LTC User's Guide

Ulrik Petersen

October 27, 2011

1 Introduction

1.1 What is LTC?

LTC is a program for analyzing text in terms of syntax trees. That is, it assists in building a rigorous hierarchy of syntactic elements on top of the words in a given piece of text. This is done by interacting with a tree-view of the text, gradually building the analysis. This can be done either bottom-up or top-down, or in a mix of both styles.

1.2 One version

As of version 3.0.0, LTC only exists in a Unicode version. This is true on all supported platforms: Windows, Mac OS X, and Linux/Unix.

1.3 Supported platforms

The following are supported:

- Mac OS X
- Some versions of Windows
- Linux
- Some versions of Unix

1.4 Supported Windows platforms

In the Windows family, LTC will run **only** on the following platforms:

- Windows 2000
- Windows XP
- Windows Server 2003,
- Windows Vista, and
- Windows 7.

The following are **NOT SUPPORTED**. It is **NOT POSSIBLE** to run LTC on the following platforms:

- Windows 95
- Windows 98
- Windows 98SE
- Windows ME

1.5 What's next?

This user's manual attempts to lay out all the knowledge needed in order to operate LTC. The layout of the document is as follows:

1. First we discuss a few linguistic issues.
2. Then we briefly talk about the file formats of LTC.
3. Then we discuss how to start a new project.
4. After that, we give a brief tour of the main window of the program, which introduces all four parts of the window.
5. After that, the bulk of the document is taken up by descriptions of how to use the various parts of the program.
6. At the end, there are appendixes on file formats.

2 Linguistic issues

In this section, we discuss some linguistic issues.

2.1 Theories supported

Three styles of trees are supported:

- "Generic" trees
- X-bar trees
- Role and Reference Grammar (RRG) trees

You can mix the various kinds of trees in the same document.

2.2 Hierarchy of "generic" trees

For the "generic" trees, the hierarchy looks like this: From bottom to top:

1. Word
2. Phrase
3. "Syntax node"

The "Syntax nodes" are defined in a file which you, the user, are free to edit. Thus you can customize the availability of "syntax nodes". By default, the kinds "Clause", "Sentence" and "Paragraph" are defined. Please refer to Appendix B for more information.

2.3 Phrase types

A phrase must be given a "phrase-type", such as "NP" or "PP". This can be selected from the side-bar, or via the "NP" button. When making a phrase with the "NP" button, you will be asked to pick a phrase-type from a list of possible phrase-types.

The list of available phrase types is stored in a special file which you can edit. Please refer to Appendix B for more information.

2.4 Text

One special object which always exists in the program is the "text." In the terminology used by LTC, the "text" simply refers to the top-level node of the tree.

That is, one must be able to trace a line from all elements of the tree back upwards through the tree to the "text" node. The program maintains this rigor automatically.

3 File formats

There are three basic file formats:

- .txt (for plain, untagged text);
- .gen (for interlinear text), and
- .ltcx (for syntactically analyzed text).

The most important format, besides .ltcx, is the .gen format, which you can use to import language data from any language, so long as a Unicode font is available to display the language.

There is also a variation on the .ltcx file format, namely .ltc. It is for legacy support of previous versions of the program (prior to 3.0.0).

3.1 *.ltcx files

The .ltcx files hold both the words and the syntactic analysis of a piece of text. They are the resulting files when you save your work from LTC.

The .ltcx file format is XML-based, meaning programmers can easily write programs to read and manipulate the contents of the file. The format is fully documented inside each *.ltcx file itself.

3.2 *.txt files

You can import plain text files using the "File –> New" menu item. You will be asked, upon loading such a file, to tell LTC which font you want to use. The name of this font will be stored in the analyzed .ltcx file so that you don't need to specify it again when you load the analyzed file. LTC will assume that the text file is encoded in either straight, 7-bit ASCII, or UTF-8.

3.3 *.gen files

The .gen files are "Generic" files. You can use this format to load your own language data, provided they are in a word-per-record, SFM form. See Appendix C for information on the file format.

