexed features to the list of the indexing entries and a file which maps the number of an entry to the head information to be retrieved.

Example :

Assume the German words *Heizanlage* (heating installation) and *Hauptstromkabel* (main electric cable) are to be stored in the database. The morphological analysis yields the following **OBJ** :

```
{lu=heizanlage,ls=heizen#anlage,
  c=noun,ss=ation#anlage;money,gra=cap}.
{lu=hauptstromkabel,ls=haupt#strom#kabel,
  c=noun,ss=a#phen#thing,gra=cap}.
```

For each lexeme *heizen* (to heat) and *anlage* (installation), *haupt* (main), *strom* (electric) and *kabel* (cable) a separate **FB** is generated as shown below. The first **FB** carries in addition information related to the head of the compound term in the *mother*-feature. This information is taken from the analyses of the term's component and collected in the *mother* feature using a set of KURD rules. Head information includes — amongst others — the POS of the entry (i.e. *mc=noun*), a graphics feature of the compound *gra=cap*, (i.e. first letter capital), the lemma, *lus=heizanlage* and *lus=hauptstromkabel* and a semantic feature<sup>5</sup>.

```
1 {lu=heizanlage,ls=heizen,c=noun,ss=ation,gra=cap,
  mother={mc=noun,gra=cap,lus=heizanlage,s=anlage;money}},
  {lu=heizanlage,ls=anlage,c=noun,ss=anlage;money,gra=cap}.

2 {lu=hauptstromkabel,ls=haupt,c=noun,ss=a,gra=cap,
  mother={mc=noun,gra=cap,lus=hauptstromkabel,s=thing}},
  {lu=hauptstromkabel,ls=strom,c=noun,ss=phen,gra=cap},
  {lu=hauptstromkabel,ls=kabel,c=noun,ss=thing,gra=cap}.
```

5. Note that the lexeme *anlage* has two meanings, a money related meaning *ss=money* and an 'anlagen'-related meaning *ss=anlage* which are separated by a semicolon.

Head information is used for checking the status of the term which will be explained later. The objects 1 and 2 are generated and indexed in an EDGAR database. As we want to store abstract representations of the terms, only lexemes (i.e. the *ls* features) and the POS feature *c* are indexed in an EDGAR database. Accordingly, when retrieving a candidate variant, only these features are matched. The head information is stored in a separate data-file. Indexes are generated by concatenating the lexeme position in the term, the attribute path and the value of the indexed feature. In addition to this the length of each term is stored as an index, #2 and #3 for *Heizanlage* and *Hauptstromkabel* respectively. For *log-time* retrieval, these indexes are stored in a binary AVL-tree [WIR 83]. The following indexes are generated together with their associated list of entries :

index	numbers of indexing entries
1/ <i>ls</i> / <i>heizen</i>	[1]
1/ <i>ls</i> / <i>haupt</i>	[2]
1/ <i>c</i> / <i>noun</i>	[1 ;2]
2/ <i>ls</i> / <i>anlage</i>	[1]
2/ <i>ls</i> / <i>strom</i>	[2]
2/ <i>c</i> / <i>noun</i>	[1 ;2]
3/ <i>ls</i> / <i>kabel</i>	[2]
3/ <i>c</i> / <i>noun</i>	[2]
#2	[1]
#3	[2]

Each term is associated with the head information of the entry as shown below, where the first column denotes the number of the entry and the second column its associated head information :

- 1 {mother={mc=noun,gra=cap,lus=heizanlage,s=anlage;money}}
- 2 {mother={mc=noun,gra=cap,lus=hauptstromkabel,s=thing}}

#### 4.3.2. Retrieving Terms with EDGAR

When matching an input **GRAPH** on a database, EDGAR generates for each feature in the **GRAPH** an index, retrieves the sets of indexing entries and calculates their intersection. A sequence in the **GRAPH** matches an entry in the database, if the intersected set contains at least one member.

