## Table des matières

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basics</td>
<td>3</td>
</tr>
<tr>
<td>Installation</td>
<td>3</td>
</tr>
<tr>
<td>User interface basics</td>
<td>3</td>
</tr>
<tr>
<td>Keyboard shortcuts</td>
<td>3</td>
</tr>
<tr>
<td>Knowledge representation</td>
<td>4</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>4</td>
</tr>
<tr>
<td>Overview</td>
<td>5</td>
</tr>
<tr>
<td>Concept types edition</td>
<td>10</td>
</tr>
<tr>
<td>Insertion</td>
<td>11</td>
</tr>
<tr>
<td>Graph arrangement</td>
<td>12</td>
</tr>
<tr>
<td>Concept type hierarchy control</td>
<td>12</td>
</tr>
<tr>
<td>Forbidden types</td>
<td>12</td>
</tr>
<tr>
<td>Concept type alteration</td>
<td>12</td>
</tr>
<tr>
<td>Relation types edition</td>
<td>12</td>
</tr>
<tr>
<td>Insertion</td>
<td>13</td>
</tr>
<tr>
<td>Relation type signatures</td>
<td>13</td>
</tr>
<tr>
<td>Relation type hierarchy control</td>
<td>13</td>
</tr>
<tr>
<td>Relation type alteration</td>
<td>13</td>
</tr>
<tr>
<td>Individuals</td>
<td>13</td>
</tr>
<tr>
<td>Rules</td>
<td>14</td>
</tr>
<tr>
<td>Constraints</td>
<td>14</td>
</tr>
<tr>
<td>Facts</td>
<td>14</td>
</tr>
<tr>
<td>Queries</td>
<td>14</td>
</tr>
<tr>
<td>Reasoning</td>
<td>14</td>
</tr>
<tr>
<td>Inspecting facts</td>
<td>15</td>
</tr>
<tr>
<td>Applying rules</td>
<td>15</td>
</tr>
<tr>
<td>Sum and normalization</td>
<td>15</td>
</tr>
<tr>
<td>Querying</td>
<td>15</td>
</tr>
<tr>
<td>Import, Export and Convert</td>
<td>15</td>
</tr>
<tr>
<td>To/From RDF(S) and OWL</td>
<td>15</td>
</tr>
<tr>
<td>To/From Datalog+</td>
<td>15</td>
</tr>
</tbody>
</table>
Basics

Installation

User interface basics

Keyboard shortcuts

Finding, Searching, and Replacing

- **Ctrl-F3**: Search word at insert point
- **F3/Shift-F3**: Find next/previous in file
- **Ctrl-F/H**: Find/Replace in file
- **Ctrl-Shift-F/H**: Find/replace in projects
- **Alt-Shift-H**: Turn off search result highlights
- **Ctrl-R**: Rename
- **Ctrl-U, then U**: Convert selection to uppercase
- **Ctrl-U, then L**: Convert selection to lowercase
- **Ctrl-U, then S**: Toggle case of selection
- **Ctrl-Shift-V**: Paste formatted
- **Ctrl-Shift-D**: Show Clipboard History
- **Ctrl-I**: Jump to quick search field
- **Alt-Shift-L**: Copy file path

Opening and Toggling between Views
Ctrl-Tab
(Ctrl-`) Switch between open documents by order used
Shift-Escape Maximize window (toggle)
Ctrl-F4/Ctrl-W Close selected window
Ctrl-Shift-F4 Close all windows
Shift-F10 Open contextual menu
Ctrl-PgUp/PgDown Switch between open documents by order of tabs
Ctrl-Alt-T Reopen recently closed file

Editing with graphical editors

Ctrl-C Copy selected vertices and edges
Ctrl-V Pasted vertices and edges
Del Delete selected vertices and edges. It also delete pending edges if any
Ctrl-Mouse To create edges between concept types or relation types into vocabulary graphical editor
Ctrl-Wheel Zoom +/-

Knowledge representation

CoGui works on a model of a knowledge base consisting of:

The ontological part
- A unique and necessary Vocabulary
- A set of Individuals
- A set of Rules
- A set of Constraints

Data organized into
- A set of Facts
- A set of Queries

Vocabulary

Overview about vocabulary
The vocabulary defined types used to qualify the data. These types are organized into two hierarchies:
- A concept type hierarchy
- A relation type hierarchy

**Overview**

CoGui is able to create multilingual ontologies designed for Conceptual Graphs (CGs). A CG Ontology is composed of exact knowledge and contextual knowledge. The vocabulary is one important part of the exact knowledge and consists of two hierarchies:
1) a hierarchy of concept types (also named concept or class or object type)
2) a hierarchy of relation types (also named relation) with arity greater or equal to 1.

The above hierarchies are respectively organized in partially ordered sets (not necessarily a tree or a lattice).

