Grisu Manual

Version 1.3

February 11, 2003

Contents

About this Manual

Grisu, the **GR**ammar V**ISU**alization Tool, is an application that allows the display and inspection of large feature structures or trees over feature structures as they are generated as the result of linguistic analyses in the **Trale** system. Although is it possible to use Grisu with other systems than Trale, this manual will focus only on using Grisu in conjunction with Trale. Grisu is Trale's viewing frontend of choice and has been very well integrated with Trale, so it is very likely that users who work with Grisu also use the Trale system. Consult the Trale manual[1] for more information on Trale.

First, this manual will cover the installation process of Grisu and its system requirements. Then it will move on and describe the complete functionality of Grisu.

[1] `http://ling.ohio-state.edu/~dm/2002/autumn/795K/trale-2.1.4-manual.pdf`

Part I

Setting up Grisu

1 System Requirements

If you want to install Grisu, you should first make sure that your system meets Grisu's requirements.

- A current version of Linux with gcc version 2.95 or higher and appropriate libc on Intel/AMD x86 systems
- Solaris 2.x with gcc version 2.95 or higher and appropriate libc on SPARC or Intel/AMD x86 systems.
- KDE 2.1 or higher (KDE 3.x is suppported). We recommend that you use a full installation of KDE including the KDE desktop. Grisu doesn't require such a full installation of KDE; it will run on systems with other user interfaces or window managers (GNOME, fvwm2, windowmaker), provided that the KDE base libraries (the `kdelibs` package) is installed on your system.
- KDE 2.x requires Qt 2.x, KDE 3 requires Qt 3.x. The Qt libraries must be present on your system because the KDE base libraries depend on them.

1.1 Grisu on Linux

The easiest way to set up a proper system if you have a PC is to install one of the current Linux distributions, which are sufficiently preconfigured in most cases. Many distributions allow you to choose at install time if you want to include development support in your system, make sure you do so since otherwise you won't be able to compile Grisu.

Grisu has been tested on these distributions:

- Debian Linux 3.0 "woody" / KDE 2

- Mandrake Linux 9.0 / KDE 3
- RedHat Linux 7.2 / KDE 2
- RedHat Linux 7.3 / KDE 3
- RedHat Linux 8.0 / KDE 3
- SuSE Linux 7.1 / KDE 2
- SuSE Linux 8.0 / KDE 3
- SuSE Linux 8.1 / KDE 3

1.2 Grisu on Solaris

Unfortunately, on Solaris things are not quite as easy. Neither Qt nor KDE are preinstalled on any Solaris system. This means that you must download, compile and install Qt and KDE on Solaris yourself. This manual can't assist you with this task, please refer to Troll Technology's[2] and KDE's[3] homepages.

Nevertheless, it can be done and once Qt and KDE are properly set up, Grisu will run fine.

2 Unpacking the Grisu package

Grisu comes in a bzip2-packed tar-archive. To unpack Grisu in a directory of your choice, simply type:

```
bzip2 grisu-v-1-3-3.tar.bz2

tar xvf grisu-v-1-3-3.tar
```

(Note that you may have a package with a higher version number, so the name of your package may differ.) This will create a new subdirectory in

[2] http://www.trolltech.com
[3] http://www.kde.org

the current directory, named `grisu-v-1-3-3`. We will refer to this directory as the "Grisu-directory" in what follows. Within the Grisu-directory, you'll find the `Makefile`, the Grisu source and header files, and another directory named `pics` which contains a number of icons used by Grisu.

The Grisu executable will also be placed in the Grisu-directory after compilation.

3 Compiling Grisu

Once Qt and KDE are set up properly, you can move on and compile Grisu. This is a semi-automated process in the sense that in this release the paths to include files and libraries cannot be determined automatically. In many cases, setting the correct will boil down to uncommenting a single line in the Makefile.

3.1 Adapting the Makefile

Before you compile Grisu, you must adapt the `Makefile` to meet your system's configuration. Grisu has been built successfully on quite a number of different platforms, for which there exist preconfigured targets in the `Makefile`. There is a good chance that your system is among those platforms, or very close to these platforms (for example, just a different version of the same Linux distribution).

Selecting the target platform

When you load `Makefile` in an editor, you'll see this the top of the file:

```
# Grisu has been built on the platforms listed below.
# Uncomment the one that meets your configuration.

# --- Solaris targets ----

# Solaris 7 or 8 with KDE2
#TARGET = Solaris_KDE2_OSU
```

```
#TARGET = Solaris_KDE2_SfS

# Solaris 7 or 8 with KDE3
# TARGET = Solaris_KDE3

# --- LINUX targets ----

# SuSE Linux 7.1 with KDE2
#TARGET = SuSe_KDE2

# SuSE Linux 8.0 with KDE3
#TARGET = SuSe_KDE3

# SuSE Linux 8.1 with KDE3
#TARGET = SuSe81_KDE3

# RedHat Linux 7.2 with KDE2
#TARGET = RedHat_KDE2

# RedHat Linux 7.3 with KDE3
#TARGET = RedHat_KDE3

# RedHat Linux 8.0 with KDE3
#TARGET = RedHat8_KDE3

# Mandrake Linux 9.0 with KDE3
#TARGET = Mandrake9_KDE3

# Debian GNU/Linux with KDE2
# Tested for Woody (stable) and Sarge (testing).
TARGET = Debian_KDE2
```

These are the preconfigured targets. If you find your system under these targets, uncomment (remove the # symbol at the beginning of) the line directly following the description of your system. For example, if you have a RedHat 8, you uncomment the line `TARGET = RedHat8_KDE3`.

Make sure to comment out **all** other target lines by putting a # to the left of the line.