Example : When matching the phrase *in die Heizanlage* on the database above, first a morphological analysis is generated for which MPRO yields the following object :

```
{lu=in,ls=in,c=w,sc=p,gra=small},
{lu=d_art,ls=d_art,c=w,sc=art,gra=small};
{lu=d_rel,ls=d_rel,c=w,sc=rel,gra=small},
{lu=heizanlage,ls=heizen,c=noun,ss=ation,gra=cap},
{lu=heizanlage,ls=anlage,c=noun,ss=anlage;money,gra=cap}.
```

Note that the German word *die* (the) has two readings which are represented in two **AVMs** separated by a semicolon ‘;’. The first reading *lu=d\_art* corresponds to the article interpretation, the second interpretation corresponds to the relative pronoun. The

word *Heizanlage* contains two lexemes which are decomposed into two successive **FBs**-analogous to the process described for compilation step.

EDGAR generates the following indexes and calculates the intersection of the retrieved sets of indexing entries.

position	generated indexes	indexing entries	intersection
1	1/1s/in	[]	[]
2	1/1s/d_art;d_rel	[]	[]
3a	1/1s/heizen	[1]	[1]
3b	1/c/noun	[1;2]	[1]
4a	2/1s/anlage	[1]	[1]
4b	2/c/noun	[1;2]	[1]
4	#2	[1]	[1]

The first two words *in* and *die* are not indexed in the database and accordingly the retrieved set of indexes is empty. From the above table can be seen that EDGAR matches a two word sequence in the object starting at position 3. This sequence is identical to entry 1 in the database with respect to the matched features 1s and c. Subsequently, EDGAR generates a mother **AVM** which spans the matched sequence, retrieves `{mother={mc=noun,gra=cap,lus=heizanlage,s=anlage;money}}`, the head information of entry 1, from the database and copies it into the generated **AVM** :

```
[1 -> 2 conf:0.0000000 tree:NULL
  {lu=in,ls=in,c=w,sc=p,gra=small}
[2 -> 3 conf:0.0000000 tree:NULL
  {lu=d_art,ls=d_art,c=w,sc=art,gra=small};
  {lu=d_rel,ls=d_rel,c=w,sc=rel,gra=small}
[3 -> 4 conf:0.0000000 tree:NULL
  {lu=heizanlage,ls=heizen,c=noun,ss=ation,gra=cap}
[4 -> 5 conf:0.0000000 tree:NULL
  {lu=heizanlage,ls=anlage,c=noun,ss=anlage;money,gra=cap}
]] [3 -> 5 conf:0.0000000 tree:NULL
  {c=noun,mother={mc=noun,gra=cap,lus=heizanlage,s=anlage;money}}
]] .
```

The above **GRAPH** has a representation identical to the one generated by KURD rules. It the following graphical representation :

```
      { ... }
     /     \
in die heizen anlage
```

This **GRAPH** is then validated by KURD rules which check whether the matched sequence is an authorized term or possibly one of its variants. We will describe this process in the next section.

#### 4.3.3. Indexing Negation in EDGAR

EDGAR also allows to store and match negated features. Storing and retrieval becomes more complex, as the set of entries which matches a sequence in an object is

calculated in a compositional manner. The set of entries which is compatible with an **FB**  $\{A=V\}$  is the union of the entries which code  $\{A=_\}$  plus those entries which code  $\{A=V\}$  plus those entries which code  $\{A\sim=W\}$ , where  $W$  is different from  $V$ . The set of entries which is compatible with a **FB**  $\{A\sim=V\}$  is the union of the entries which code  $\{A=_\}$  plus the entries which code  $\{A\sim=W\}$ , where  $W$  has any value, plus those entries which code  $\{A=W\}$ , where  $W$  is different from  $V$ .

The table below shows in its right side the set of database entries which are to be retrieved for matching the **FB** on the left side. The symbols  $\cup$  and  $\setminus$  mean intersection of retrieved entries and complement of intersection respectively.

<b>FB</b>	set of entries matching the <b>FB</b>
$\{A=V\}$	$[\{A=_\}] \cup [\{A=V\}] \cup [\{A\sim=W\}] \setminus [\{A\sim=V\}]$
$\{A\sim=V\}$	$[\{A=_\}] \cup [\{A\sim=W\}] \cup [\{A=W\}] \setminus [\{A=V\}]$

Negated indexes are currently not needed in TTTT. However, as the tendency has been to shift from the rule-based to the example-based strategy, it might be desirable to store more and also more complex entries.

