Universal Terminology

Chapter 18: Management of terms

This chapter is about the implementation of the $\mathbf{T}_{logy}$. The implementation has 3 facets: the database, the generating software and the website. The software itself has different aspects: The management of terms, the management of lists and the management of web pages. The management of terms is considered in this chapter.
This document is the chapter 18 of the book Universal Terminology which presents a global documentation on the $\mathbf{T}_{logy}$.

Contents

18.1 The role of software

The implementation of the $\mathbf{T}_{logy}$is a complex process based on essentially three facets: the database, the generating software and the website.

The database is the recipient of the raw content of the $\mathbf{T}_{logy}$, but the terms as visible by the casual users are possibly absent of the database depending on its form and its language. In order to optimize the maintenance of a collection of several hundred thousands terms in multiple languages, it has been necessary to create an abstract presentation of the $\mathbf{T}_{logy}$, which alone is present in the database. This abstract representation is the core of the Universal Terminology. It is the guarantee of the coherence between the different language versions.

Thanks to this universal representation, the real number of terms in the database is around 16000 items, representing up to 200000 virtual items as presented to the users of the $\mathbf{T}_{logy}$(values estimated in March 2023). This means a reduction of the number of terms by a factor 12. To the authors opinion, this figure is probably underestimated!

The website is the visible part of the $\mathbf{T}_{logy}$. It is the primary source of access, where any users in the world gain access to this anatomical knowledge.

The generating software is the link between the abstract database and the visible presentation of the $\mathbf{T}_{logy}$. The aspects of this software are about three categories of information: the terms, the lists and the web pages. The present chapter is about the management of terms.

18.2 Generate a term

A term belongs to an entity and each entity is the owner of several terms in several languages. Most of the terms belonging to an entity can be derived from the raw information about this entity, in particular one or more universal formulas.

A single procedure builds any term, without exception:
tEntity.SearchTerm

18.2.1 General architecture

This procedure is at the center of a general search schema for the generation of all terms in all forms of the $\mathbf{T}_{logy}$. In front of this procedure are several interfaces, easing the access to the search for the most common situations. The core of the search is a tree steps process, which are the translation of a formula, the expansion of term and the build of missing terms. This core process can also be viewed as a language independant part by definition common to all languages, and as many as necessary language dependant parts for the proper treatment of any language of the $\mathbf{T}_{logy}$. The general search schema is illustrated in figure 18.1.