3.4 *.ltc files

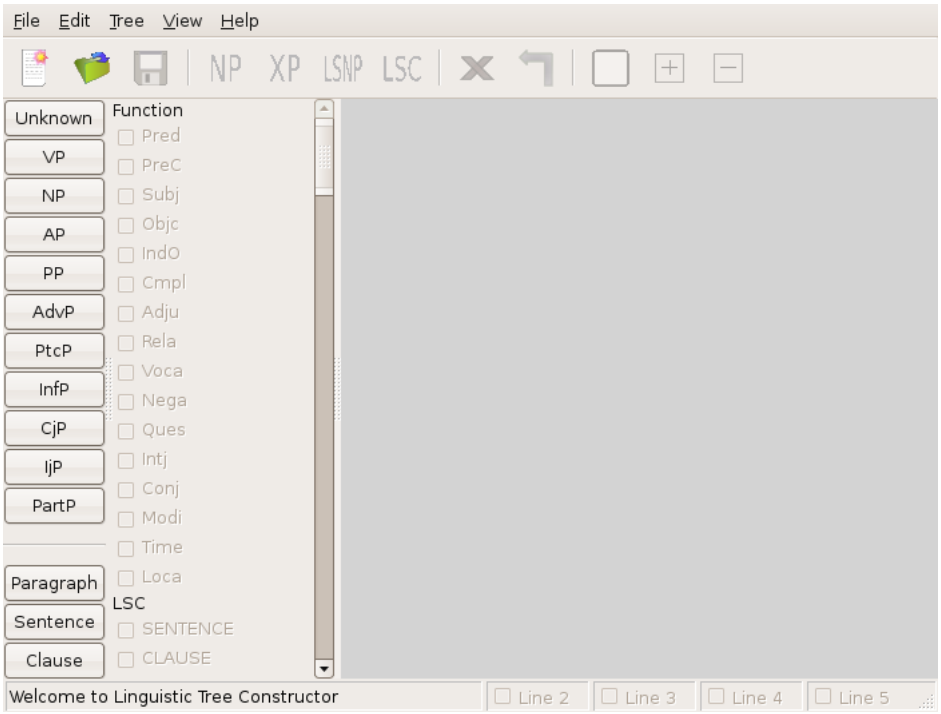
The .ltc files are the equivalent of the *.ltcx files, only they have been saved with a version of LTC prior to version 3.0.0. You can import these into LTC with the "File –> New" menu item. You cannot, however, save work to the .ltc format; only load it, then save as *.ltcx.

4 Using LTC

In this section, we describe how to use LTC.

4.1 Starting LTC

When you start LTC, you will see a window similar to this:



You can now start a new document by pressing the "New" button: , or by pressing the "Open" button .

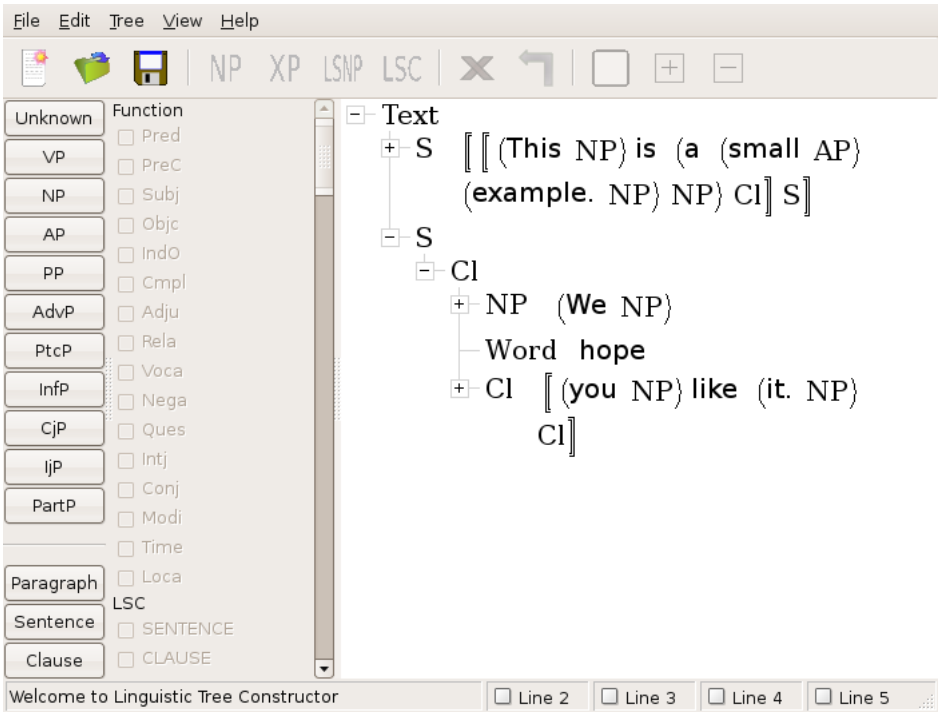
The "New" button  allows you to create a new document from one of the file formats described above.