The exact knowledge of the ontology, apart from the vocabulary, consists of:
* a collection of individuals and eventually associated individual graphs
* rules

Editors allow end users to navigate through the ontology and edit graphically its structure and content. The ontology is controlled and, if necessary, tools are provided to correct it.

Graphically, types are displayed as vertices. An arc connecting vertex A to vertex B means that the type A is a kind of type B (or A is a specialization of B or B is a generalization of A):

In most cases the ordered set looks like this:

In this case, the hierarchical structure is a tree. But the model accepts extra connections. Two examples below illustrate hierarchies that not have a tree structure:
The edit operation is not heavily constrained by the model, in practice, the only critical error occurs when a circuit is detected. More details can be found in following chapters.

**How to browse through type hierarchies**

When a project is opened (or created) a vocabulary panel appears on the right part of the main window. Concept types, relation types and Individuals are displayed in three separated tabbed panels. An arborescent representation containing every path between maximal type and others. Types are alphabetically sorted, relation types are also sorted by arity.
The tree representation is useful to create vertices in conceptual graphs by dragging types into the graph editor (see Graph Edition chapter). Please remember that the type's order is not necessarily a tree. That's why the same type may be retrieved several times in the tree representation. For the same reasons tree is not automatically expandable if hierarchy contains at least one circuit.
Click right button and choose ‘Graphical Editor’

The graphical editor is opened
two synchronized editors for a same type hierarchy (both concepts and relations)

you can navigate between both representations:

- a simple double click on a vertex in left panel select and show the (unique) corresponding vertex in the hierarchy view

- a right click on vertex displays a popup menu: the 'Navigate/Show type in tree' action selects (and scrolls if necessary) the corresponding node(s) int the left panel.
Double click will open graphical editor and scroll to make the vertex visible.

Two other options show parents or children inside the graph (the scrolling process is automatically performed).

- Shows parent vertex in the graph representation and select them
- Shows children vertex in the graph representation and selected them

Concept types edition
- Insertion
- Graph arrangement
- Concept type hierarchy control
- Forbidden types
- Concept type alteration
Insertion

A newly created concept type hierarchy contains only one type named ‘Top’.

The ‘Insert concept type’ button on the toolbar of the graphical editor can be used to insert the desired type at the mouse location.

A concept type can also be created into the tree representation. ‘New concept type’ option in the type view popup menu creates a new concept type as a type of selected item.

Into the graph editor, the following vertex is displayed: conceptType1. Click twice on the vertex to edit.
type name. Concept types can be renamed directly on the concept type tree. Click once on the tree item to edit type name. Both actions have same effect and are synchronized.

**Naming convention**
Homonymous types are not allowed in the same type hierarchy. The case is respected but comparisons are case insensitive. For instance user can decide to write 'Dog' or 'dog' but cannot define both words in the concept type hierarchy. Blank spaces are allowed.

**Graph arrangement**
A vertex is moved by dragging its box with the mouse. Another way to place vertices is to run an automatic arrangement with the layout algorithms.

**Concept type hierarchy control**

**Forbidden types**

**Concept type alteration**

**Relation types edition**
Individuals

An individual is an identifier which is a surrogate for a precise entity of the discourse universe. For instance, if Town is a concept type then Budapest is an individual of type Town. A concept type may have subtypes, e.g. SmallTown could be a subtype of Town, an individual cannot have subtypes, e.g. SmallTown could be a subtype of Town, an individual cannot have subindividuals. The ontological individuals are the individuals about which all the users agree, i.e. for all users an ontological individual must represent the same entity in the discourse universe. An ontological individual is entered into a COGUI-ontology with a primitive concept type called its privileged type. For instance, if the COGUI-ontology concerns Modern Art, and if Picasso is an ontological individual of privileged type Artist representing the famous artist Pablo Picasso, then it is impossible to use the identifier Picasso for representing a Citroën car (unless the conjunctive type Car, Painter is not forbidden). COGUI checks that an ontological individual appearing in an annotation has a type which is compatible (i.e. not forbidden) with the privileged type of the individual.

All individuals appearing in a COGUI-ontology (e.g. in rules, constraints or prototypical knowledge) must be ontological individuals. Thus, the set of ontological individuals can be completed only whenever all knowledge representing in a COGUI-ontology have been considered.

The individual view lists all individuals in a sortable table. Needless to complete, the list of individuals automatically updates when the user references individuals in different graphs. Select individuals in the list to drop them to the graphs and right click to popup the contextual menu to rename, change the privileged type.
Individuals tab in vocabulary view displays the complete list of individuals

Since 3.0 CoGui integrates the notion of namespace. Then a namespace can be associated to each individual. It can be selected directly on the list:

Rules

Created with the Personal Edition of HelpNDoc: Produce Kindle eBooks easily

Constraints

Created with the Personal Edition of HelpNDoc: Produce electronic books easily

Facts

Created with the Personal Edition of HelpNDoc: Easily create PDF Help documents

Queries

Created with the Personal Edition of HelpNDoc: Free help authoring tool

Reasoning