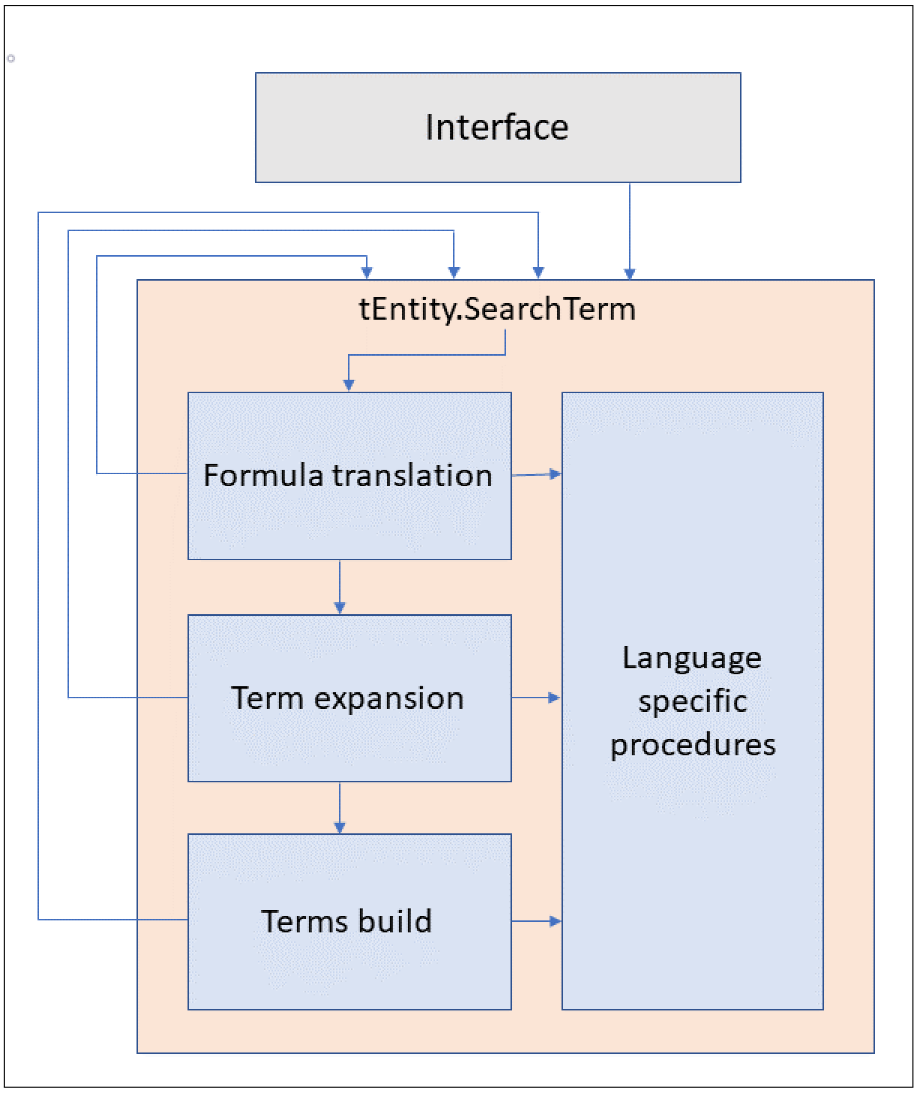

Figure 18.1: The overall generation of terms. The SearchTerm procedure is made of three phases: the formula translation occurs when a formula must be explicited in a language of the $\mathbf{T}_{logy}$; the term expansion is executed when a term has been found; if not found, the term build is executed. All three phases call the language specific procedures, developed for each language according to a strict list of linguistic functions. The recursive aspect of this process is visible on this schema.

Property	**Type**	**Description**
MyType	tSynType	type of term
Direct	boolean	present terms only
IsUniv	boolean	universal terms only
SynPos	Integer	position in case of multiple occurences

Table 18.1: Description of the entity **tEntity.Query** object, which defines the search to be performed. The search is on all possible terms for the entity, searching for MyType, in position SynPos if multiple occurences are possible. The boolean parameters specifically limit the search when true.

18.2.2 Query for the search of a term

The retrieval, search or creation of a term is driven by a property of the entity **tEntity.Query**, an object of type tSearch. This object has 4 properties that define the query for search of the term. These 4 properties must be set before calling the procedure and will govern the search. These properties are discussed in table 18.1.

The specification of the search, as defined by the Query object is completed by the property LgCurrent of the entity, giving the language of terms to be searched for.

Type	Value	Description
st_bas	bas	A basic term considered as the main term
st_dis	dis	A main term in display form
st_ofd	ofd	A basic term considered as the first official synonym
st_od2	od2	A basic term considered as the second official synonym
st_od3	od3	A basic term considered as the third official synonym
st_for	for	Universal formula of the main term
st_fr1	fr1	Universal formula of the first synonym
st_fr2	fr2	Universal formula of the second synonym
st_fr3	fr3	Universal formula of the third synonym
st_mod	mod	A model formula for the main term
st_md1	md1	A model formula for the first official synonym
st_md2	md2	A model formula for the second official synonym
st_md3	md3	A model formula for the third official synonym
st_syn	syn	A related term, not official
st_ori	ori	Original term in TA98
st_epo	epo	An eponym term
st_fma	fma	Name of the term in FMA
st_hrm	hrm	An inherited term from the main formula
...		Etc
st_xxx	xxx	Any other term, generally having a litteral value, attached to this term, in the current language

Table 18.2: Description of the principal values for the type of term. Each value is related to a single language or to the universal model.

