Technische Universität Braunschweig

Institut für Verkehrssicherheit und Automatisierungstechnik iVA
Prof. Dr.-Ing. Dr. h.c. mult. E. Schnieder

User Manual

Contents

List of illustrations

Index

Dear *iglos* user,

this manual is supposed to give you a simple introduction to using *iglos.* With short and lucid chapters we want to provide you with all relevant information for developing and modelling a terminology in *iglos*. For this purpose you may consult the „What is … ?“ chapters.

This knowledge will enable you to manage your own terminology. In order to quickly orient yourself without long times of searching, each “What is …?” chapter is usually followed by a “How to … ?” chapter, where you can look up the according steps.

If there are any unanswered questions, please do not hesitate to send us an email (*info@iglos.de*).

Your *iglos* team

1. What is an *iglos*-terminology?

iglos terminologies are lists of lexemes. Lexemes consist of a designation, a definition (a description of meaning) and a specification of the technical language the lexeme belongs to and in which the combination of designation and definition are commonly used (= variety). In everyday communication designations are used to implicate meanings. Participants of everyday communication assign a familiar meaning to designations which appears to be most suitable for the conversation at hand. Therefore, a listener's or reader's understanding is probably not the same a speaker or writer wants him to understand – misunderstandings may result easily.

In professional conversation such an unreliable method of communication is undesirable. Terminologies try to avoid those impairments of communication by making meanings explicit and by determining what should be understood while hearing or reading a designation. This is supposed to allow a common comprehension among all participants of communication.

2. What is a lexeme?

Lexemes consist of three components. It is necessary to describe all of these three parts, in order to differentiate lexemes from each other.

1 Designation
Designations are names for existing objects, persons and ideas (= entities).

2 Definition
Definitions are short descriptions of meaning, determining the understanding while reading, listening, articulating or writing a designation.

3 Variety
Subject fields often use their own designations and definitions to denote the entities they theorize on. Languages of subject fields are called functional varieties.

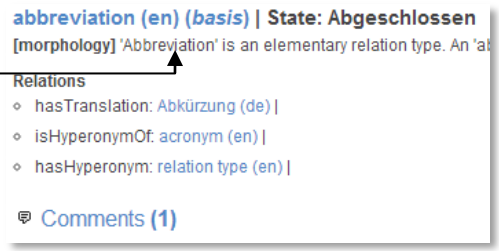

Illustration 1 Example for a lexeme

Lexemes of a terminology are never isolated, but are always situated within a terminological system and thereby have **relations** to other lexemes. In *iglos* those relations are understood as an important part of the lexeme.

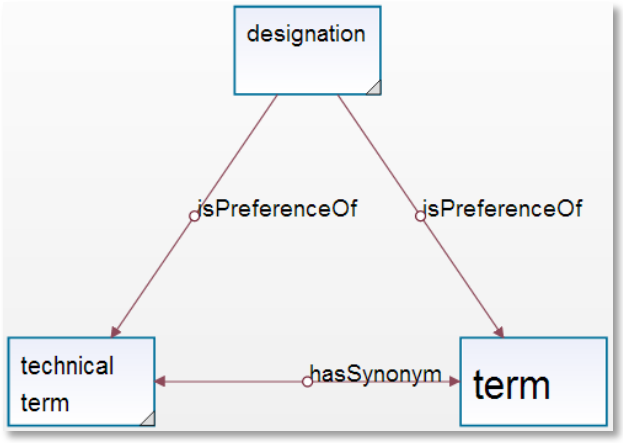

Illustration 2 Example for related, visualised lexemes

Further information:

How to create a lexeme? (p. 15)
What is a lexeme relation? (p. 22)

3. What is a variety?

As noted above, common language designations are used to implicate meanings. Experts in economy and science, however, deploy them to communicate about very special knowledge. The knowledge of these experts is significantly more comprehensive and describes entities far more precisely than non-professionals would do. Meanings they associate with designations therefore differ considerably from those meanings non-professionals associate with designations – even if both use an identical designation!

Even though experts communicate in one of the 6.000 natural languages (e.g. English) it can be stated that they have their own language. This own language is embedded in a special subject field and is used to communicate about the objects of consideration.

In *iglos* the specification of the technical language has an important function: To locate a designation within a technical language limits its possible meanings and defines it to some extent. Therefore specifying the technical language reduces the inherent ambiguity of a designation. Various other meanings associated with a designation, are therefore eliminated from the outset.

Technical languages are represented as a variety tree in *iglos*. Five levels are recommended to distinguish meanings:

1 **Super variety:** National languages like English constitute the upper node of the linguistic hierarchy. All technical communication is located within a natural national language. It is **not** part of a superior variety.

2 **Functional variety:** Functional varieties are part of a natural language. They are used in particular subject fields and include its entire terminology, e.g. linguistics. Varieties below this level are part of a superior variety.

3 **Varieties of sub-disciplines:** Each subject field is divided into further particular fields, so that language peculiarities can arise as well. Occasionally terminologies of particular fields can come into conflict with each other. For example the functional variety **of linguistics** includes the varieties of the sub-disciplines syntax, phonetics, special language linguistics

4 **Topic varieties**: The particular varieties of sub-disciplines may include further varieties, especially concerning very complex subject areas. This distinction, however, should only be necessary, if the terminology of a subject field is in conflict with its coordinate varieties. For example: terminology science as a part of special language linguistics

5 **Project varieties** are on the lowest level of the linguistic hierarchy. They contain terminology that has not been used in its particular combination in a subject field before.

Additionally, regardless of our recommendation *iglos* allows a free use of the variety, so you can determine the criteria by which you want to distinguish your lexemes from each other all by yourself. The number of levels can be freely chosen as well.

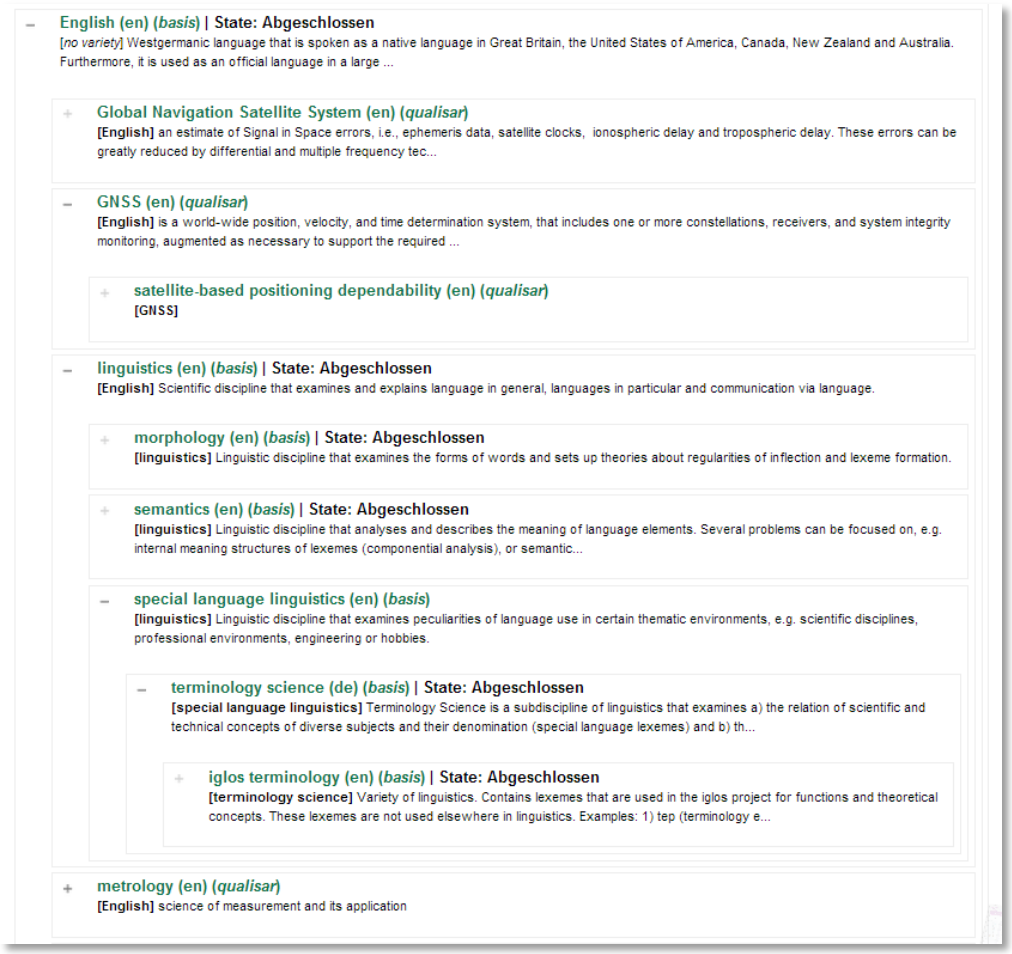

Illustration 3 Example for a variety tree

More information:
How to create a variety? → See How to create a variety? (page 19)

4. What is a definition?

A definition is a short description, determining the meaning of a lexeme.
Example:

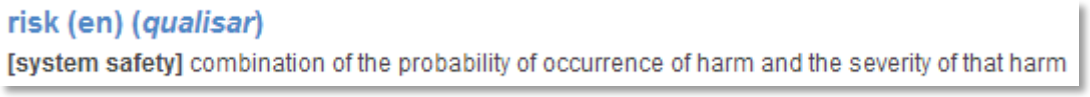

Illustration 4 Example for a definition

Definitions may contain:

- the class a defined entity belongs to
- what an entity consists of
- the properties of an entity
- additional knowledge for handling an entity

Definitions should **not** be:

- too long, but neither too short, i.e. they should contain enough information which can be used to differ one entity from another
- tautological, i.e. the definition should not contain the designation it tries to define
- circular, i.e. two corresponding definitions should not refer to each other

How to create a definition? → See *How to create a lexeme?* (Page 15)

5. What is an instance?

iglos is organizing terminologies in instances. User groups (e.g. research teams, institutes or companies) usually have a shared instance for their terminology:

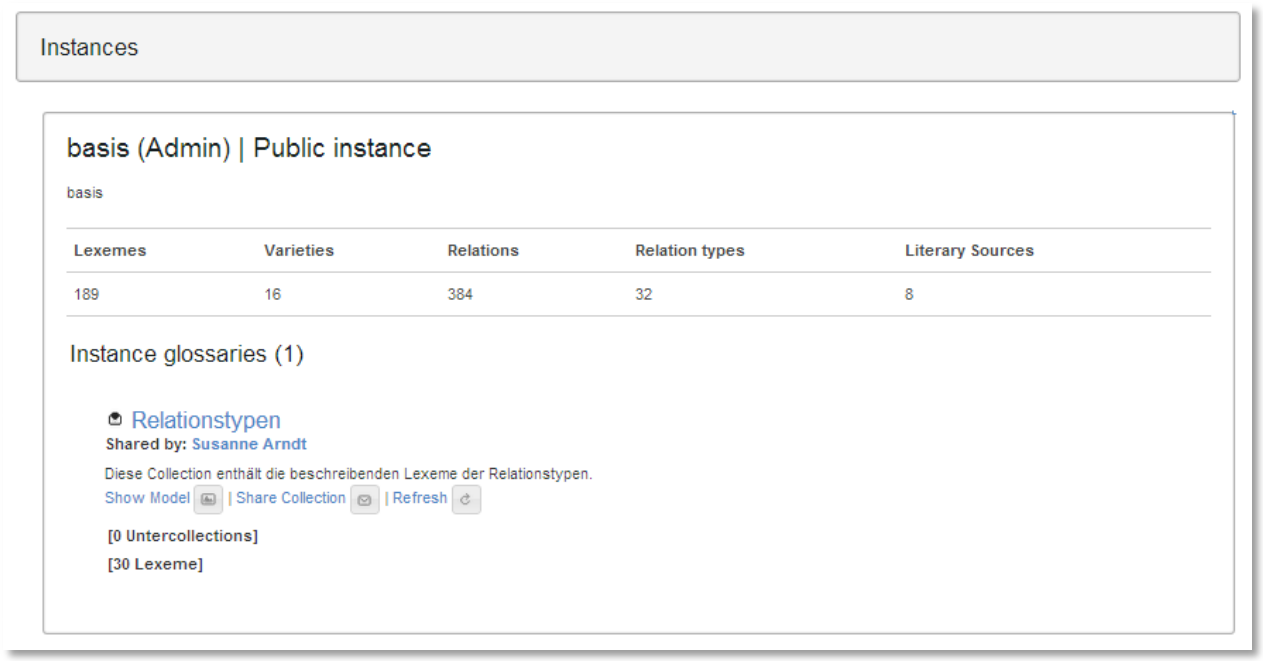

Illustration 5 *iglos* instance overview

An instance allows the separation of data, which means different terminologies can be modelled separately. This is especially important if terminologies are to have limited access for different user groups. Unauthorized access to terminologies will be denied by different access rights. These available rights are *user* (read access), *editor* (write access), *admin* (extended write access).

Even if you do not have permission to edit an instance, you can still have read access for foreign instances (*user*). It is possible to search inside multiple instances at a time. To narrow the focus of a search an instance can be applied to the filter.