## 5. Generating and Retrieving Variants of Terms

In the previous sections we have presented the partial parser KURD, we have seen how an EDGAR database is compiled from a list of terms and how terms are retrieved. In this section we shall show how variants of terms are recognized. This section illustrates the retrieval of variants generated from the terms *Heizanlage* (heating installation) and *Hauptstromkabel* (main electricity cable). Currently, EDGAR checks for five types of variants :

- writing variants as in *Heizanlage* vs. *Heiz-Anlage*.
- derivational variants as in *Heizanlage* vs. *Heizungsanlage* (heating installation).
- synonyms as in *Heizanlage* vs. *Heizsystem*, where *system* is a synonym of *anlage*.
- permutation of head and expansion as e.g. in *Heizanlage* vs. *Anlage zum Heizen* (Installation for heating).
- omissions of elements as in *Hauptstromkabel* vs. *Hauptkabel* (main cable)

Combinations of these variants are equally recognized as, for instance *Einrichtung zum Heizen* (facilities for heating) or *Hauptleitung* (main line). To see how we derive *Einrichtung zum Heizen* from *Heizanlage* and *Hauptleitung* from *Hauptstromkabel* consider the following three steps, 1. lexemization, 2. Synonymization and 3a. permutation or 3b deletion :

- |     |                                    |   |  |
|-----|------------------------------------|---|--|
| 1.  | <i>Heizanlage</i>                  | → | <i>heizen#anlage</i>                   |
| 2.  | <i>heizen#anlage</i>               | → | <i>heizen#(einrichten ;anlage)</i>     |
| 3a. | <i>heizen#(einrichten ;anlage)</i> | → | <i>(einrichten ;anlage) zum heizen</i> |
|     |                                    |   |  |
| 1.  | <i>Hauptstromkabel</i>             | → | <i>haupt#strom#kabel</i>               |
| 2.  | <i>haupt#strom#kabel</i>           | → | <i>haupt#strom#(kabel ;leiten)</i>     |
| 3b. | <i>haupt#strom#(kabel ;leiten)</i> | → | <i>haupt#(kabel ;leiten)</i>           |

A set of KURD rules is used to perform the transformations 2 and 3 and to generate the head information as described above. The sequences of lexemes and the POS information of the authorized terms and their variants are indexed in an EDGAR database. Variants of terms are recognized by abstraction and re-instantiation : while the abstracted forms (i.e. lexemes) are indexed and matched in the database, variations are recognized as the re-instantiation differs from the head information of the authorized form.

The method implies that only the results of step 2 and step 3 need be stored in an EDGAR database. As eventually further variants are to be generated, the list of transformations will become longer. However, the number of recognized variants will grow much faster than the number of term transformations. On the other hand, the number of indexes will grow much more slowly than the number of generated entries. While coverage can thus be enhanced at a low computational price, precision remains stable as authorized terms and their variants can be retrieved and recognized based on the available information.

### 5.1. Terms and Synonyms

In the previous TTTT implementation [SCH 98], an initial set of synonyms was generated by back-and-forth translating : a set of terms were translated from German into English and all English translations were translated back into German. This initial set of synonyms turned out to be much too large and was incrementally reduced until now. Synonyms also tend to be user specific. TETRIS thus foresees different sets of synonyms for different user. The BMW set of synonyms — the biggest set in the current TETRIS project — contains 628 different lexemes with 2680 synonyms. Synonyms are indexed together with the authorized lexemes of the terms. Synonyms are also recognized as they have different lemmas.

Example :

Assume the lexeme *anlage* (plant,system,equipment,investment) has the synonyms : *aggregat* (aggregate), *einrichten* (installation,institution,set up), *einheit* (unit), *system* (system), *vorrichten* (device,equipment,apparatus). When generating synonyms in processing step 2 above, the KURD rule *syn1* replaces all occurrences of the lexeme *anlage* by their synonyms. Similarly, the rule *syn2* replaces all occurrences of *kabel* by a set of synonyms (*kabel ;leiten*).

```
syn1 = Aa{ls=anlage}
      : Ar{ls=aggregat;anlage;einrichten;einheit;system;vorrichten}.
```

```
syn2 = Aa{ls=kabel}
      : Ar{ls=kabel;leiten}.
```