18.2.3 Type of term

The type of terms tSynType is on control of all the terms, each one being assigned a unvariable type when created. The main values of interest in a search for a term are documented in table 18.2.

```
// Define the model formula
case AType of
  st_For: NewCateg := st_Mod;
  st_Fr1: NewCateg := st_Md1;
  st_Fr2: NewCateg := st_Md2;
  st_Fr3: NewCateg := st_Md3;
  else NewCateg := st_undef;
end;

// Search for a universal formula
Self.LgCurrent := lt_Universal;
Self.Query := tSearch.Create( NewCateg );
Self.SearchTerm;
ATerm := Self.CurrSingle;
if ( ATerm <> nil ) then
begin

  // Build the basic part in the specified language from formula
  MyTerm := tTerm.Create;
  MyTerm.Language := Lang;
  MyTerm.Formula := ATerm.Mandat;
```

Figure 18.2: An example of the search for a term: the search is for a Universal formula. This example is issued from the procedure tEntity.MakeFormula. Here, the variable AType has the value st_for, Self represent the instantiated entity, the local variable NewCateg receives the value st_mod, the current language is set to lt_universal, the local variable ATerm collects the resulting term, which property Mandat is the searched universal formula. The assignment of a new value to the variable MyTerm.Formula results in a call to the universal function UnivSetFormula. The result is obtained in the property CurrSingle that is the current term, in fact the term resulting from the translation of the formula.

18.2.4 Example of search

An example of a call to SearchTerm is visible on figure 18.2. The general strategy when building a search is: 1) select an instantiated entity; 2) build a query object for that entity; 3) be sure of the current language; 4) launch the search; 5) get the result on the specific properties of the entity.

18.3 Formula translation

This is one of the three phases of the generate a term process realized by the searchTerm procedure presented above. Any term of type st_for and relatives is by definition to be translated in the entity current language from its corresponding formula. As all the three phases, this process is dependant on the selected language of the term, governed by the language dependant procedures in place.

The formula translation procedure is uniquely call by the procedure tEntity.MakeFormula, that has been documented in figure 18.2. The assignment of the variable tTerm.Formula generate a call to the universal function UnivSetFormula. This function dispatches the controle to a language specific procedure, typically SetFormulaLA for the Latin language or SetFormulaFR for the French

Type	Lem	Value	Grammatical argument
noun	vena	vena	noun, nominative, feminine, singular
prefix	supra	supra	prefix, -, -, -
adjective	trochlearis	trochlearis	adjective, nominative, feminine, singular
prefix	supra	supra	prefix, -, -, -
adjective	trochlear	trochlear	adjective, -, -, -
noun	vein	vein	noun, -, -, singular

Table 18.3: The accompanying grammar of each term. This example is based on LA:*vena supratrochlearis* and EN:*supratrochlear vein*. When in the grammatical argument a part is invariable, the place is indicated by a dash.

language.

The processing of a universal formula is limited to its base part. The expansion part of the formula will be processed by the term expansion phase.

18.3.1 Translation procedure

Each formula is examined from left to right, corresponding to the natural order of words in Latin. But the generated term will be generated in the order corresponding to its specified language: Latin, French and Spanish follow the Latin order, but English and Russian have an inverted order of words.

The processing of a formula is a two steps process: first, the extraction of the next item of the formula and second, the retrieval of a word and its insertion in the translated term. In addition, an adjustment of the syntax between words of the term is necessary, depending on the specified language.

The extraction of an item of the formula is rather straightforward. A typical item is N(13097): N means according to the formal grammar described elsewhere that this item is a noun and the integer in round parentheses is the identifier of a vocabulary unit. This identifier, once it has been extracted, gives immediate access to the unit, which by definition is a container of words. Because the formula want a noun, the vocabulary unit is expected to contain a noun. If not an error is generated. In the item given as an example, the noun for 13097 in Latin is caput.

The insertion of the retrieved word in the translated term is a more complex process, largely dependant on the specified language. The main reason is the natural order of words in the specified language, which may differ from the order of words of the formula. A detailed discussion of this process is left to the documentation of the language specific procedures.

A translated term is accompanied by a sequence of grammatical cells documenting the syntax of each word contribution. Each cell contains the basic word or lem, the actual word and the grammatical argument. This argument possibly has four values: type of word, case, gender and number, some of them being absent in some languages. An example of such a grammatical sequence is given in table 18.3. When later on the term is expanded and modified, the grammatical argument is permanently updated to the new situation.

18.3.2 Grammatical concordance

When a new term has been translated from a universal formula, an adjustment of the syntax is generally necessary in order to adjust the words together in case, number and gender, depending obviously on the specified language.