The "Open" button  allows you to open a .ltcx file that you have saved with LTC before.

4.2 Layout of the main window

When working with an LTC document, you will see a main window which is divided into six areas:

1. The main part of the window, which is the tree-view.
2. The far left, which contains a number of buttons for making phrases and other nodes.
3. The middle left, which contains checkboxes for selecting labels for a node.
4. The toolbar, which is where most of the program's commands are activated from.
5. The status-bar, which is where you can see messages and manipulate the number of interlinear lines displayed.
6. The menu-bar.



These will be described in turn below.

4.2.1 The tree view

The tree-view is where most of your interaction with the program goes on. It contains a tree-view of the text much as the one you see in Windows Explorer. The things you can do to this view are:

1. Expand a node.
2. Collapse a node.
3. Select a node.
4. Deselect a node.
5. Select one node and simultaneously deselect all others.
6. Select a range of nodes.

To expand or collapse a node, click on the "plus" or "minus" next to the node itself:.

To select or deselect a node, press the left mouse-button on the node itself. This toggles its selected state on or off.

To select one node and simultaneously deselect all others, click on a node with the right mouse-button.

To select a range of nodes, hold down the shift-key and left-click a node. The selection will extend backwards to the nearest selected node, or, if there is no selected node at the same level in the backwards direction, forwards to the nearest selected node.

4.2.2 The tool-bar

The tool-bar is on the top of the window, beneath the menu-bar. It consists of a number of buttons. These buttons are for executing commands, such as "save file" or "make phrase" NP. You will learn more about these below.

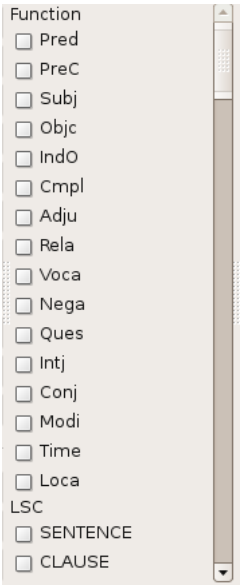
4.2.3 The node-panel

The node panel shows the phrase-types which are in the phrase_types.txt file (edit this file to change the phrase types), along with the "syntax nodes" which are in the syntax_nodes.txt file.

You can press any of the buttons to quickly create a phrase or a syntax node.

Please see Appendix B for more information on "phrase_types.txt" and "syntax_nodes.txt".

4.2.4 The label-panel



The label panel shows the labels which are in the labels.txt file (edit this file to change the labels).

The panel is inactive ("greyed out") at all times, except when there is precisely one node in the selection. In other words, if you have selected one node, and one node only, then that node's labels will be editable with this panel.

Simply click on a label's check box to turn it on or off. The tree-view will update itself with the new (or removed) label.

You can scroll the panel vertically to reveal more labels, if present in labels.txt.

You can turn the panel on and off in the View menu (View—Show label panel).

Please see Appendix B for more information on "labels.txt".

4.2.5 The status-bar

The status-bar is the line at the bottom of the window. Its function is twofold: First, on the left-hand-side, it displays a message pertinent to the current situation. This includes giving hints on what a particular command does, when hovering the mouse over its button.

Second, on the right-hand-side are four buttons labelled "Line 2" to "Line 5". These are for switching individual interlinear lines in the tree-view on or off.