Only one instance is editable at a time. This is set as **standard instance** and can be switched in *options*:

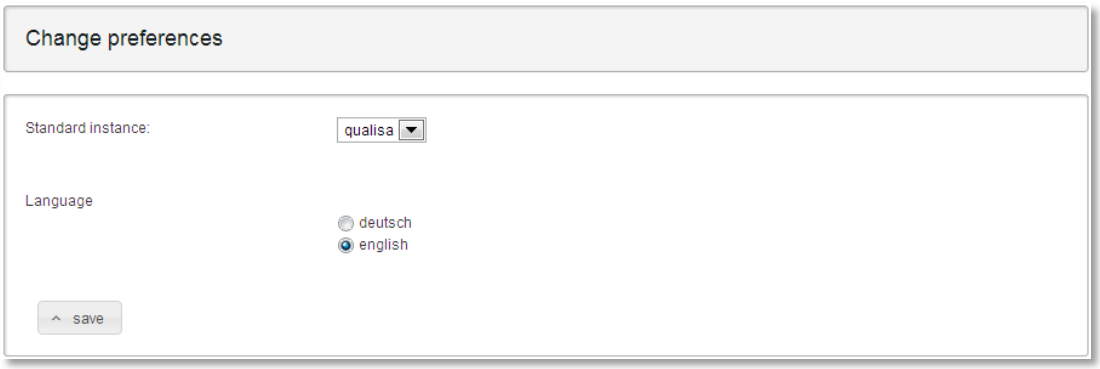

Illustration 6 Switching the standard instance in *Options > Preferences*

Editing an instance means: To add lexemes, relate lexemes, comment lexemes or to remove lexemes – that is, the construction or reduction of a terminology. To set another but the standard instance, an instance can be selected from the drop-down menu next to the item to be edited:

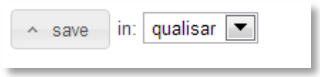

Illustration 7 Selection of instances when saving data

An instance, for which access rights are available, can be chosen can be selected from the list of available instances. Access rights are displayed on the start page or in the list of instances next to the instance name.

6. How to find lexemes?

1. Enter the designation of the lexeme in the text box next to the *iglos* logo.
2. Set the filter for
   - language code
   - instance
   - variety
   - state
   - modification period

   Click on *apply*.
3. Apply the search to current or deleted lexemes.

   Notice: In *iglos* lexemes are not deleted but marked as deleted. This way you can still find previous versions.

4. Perform a search for similar items or for exact matches.
5. Perform the search by clicking on the magnifiying glass next to the text box.

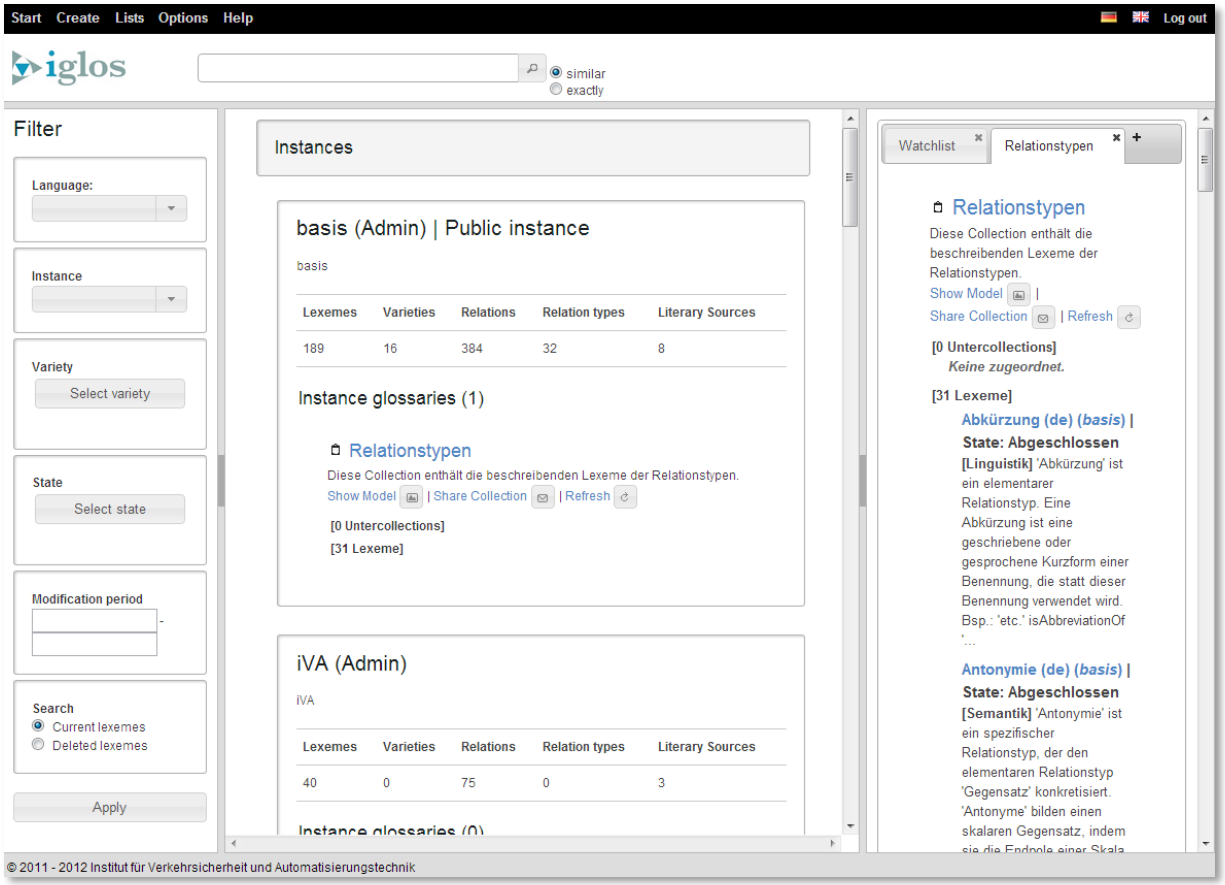

Illustration 8 Filter and search

6. Group the search results by
   - designation
   - definition
   - variety
7. Switch to full view of a lexeme by clicking on the arrow icon on the upper right edge of the lexeme.

   Notice: The full view of a lexeme contains comments to the lexeme.

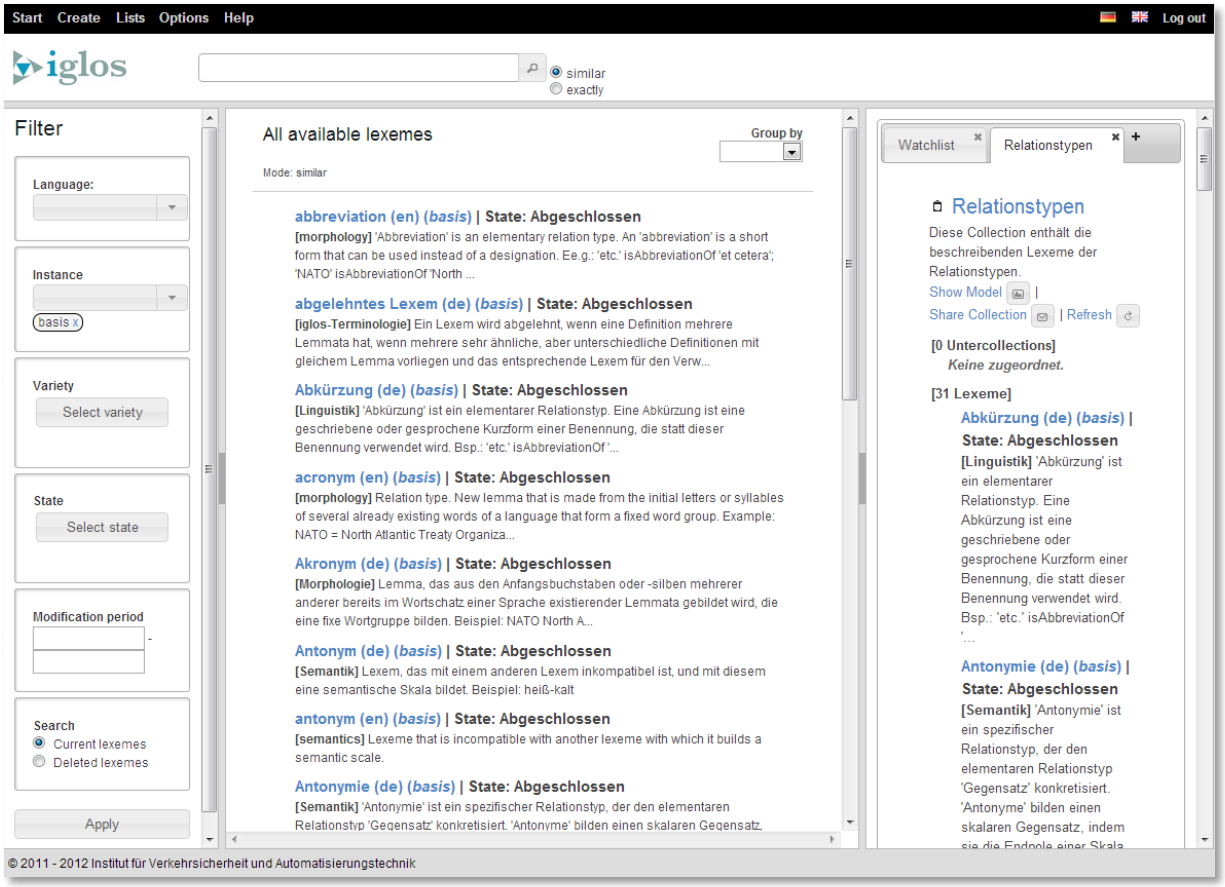

Illustration 9 List of search results

7. How to create a lexeme?

1. Click on *Create* in the menu, then click *Lexeme*.

   Notice: The lexeme assistant opens in the middle frame.

2. Enter the data of the lexeme into the tab *Lexeme*.

   – the designation
   – the language code

   Notice: The language code *international* is used for codes from classification systems etc.

   – the definition
   – the variety: Select the variety by clicking the check mark button in the varietiy list.

   Notice: Varieties are lexemes themselves and need to be created before they can be set as a variety of other lexemes.

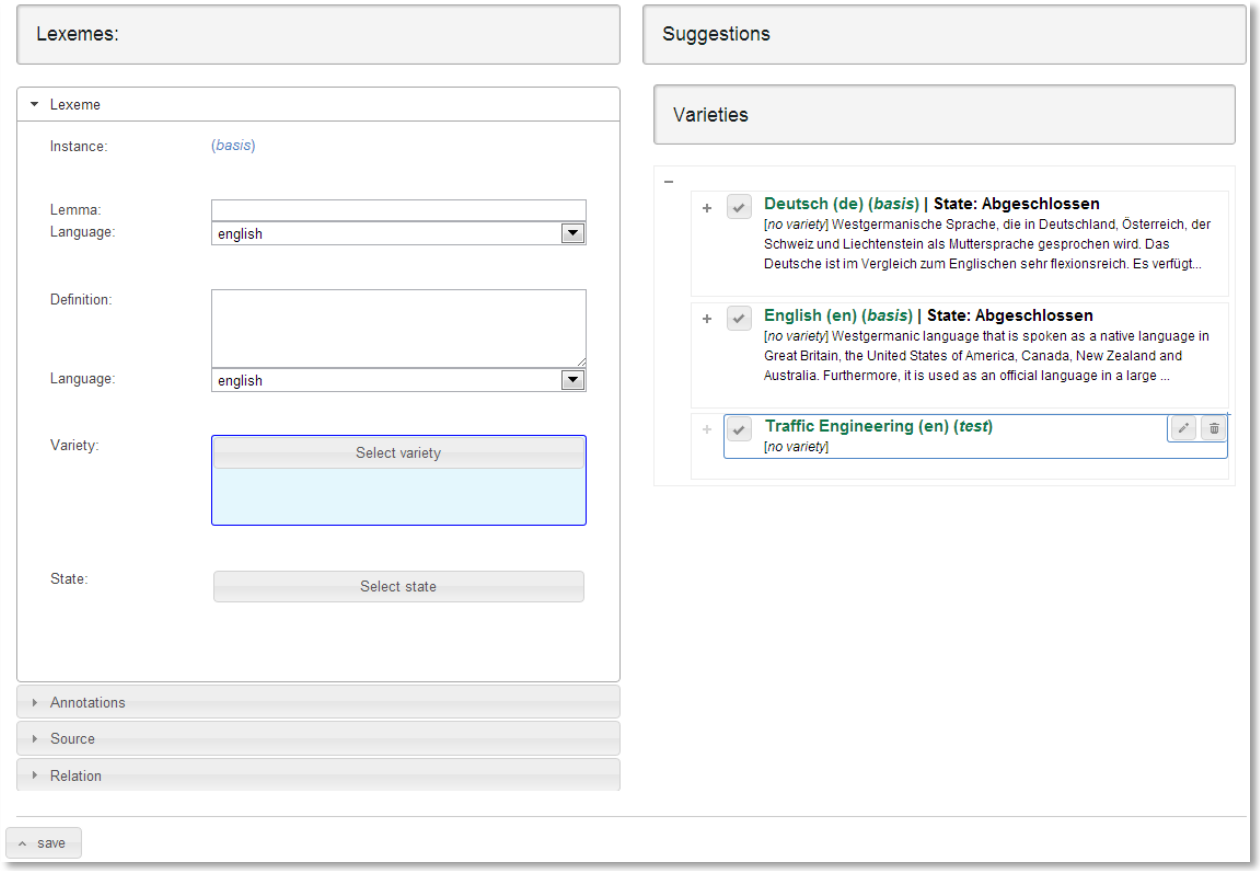

Illustration 10 Lexeme Assistant, tab *Lexeme*, selecting the variety

– State: Select the state in the state list by clicking the check mark button.