Setting ICONDIR

Right below the list of targets you will find the `ICONDIR` entry:

```
# Directory in which the icons for our application live
# Note: most targets below change this setting
ICONDIR = '"/this/is/the/path/to/the/pics/directory"'
```

Grisu needs to know the directory which contains its icons. By default, this is the `pics` directory, a subdirectory of the Grisu-directory. There are two ways of how to tell Grisu about this directory. The first one, explained here, is to set `ICONDIR` in the Makefile. The second way is to set the `GRISU_RESOURCE_PATH` environment variable, described later.

The value of `ICONDIR` will become hardwired in the Grisu executable, and can't be changed without recompiling Grisu. Although not very flexible, this is useful if you want to set up a site-install of Grisu in a central location with multiple users accessing it. By using `ICONDIR`, it is not necessary for every user to set up their own environment.

So if Grisu is installed in `/usr/grisu-v-1-3-3`, `ICONDIR` should have the following value:

```
# Directory in which the icons for our application live
# Note: most targets below change this setting
ICONDIR = '"/usr/grisu-v-1-3-3/pics"'
```

The strange combination of single- and double-quotes is necessary. Don't change it!

Verifying the TARGET settings

Next in the makefile are the actual settings for the target platforms. You should check if the paths specified in your target are really correct, and their are some targets which locally redefine the `ICONDIR` value. So if you use `ICONDIR`, and it is redefined in your target, make sure to fill in the correct value here, too.

This is the target for the Madrake Linux 9 distribution:

```
#Mandrake Linux 9.0, KDE3
ifeq ($(TARGET), Mandrake9_KDE3)
INCLUDE = -I/opt/kde3/include -I/usr/lib/qt3/include -I/usr/X11R6/include -I.
LIBDIR = -L. -L/usr/X11R6/lib -L/opt/kde3/lib -L/usr/lib/qt3/lib
LIBS    = -lkdeui -lkdecore -lkio
DEFINES = -D_LINUX -DRESOURCE_PATH=$(ICONDIR)
MOCCOMPILE = /usr/lib/qt3/bin/moc
ICONDIR = '"/home/wunsch/grisu/pics"'
endif
```

The INCLUDE line contains all paths to directories containing include files. Each path must be preceeded by `-I` (the options for include paths in gcc).

The LIBDIR line contains all library paths. Each path must be preceeded by `-L`.

The LIBS line contains all additional libraries Grisu must be linked against. The basic C libraries will always be linked automatically. Especially Solaris requires a much larger number of libraries (as you can easily see in a Solaris target).

It shouldn't be necessary to change the DEFINES line.

The MOCCOMPILE line contains the path to the `moc` executable, which is a part of Qt.

Finally in this target, there is also an ICONDIR line which overwrites the `ICONDIR` value before.

Adding new targets

If you can't find your system in the list of the preconfigured targets, there are several things you can do. First, select the target which is the closest to your system. So if you use SuSE Linux 7.3, try the target for SuSE Linux 7.1.

If this doesn't work (that means, if you get errors while compiling), try the other targets. Most newer Linux distributions adhere to the "Linux Standard Base", which means that they all use the same directory structure and libraries.

If you're still not able to compile Grisu, then must add your own new target. The best idea to copy-and-paste and existing target and then modify

the lines you need. Tell us, if you successfully ported Grisu to a new platform!

Building Grisu

Now you're ready to built the Grisu executable. In the Grisu-directory, type

```
grisu-make
```

which will start the compilation process. This will create the Grisu executable.

Part II

Using Grisu

4 Starting Grisu

4.1 Starting Grisu together with Trale

In most cases, you will use Grisu together with Trale, with both applications running on the same machine. In this case, starting Grisu is simple. Just type

```
trale -g &
```

This will start Trale with Grisu support. Both Trale and Grisu will start, and the two systems will be connected such that any output generated by Trale will be sent to Grisu. If you already used Trale and Grisu before you started to read this manual, you probably haven't even noticed that Grisu is actually an independent application.

4.2 Starting Grisu and Trale on different machines

It is possible to run Trale and Grisu on two different machines, with Trale sending its output over the internet. In order to do this, first start Grisu on the first machine, by typing

```
grisu &
```

Grisu will come up with its main window. In the lower right corner of the window, Grisu displays the name of the host and the port it is running on. You will need to pass these to Trale.

Then, start Trale on the other machine, by typing

```
trale -g <hostname> <port>
```

replacing `<hostname>` with the hostname displayed by Grisu and `<port>` with the port number displayed by Grisu.

Note that on some systems, only the plain hostname is displayed without the domain name. Trale expects a fully qualified hostname, that is a hostname that includes a domain name.

For example, if Grisu displays

```
myhost.mydomain.net 5001
```

then call Trale with

```
trale -g myhost.mydomain.net 5001
```

If Grisu only displays

```
myhost 5001
```

then call Trale with

```
trale -g myhost.mydomain.net 5001
```

(If you don't know the domain your machine is in, please ask your system administrator).

4.3 Starting Grisu without Trale

Finally, you can also start Grisu without Trale. This can be useful if you want to look at some structures you previously saved, or you have a program other than Trale that can send Grisu data it understands.

You can start Grisu alone by typing

```
grisu &
```

5 The Main Window

What you see first after Grisu has come up is the *main window* (figure 5). It is arranged in three major parts like most other modern applications with a graphical user interface: The topmost part consists of the *main menu* and the *toolbar*, the large area is the *list of data structures*, and the *status line* is located at the bottom of the main window.