By expanding the *ls* features these two rules modify objects 1 and 2 as presented in section 4.3.1.

```
1 {lu=heizanlage,ls=heizen,c=noun,ss=ation,
  mother={mc=noun,gra=cap,lu=heizanlage,s=anlage;money}},
  {lu=heizanlage,c=noun,ss=anlage;money,
  ls=aggregat;anlage;einrichten;einheit;system;vorrichten}.
2 {lu=hauptstromkabel,ls=haupt,c=noun,ss=a,gra=cap,
  mother={mc=noun,gra=cap,lu=hauptstromkabel,s=thing}},
  {lu=hauptstromkabel,ls=strom,c=noun,ss=phen,gra=cap},
  {lu=hauptstromkabel,ls=kabel;leiten,c=noun,ss=thing,gra=cap}.
```

When indexing the objects 1 and 2 in an EDGAR database the head information is extracted and stored in a separate file.

```
1 {mother={mc=noun,gra=cap,lu=heizanlage,s=anlage;money}}
2 {mother={mc=noun,gra=cap,lu=hauptstromkabel,s=thing}}
```

For each variant which matches an entry in the database, the head information is retrieved and allocated in a *mother* node as described in section 4.3.2. The partial derivation is then validated by KURD rules. The matched sequence is a variant, if any of the head features differs from the appropriate feature in the matched sequence. A synonym and the authorized form have different lemmas. The lemma of the variant *Heizsystem* i.e. *lu=heizsystem*, is not unifiable with the lemma of the authorized term as provided in the head information i.e. *lus=heizanlage*. While now all forms *Heizanlage*, *Heizaggregat*, *Heizeinrichtung*, *Heizeinheit*, *Heizsystem* and *Heizvorrichtung* match entry 1, only *Heizanlage* is recognized as an authorized term. By means of a KURD rule, the variants are accordingly marked by a flag.

## 5.2. Writing Variants

Writing variants of authorized terms are characterized by use of different upper case/lower case letters or different hyphenation pattern. The variant *heizanlage*, for instance, has a lower case initial letter while the variant *Heiz-anlage* has a hyphen which separates both lexemes. The sequence of lexemes in these variants is identical with the authorized term *Heizanlage*, the forms differ, however, in their lemmas *lu=heiz-anlage* and graphical feature *gra=small* respectively.

The MPRO analysis of *heizanlage* and *Heiz-anlage* yields the following objects :

```
{lu=heizanlage,ls=heizen,c=noun,ss=ation,gra=small},
{lu=heizanlage,ls=anlage,c=noun,ss=anlage;money,gra=small}.

{lu=heiz-anlage,ls=heizen,c=noun,ss=ation,gra=cap},
{lu=heiz-anlage,ls=anlage,c=noun,ss=anlage;money,gra=small}.
```

To detect a writing variant, a KURD rule checks whether the lemmas of the matched variants are identical with the lemma in the head information of the retrieved entry. Another KURD rule checks whether the *gra* features have the same value. If all features of the head information are unifiable with the corresponding features of the matched sequence, the sequence is considered to be identical to the authorized term. Notice that both forms can still be different in number and case as these variations are not considered to be discriminative for term formation.

Since in the above examples the head information is different from the matched sequence, the sequence is marked with an appropriate flag allowing for further processing dependent on whether the variant was detected in TemLint or in the author's tool.

### 5.3. Derivational Variants

Derivational variants of authorized terms are characterized by a differing derivational pattern of one or more expansion components in the term. While the derivational variants share the same lexeme with their authorized form, they differ in their lemma. MPRO recognized a number of derivational patterns, amongst this :

- action derivation as e.g. *Heizung* (heating)
- agent derivation as e.g. *Heizer* (heater) ; *Heizerin* (female heater)
- state derivation as e.g. *Heizbarkeit* (heatability)
- adjective derivation as e.g. *heizbar* (heatable)

Recognition of these variants is similar to recognition of writing variants by comparing the lemma of the authorized term and its variant :

MPRO object of *Heizungsanlage* :

```
{lu=heizungsanlage,ls=heizen,c=noun,ss=ation,gra=small},
{lu=heizungsanlage,ls=anlage,c=noun,ss=anlage;money,gra=small}.
```

As can be seen, the lemma *lu=heizungsanlage* is different from the lemma of the retrieved head information *lus=heizanlage* as shown above. Accordingly the variant *Heizungsanlage* is marked in the input object.