4.2.6 The menu bar

The menu-bar is at the top of the window. It has five menus:

- File
  - (standard file-menu entries)
- Edit
  - Undo (Ctrl-Z)
  - Clear selection (Ctrl-D)
  - Copy (Ctrl-C)
  - Always clear selection after action
- Tree
  - Make Phrase
  - Make X-Bar
  - Make RRG LNSP
  - Make RRG LSC
  - Make Syntax Node
  - Make Delete selection

  - Move one upwards
  - Insert Null Element
  - Delete a word

- View
  - Colors
  - Interlinear Lines
  - Magnification
  - Brackets
  - Show label pane (Toggle)
  - Open horizontal tree (Ctrl-H)
  - Next horizontal tree magnification

- Help
  - Help Contents (F1)
  - About LTC

File menu The File menu is for manipulating files, and uses standard terminology.

Edit menu The Edit menu has three items: "Clear selection" (Ctrl-D), "Copy" (Ctrl-C) and "Undo" (Ctrl-Z).

The Copy command only works if you have selected exactly one node in the tree. It places the node and its descendants on the clipboard as a graphics image file. You can either paste it into your favourite word processor, or paste it into a graphics-manipulation program, such as the Paint application that comes with Windows.

You can toggle the "Always clear selection after action" item on and off. If on, LTC will always clear the selection after an action has been taken, e.g., after making a phrase, making a clause, or moving the selection once upwards. If this item is off, on the other hand, then the selection will, for most actions, not be cleared. Instead, after making a node, the new node will be selected. After moving the selection once upwards, the node(s) will still be selected, etc. The state of this item (on or off) will be saved between sessions, so this is really a configuration menu item. Which state best suits your needs and analysis-habits is up to you to decide.

Tree menu The Tree menu replicates most of the functionality of the buttons that make, delete, or move nodes.

Two menu-items in the Tree menu do not have toolbar-button-equivalents: "Insert Null Element" and "Delete a word".

The "Insert Null Element" inserts an item at word-level which looks like the symbol used in many branches of linguistics for null elements (a ring with a slash through it). The selection must contain precisely one node for this to work. The node need not be a word, but can be any node other than the Text. You will be

prompted to choose whether the new null element should be inserted "Before" or "After" the selected node. The new null element node will be inserted at word-level as a sibling of the selected node, i.e., it will have the same parent as the selected node, and will be inserted "Before" or "After" the selected node. This operation can, of course, be undone with the "Undo" functionality.

The "Delete a word" menu-item deletes a node at word-level. The selection must contain exactly one word for this to work, and the word must not be the only child of its parent. You will be asked whether you really, really want to delete the word. This operation can, of course, be undone with the "Undo" functionality.

View menu The View menu has seven items.

The first option, "Colors", lets you configure the colors of the various parts of the display. When you choose one of the sub-items of the "Colors" menu, you will be presented with a dialog box in which you can choose the name of a color. If you click "OK", the color of the chosen part of the display will be set to whatever you chose. If you click "Cancel", then nothing happens.

The second option, "Interlinear Lines", turns individual interlinear lines on and off. This does the same as toggling the checkboxes on the status bar.

The third option, "Magnification", is for setting the magnification of the tree-view. Thus with this menu, you can make the typeface larger or smaller in the tree-view.

The fourth option, "Brackets", is for setting the bracketing-options for the tree-view. Thus with this menu, you can turn bracketing off and on, and you can choose the highest and lowest level for which you wish to have brackets.

The fifth option, "Show label panel", is for toggling the label panel on and off.

The sixth option, "Open horizontal tree", is for viewing the current selection as a horizontal tree in a separate window. This command is only available if you have selected precisely one node, no more, no less. The new window produced by this command is independent of the main window, and can be moved, scrolled, and resized at will. The new window has a File menu which you can use to draw the window contents to the clipboard, or to close the window.

The seventh option, "Next horizontal tree magnification", is for changing the magnification of the next horizontal tree to be produced by selecting the previous menu-item. Think of it as a "default size" for the horizontal tree. Once the horizontal tree has been created, you can change its size "on-the-fly" with the View menu in the new window.