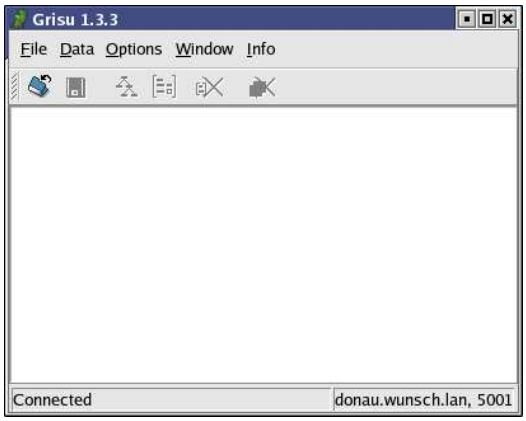

Figure 1: The main window

In the list of data structures, Grisu displays descriptions[4] of all data structures (that is feature structures or trees) that are available for inspection.

The status line displays information about the current status of the connection between Grisu and Trale. During normal operation, the status should be `Connected`.

6 The Trale Window

When started with the `trale -g` command as shown in section 4.1, a second window will come up which is the *Trale window*. The Trale window is actually an XEmacs window in shell mode, in which Trale is running.

[4] If you use Trale, the description is the same as what you passed to Trales `rec` or `lex` commands.

7 Inspecting Feature Structures and Trees

This section will give a tutorial on how to inspect feature structures and trees that were the result of linguistic analyses in Trale. Note that the examples shown here depend on the grammar loaded into Trale, so if you want to reproduce this tutorial, replace the examples with ones that are well-formed with respect to the grammar you use.

You can parse and analyze a sentence in Trale with the `rec` command. Activate the Trale window, and at Trale's prompt, type:

```
rec [peter, walks]
```

As the result of this, one item is added to the list of data structures in the main window, named `[peter, walks]` (figure 2).

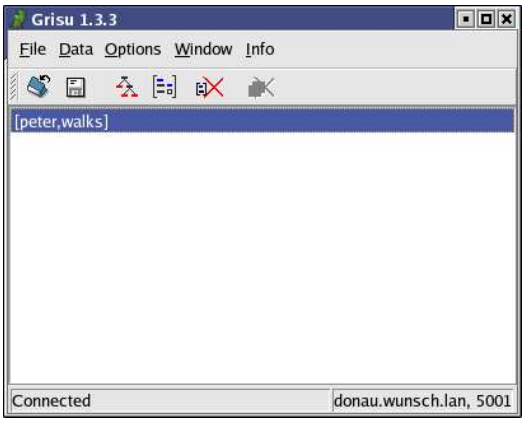

Figure 2: A new item in the list of data structures

At the same time, a new window should open which displays the data structure which contains the analysis of the `[peter, walks]` sentence[5]. This is a *data window* (figure 3).

You can see a simple tree with three *nodes*, the root node which represents the phrase "peter walks", and two daughters. The left daughter represents the word "peter", the right daughter represents the word "walks".

[5]If the window doesn't open, double-click the `[peter, walks]` item in the list of data structures.

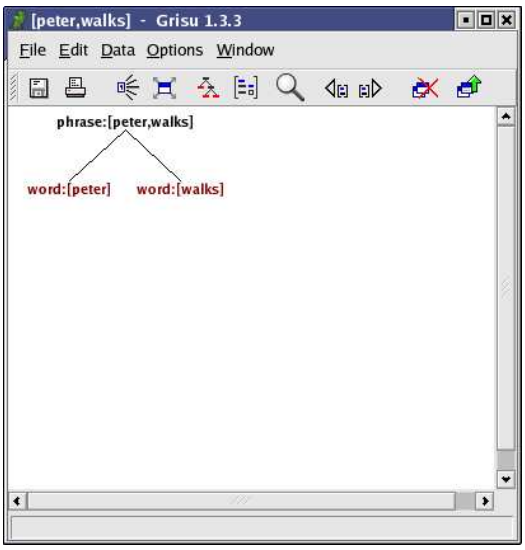

Figure 3: Data window displaying an analysis of `[peter, walks]`

Clicking on any node will show the associated feature structure (trees in an HPSG framework are just a more intellegible way of displaying large feature structures), as you can see in figure 4.

Clicking again on the same tree node will hide the feature structure again.

Structures and trees get huge quickly. Clicking on the `phrase` node reveals the feature structure shown in figure 5.

With Grisu, you can selectively show and hide parts of a large feature structure. In feature structures, a "tag" (also called a "reference") in front of a substructe indicates that this substructure occurs at multiple positions in the feasture structure. In figure 6, you can see such a tag, which is part of the feature structure shown in figure 5. By clicking on this tag, you can "turn off" the complete substructure. This makes the displayed part of the overall feature structure much smaller, as you can see in figure 7. Clicking the tag again brings back the substructure.

In addition to expanding and collapsing tags it is possible to hide any node in a feature structure or tree. In Grisu, everything that is displayed is a node: A type, a feature, a tag, a whole feature structure. Of course, a tree node is also a "node". By Ctrl-Clicking (clicking the mouse while you hold down the CTRL key), you can hide any node. Figure 8 again shows our large feature structure, with the DTRS feature hidden.