### 5.4. Permutation Variants

Variation by permutation is a kind of syntactic term variation. Currently TTTT produces noun phrase variants from compound nouns of the form  $N_1 N_2 \rightarrow N_2 W N_1$  and compound nouns variants from noun phrase in the style  $N_1 W N_2 \rightarrow N_2 N_1$  where  $N$  are nouns and  $W$  is a function word. A KURD rule performing the former type of variation is discussed in section 4.2.5.

Since the number and order of lexemes changes with syntactic variants, a new database entry 3 is generated. For the two lexeme term *Heizanlage*, for instance, the head noun and expansion are inverted and a **FB** {ls=\_,c=w} is inserted in between which matches function words :

```
3: {lu=heizanlage,c=noun,ss=anlage;money,
    ls=aggregat;anlage;einrichten;einheit;system;vorrichten,
    mother={mc=noun,gra=cap,lus=heizanlage,s=anlage;money}},
    {ls=_,c=w},
    {lu=heizanlage,ls=heizen,c=noun,ss=ation}.
```

This entry matches a large number of variants where both expansion and head may vary in number and case, the expansion may match a number of synonyms and both parts are joined with a unspecified function word.

As all instantiations of entry 3 are variants of *Heizanlage*, the head information of entry 3 is inherited from the authorized term and stored in the EDGAR database together with the head information of entry 1 :

```
1,3 {mother={mc=noun,gra=cap,lus=heizanlage,s=anlage;money}}
```

Apart from omission, these are the only syntactic variations currently implemented in TETRIS. As Royauté [ROY 96, ROY 99] show, there are many productive ways for syntactic variation and we believe that this is an area for future in TTTT.

### 5.5. Omission in compound nouns

Similar to Permutation variants, also omission variants are syntactic variants. While transforming a compound into a noun phrase requires inversion of the head noun and the expansion and an insertion of a function word, recognizing the omission of a part in a term requires deletion of a part. Thus, to recognize *Hauptkabel* (main cable) as an omission variant of the authorized term *Hauptstromkabel* (main electric cable) a new database entry 4 is generated.

```
4: {lu=hauptstromkabel,ls=haupt,c=noun,ss=a,gra=cap,
    mother={mc=noun,gra=cap,lus=hauptstromkabel,s=thing}},
    {lu=hauptstromkabel,ls=kabel;leiten,c=noun,ss=thing,gra=cap}.
```

As variants matching entry 4 are, actually variants of entry 2, the head information in entry 4 is inherited from entry 2 :

```
2,4 {mother={mc=noun,gra=cap,lus=hauptstromkabel,s=thing}}
```

Storing entries 1 to 4 in an EDGAR database, a huge number of variations for the terms *Heizungsanlage* and *Hauptstromkabel* can be matched and recognized while the database contains only 34 indexes.

It is however, likely that not all recognized variants are actually variants of the two terms. The database can easily be extended with further authorized terms. If, for instance, *Einrichten der Heizung* (set up of the heating) is not to be recognized as a variant of *Heizanlage* (heating installation), it can easily be added to the database as a further authorized term overruling its variant reading.

### 5.6. Filtering out Non-variants

Terms and their variants are recognized based on their sequence of lexemes. In addition to this, a variant of a term should also have a compatible head-semantic with the base form from which it was derived. As Jing and Tzoukerman [JIN 01] point out, reducing morphological variants to the same root (or lexeme, as in our case) enhances recall and — as an un-wanted side-effect — generates links between words which are not variants of each other. A sense-based retrieval in contrast is a precision enhancing procedure; it links words based on their semantics. Even though the words *heizen*, *Heizer* and *Heizung*, for instance, are derived from the same lexeme `ls=heizen`, they are not variants of each other : *heizen* is a verb or its nominalization (to heat, heat), *Heizer* is the agent derivation (fireman) and *Heizung* (heating) is an action derivation. Accordingly all forms receive different semantic features in the MPRO analysis :

Heizung : (action derivation)

```
{lu=heizung,ls=heizen,c=noun,ss=ation,gra=cap}.
```

Heizer : (agent derivation)

```
{lu=heizer,ls=heizen,c=noun,ss=er,gra=cap}.
```

heizen : (verb)

```
{lu=heizen,ls=heizen,c=verb,vtyp=inf,ss=v,gra=small};
```

```
{lu=heizen,ls=heizen,c=verb,vtyp=fiv,ss=v,gra=small};
```

```
{lu=heizen,ls=heizen,c=noun,ss=process,gra=small}.
```

Semantic features are categorized into three classes :

```
agent: {ss=er;agent}
```

```
state: {ss=quality;state;law}
```

```
action: {ss=process;ation;massnahme;result;act;event}
```

As a word form is only a variant of an authorized term if they don't have incompatible head semantics, only the nominalized reading of *heizen* (i.e. *Heizen*) is a variant of *Heizung* as both are action derivations.