Notice: *iglos* provides a state system with its basis instance. As an editor you can additionally create your own state system in your instance.

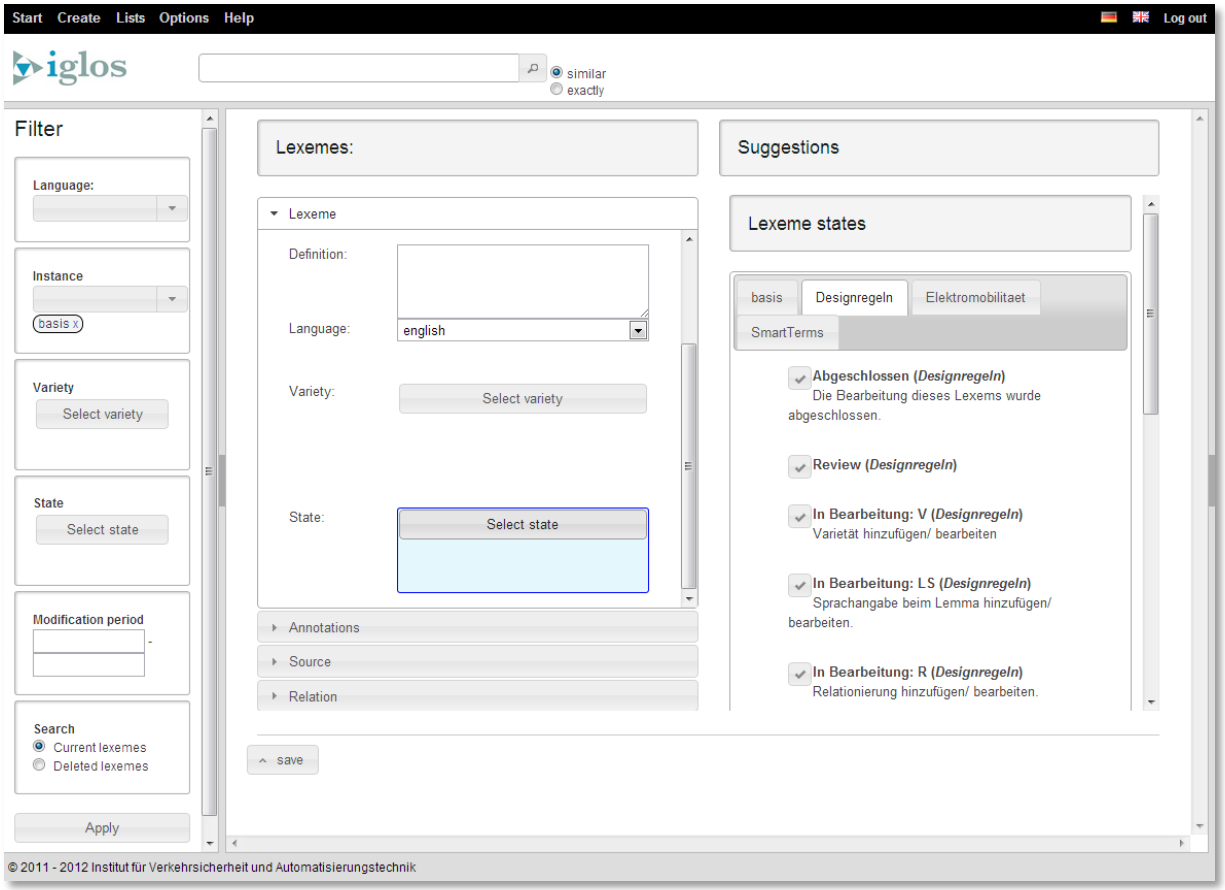

Illustration 11 Lexeme Assistant, tab *Lexeme*, selecting the state of a lexeme

3. Enter notes that supplement the definition of the lexeme to the tab *Annotations*.

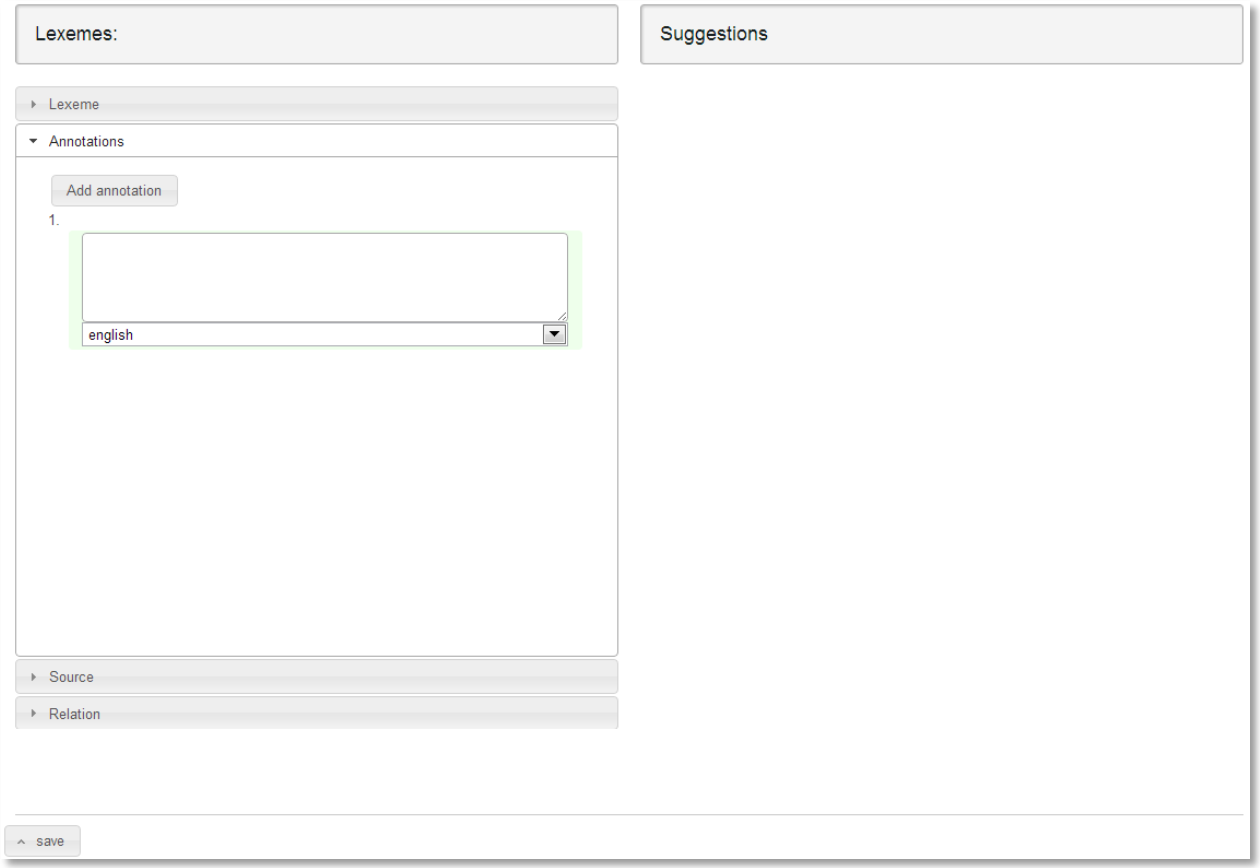

Illustration 12 Lexeme Assistant, tab *Annotations*

4. The tab *Source* requires the following data:
   - the source:
     1. Click on *add*.
     2. Click on *Select literary source* to see the list of available sources.
   - page numbers
   - the chapter or passage

5. The tab *Relation* is used to create relations.

   Notice: See more information about adding relations in section How to relate lexemes? (page 29)

6. Save all data by clicking the *save* button.

   Notice: You can delete each singular information related to the current lexeme in the lexeme assistant (e.g. the relation, but not the [related lexeme).

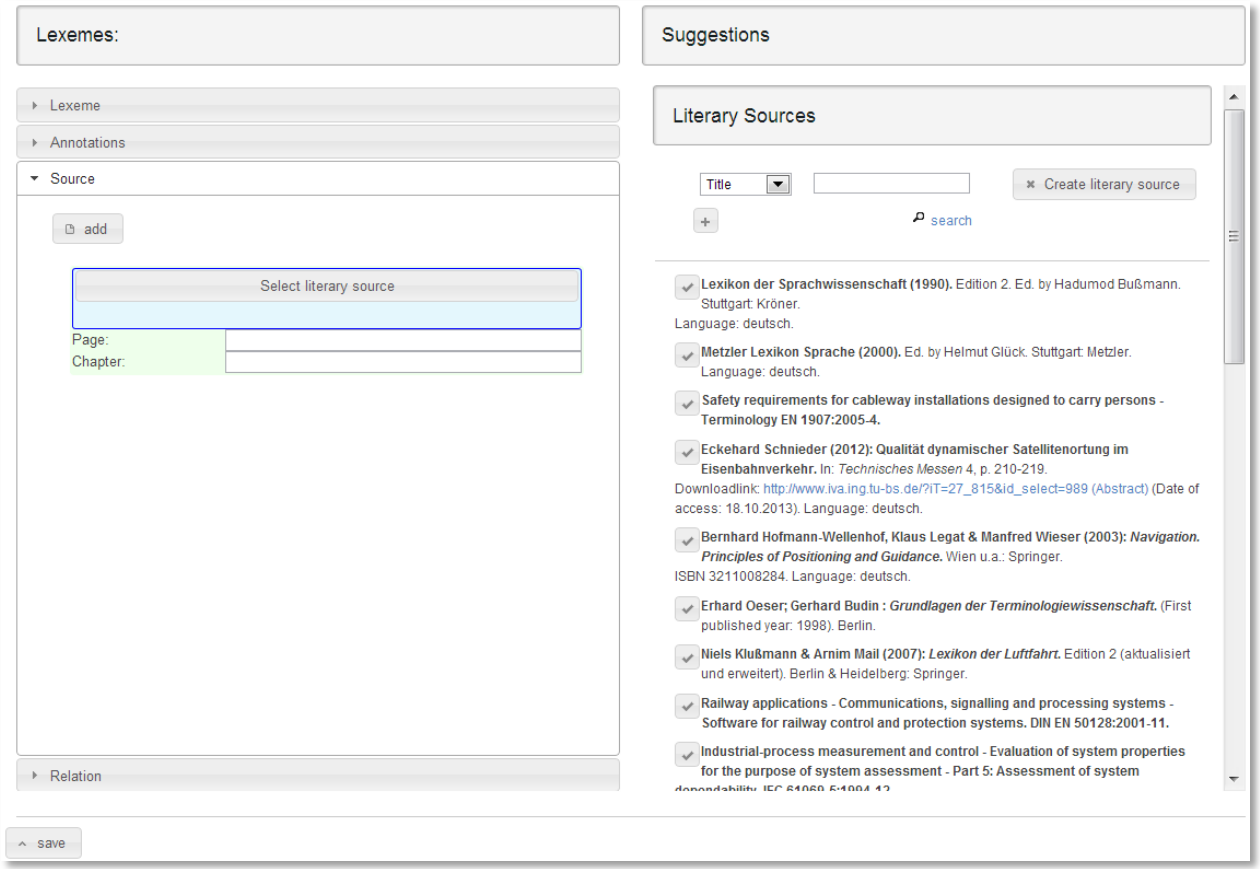

Illustration 13 Lexeme Assistant, tab *Source*

8. How to create a variety

1. Click on *Create* in the menu, then click *Variety*.

   Notice: The lexeme assistant opens in the middle frame.

2. Enter the data of the variety as described for lexemes in section 7 *How to create a lexeme?*

9. How to manage literary sources?

1. Open the list of available sources by clicking on *Lists* in the menu.

Searching for literary sources

1. Search the list of literary sources by selecting relevant fields.
2. Add additional fields using the + button as needed.

   Notice: If there are no results searching for *author,* you may try *editor*.