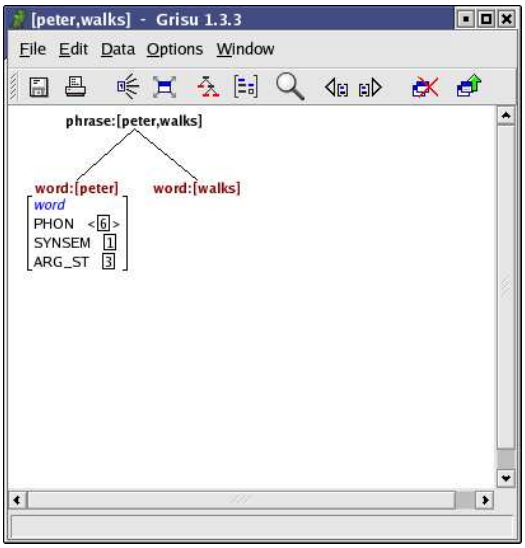

Figure 4: A tree node with its associated feature structure

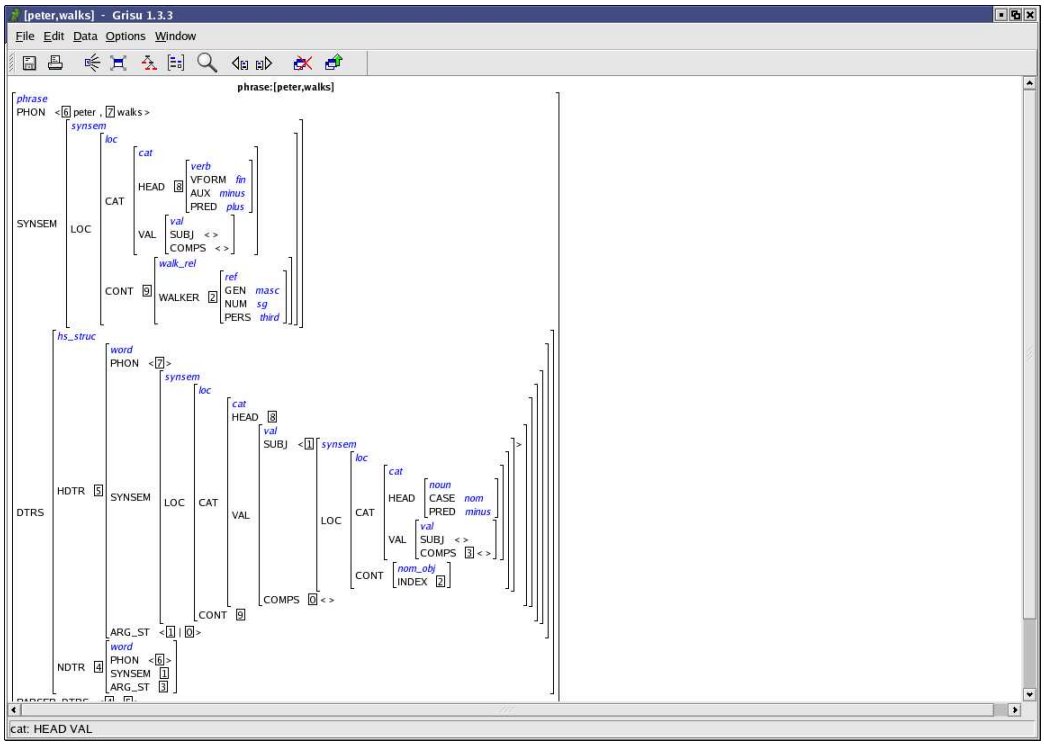

Figure 5: A large feature structure

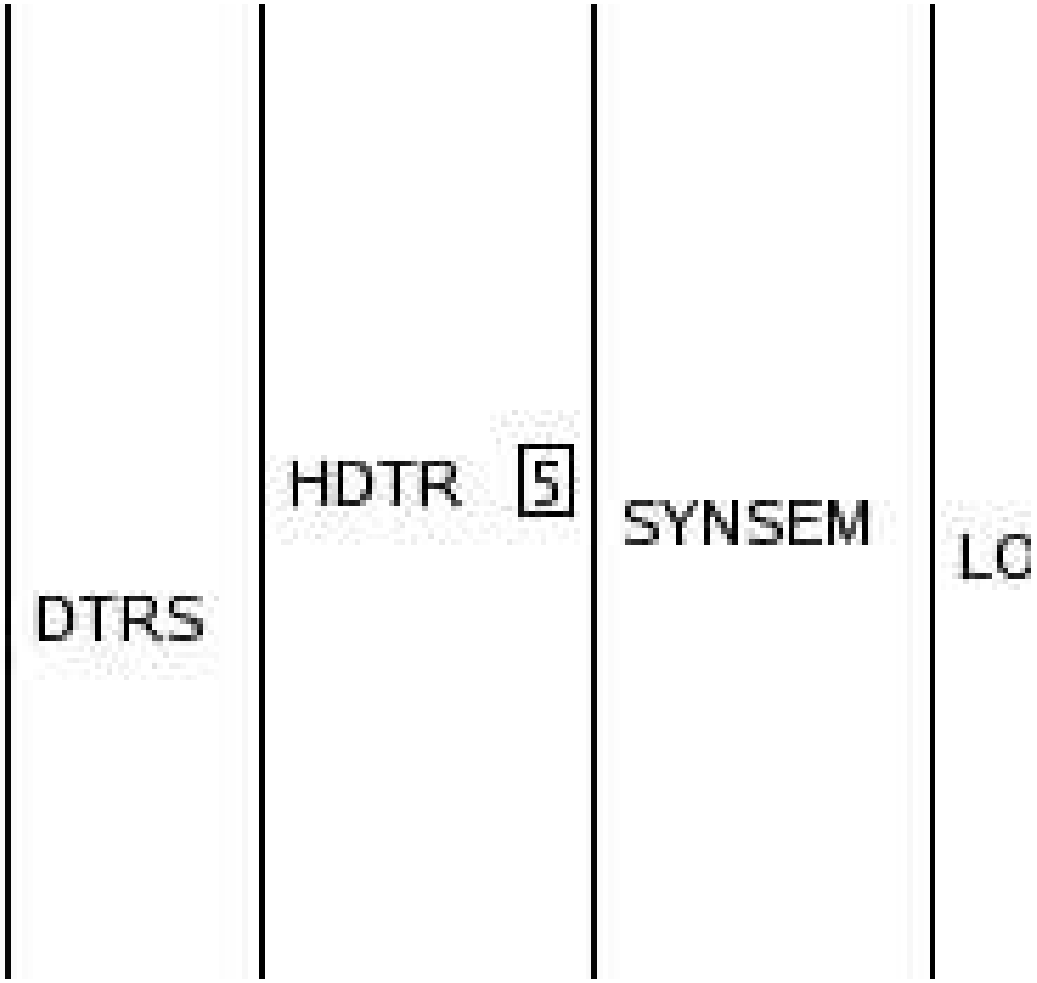

Figure 6: A tag

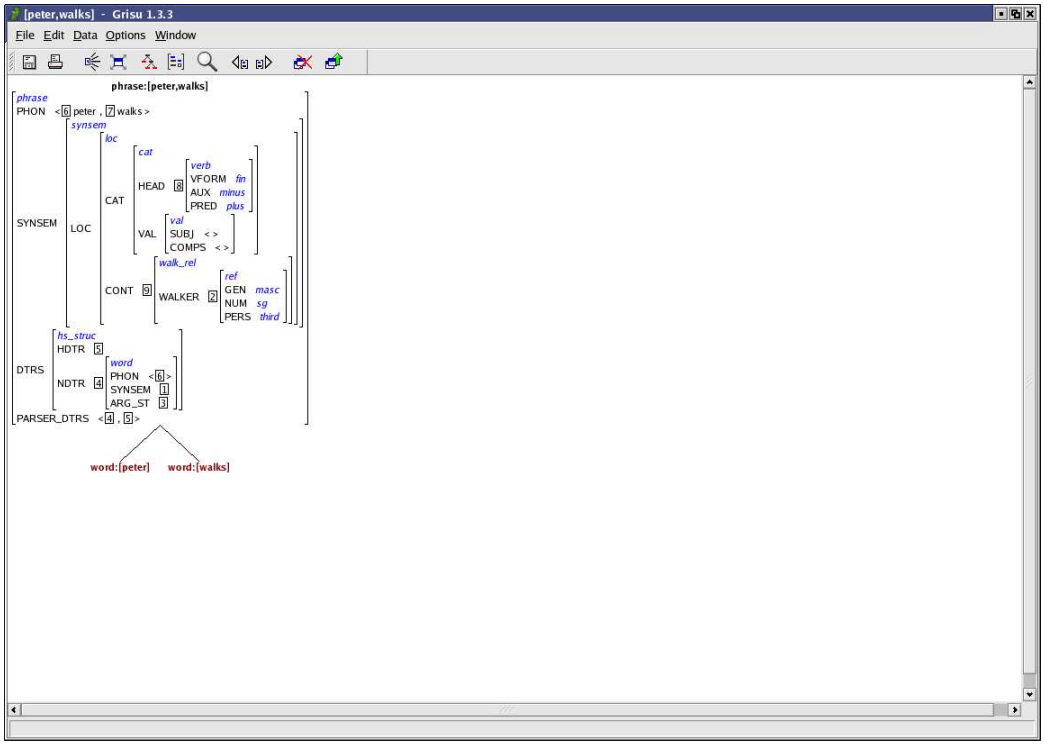

Figure 7: A collapsed tag

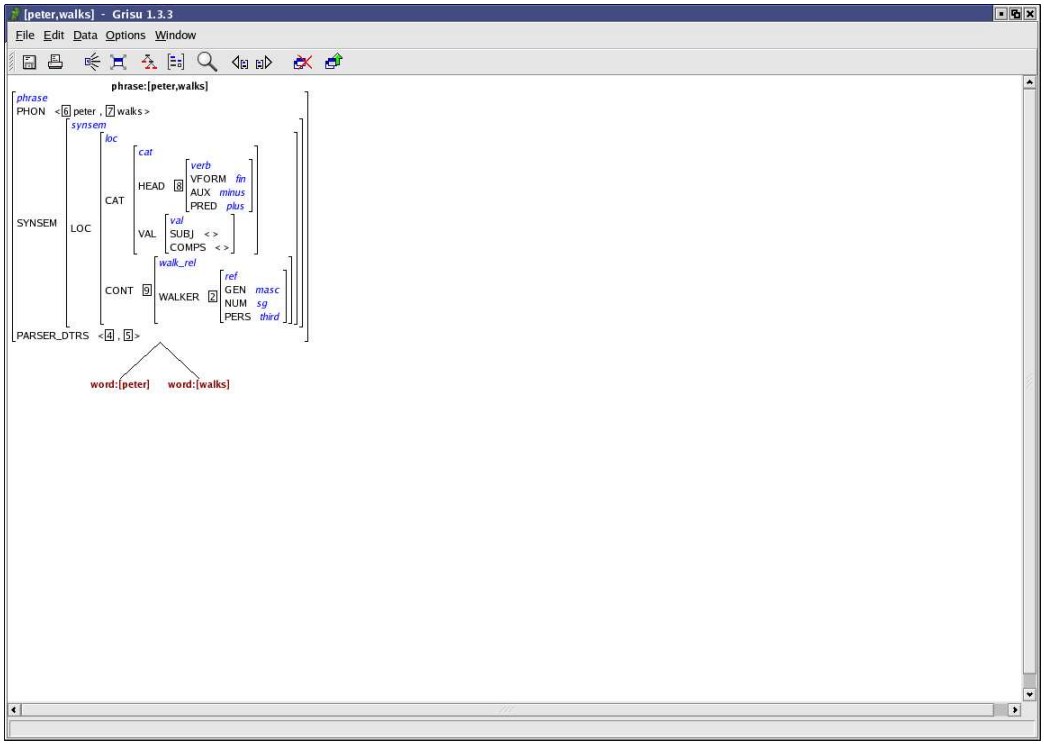

Figure 8: The DTRS feature is hidden

Hidden nodes are invisible, so you can't click on them to bring them back. However, there is an option in Grisu to "show hidden nodes". This means that instead of totally hiding the node, a placeholder for the node is displayed in gray color. Clicking this placeholder will unhide the node again. "Show Hidden Nodes" is turned off by default, you can turn it on by clicking SHOW HIDDEN NODES in the OPTIONS menu.