## 6. Conclusion

In this paper we have presented a hybrid rule and example-based system for terminology management, the TETRIS Terminology Tool (TTTT). The aim of this tool is to provide an incremental approach for the elaboration of a prescriptive terminology and a means for checking consistent use of this terminology in documents. To accommodate both requirements, TTTT has two interfaces. The terminologist tool is designed to acquire and check new terminology and to update and modify the terminology pool. The author interface checks documents for consistent use of terms. Both interfaces are integrated in the company document flow.

Since its first implementation [HAL 96, Mul95, REU 98, SCH 98] the tool has undergone a number of changes. A major change in the underlying strategy was to

switch from a more rule-based approach to a more example-based approach which is documented in this paper. Using the example-based approach opens up completely new possibilities for the detection of syntactic variants of terms which was previously unthinkable due to computational intractability.

Work on TTTT has been in line with the methodology of iterative refinement as outlined by Meyer [MEY 01] :292

- 1) identify an initial set of knowledge patterns that the system will handle
- 2) analyze the output to identify other patterns that should be added or restricted
- 3) make the required changes to the system
- 4) repeat steps 2 and 3 as often as necessary

The set(s) of synonyms, for instance, have iterated the refinement loop numerous times and will most likely keep iterating as long as the terminology is incomplete and new concepts are added. The example-based strategy does not change this refinement cycle. The example-based strategy, however, changes the knowledge patterns that the system will handle. Thus, TTTT now provides also the possibility to refine pattern for syntactic variation which was previously impossible.

There are a number of unsolved problems in TTTT which will be tackled in the near future. To name just two : Whether or not a given word or sequence of words is a term also depends on the syntactic context in which the word or sequence occurs. For instance, in its nominal reading the expression might be a term but it is not a term when used as a verb. Parsing of the syntactic structure becomes important in order to figure out the syntactic context of a candidate term. Although TETRIS performs a shallow syntactic analysis<sup>6</sup> in its style and grammar checking tools cf. [HAL 00], the results of these processing steps are not fully exploited in TTTT.

A more complex processing device is also required when adding discontinuous terms to the terminology, or when special verbs are required in the context of particular terms. Discontinuous terms consist of several parts which are distributed in a sentence, while frame information of verbs select and control their arguments. Also here, in order to relate the different parts of the term or to select appropriate arguments for the verb, a syntactic analysis is required for a proper treatment.

Future research in TETRIS and in TTTT will focus on the exploitation of empirical knowledge, such as technical texts, manuals and/or term lists. These knowledge resources are to be joined with analytical knowledge of the language such as morphological analyses, knowledge of word and term formation, syntax and semantics and integrated into a human supported prescriptive authoring system.

---

6. This syntactic analysis is achieved with a set of KURD rules.

### Acknowledgments

We would like to thank the anonymous reviewers for their helpful and constructive comments. We also thank the IAI-crew in particular those people who are or were involved in TTTT and its predecessors, without whom this work would not have been possible.