18.4 Term expansion

This is one of the three phases of the term generating process realized by the tEntity.SearchTerm procedure presented above. Any term which has been translated from a formula, or created by the adhoc procedures and retrieved as belonging to the entity, needs to be processed for all its possible expansions. As all the three phases, this process is dependant on the selected language of the term, governed by the language dependent procedures in place.

There are four types of expansion: the adjective expansion, the preposition expansion, the mandatory expansion and the optional expansion. The preposition expansion is not yet implemented. Another special expansion is the adjunction of lateral modifiers for the left and right members of a pair. All expansions are extending the base term as generally issued from a universal formula. An expansion being specified by a target unit, and each unit being able to get its own expansions, the process is recursive. Up to four levels of recursion have been observed in the $\mathbf{T}_{logy}$.

18.4.1 Adjective expansion

This kind of expansion generates an adjective not being part of the base part of the term, but inserted into this base part at a convenient location. The adjective expansion corresponds to a referred unit, that has been declared to be represented by this adjective. Such a representation link is established between an ordinary physical anatomical unit and a vocabulary unit. This is this last mentioned unit that declares itself as the representative of the physical unit. This means that a vocabulary unit may represent one physical unit at maximum or none.

A typical example comes from LA:*cor*. The vocabulary unit is LA:*nomen cor* that declares itself with the reference 3614 as the representation of this physical entity. When looking at the values of the vocabulary entity, one find the adjective *cardiacus* together with the noun *cor* and the prefix *cardio* for Latin, and the equivalent values for all other languages.

An adjective expansion may include a prefix, handled as the adjective and referring to a convenient physical unit. In this situation, the extracted word from the vocabulary entity is the prefix.

The main difference between an adjective declared in the base part or an adjective issued from an expansion is the fact that the former is purely lexical when the later refers to an anatomical entity, The overall $\mathbf{T}_{logy}$is in fact, together with all other types of expansion, a network of links between entities, totally distinct of the traditional hierarchies like the partonomy or the taxonomy. The appartenance to this network is important and must preferably be made explicit. For this reason, the authors of the $\mathbf{T}_{logy}$are mandated to implement a maximum of adjective expansions.

Another aspect is relevant in favor of the adjective expansions: when formulated in a universal formula, such an expansion may be inherited in a specific language with a transformation to a mandatory expansion (or vice-versa). This mechanism is quite useful, because there is a well known divergence between modern languages about the use of an adjective instead of a noun complement. This possible inheritance conciliates the opposed points of view.

The management of the adjective expansion is rather simple. First, the adjective is retrieved from the vocabulary entity. Second, the adjective is inserted at its good location (dependent on the language). Third, the syntax of the adjective is adjusted.

The good location for insertion of an adjective in a base term is immediately close to the noun. But this location may be discussed in some languages in presence of an apposition or a noun complement. This position is automatically calculated and cannot be adjusted at will.

18.4.2 Mandatory expansion

18.4.3 Optional expansion

18.4.4 Lateral expansion

18.5 Term creation

This is one of the three phases of the generate a term process realized by the tEntity.SearchTerm procedure presented above. Any term not explicitly represented in the database of the $\mathbf{T}_{logy}$must be created on the basis of the existing raw information. This raw information is either the properties of the entity itself, or other already present terms of this entity. As all the three phases, this process is dependant on the selected language of the term, governed by the language dependant procedures in place.

This phase is executed because the search query mentions a non-existing type of term in the current langauge. This is the responsability of this process to attempt to create this new term and to store it in the entity. For example, if the query is for a st_bas type of term for an entity uniquely specified by a universal formula, this term must be created by a call to the formula translation phase. Another example is when we want a term for a member of a pair, where nothing is explicitely present in the database and all needed terms must be created, starting from the generic entity.

18.5.1 Select a creation procedure

The selection of the adequate creation procedure is principally based on the type of term that is searched for. There are 11 distinct procedures, which can be grouped in the procedures generating a term in formal form and those generating a term in display form.

The terms of the $\mathbf{T}_{logy}$may exist in two different forms of presentation. The canonical form is named **formal form**, and it applies all the theoretic rules of the terminology. In particular all terms are presented at nominative singular followed by a genitive part. Plural terms are forbidden. But, because the formal terms are odd and complicated and not adequate in a general presentation, they can be replaced by the **display form**, that is simplified and adapted to current usage. Both forms are strictly synonym, and a term in one form can always be replaced by the other form. Notice that if the formal form and the display form are equal, we speak of a **single form**.