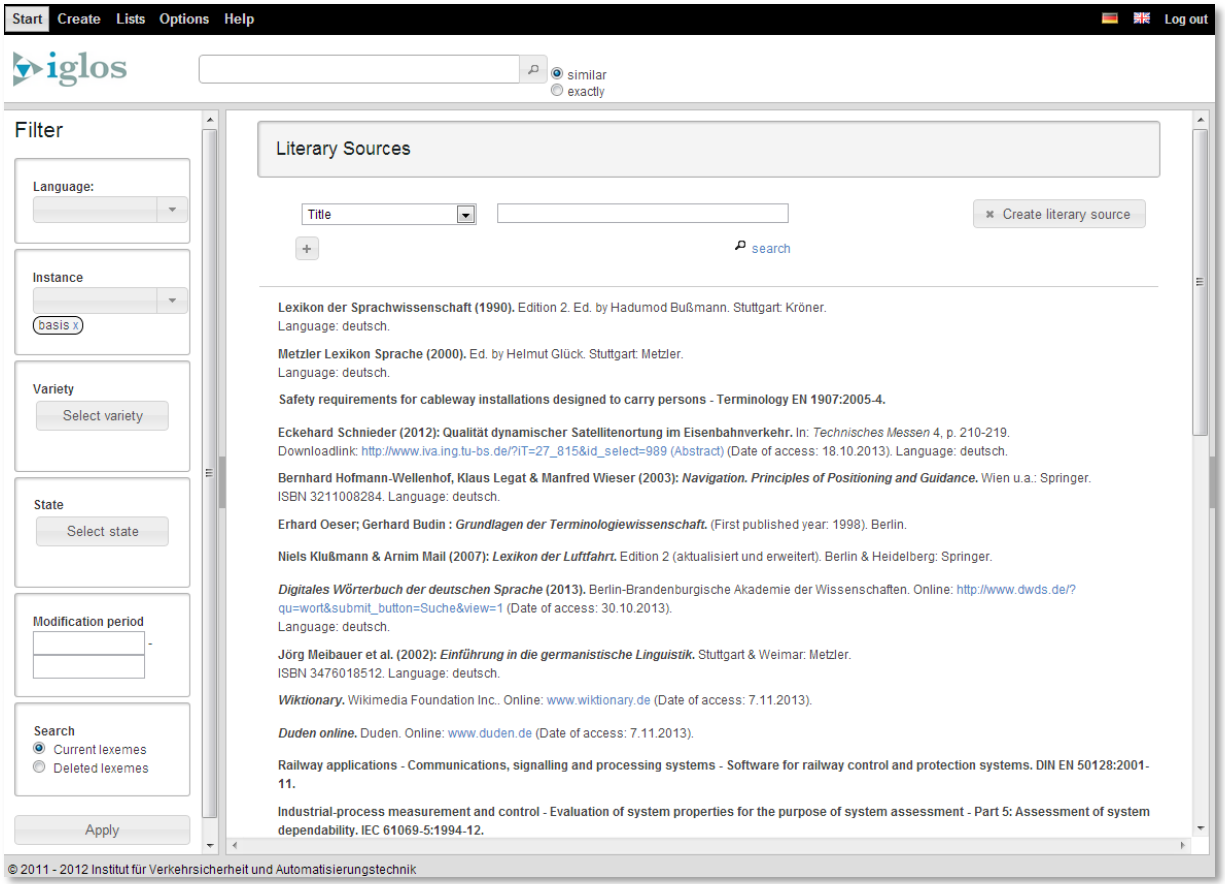

Illustration 14 List of literary sources

Add new literary sources

1. Click on *Create literary source*.
2. Select the type of the publication by using the drop-down menu *Publication type*.
3. Fill in at least the text fields marked as *required*.
4. Save the source.

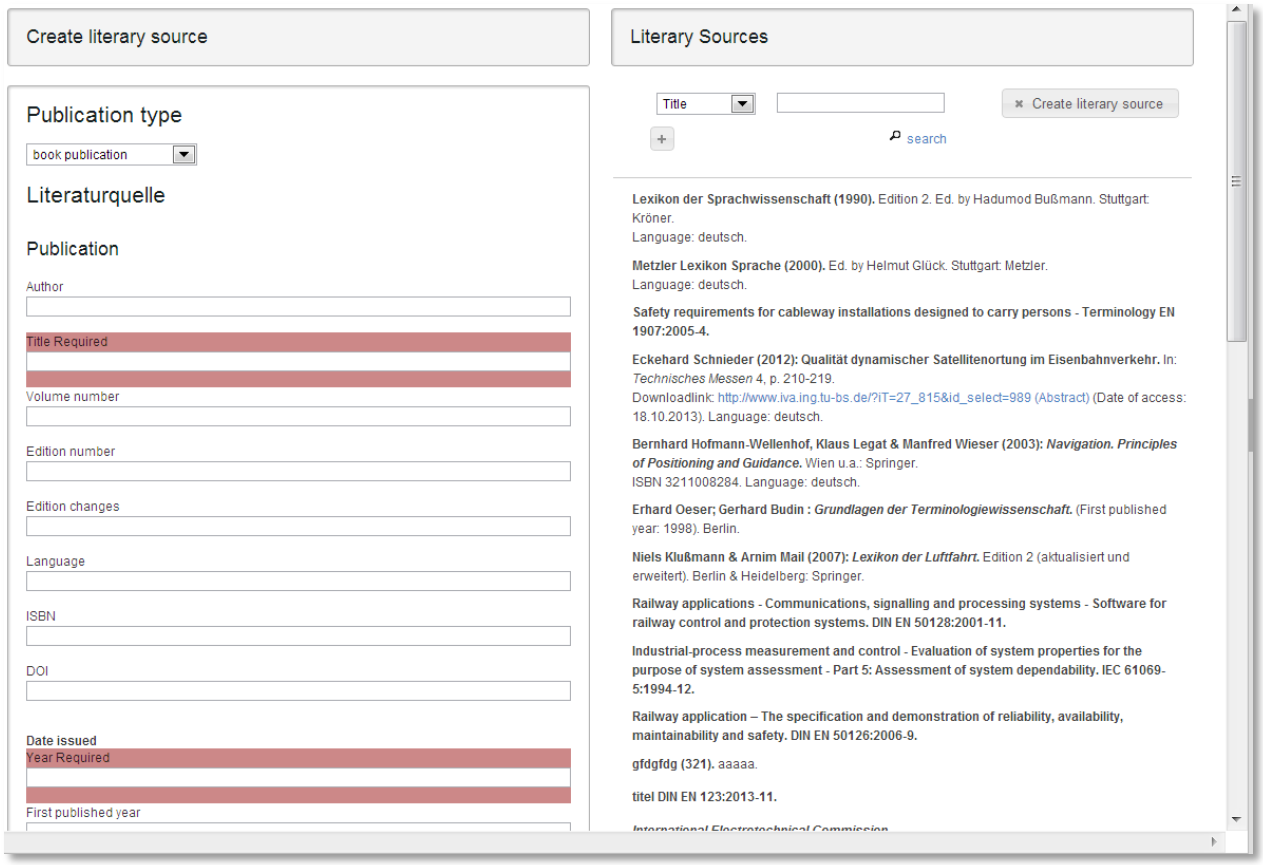

Illustration 15 Adding literary sources

10. What is a lexeme relation?

Lexemes never stand for themselves: In most cases, they refer to each other. A lexeme relation always consists of exactly two lexemes: One of the lexemes performs the role of a subject, while the other performs the role of an object. The connection between the two lexemes is realized by specifying a relation type.

In *iglos* there is a number of different relation types (see Illustration 17 *iglos* relation types on page 28). These relation types can be used to relate lexemes and are provided by the basis instance.

The individual relation types can be viewed in *iglos* under the menu item *Lists* > Relation types.

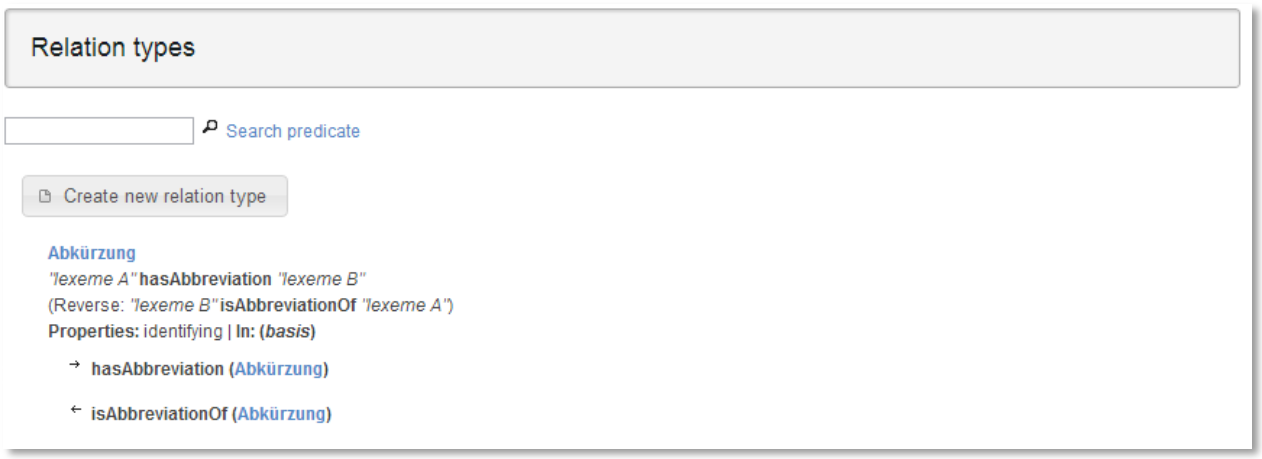

Illustration 16 Example for a relation type

Each relation type contains a **describing lexeme**, which defines the relation type.

Each relation type has a unique ID consisting of a short English description. These start with *is*… or *has*….

The relation types are determined by logical properties:

- **Symmetric** relation types are their own inverse function, which means they apply to both directions (from the subject lexeme to the object lexeme and from the object lexeme to the subject lexeme).
- Some relations are not symmetric and do therefore not apply to both directions. Such relation types need an **inverse function**, which is able to describe the relation when subject and object are inversed.
- Some relation types are **transitive**, i.e. two lexeme relations can imply a third one. If a relation type is transitive and is applied by two relations (x,y) and (y,z), then it can be implied that there is a relation (x,z) which applies that relation type as well.

It is also possible to add new relation types to a terminology. These are stored in the user's instance (see *How to create relation* types? on page 32).

More Information:

What is an instance? (p. 11)
How to relate lexemes? (p. 29)

The following pages provide a description of the different relation types:

Relation is the relation type, which is superior to all other types of relations. It is an unspecified connection between lexemes. ***Relation*** simply indicates that there is a connection existing between two lexemes, but not of which sort it is. By the subordinate relation types, this relation can be further specified. If there are uncertainties concerning the type of relation between two lexemes, but simultaneously assurance about the existence of a relation, this may be indicated by ***relation***.
(ID: hasRelation) [Properties: -transitive, +symmetric, - identifying]

All more concrete relation types of the instance **basis** are classified into elementary and specific relation types:

a) Elementary relation types serve as the first relationing of lexemes and the construction of a rough terminological structure that relates lexemes linguistically and conceptually. This addresses the question in which context lexemes stand and whether there are lexemes with similar meanings or similar designations. First normative regulations can be made with these elementary relation types. Elementary relation types are supposed to facilitate the first steps in terminology work.
b) Specific relation types can be used to give a more accurate model of the terminological system. Some of the specific relation types specifiy the elementary relation types so that the terminology can be modelled more precise. Furthermore, there are relation types in this group which help to make factual relations and world knowledge explicit within the terminology.

Elementary relation types

Abbreviation is an elementary relation type. An ***abbreviation*** is a short form that can be used instead of a designation.

E.g.:
etc. isAbbreviationOf *et cetera*
NATO isAbbreviationOf *North Atlantic Treaty Organization*

(ID: hasAbbreviation, isAbbreviationOf) [Properties: -transitive, -symmetric, +identifying]

Translation is an elementary relation type. Two lexemes are in ***translational relation*** when they are considered to be equivalents in two natural languages and therefore variety and meaning are identical. If this is not the case and there are preferences or reservations towards a particular translation candidate, see also ***rejected translation***, ***approved translation*** and ***mistranslation***.
(ID: hasTranslation) [Properties: + transitiv, +symmetrisch, + identifying]

Risk of confusion is a elementary relation type. Two lexemes are in ***risk of confusion*** when they have the same denomination, but their meanings are different or only similar but not identical to each other. ***Risk of confusion*** is specified by the specific relation types ***polysemy*** and ***homonymy***.
(ID: isMixesUpWith) [Properties: -transitive, +symmetrical, -identifying]

Context is an elementary relation type. ***Context*** is the relation of a lexeme to another lexeme of the same context. This serves to show the topical relations of a lexeme.