Help menu The "Help" menu at lets you see the Help file, and lets you see the "About-box". (On Mac OS X, the "About LTC" menu item is under the "LingTreeConstructor" menu, where you would expect it!)

4.3 Daily usage

In this section, we describe daily LTC usage.

4.3.1 Starting afresh (File—New)

This command can be reached in two ways: Either with the "New" button on the tool-bar  , or with the "File –> New" menu-item.

It clears the document at hand and loads in a new document. If changes have been made to the document, the program will ask first if you want to save the present document.

The following can be loaded:

- *.txt (Plain text files)
- *.gen (Interlinear text; see Appendix C)
- *.ltc (syntactically analyzed legacy LTC format)
- *.ltcx (the syntactically analyzed format of current versions of LTC)

4.3.2 Opening a previous session (File—Open)

This command can be reached in two ways: Either with the "Open" button on the tool-bar  , or with the "File –> Open" menu-item.

You can only open .ltcx files with this command, not, say, .txt or .gen files. If you want to start afresh from a .txt or a .gen file, use the "New" command  .

If changes have been made to the present document, you will be asked if you want to save them before loading the new document.

The side-panel may be affected by loading a *.ltcx file, since the labels, phrase categories and syntax node categories present in the *.ltcx file is used in the side panel after loading. Upon closing the document, the original, default values in the side panel are restored.

4.3.3 Saving your work (File—Save)

This command can be reached either from the tool-bar  or from the "File" menu. It lets you save your work in .ltcx format.

4.3.4 Exporting to MQL

LTC can export a document to Emdros MQL. Emdros (http://emdros.org) is a database engine for linguistically analyzed text. As such, it is ideally suited to making LTC documents searchable.

You export an LTC document by accessing the "File –> Export to Emdros MQL..." menu item. Since it exports a document, it is only available if a document is open. Otherwise, it is grayed out.

Accessing this menu item will bring up a dialog box in which you can choose the following:

- Output filename: Here you can set the name of the output MQL file. It defaults to the filename of the document, with the extension set to ".mql".

- Starting id_d: All objects in a given database need to have a unique integer id (id_d). This setting says which id_d to start the export at. This integer can be between 1 and 2,100,000,000. It is stored between sessions, such that the next export always starts at the highest id_d used for the last export, plus 1. That way, a series of exports will produce MQL files in which the different objects are unique across the database.

- Starting monad: All words in a document need a unique monad (integer). This setting says which monad to start the export at. This integer can be between 1 and 2,100,000,000. It is stored between sessions, such that the next export always starts at the highest monad used for the last export, plus 1. That way, a series of exports will produce MQL files in which the different documents are adjacent and non-overlapping.

- Whether to emit the schema or not. This should only be done for the first document to be imported into a new database. All other documents to be imported into a database should have this checkbox "unchecked" before exporting.

In order to use the MQL files, you should run the produced MQL files through the mql program (mql.exe on Windows). See the manual page for mql for how to use it.

In order to use a database produced from this output with the Emdros Query Tool, you need a .cfg file which matches the output data. A sample .cfg file for the Emdros Query Tool is provided with LTC.