The procedures are summaried in the table 18.4.

Each group must further dispatch to distinct procedures, generally depending on either the type of entity or the type of partonomic link to the ancestor of the entity.

The new term is locally created of type tSingle and the dispatched procedure is necessarily a function attached to this type. Each function is of type boolean, that indicates the success of the creation or the failure. On return, the flag is tested, resulting either in the addition of the new term to the entity, or a nil result with a possible error message.

Group	Description	Procedure
single form	missing base part for single term	CreateBaseSingle
		CreateFormulaLateral
formal form	missing formal term	CreateFormalPair
		CreateFormalPset
		CreateFormalSet
		CreateFormalSetLateral
		CreateFormalLateral
display form	missing display form	CreateDisplayGeneric
		CreateDisplaySpecific
		CreateDisplayLateral
		CreateDisplayPair

Table 18.4: Branches for redirecting the process of creation of missing terms. Each branch is briefly described for its usefullnes in specific situations. There are 11 distinct procedures, of which 2 are single form (simultaneously formal and display form).

18.5.2 Model of a creation procedure

There are currently 11 procedures for the creation of missing terms, as visible on table 18.4. All these procedure are language independant. However, they may present a separate treatment for different languages. In addition they can call some generic procedures that are dispatching to language specific procedures.

The model is rather simple and summarized in a 3-step process:

- Retrieve another term from which the present term is built.
- Transform the retrieved term to the new term.
- Prepare the new term for storage in the entity. This step generally call a universal function, which dispatches the controle to the adequate language-specific procedures, as prepared for each language of the $\mathbf{T}_{logy}$.

18.5.3 All creation procedures

In this subsection, we give a short presentation of each creation procedure and in which circumstances it is call. A procedure will generally call a universal function, which has the role of dispatching to the language specific procedures necessary for the construction of the new term. This universal function is explicitly mentioned. In addition, examples of units using this procedure are given.

- **CreateBaseSingle**
  This procedure is call when one want to build the main term of a single unit, that has a formula and no st_bas term in the specified language. Examples are LA:*cor*, LA:*cauda equina*, LA:*lingua*. It works also for a deleted unit like LA:*organum subfornicale*.
- **CreateFormalLateral**
  This procedure is call when one want to build the main term of a lateral member of a pair unit in the specified language. Example is LA:*arteria*

subclavia sinistra. Example for a synonym is LA:*fibrae meridionales corporis ciliaris sinistri*. The st_bas term of the generic subclavian artery is first retrieved in the specified language. Notice here that this term possibly needs to be created if it does not exist. Then, this term is modified by the universal function **tTerm.Lateral**, which will call the adequate version for the specified language. If the st_bas term is irregular, an attempt to load in place a possibly present universal term is performed. If this is not possible, the creation fails. In case of the presence of an optional expansion, the laterality must also be tentatively applied on this part of the term, preparing the term for two different solutions, one with the optional expansion and one without it. Typically with LA:*equator lentis*, the left member is either equator sinister or equator *lentis sinistri*, but equator sinister *lentis* is not correct! This rule is without exception and is valid for all languages.

- **CreateFormalPair**
  This procedure is call when one want to build the main term of a pair entity, which is part of a pair unit in the specified language. This procedure creates the formal form. Example is LA:*par arteriarum subclaviarum*. Example for a synonym is LA:*substantia propria corneae* with synonym *stroma corneae*. The st_bas term of the generic subclavian artery is first retrieved in the specified language. Notice here that this term possibly needs to be created if it does not exist. Then, this term is modified by the universal function **tTerm.FormalPair**, which will call the adequate version for the specified language. If the st_bas term is irregular, an attempt to load in place a possibly present universal term is performed. If this is not possible, the creation fails.

- **CreateFormalPset**
  This procedure is call when one want to build the main term of a pset entity, which is part of a pset unit in the specified language. This procedure creates the formal form. Example is LA:*venae supratrochleares*. Example for a synonym is LA:*fibrae meridionales corporis ciliaris* with synonym *fibrae longitudinales*. The st_bas term of the generic vena supratrochlearis is first retrieved in the specified language. Notice here that this term possibly needs to be created if it does not exist. Then, this term is modified by the universal function **tTerm.FormalPset**, which will call the adequate version for the specified language.