E.g.:
virus hasContext *human body*
virus hasContext *computer*

(ID: hasContext; isContextOf) [Properties: -transitive, -symmetrical, -identifying]

Opposition is an elementary relation type. Two lexemes are ***opposites*** when their meanings are contrary to each other. Their meaning relation can be characterised by incompatibility or by a reversed perspective of a circumstance. ***Opposition*** is specified by the more specific relation types ***complementarity***, ***antonymy*** and ***converseness***.
(ID: hasOpposite) [Eigenschaft: -transitive, +symmetrical, -identifying]

Approved lexeme is an elementary relation type. The relation shows which of two possible lexemes needs to be preferred and which needs to be rejected.
(ID: hasPreference; isPreferenceOf) [Properties: -transitive, -symmetrical, -identifying]

Hierarchy relation is an elementary relation type. Two lexemes are in hierarchical relation when one is superior over the other. ***Hierarchy relation*** can be specified by ***hyperonymy*** and ***holonymy***.
(ID: hasInferior; isInferiorOf) [Properties: +transitive, -symmetrical, -identifying]

Meaning identity is an elementary relation type. Two lexemes are in relation of ***meaning identity*** when they refer to exactly the same concept and do not mean more or less than the other lexeme. It is possible, however, that these lexemes either differ in designation or in definition. ***Meaning identity*** is specified by ***synonymy*** and ***equivalence of definition***.
(ID: hasSameMeaningAs) [Properties: +transitive, +symmetrical, +identifying]

Specific relation types

Mistranslation is a specific *translation* relation type. ***Mistranslation*** relates lexemes which are by many speakers assumed to be translations of each other because of their formal similarity. This assumption and thus the translation, however, is not valid since the meanings of the two lexemes completely differ from each other.

E.g.:

gift (en) hasFalseFriend *Gift* (de, „poison“)

(ID: hasFalseFriend) [Properties: +transitive, +symmetrical, -identifying]

Approved translation is a specific *translation* relation type. It identifies one of the possible or commonly used translations as the ***preferred translation***.
(ID: hasPreferredTranslation) [Properties: -transitive, +symmetrical, +identifying]

Rejected translation is a specific *translation* relation type. It identifies one of the possible or commonly used translations as unwanted.
(ID: hasForbiddenTranslation) [Properties: -transitive, +symmetrical, +identifying]

Polysemy is a specific *risk of confusion* relation type. Two lexemes are ***polysemes*** of each other when their meanings partially differ from each other while their designations are the same. In contrast to *equivalence of definition*, this is not just based on formality but on meaning. The difference in meaning may be strong or weak but always partial, which means that a common core of meaning always exists. ***Polysemy*** thus also differs from the relation type *homonymy* where two lexemes have totally different meanings but a common denomination, (designation A = designation B, meaning A ≠ meaning B, BUT: core of meaning A = core of meaning B)

E.g.:
screw hasPolysem *screw*
(ID: hasPolysem) [Properties: +transitive, +symmetrical, -identifying]

Homonymy is a specific *risk of confusion* relation type. Two lexemes are ***homonymous*** when they have identical designations but completely different meanings. This characteristic distinguishes ***homonymy*** from *polysemy* where a common core of meaning exists, (designation A = designation B, meaning A ≠ meaning B AND: core of meaning A ≠ core of meaning B)

E.g.:
bank hasHomonym *bank*
(ID: hasHomonym) [Properties: +transitive, +symmetrical, -identifying]

Complementarity is a specific *opposition* relation type. Two lexemes are ***complementary*** when they are incompatible. One of these lexemes is always true but never both at the same time.

E.g.:
dead hasComplement *alive*
(ID: hasComplement) [Properties: -transitive, +symmetrical, -identifying]

Antonymy is a specific *opposition* relation type. ***Antonyms*** stand in *scalar opposition* by denoting end poles of a gradual scale. When one of the lexemes is not true this does not necessarily mean the other lexeme is true.

E.g.:
hot hasAntonym *cold*
The weather is not cold. ≠ The weather is hot.
(ID: hasAntonym) [Properties: -transitive, +symmetrical, -identifying]

Converseness is a specific *opposition* relation type. Two lexemes are reversed perspectives of each other when they express a relative relation. The chosen perspective determines which of the lexemes is used.

E.g.:
*The house is **in front of** the garden.*
*The garden is **behind** the house.*
in front of hasConvers *behind*
The described circumstance is the same in both cases.
(ID: hasConvers) [Properties: -transitive, +symmetrical, -identifying]

Hyperonymy is a specific *hierarchy* relation type. Two lexemes are in ***hyperonymy*** when they describe entities that have different degrees of abstraction. The lexeme which is superordinate (= *hypernym*) is less specific than the lexeme which is subordinate (= *hyponym*). The superordinate includes the subordinate lexeme.

E.g.:
flower isHyperonymOf *rose*, *tulip*, *daffodil*, etc.
Alsatian, *dachshund*, *poodle*, *mastiff*, etc. hasHyperonym *dog*

Whether a relation is a hyperonymy can be tested by asking:

Is X a kind of Y? (Y isHyperonymOf X / X hasHyperonym Y)

Is an *Alsation* a kind of *dog*? → *hyperonymy*

(ID: hasHyperonym; isHyperonymOf) [Properties: +transitive, -symmetrical, -identifying]

Holonymy is a specific *hierarchy* relation type. Two lexemes are in ***holonymy relation*** when one of them describes an entity (= *holonym*) and the other one an entity which describes a part of it (= *meronym*).

E.g.:

head, *neck*, *shoulder*, *chest*, *arm*, *finger* etc. isPartOf *body*

body hasPart *head*, *neck*, *shoulder*, *chest*, *arm*, *finger* etc.

Whether a ***holonymy relation*** exists can be tested by asking:

Is X a part of Y?/ Does Y have a part designated X?

Is *head* a part of the *body?*/ Does *body* have a part designated as *head*? → ***holonymy***

Is *rose* a part of *flower*/ Does *flower* have a part denominated as *rose*? ≠ ***holonymy***

(ID: hasPart; isPartOf) [Properties: +transitive, -symmetrical, -identifying]

Synonymy is a specific *meaning identity* relation type. Two lexemes are ***synonym*** when they have different designations but identical meaning, (designation A ≠ designation B, meaning A = meaning B)

E.g.:

quickly hasSynonym *speedily*

(ID: hasSynonym) [Properties: +transitive, +symmetrical, +identifying]

Equivalence of definition is a specific *meaning identity* relation type. Two lexemes are ***equivalent of definition*** when the texts of their definitions differ only in formulations but not in their general content. Thus they have the same meaning and the same denomination, (designation A = designation B, meaning A = meaning B, BUT: definition A ≠ definition B)

(ID: hasEquivalentDefinition) [Properties: +transitive, +symmetrical, +identifying]

Output is a specific relation type. Two lexemes are in ***output relation*** when one of them describes an entity which produces another entity as its product and result. This other entity is the ***output***.

E.g.:

coffee machine hasOutput *coffee*

(ID: hasOutput; isOutputOf) [Properties: -transitive, -symmetrical, -identifying]

Input is a specific relation type. Two lexemes are in ***input relation*** when one of them describes an entity that processes another entity. This other entity serves as the ***input***.

E.g.:

coffee machine hasInput *coffee powder*

(ID: hasInput; isInputOf) [Properties: -transitive, -symmetrical, -identifying]

Function is a specific relation type. Two lexemes are in ***function relation*** when they each designate an entity and a purpose of this entity.

E.g.:

Calculator hasFunction *calculate*

(ID: hasFunction; isFunctionOf) [Properties: +transitive, -symmetrical, -identifying]

Sequence is a specific relation type. Two lexemes are in ***sequence relation*** when they designate entities, which appear in temporal succession. Sequence relation should only be used for elements that directly follow each other. By using sequence a timeline can be generated.

E.g.:

January hasSuccessor *February* hasSuccessor *March*

(ID: hasSuccessor; hasPredecessor) [Properties: -transitive, -symmetrical, -identifying]

State is a specific relation type. Two lexemes are in ***state relation*** when they each designate an entity and a potential state of this entity. An engine, for example, can be in the *operating state*. When an entity has several potential states the state will be denominated by an abstract generic term which classifies the single potential states the entity can take.

E.g.:

water hasState *state of matter*

water hasState *purity grade*

(ID: hasState; isStateOf) [Properties: -transitive, -symmetrical, -identifying]

Causality is a specific relation type. Two lexemes are in ***causal relation*** when one of them designates a result whose cause is described by the other lexeme.

(ID: hasCause; isCauseOf) [Properties: +transitive, -symmetrical, - identifying]

Individual occurrence is a specific *hyperonymy* relation type. It relates a lexeme describing a class or a category with a lexeme which is unique individual of that class.

E.g.:

physicist hasInstantiation *Albert Einstein*

(ID: hasInstantiation; isInstantiationOf) [Eigenschaften: -transitive, -symmetrical, -identifiying]

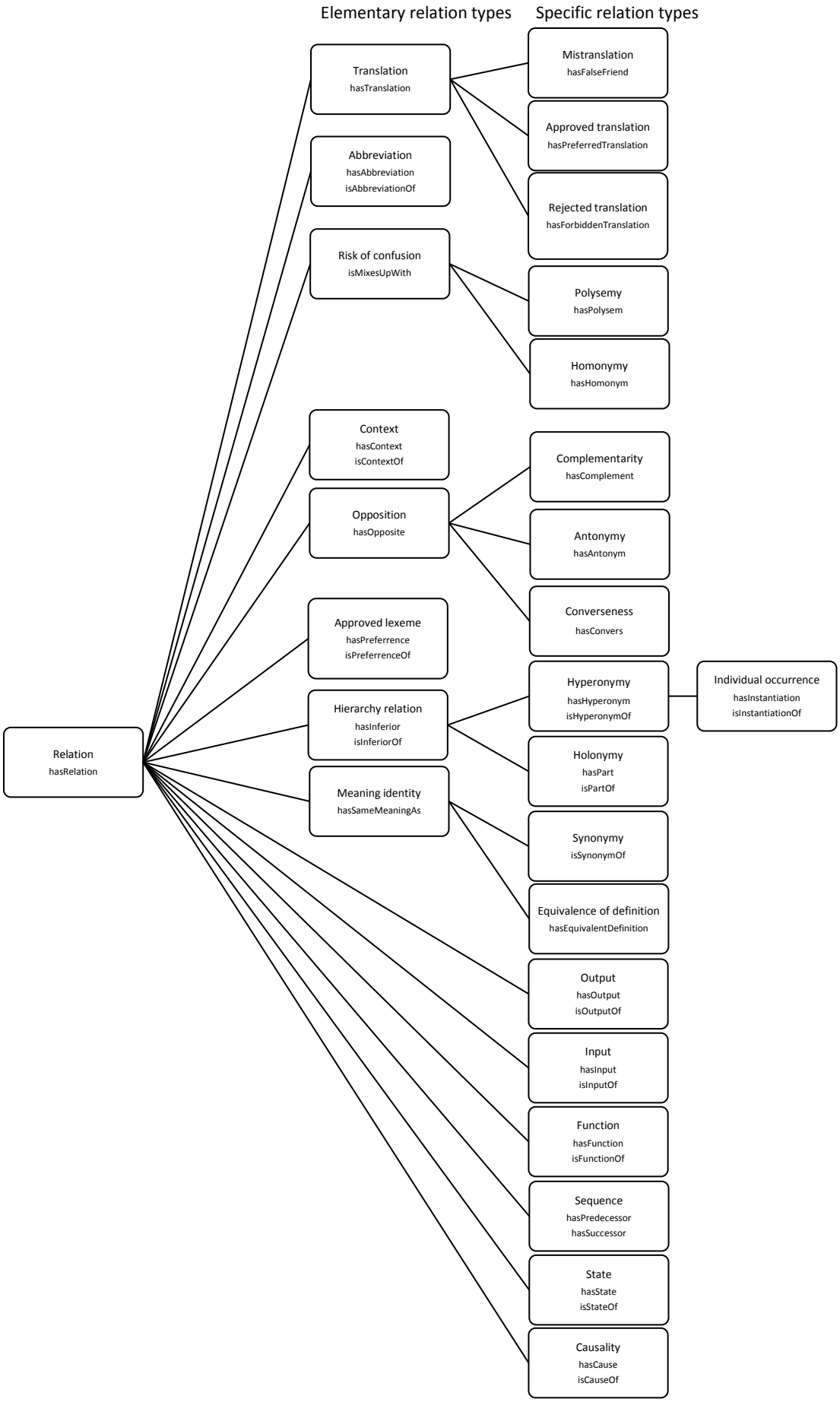

Illustration 17 *iglos* relation types

11. How to relate lexemes?

Lexemes can be related in two different ways:

A) Using the lexeme assistant

1. Open the lexeme assistant for a new lexeme.

   or:

   Open the lexeme assistant for an available lexeme.
2. Open the tab *Relation* in the Lexeme assistant.
3. Click the *add* button to add a relation.
4. Select the predicate of the relation in the relation type list. You may search for the description of the predicate (e.g. hasAbbreviation) or the describing lexeme in German (e.g. Abkürzung), if needed. Select the predicate by clicking the check mark button.

   Notice: A description of a relation type is available by clicking on the describing lexeme in German. This opens the full entry where you can also load descriptions in other languages.