## 7. Bibliographie

- [BOU 01] BOURIGAULT D., JACQUEMIN C., L'HOMME M.-C., *Recent Advances in Computational Terminology*, John Benjamins Publishing Company, Amsterdam/Philadelphia, 2001.
- [CAB 01] CABRÉ CASTELLVÍ M. T., ESTOPÀ BAGOT R., VIVALDI PALATRESI J., « Automatic term detection : A review of current systems », in [BOU 01], 2001, p. 53-89.
- [CAR 97] CARL M., SCHMIDT-WIGGER A., HONG M., « KURD - a Formalism for Shallow Postmorphological Processing », *Proceedings of the NLPRS*, Phuket, Thailand, 1997.
- [CAR 99] CARL M., « Inducing Translation Templates for Example-Based Machine Translation », *MT-Summit VII*, 1999.
- [HAL 96] HALLER J., « MULTILINT, A Technical Documentation System with Multilingual Intelligence », *Translating and the Computer 18*, London, 1996, Aslib, The Association for Information Management, Information House.
- [HAL 00] HALLER J., « MULTIDOC -Authoring Aids for Multilingual Technical Documentation », *First Congress of Specialized Translation*, Barcelona, 2000, <http://www.iai.uni-sb.de/~munpyo/IAI/bcn.doc>.
- [HAM 98] HAMON T., NAZARENKO A., GROS C., « A step toward the detection of semantic variants of terms in technical documents », *COLING'98*, 1998, p. 498-504.
- [HAM 01] HAMON T., NAZARENKO A., « Detection of synonymy links between terms : Experiment and results », in [BOU 01], 2001, p. 185-208.
- [HON 01] HONG M., FISSAHA S., HALLER J., « Hybrid Filtering for Extraction of Term Candidates from German Technical Texts », *TIA-2001*, 2001, <http://www.iai.uni-sb.de/de/pub.html>.
- [JAC 96] JACQUEMIN C., « A Symbolic and Surgical Acquisition of Terms Through Variation », *Connectionist, Statistical and Symbolic Approaches to Learning for Natural Language Processing*, 1996, p. 425-438.
- [JAC 01] JACQUEMIN C., *Spotting and Discovering Terms through Natural Language Processing*, The MIT Press, Cambridge, Massachusetts, 2001.
- [JIN 01] JING H., TZOUKERMANN E., « Determining semantic equivalence of terms in information retrieval : An approach based on context distance and morphology », in [BOU 01], 2001, p. 245-261.
- [MAA 95] MAAS H.-D., « Documentation of the Features used in MPRO », Software documentation, 1995, IAI, Saarbrücken.
- [MAA 96] MAAS H.-D., « MPRO - Ein System zur Analyse und Synthese deutscher Wörter », HAUSSER R., Ed., *Linguistische Verifikation, Sprache und Information*, Max Niemeyer Verlag, Tübingen, 1996.

- [MEY 01] MEYER I., « Extracting knowledge-rich contexts for terminography », *in [BOU 01]*, 2001, p. 279-302.
- [Mul95] IAI, Saarbrücken, Germany, « Multilinguale Intelligenz für die technische Dokumentation », 1995, <http://www.iai.uni-sb.de/de/multi-de.html>.
- [POL 95] POLANCO X., GRIVEL L., ROYAUTÉ J., « How to do Things with Terms in Infometrics : Terminological Variation and Stabilization as Science Watch Indicators », *Fifth International Conference of the International Society for Scientometrics and Infometrics*, 1995, p. 435-444.
- [REU 98] REUTHER U., SCHMIDT-WIGGER A., FOTTNER-TOP C., Multilint Abschlussbericht, Rapport, 1998, IAI, Saarbrücken, <http://www.iai.uni-sb.de/docs/papers/bericht5.ps>.
- [ROY 96] ROYAUTÉ J., MULLER C., POLANCO X., « Une approche Linguistique Infométrique de la Validation Terminologique pour l'Analyse de l'Information », *Informatique & Langue naturelle, ILN'96*, 1996.
- [ROY 99] ROYAUTÉ J., « Les groupes nominaux complexes et leurs propriétés : application à l'analyse de l'information », PhD thesis, Université Henri Poincaré-Nancy 1 College, Nancy, 1999.
- [SCH 98] SCHMIDT-WIGGER A., « Building Consistent Terminologies », *Proceedings of COMPUTERM'98*, 1998.
- [TET99] IAI, Saarbrücken, Germany, « Technologie-Transfer intelligenter Sprachtechnologie », 1999, [http://www.iai.uni-sb.de/tetris/tetris\\_home.htm](http://www.iai.uni-sb.de/tetris/tetris_home.htm).
- [WIR 83] WIRTH N., *Algorithmen und Datenstrukturen*, B.G. Teubner, Stuttgart, 1983.
- [WRI 97] WRIGHT S. E., BUDIN G., Eds., *Handbook of Terminology Management*, John Benjamins Publishing Company, Amsterdam/Philadelphia, 1997.