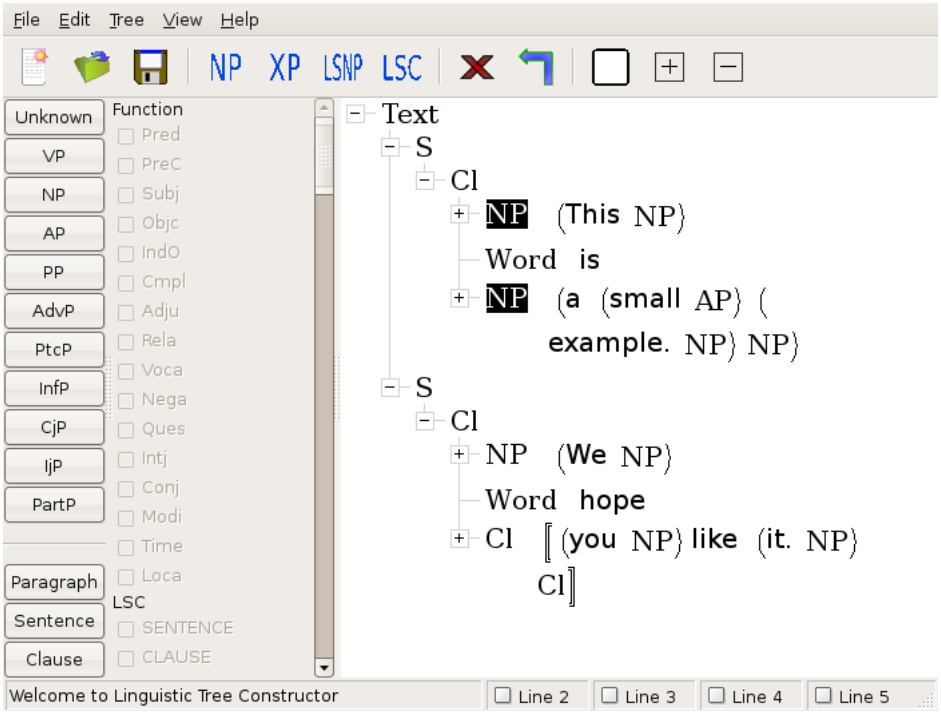
- On Windows, it is located in Files\LingTreeConstructor\examples\).

- On Mac OS X, it is located in the distribution .dmg image file, in examples/EmdrosQueryTool.cfg.

- On Unix/Linux, this is located in /usr/local/share/ltc/examples/EmdrosQueryTool.cfg.

4.3.5 The selection

The selection is all the highlighted nodes on the tree-view.



(In the above example, the two "Sentence" nodes are highlighted.

Selecting individual nodes You select or deselect individual nodes in the tree by clicking on them with the left mouse-button.

You can select one node and deselect all others by clicking on it with the right mouse-button. This is useful, for example, when going through the tree to see if everything is OK. It is also useful when you want to start a new grouping and wish to clear any previous selection before selecting the first node.

The importance of the selection The importance of the selection cannot be underestimated: Everything that happens in the program happens with or because of the selection. You make a phrase from the selection. You delete the nodes in the selection. You expand the selection recursively, and so on.

What you can't do Because of the nature of the immediate constituent model of language, it is impossible to make a new node from nodes that don't already **have the same parent**. Thus what happens when you make a node is that you push a node in between the nodes of the selection and their previous parent.

The program maintains this rigor by not letting you select nodes that don't have the same parent. If you try to select a node that doesn't have the same parent as the nodes that are already in the selection, a dialog-box will pop up with a brief explanation of the problem, and the program won't let you select the node you were trying to select.

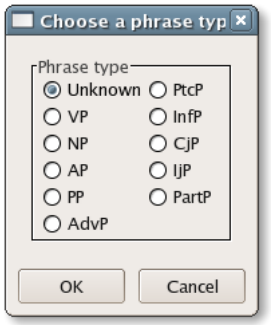
An easy way to overcome the problem is to right-click on the node you wish to select. Then the selection is cleared before the node is selected, thus getting around the problem.

4.3.6 Making a phrase

The "Make phrase" button on the tool-bar looks like this: NP.

This is not to suggest that all phrases are NPs, but the button had to have some kind of look, and NPs are almost the prototypical phrases.

To make a phrase, simply select the nodes in the tree-view which should be immediate constituents of the phrase, and click the "Make phrase" button. This will bring up a dialog-box with all the available phrase-types.



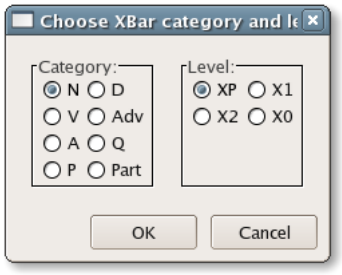
Select the appropriate phrase type. The click OK to make the phrase, or "Cancel" to return to the program without making the phrase.

You can control the categories which appear in this dialog box by editing the "phrase_types.txt" file. Please see Appendix B for more information.