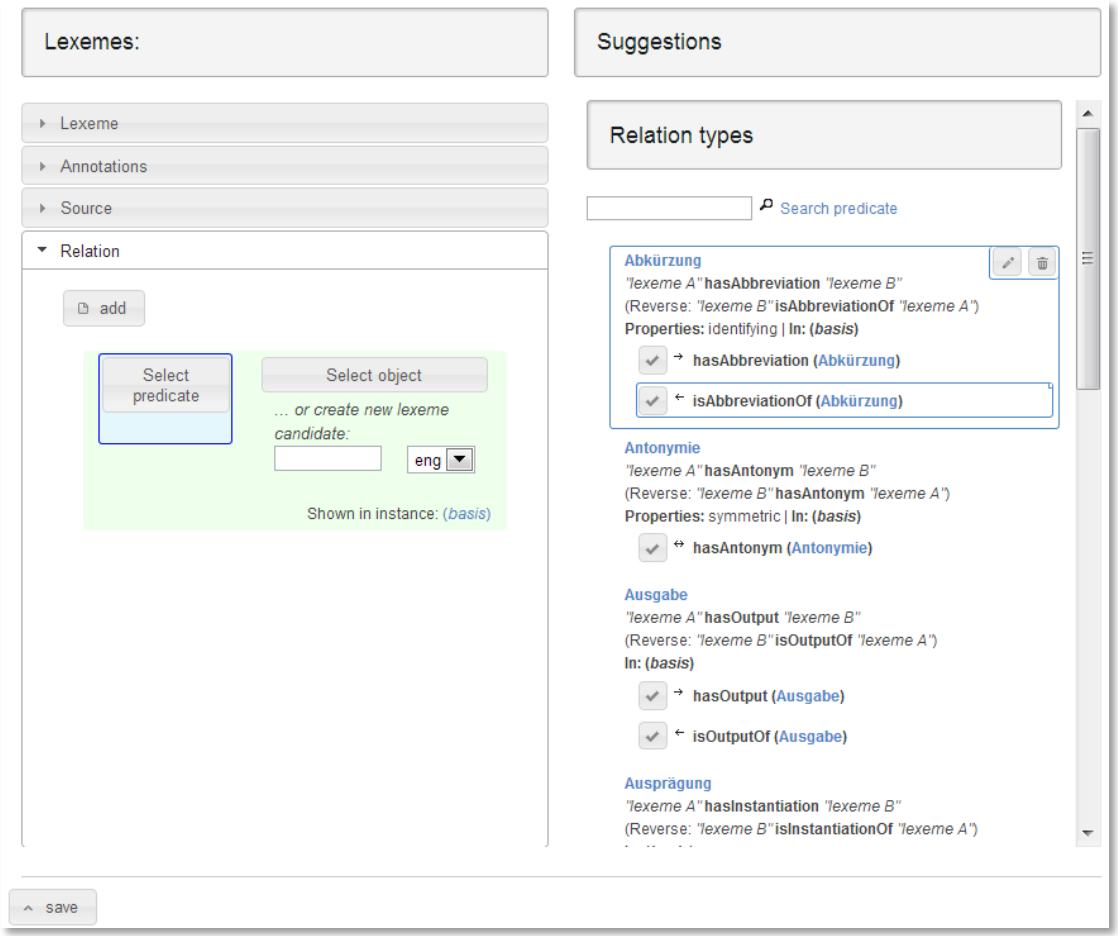

Illustration 18 Selecting a predicate to relate lexemes

5. Click on *Select object*. If necessary, you may search the suggestion list for the required lexeme. Select the required lexeme by clicking on the check mark button.

   Notice: If the required lexeme does not yet exist, you can enter the designation of a lexeme candidate to the text box instead. It will be created by saving the current lexeme.

6. Save the relation by clicking on *save*.

   Notice: This way you also save all other changes you have made!

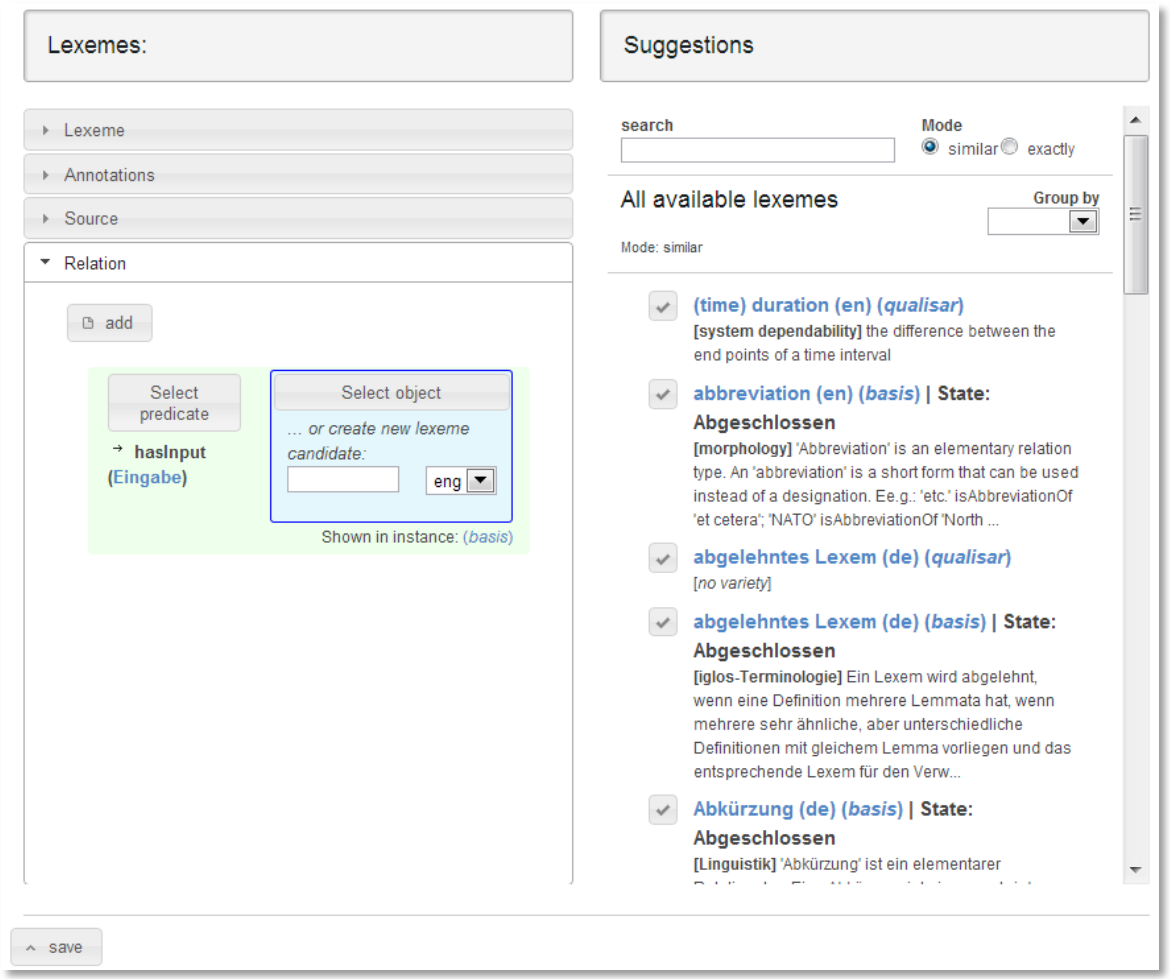

Illustration 19 Selecting a lexeme for creating a relation

B) Using the visualisation

Notice: The lexeme to be related needs to be part of a collection!

1. Visualise the Collection which contains the lexeme to be related.
2. Click the button *Relation* (Illustration 20). The cursor then appears as a cross in the visualisation.
3. Click on the subject lexeme.
4. Drag the cursor to the object lexeme. Release the mouse button at the position of the object lexeme.
5. Select the predicate of the relation in the appearing pop-up.
6. Set the instance where the relation shall be stored.
7. Save the relation.

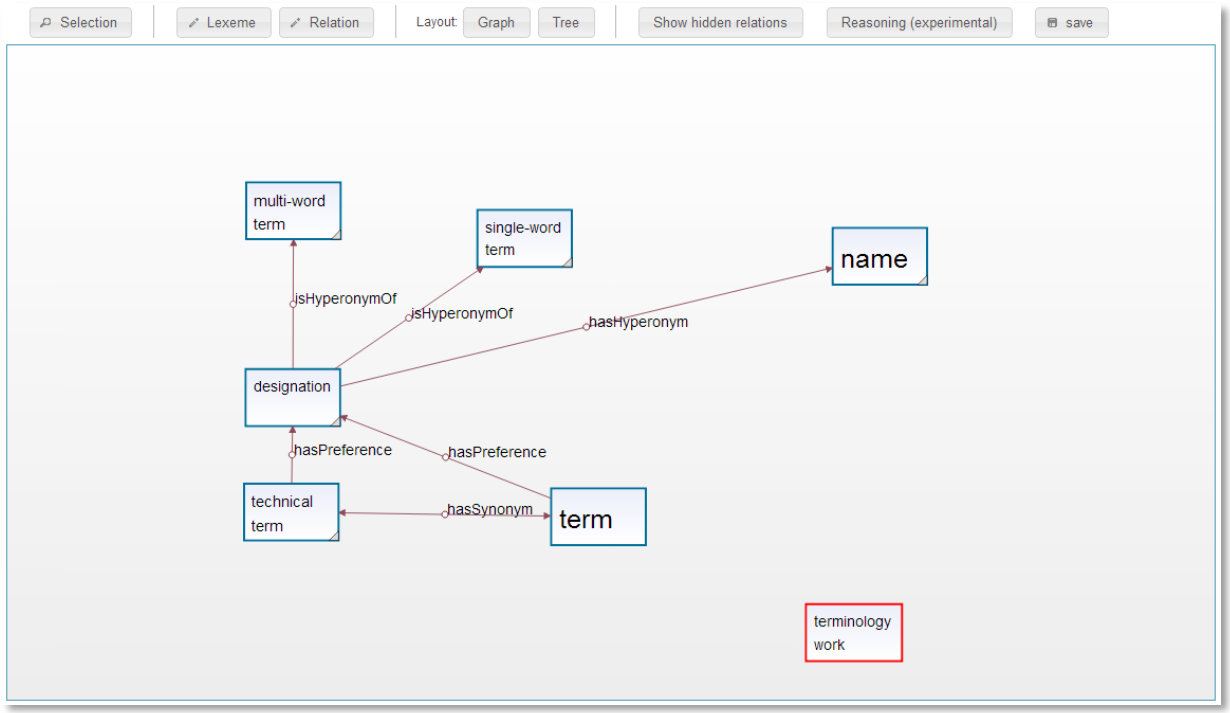

Illustration 20 Visualisation

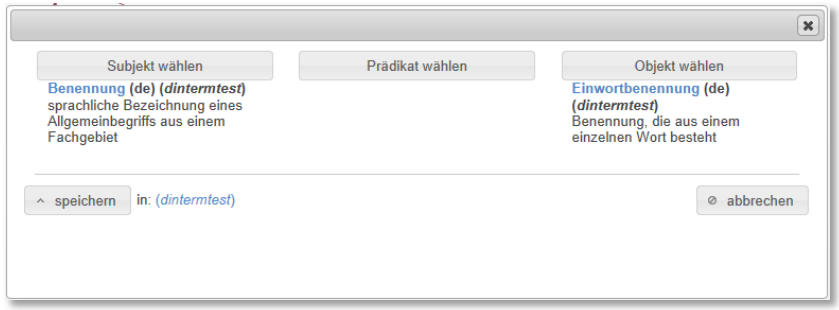

Illustration 21 Pop-up to define a relation type

12. How to create relation types?

1. Click on *Create* in the menu, then click *Relation type*.
2. Enter a describing lexeme by typing its designation into the upmost text box on the left. A pop-up with suggestions will open. Select the lexeme by clicking the check mark button.

   Notice: If a suitable lexeme does not exist yet, you can enter the designation of a lexeme candidate into the text box instead. It will be created by saving the current lexeme.

3. Enter the designation for the predicate of the relation type.
4. Enter the designation of the reverse predicate.
5. Define the properties of the lexeme.
6. Set the options for the visualisation of the relation type in the tree layout.
7. Set the instance where the relation type shall be stored.
8. Save the relation type.