We have come to the end of our short tutorial. The next sections will describe all Grisu features in detail.

8 Features Accessible from the Main Window

The main window is Grisu's control center. With the functions provided by the main window, you can manage your data structures and data windows, load and save data structures and set options that apply to all data windows.

8.1 The list of data structures

The list of data structures (the large area in the center of the main window) displays descriptions of all data structures which are available in Grisu for inspection. By **Double-clicking** on an item in the list you can bring up a data window that displays the corresponding data. Grisu automatically determines whether to display a feature structure or tree.

You can highlight multiple items by **Ctrl-clicking**. Highlighting an item, then **Shift-clicking** another item highlights all items between these two.

8.2 The FILE menu

8.2.1 LOAD DATA

 **Alt+L**

With the LOAD DATA command you can load a data structure that you previously saved in a file. A file dialog will appear allowing you to select a file. Grisu data files have the extension `.grisu` by default.

The data structure appears in the list of data structures after it was loaded.

8.2.2 SAVE DATA

 **Alt+S**

With the SAVE DATA command you can save the data structure that is currently highlighted in the list. In the file dialog that appears, specify the name of the file you want to save the data structure in. Grisu data files have the extension ġrisu by default.

8.2.3 EXIT GRISU

Alt+X, Alt+Q

The EXIT GRISU command terminates Grisu. If you changed any options throughout the Grisu session, you will be asked if you want to save these options.

Caution: If you use Grisu together with Trale, make sure you leave Trale *before* you leave Grisu. Otherwise, the TCP/IP port used in the connection between Trale and Grisu cannot be freed properly.

8.3 The DATA menu

8.3.1 SHOW TREE

 **Ctrl+T**

The SHOW TREE command opens a new data window that displays the data structure currently highlighted as a tree. If the data structure doesn't contain any tree data, a feature structure is displayed instead.

If multiple data structures are highlighted, multiple data windows are opened.

8.3.2 SHOW STRUCTURE

 **Ctrl+S**

The SHOW STRUCTURE command opens a new data window that displays the data structure currently highlighted as a feature structure.

If multiple data structures are highlighted, multiple data windows are opened.

8.3.3 DELETE DATA

 **Ctrl+D**

The DELETE DATA command deletes all highlighted data structures from the list and closes all associated data windows.

8.3.4 DELETE ALL DATA

Ctrl+Shift+D

The DELETE ALL DATA command deletes all data structures from the list and closes all associated data windows.

8.4 The OPTIONS Menu

8.4.1 AUTO-EXPAND TAGS

If the AUTO-EXPAND TAGS option is turned on (a checkmark is visible next to the menu item), any tag that occurs for the first time in a feature structure is expanded (i.e. the associated substructure is displayed) when a new data window is opened. All subsequent tags are displayed collapsed.

8.4.2 WINDOWS FIT STRUCTURE SIZE

If the WINDOWS FIT STRUCTURE SIZE option is turned on, the size of a data window will always match the size of the data structure it displays.

Changing this option only affects data windows opened after the change, not data windows already open.

8.4.3 HIGHLIGHT STRUCTURES

If the HIGHLIGHT STRUCTURES option is turned on, a border is drawn around the node the mouse cursor points to. In large feature structures, this can be useful to find out where a node belong to.

8.4.4 AUTOMATICALLY OPEN WINDOWS

If the AUTOMATICALLY OPEN WINDOWS options is turned, a data window is opened as soon as new data comes in.

8.4.5 STRUCTURE SPACING

The STRUCTURE SPACING command brings up the "Structure Spacing" dialog.

The structure spacing value determines the amount of space between two nodes in a feature structure. The larger the value, the more space is left between nodes. The default value is 2.

8.4.6 COLORS

You can customize the color of most different types of nodes displayed in data windows. Each of the items in the COLOR menu brings up a color selection dialog.

FEATURES

Changes the color of features.

Types

Changes the color of types.

Text

Changes the color of atoms, as they occur in phon-lists, for example.

Inner Tree Nodes

Changes the color of inner nodes in a tree.

Edges

Changes the color of edges (the lines connecting two nodes) in a tree.

Leaves

Changes the color of leaves (nodes with no children) in a tree.

Edge Labels

Changes the text color of the labels on edges in a tree.

Edge Label Background

Changes the background color of the labels on edges in a tree.

Hidden Nodes

Changes the color of placeholders for hidden nodes.

HIGHLIGHTED STRUCTURES

Changes the color of the border drawn around highlighted nodes.

FOUND STRUCTURES

Changes the color of the border drawn around the next found structure.

DIFFERENT STRUCTURES

Changes the color of the border drawn around two structures that are different (when comparing two structures).

ERRORS

– This color is currently not used. –

8.4.7 Fonts

You can customize the color of many different types of nodes displayed in data windows. Each of the items in the FONTS menu brings up a font selection dialog.

FEATURES

Changes the font of features in a feature structure structure.

TYPES

Changes the font of types in a feature structure structure.