- **CreateFormalSet**
  This procedure is call when one want to build the main term of a set entity, which is part of a set unit in the specified language. This procedure creates the formal form. Example is LA:*classis septorum cordis*. Example for a synonym is LA:*venae anteriores ventriculi quarti* with synonym *venae cardiacae anteriores*. The st_bas term of the generic septum cordis is first retrieved in the specified language. Notice here that this term possibly needs to be created if it does not exist. Then, this term is modified by the universal function **tTerm.FormalSet**, which will call the adequate version for the specified language.

- **CreateDisplayGeneric**
  This procedure is call when one want to build the generic term of a set,

mset or pset entity. Example is LA:*musculi pectinati atrii dextri*.

- **CreateDisplaySpecific**
  This procedure is call when one want to build the specific term of a pset entity. Example is LA:*venae thoracoepigastricae*.

- **CreateDisplayLateral**
  This procedure is call when one want to build the display lateral term of a pset entity. Example is LA:*venae supratrochleares*. The st_bas term of the trochlear vein is first retrieved in the specified language. Notice here that this term possibly needs to be created if it does not exist. Then, this term is modified by the universal function **tTerm.LatPlural**, which will call the adequate version for the specified language.

- **CreateDisplayPair**
  This procedure is call when one want to build the main term of a pair unit, which is part of a pair unit in the specified language. This procedure creates the display form. Examples is LA:*arteria subclavia (par)*. The st_bas term of the generic subclavian artery is first retrieved in the specified language. Notice here that this term possibly needs to be created if it does not exist. Then, this term is modified by the universal function **tTerm.DisplayPair**, which will call the adequate version for the specified language. In fact, here, the function simply generates the word pair in round bracket, which is added to the right of the term.

- **CreateFormulaLateral**
  This procedure is call when one want to build the lateral member of a pair unit in the specified language. Examples is LA:*venter anterior musculi digastrici sinistri*. The st_for term of the generic entity of the pair is first retrieved in the specific language. Notice here that this term possibly needs to be created if it does not exist. Then, this term is modified by the universal function **tTerm.Lateral**, which will call the adequate version for the specified language.

18.6 Universal linguistic functions

The universal linguistic functions is a set of language independant procedures in charge of the dispatching to the proper language dependant procedures, depending on the currently specified language. These universal functions totally separate the linguistic aspects in the main stream procedures from the language specific implementations.

The idea when calling one of these functions is to ask for a universal aspect in a given language when ignoring the actual implementation of that language. In this way, the authors of the terminology can work on the universal $\mathbf{T}_{logy}$, which is closed to Latin version, without being disturbed by the intricacies of the vernacular languages. At the same time, the persons responsible for a particular language are free to implement their own specific point of view on the terminology, even if the general recommendation is to follow as much as possible the universal version. Indeed, this is a perpetual compromise to navigate between a strict formal terminology and authenticity of a natural language.

A chapter of the present documentation is specifically dedicated to the implementation of any other language of the $\mathbf{T}_{logy}$. It will be seen that any new language implementation is made in four parts:

- **Vocabulary**
  Defining all the words of the $\mathbf{T}_{logy}$in the specified language.
- **Grammar rules**
  Defining the exhaustive list of rules for the correct grammatical treatment of the terms, as well as all exceptions to these rules.
- **Test procedures**
  Implementing the test functions for the grammar rules.
- **Universal functions**
  Implementing the universal functions for the specified language.

There are 17 universal functions, that must be implemented for each language of the terminology. All the function names are starting with the letters 'Univ'. The table 18.5 gives an overview on these functions. A short description is provided below:

- **UnivPlural**
  This universal function makes a plural of a term in the specified language. A plural term is obtained by transforming each noun or adjective at nominative singular to the plural. This procedure is dependant on each language. The formal equivalent function is UnivFormalSet.
- **UnivLateral**
  This universal function makes a lateral member of a pair unit in the specified language. A lateral member is obtained the by the addition of the lateral adjective left or right in the adequate position of the term. This position is dependant on each language.
- **UnivLatPlural**
  This universal function makes a lateral member of a pset unit in display