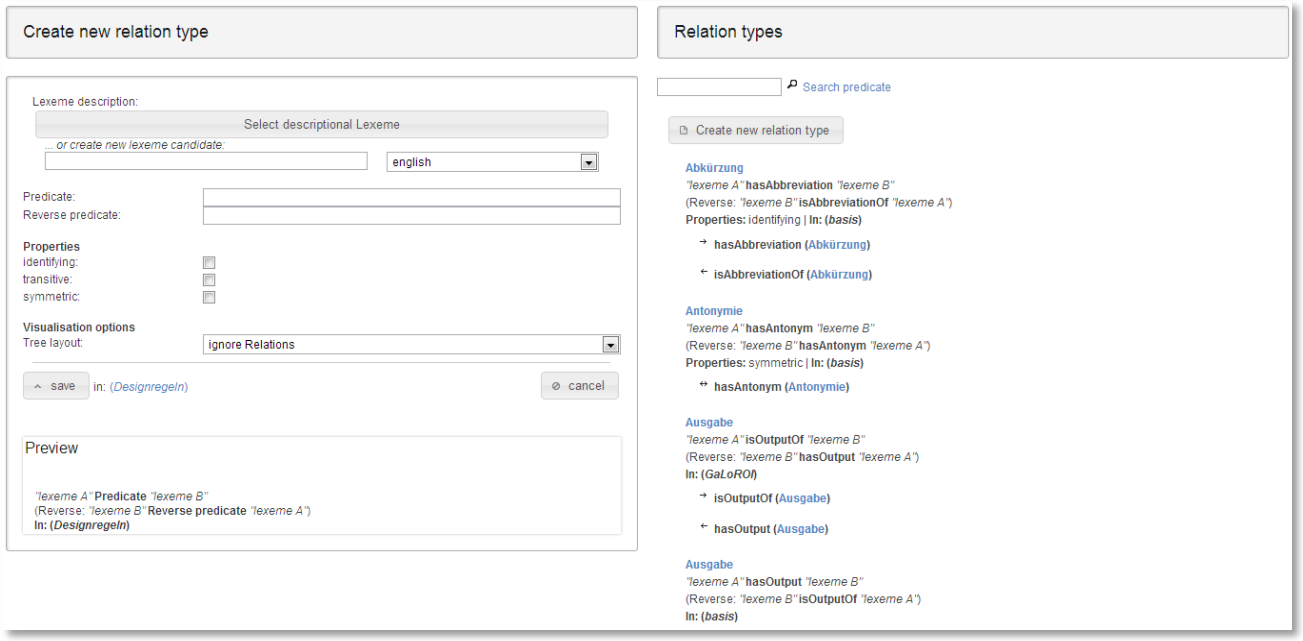

Illustration 22 Creating a relation type

13. How to create instances?

In order to create new instances please contact the technical support of *iglos*:

E-Mail: info@iglos.de
Tel.: +49 (0) 531 391 3307

14. How to edit lexemes?

1. Search for the lexeme to be edited by using the search box.
2. Click on the *Edit* button on the upper right edge of the lexeme.
3. Change the data of the lexeme as described in section 7 *How to create a lexeme*? (page 15).

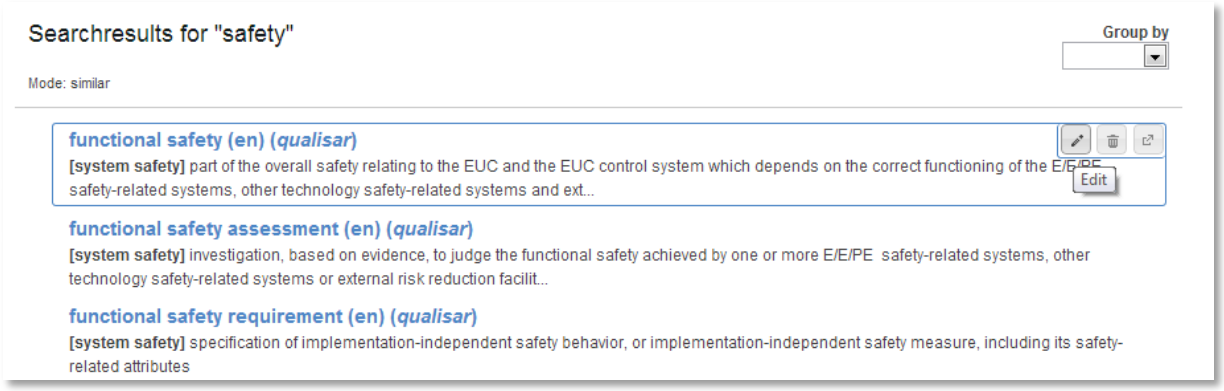

Illustration 23 Searching result, preview

15. What is a collection?

Collections are lists that will help you to arrange excerpts of a terminology for different target groups and uses. Collections can contain lexemes and other collections. They can be visualised and modelled graphically.

By default each user has a *watchlist* which is always opened. It can be closed, opened, edited and also be deleted.

As a user you can create further collections with a freely chosen name and a description.
Collections can be filled with all lexemes for which you have read access.
Collections can be shared with other registered *iglos* users. The recipient of a collection needs to have read access to the collection element. If read access for that user is not available the collection element will be represented by a subsitute representation.

16. How to use collections?

Load the list of collections

1. Click on *Lists* in the menu.
2. Click on *Collections*.

Create a collection (Illustration 25):

1. Click on *New Collection*.
2. Enter a designation.
3. Enter a description.
4. Save the collection.

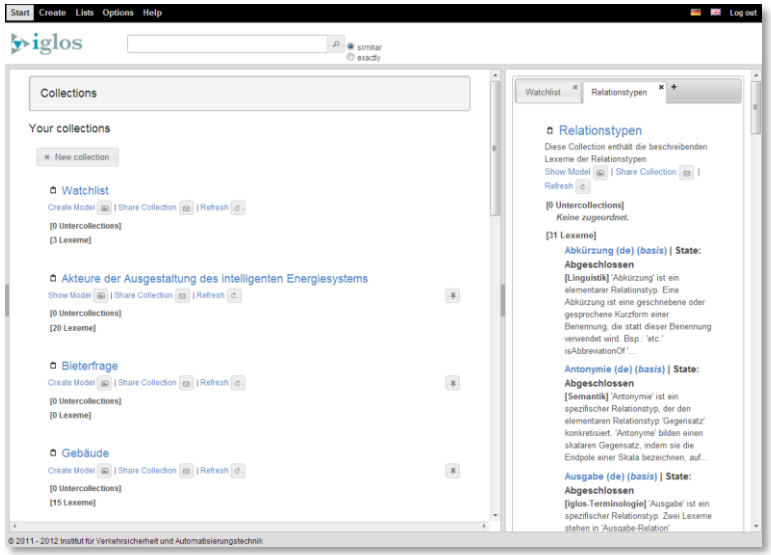

Illustration 24 Collection list

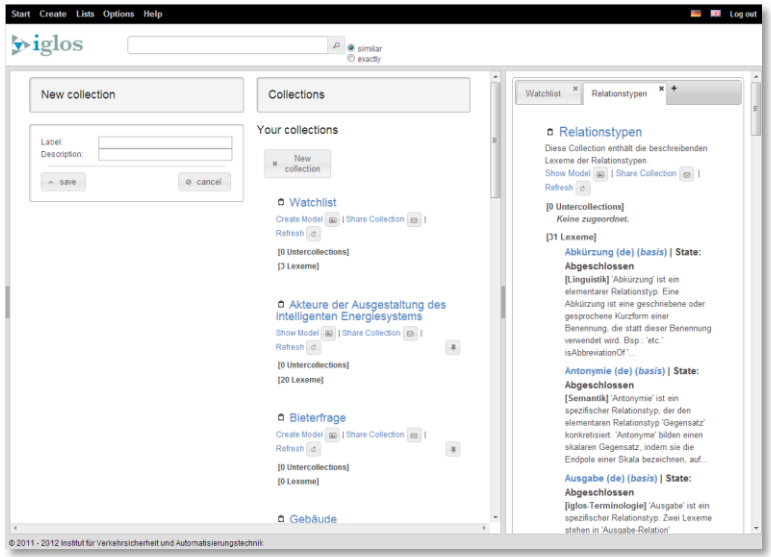

Illustration 25 Creating a collection

How to use collection buttons

	Edit collection Edit the name and the description of the collection.
	Delete Delete the collection.
	Component full view Load the collection with its elements or a lexeme contained in the collection in the middle frame.
	Visualisation Load the collection in the *iglos*-visualisation.
	Open collection in tab Open the collection in the right frame.
	Remove Remove a subordinate collection or a lexeme from a collection.
	Share Collection Share a collection by sending a mail to a registered *iglos* user.
	Open/Close Open and close the list of collections.
Designation	Click on a designation, to open and close the lexeme.
Lexeme box	Click the lexeme box in the collection to highlight the lexeme. By drag & drop you can move it into other collections or highlighted form fields.
Object designation	Click on a designation of an object to load the object of a lexeme relation in the middle frame.

How to use collection tabs

Notice: In the right frame you will find the collection tabs. Here you can open your collections and drag search results listed in the middle frame or elements of collections to other collections. Switch collections by clicking on the tabs.

Open a collection in a tab:

1. Open the list of collections.
2. Open the collection in a tab by clicking on .

Adding elements to a collection:

1. Search the required element using the search box, if necessary by applying the filter options.

   or:

   Load the list of collections by using the menu.

2. Click on the required element and drag it into the collection opened in the collection tab.

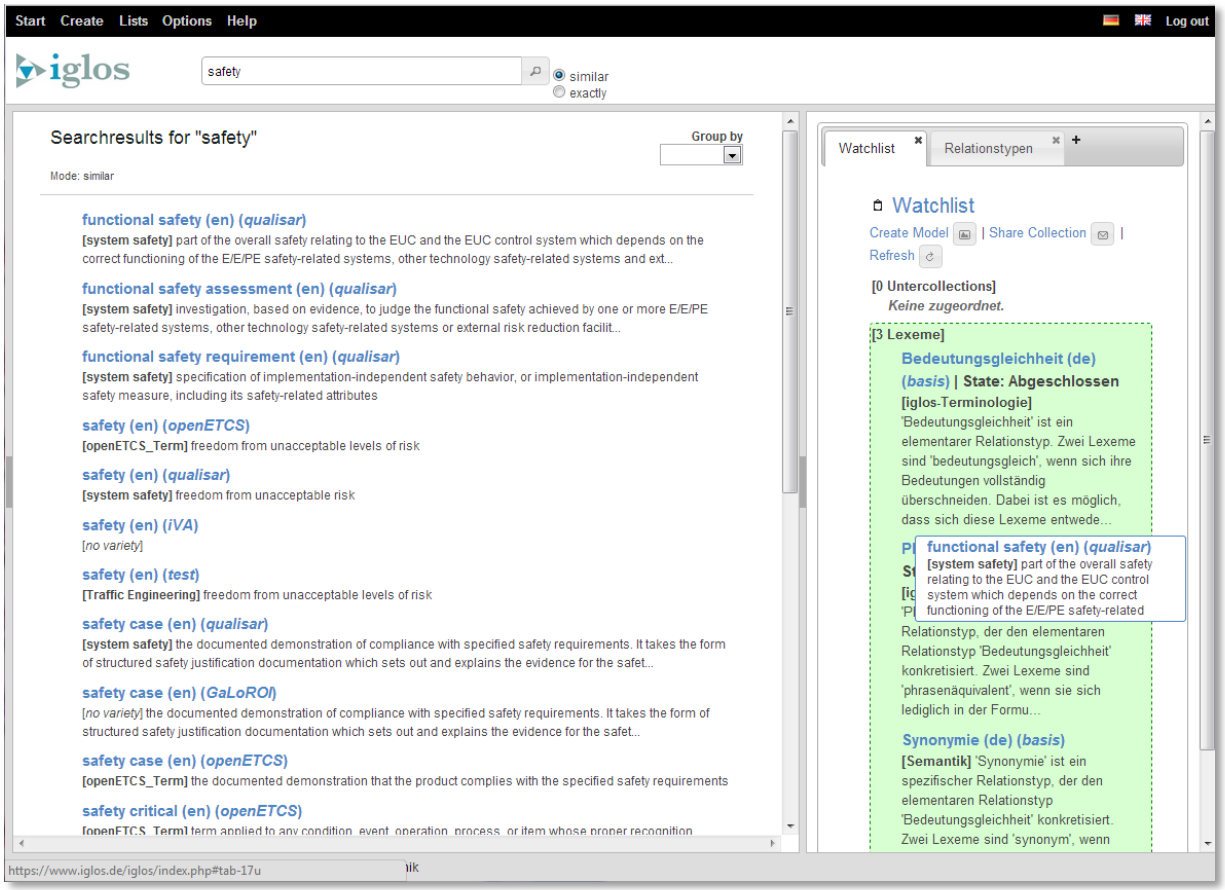

Illustration 26 Adding lexemes to a collection in collection tabs using drag & drop

17. What is an *iglos* visualisation?

The *iglos* visualisation is a graphical representation of collections. However, the visualisation is not only representing the lexemes, but also the relations between lexemes, which have been defined by relationing.

By default the visualisation represents the terminology system as a graph, the visualisation can also be transformed into a tree layout. Hierarchical relations can thus be adequately represented as well.

This facilitates access to a topic. Additionally the visualisation allows graphical modeling of special language items that represent specialised knowledge items.