INNER TREE NODES

Changes the font of inner nodes in a tree.

LEAVES

Changes the font of leaves (nodes with no children) in a tree.

EDGE LABELS

Changes the font of labels on edges in a tree.

TEXT

Changes the font of atoms.

ERRORS

– This font is currently not used. –

8.5 The WINDOW Menu

8.5.1 CLOSE ALL DATA WINDOWS

Closes all data windows currently open.

8.6 The INFO Menu

8.6.1 ABOUT GRISU

Displays version information about Grisu.

8.6.2 ABOUT KDE

Displays information about KDE.

9 Features Accessible From Data Windows

9.1 The status line

The status line in data windows display the type and all appropriate features of the feature structure mouse cursor currently points to. All features are displayed, including those who are currently hidden.

9.2 Navigating in Data Windows

Grisu provides a several different ways of how you can move around in large data structures. You can either use the mouse or the keyboard, whatever you prefer.

To the right and to the bottom of any data window, you will find a set of **scrollbars**. The scrollbars serve two purposes: They show you which part of a data structure is currently visible, and you can use them to view other parts of a data structure by using the mouse.

A scrollbar consists of three parts: An **up-arrow** or a **left-arrow**, a **down-arrow** or a **right-arrow** and the **body**. Part of the body is the **thumb**, which moves up and down (or left to right), and changes its size. The relative size of the thumb to the total size of the scrollbar's body represents how much of the data structure is visible, while the thumb's position indicates which part is currently displayed.

9.2.1 Navigating Vertically

- You can click on the **up-arrow**, or press the **Cursor Up key** to scroll the structure one line up.
- You can click on the **down-arrow**, or press the **Cursor Down key** to scroll the structure one line down.
- You can click **above the thumb, but still within the body**, or press the **Page Up key** to scroll the structure one page up (**Ctrl+Cursor Up** works, too).

- You can click **below the thumb, but still within the body**, or press the **Page Down key** to scroll the structure one page down (**Ctrl+Cursor Down** works, too).
- If you have a wheel mouse, you can use the **mouse wheel** to scroll up or down.

9.2.2 Navigating Horizontally

- You can click on the **left-arrow**, or press the **Cursor Left key** to scroll the structure one line to the left.
- You can click on the **right-arrow**, or press the **Cursor Right key** to scroll the structure one line to the right.
- You can click **to the left of the thumb, but still within the body**, or press the **Ctrl+Cursor Left key** to scroll the structure one page to the left.
- You can click **to the right of the thumb, but still within the body**, or press the **Ctrl+Cursor Right key** to scroll the structure one page to the right.
- If you have a wheel mouse, you can use the **mouse wheel while holding down the Ctrl key** to scroll left or right.

9.2.3 Dragging

An alternative way of moving around in a structure is by holding down the **middle mouse button**, and then **dragging** the mouse. The structure will move in the same direction. This way, you can change the horizonal and vertical position simultaneously.

9.3 Mouse Functions and Context Menus

Many functions in data windows by using the mouse. Feature structures and trees are composed of "nodes" in Grisu. Types, features or tags are examples of nodes. While some features are available for all types of nodes

(hiding, for example), others are only available for specific nodes (like expanding a tag node).

Clicking on a node with the **left mouse button** will activate the "default command" for the node. Which command this is depends on the type of node, as you can see in the following table.

Node Type	Default Command
Types	Hide/Unhide type.[6]
Features	Hide/Unhide feature.[6]
Tags	Expand/Collapse tag.
Tree nodes	Show/Hide associated feature structure.

Clicking on a node with the **left mouse button** while holding down the **Shift** key will search for the next occurrence of this node (see the description of the find feature).

Clicking on a node with the **right mouse button** will bring up a context menu. The commands offered here again depend on the type of the node. This section will describe the possible commands.

9.3.1 SHOW/HIDE <NODE>

The SHOW/HIDE command hides or unhides the node.

Supported by: types, features, lists, sets, tags, tree nodes, atoms

9.3.2 SHOW ATTRIBUTE: <NAME>

The SHOW/HIDE command displays a placeholder for the hidden feature with the name NAME.

Supported by: types, features

[6] In order to hide the node, you must hold down the **Ctrl** while clicking. This is a security feature which prevents accidental hiding of nodes.

9.3.3 Don't Show This Hidden Feature

Hides the placeholder for the feature previously turned on with Show Attribute.

Supported by: types, features

9.3.4 Show Type: <name>

The Show/Hide command displays a placeholder for the hidden type with the name NAME.

Supported by: types, features

9.3.5 Don't Show This Hidden Type

Hides the placeholder for the feature or type previously turned on with Show Type.

Supported by: types, features

9.3.6 Show All Hidden Types/Features On Level

Shows placeholders for all features and types on the same level (in the same feature structure the mouse cursor points to).

Supported by: types, features

9.3.7 Hide All Hidden Types/Features On Level

Hides the placeholders for all features and types on the same level (in the same feature structure the mouse cursor points to).

Supported by: types, features

9.3.8 FIND NEXT OCCURRENCE OF <TEXT>

Starts a search for "Text" at the current node. See the description of the find feature.

9.3.9 EXPAND REFERENCE

Expands the tag.

Supported by: tags

9.3.10 SHOW/HIDE STRUCTURE

Shows or hides the associated feature structure of a tree node.

Supported by: tree nodes

9.4 The FILE Menu

9.4.1 SAVE

 **Alt+S**

With the SAVE command you can save the data structure. In the file dialog that appears, specify the name of the file you want to save the data structure in. Grisu data files have the extension ġrisu by default.

9.4.2 PRINT

 **Alt+P**

With the PRINT command you can print the data structure.