Function	Description
UnivPlural	Make the nominative plural of a term
UnivLateral	Make the lateral member of a term
UnivLatPlural	Make the display lateral plural member of a term
UnivFormalLatPlural	Make the formal lateral plural member of a term
UnivGenitive	Make the genitive singular of a term
UnivGenPlural	Make the genitive plural of a term
UnivGenLatPlural	Make the lateral genitive plural of a term
UnivFormalPair	Make the formal pair term
UnivDisplayPair	Make the display pair term
UnivFormalSet	Make the formal set term
UnivFormalPset	Make the formal pset term
UnivMakeMandat	Make a mandatory expansion
UnivMakeAdjective	Make an adjective expansion
UnivMakeOption	Make an optional expansion
UnivMakeLateral	Make a lateral placeholder
UnivSetFormula	Translate a universal formula
UnivSetNominative	Enter a nominative term

Table 18.5: List of all universal linguistic procedures. Each procedure is call when some linguistuic aspect must be dispatched to the proper language procedure.

form in the specified language. Such a member is in fact a set. A lateral member is obtained the by the addition of the lateral adjective left or right in the adequate position of the term. Because it is a set, it may be presented in display form or in formal form.

- **UnivGenitive**
  This universal function makes a genitive singular term. In particular, it is used for the mandatory and optional expansions.

- **UnivGenPlural** This universal function makes a genitive plural term. In particular, it is used for the formal terms and sometimes for the mandatory expansions, that can be specified at plural.

- **UnivGenLatPlural**
  This universal function makes a lateral genitive plural term.

- **UnivFormalPair**
  This universal function makes a formal pair term in the specified language. The standard form of such a formal pair term is the singular nominative part made of the single word pair, followed by the given term at genitive plural. However, in order to be close to the practice of casual users, the UnivDisplayPair function provides an ordinary plural form, despite it is formally not totally correct!

- **UnivDisplayPair**
  This universal function makes a display pair term in the specified language. The standard form of such a display pair term is the nominative plural of the term followed by the word pair in round brackets.

- **UnivFormalSet**
  This universal function makes a formal set term in the specified language. The standard form of such a formal set term is the singular nominative part made of the single word set, followed by the given term at genitive plural. This is how to avoid the plural terms which are vorbidden in a modern terminology. However, in order to be close to the practice of casual users, the UnivPlural function provides an ordinary plural form, despite it is formally not totally correct!

- **UnivFormalPset**
  This universal function makes a formal pset term in the specified language. The standard form of such a formal pset term is the singular nominative part made of the single word pair, followed by the genitive plural wors for set, followed by the given term at genitive plural. However, in order to be close to the practice of casual users, the UnivPlural function provides an ordinary plural form, to which the word pair in round parentheses is appended, despite it is formally not totally correct!

- **UnivMakeMandat**
  This universal function prepares a mandatory expansion.

- **UnivMakeAdjective**
  This universal function prepares an adjective expansion.

- **UnivMakeOption**
  This universal function prepares an optional expansion.

- **UnivMakeLateral**
  This universal function prepares a bilateral term with the introduction in their decriptive grammar of a lateral placeholder, to be possibly replaced on need by a lateral adjective, left or right. The position of the placeholder, as well as its grammatical definition, is largely language specific.

- **UnivSetFormula**
  This universal function translates the base part of a universal formula in a term of the specified language.

- **UnivSetNominative**
  This universal function accepts a free text and prepares its descriptive grammar. This is performed under the condition that the free text is regular respective to the formal grammar of the $\mathbf{T}_{logy}$. If not, the term is irregular and consequently it is hard to be transformed and its usage is problematic.

18.7 Log of updates

01 Mar 2023 Creation of the file.

18.8 Credentials

This document is part of the book "Universal Terminology" accompanying the website on Terminologia Anatomica. It expresses the vision of the authors of the $\mathbf{T}_{logy}$about the foundations of the science of ontology, supporting the here presented terminology. Despite it is as exact as possible, close to the reality of the database of the terminology and the surrounding software, approximations, errors and ambiguities are possible and should be considered as independent of their willingness and intents.

Identified comments about the content of the website and its presentation are welcome. An appropriate answer will be given when pertinent.

Authentic URL of this file: `https://ifaa.unifr.ch/Public/TNAEntryPage/help/Chap14.pdf`