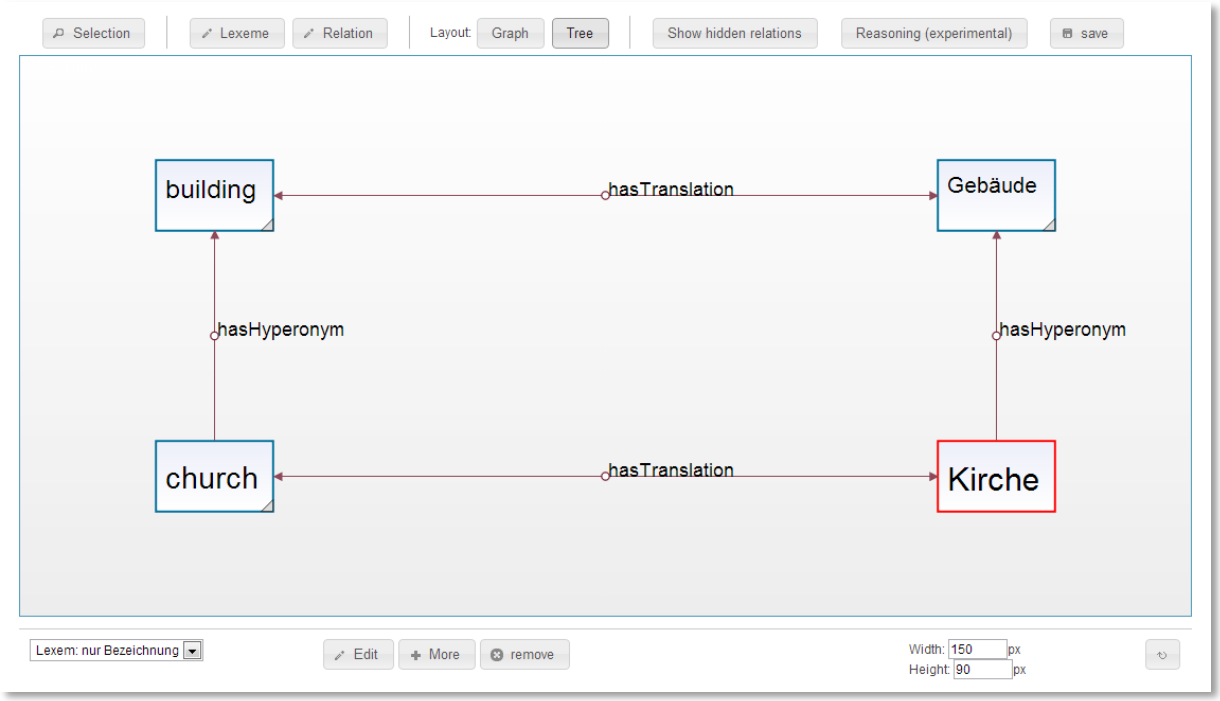

Illustration 27 Visualisation of a terminology

The content of a collection, as well as the layout of its visualisation can be edited individually to optimize the optical comprehensibility of complex concepts (see section *How to use the iglos visualisation*? on page 40). A layout specified by the user can be saved for the visualization of the corresponding collection as well. To do so click on *save* in the context menu of the collection.

18. How to use the *iglos* visualisation?

Notice: To use the visualisation the application of a mouse is essential.

Zoom

1. Place the cursor on the position you like to zoom.
2. Turn the mouse wheel to zoom.

Selection of multiple lexemes

1. Click on *Selection*.
2. Click on the visualisation area using the cursor.
3. Drag the cursor over the lexemes to be selected.

Create a lexeme

1. Click on *Lexeme*.
2. Enter the data of the new lexeme in the Lexeme assistant.

Create a relation

1. Follow the instructions in section 10 *How to relate lexemes?* on page 29.

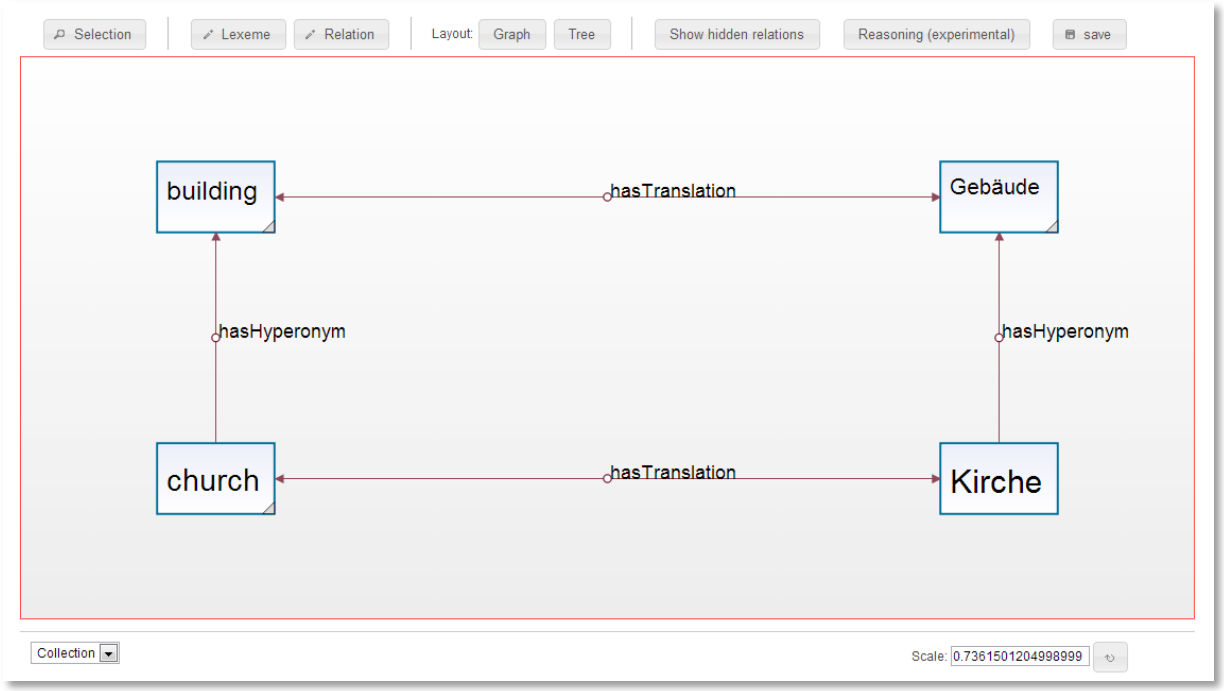

Illustration 28 Visualisation with collection context menu

Set the tree layout

1. Click on *Tree*.

Set the graph layout

1. Click on *Graph*.

Edit lexemes

1. Mark the appropriate lexeme.
2. Click on *Edit* in the context menu of the lexeme (Illustration 29).
3. Change the data of the lexeme in the Lexeme assistant.

Removal of lexemes

1. Mark the lexeme.
2. Click on *remove* in the context menu of the lexeme.

   Notice: This way the lexeme is only removed from the collection. It is not deleted.

Resizing of lexemes

1. Click the lower right edge of the lexeme.
2. Drag the lexeme to the desired size.

or:

1. Mark the lexeme.
2. Enter the width and height in the context menu of the lexeme.

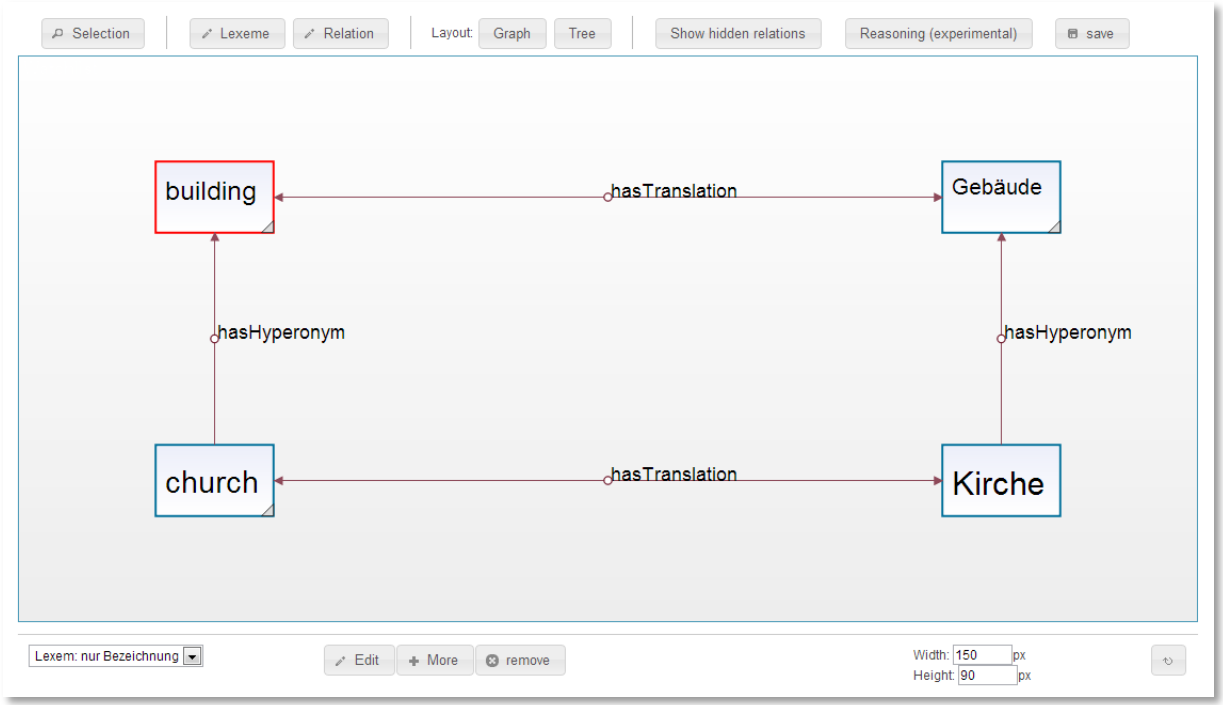

Illustration 29 Visualisation with lexeme context menu

Edit relations

1. Mark the required relation, then click on *Edit* in the context menu of the relation (illustration 30).
2. Change the subject lexeme, the object lexeme or the predicate in the appearing pop-up.

Delete relations

1. Mark the relation to be deleted, then click *Delete* in the context menu of the relation.
2. Click on *Delete* in the appearing pop-up.

   Notice: This way the relation is deleted permanently.

Hide and show relations

1. Mark the relation to be hidden.
2. Click on *Hide* in the context menu of the relation.
3. Click on *Show hidden relations*, to display all hidden relations.

Reverse predicate

1. Mark the corresponding relation.
2. Click on *Reverse predicate*, to reverse the perspective of the relation.

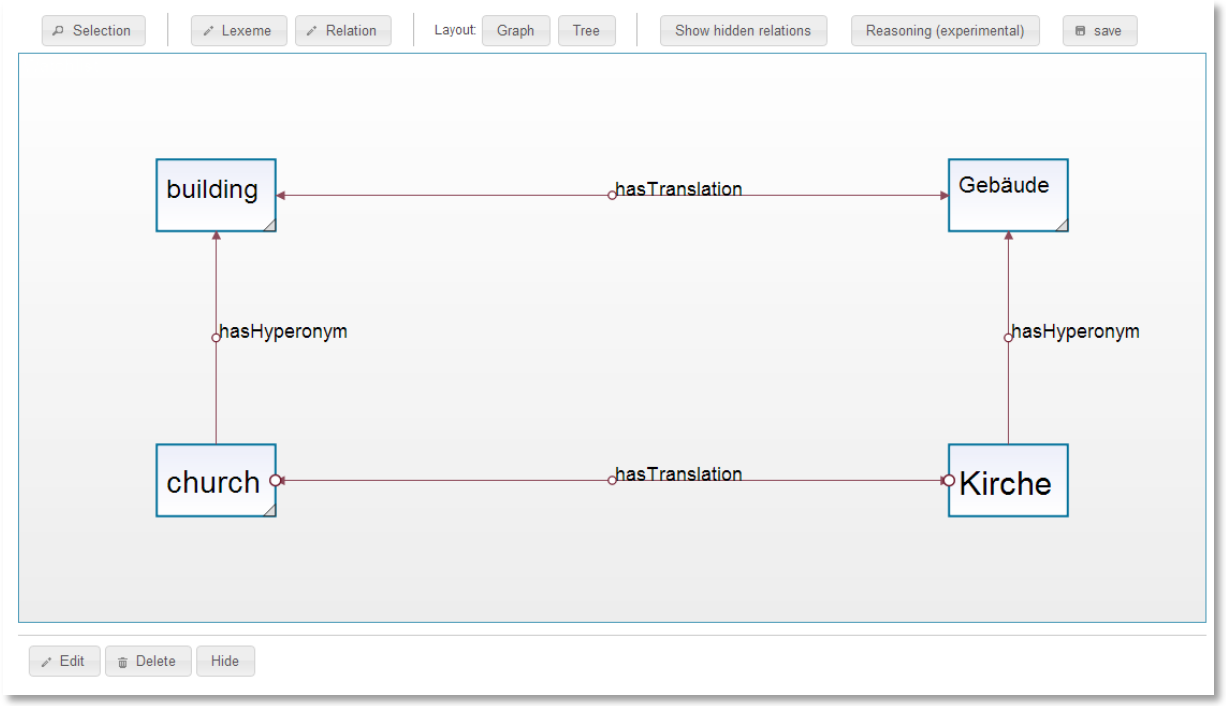

Illustration 30 Visualisation with context menu of a relation

Technische Universität Braunschweig
Institut für Verkehrssicherheit und Automatisierungstechnik

Langer Kamp 8
38106 Braunschweig

Tel. +49 (0) 531 391 3309
Fax +49 (0) 531 391 5197
Web: www.iglos.de
E-Mail: info@iglos.de