9.4.3 Close This Window

Alt+W, Alt+F4

The Close Window command closes this data window.

9.5 The Edit Menu

9.5.1 Expand Reentrancies

 **Ctrl+E**

With the Expand Reentrancies command you can expand all first the first occurrence of every tag. If Auto Expand Tags option is turned on in the main window, this command is automatically executed when a new data window is opened.

9.5.2 Reset Structure

Ctrl+R

With the Reset Structure command, you can reset the structure to its initial state, undoing all expand and hide operations.

9.5.3 Find

 **Ctrl+F**

The Find command invokes the find function.

9.6 The Data Menu

9.6.1 Show Tree

 **Alt+T**

The SHOW TREE command displays the current data structure as a tree. This requires that tree data is included in the data structure. If this is not the case, the data is displayed as a feature structure.

9.6.2 SHOW STRUCTURE

 **Alt+S**

The SHOW STRUCTURE command the current data structure as a feature structure.

9.6.3 SHOW TREE IN NEW WINDOW

The SHOW TREE IN NEW WINDOW opens a new data window, and displays the current data structure as a tree in this newly opened window. This requires that tree data is included in the data structure. If this is not the case, the data is displayed as a feature structure.

9.6.4 SHOW STRUCTURE IN NEW WINDOW

The SHOW STRUCTURE IN NEW WINDOW opens a new data window, and displays the current data structure as a structure in this newly opened window.

9.6.5 PREVIOUS DATA

 **P**

The PREVIOUS DATA command replaces the data structure currently displayed in this data window with the previous structure in the list of data structures. If the current data structure is the first in the list, the last structure will be displayed.

9.6.6 NEXT DATA

 **N**

The NEXT DATA command replaces the data structure currently displayed in this data window with the next structure in the list of data structures. If the current data structure is the last in the list, the first structure will be displayed.

9.7 The OPTIONS Menu

The options available in this menu apply to this data window only, unlike the options available in the main window, which apply to all data windows opened after the changing an option.

9.7.1 WINDOW FITS STRUCTURE SIZE

If the WINDOW FITS STRUCTURE SIZE option is turned on, the size of the data window will always match the size of the data structure it displays.

9.7.2 HIGHLIGHT STRUCTURES

If the HIGHLIGHT STRUCTURES option is turned on, a border is drawn around the node the mouse cursor points to. In large feature structures, this can be useful to find out where a node belong to.

9.7.3 SHOW HIDDEN NODES

If the SHOW HIDDEN NODES option is turned on, placeholders are displayed for **all** hidden nodes.

This option overrides the similar commands in context menus.

9.7.4 STRUCTURE SPACING

The STRUCTURE SPACING command brings up the "Structure Spacing" dialog.

The structure spacing value determines the amount of space between two nodes in a feature structure. The larger the value, the more space is left between nodes. The default value is 2.

9.8 The WINDOW Menu

9.8.1 CLOSE ALL DATA WINDOW

 **Alt+A**

The CLOSE ALL DATA WINDOWS command closes **all** data windows currently open.

9.8.2 RAISE MAIN WINDOW

 **Alt+M**

The RAISE MAIN WINDOW activates the main window so that it is shown on top of all other windows. This is useful if the desktop is cluttered with many open data window and/or other windows.

9.9 Finding Nodes

In large data structures, it can be very hard to find a specific node of interest. With Grisu's find function, you can look for a node by its name (that is, the type name for types, the feature name for features, and so on). If Grisu finds a node that matches the name, it will scroll this node into view and draw a border around it.

You can activate the find function by selecting the FIND menu item in the EDIT menu of a data window. The find dialog will open. In the title bar, the find dialog will show which data window it belongs to, which is useful if you have multiple data windows and multiple find dialogs.

You can enter your search term in the text field labeled **Search for**. Grisu will seach all nodes in the structure and mark the first matching node. Only exact matches are returned. For each type of node, the search term is interpreted appropriately: In type, feature and tree nodes, the search term is compared the type name, feature name or tree label.

So if you are looking for all types *bool*, type in `bool` in as the search term and Grisu will find all *bool* types (it wouldn't find types named *boolean* though, since only exact matches are returned).

If you are looking for SYNSEM features, use `synsem` as the search term. Grisu will find all SYNSEM features. Grisu will also find *types* with name *synsem*. This is because all nodes are searched regardless of their type.

To find a tag, just type in its number as the search term.

If the **Ignore Case** checkbox is turned on, Grisu will not take into account case while comparing. If you turn off the checkbox, only those nodes will be found, that match the search term exactly – including case. You can use this as a trick to look for types or features only. It is common style to notate types in feature structures all lowercase, while feature names are notated uppercase. So by typing in `synsem` with "Ignore Case" turned off, you'll only get lowercase types named *synsem*. The search term `SYNSEM`, again "Ignore Case" turned off, will give you the features named SYNSEM. Of course, this depends on the way types and features are named in the grammar – at least feature names are always capitalized by Grisu.

You start the search by clicking on the **Find** button. By clicking on **Find Next** Grisu will show you the next node and so on.

Clicking Find instead of Find Next will restart the search from the beginning!. If you want to see all nodes, make sure you click "Find" only once and then "Find Next".

F3 and **Ctrl+G** are shortcuts for "Find Next".

9.9.1 Using the Context Menu to Find Nodes

If you click on a node with the **right mouse button**, you can bring up the context menu for any node. The menu item FIND NEXT OCCURRENCE OF ... will search for the next occurrence of a node with the same name. This seach is case-sensitive by default, but you can change this by unchecking

"Ignore Case" in the find dialog

So if you click on a type with the name *synsem* you'll find the next type named *synsem*.