4.3.7 Making an X-bar node

The "XP" button XP is for making "X-bar"-style tree nodes.

It works similarly to making a phrase, except that you get to choose both node category and node level.

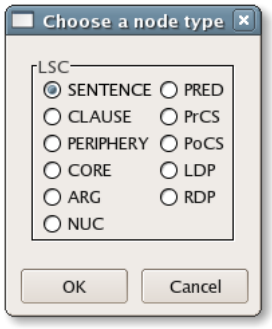


The intention is that **X'** should be read "X-bar", whereas **X"** should be read "X-double-bar".

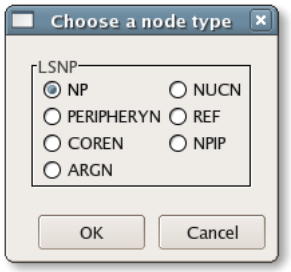
4.3.8 Making an RRG node

There are two RRG buttons:  for making the "Layered Structure of the Clause", and  for making the "Layered Structure of the NP".

The "LSC" button  brings up this dialog:



The "LSNP" button  brings up this dialog:



Both work similarly to making a phrase.

4.3.9 Deleting the selection

When a mistake is made in the analysis, it can be rectified by first deleting the offending nodes from the tree. This is done by selecting the nodes to be deleted, and then pressing the "Delete" button  .

What you can't do It is impossible to delete words with this command, and it is impossible to delete the Text. LTC will not let you delete the Text node. However, words can be deleted with the "Tree –> Delete a word" menu-item.

4.3.10 Moving the selection once upwards

The "Move once upwards" button on the tool-bar looks like this:  .

What it does This command moves the selection once upwards. Say that a node, C, is an immediate constituent of node B, which is an immediate constituent of node A. When moving C once upwards, C is first taken out of node B and is then placed in the tree so that its new mother is node A.

This command applies this procedure to all of the members of the selection, except the text node and those nodes which are already immediate constituents of the text node (if such nodes happen to be part of the selection). It doesn't make sense to move these once upwards, so it cannot be done. The computer will just silently not do it.

When it is useful This command can be useful when, for example, a post-positive conjunction, is embedded one step too low, and needs to be "taken out" at an outer level.

If you make a mistake If you make a mistake with this command, there are two ways to recover: Either:

- choose the "Undo" command, or perform these steps:
- Delete the constituent which previously was the mother of the constituent you moved, then make the mother again, this time including the constituent you moved.

4.3.11 Clearing the selection

This command can be reached in three ways:

1. From the "Clear" button on the tool-bar  ,
2. From the "Edit" menu, or
3. with the "Ctrl-D" key combination.

Nothing happens structurally to any node; this simply deselects all selected nodes.

4.3.12 Expanding the selection

When one thinks one is finished with one's analysis, it might be a good idea to check it. This can be done by selecting top-level nodes (or, indeed, the text node) and "Expanding the selection recursively downwards". The button for doing this looks like this:  .

Clicking this button expands all selected nodes right down to the level just above the word-level, and clears the selection. One can then step through the tree using the right mouse-button and check that all constituents are as they should be.

4.3.13 Collapsing the selection

The opposite of expanding recursively downwards is "Collapsing recursively downwards". The button for doing this looks like this:  .

Clicking this button collapses all selected nodes right down to the word-level, and clears the selection.

4.3.14 Undo (Edit—Undo)

It is possible to undo anything you have done to the tree. This can be done right back to when you started the program. Press Ctrl-Z or choose Undo from the Edit menu.

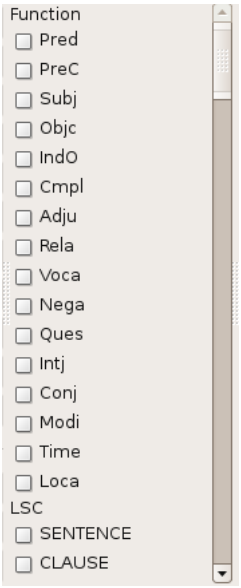
4.3.15 Adding or removing labels

There are two ways to add or remove labels from a node in the tree:

1. If you shift-right-click on any node in the tree, you will get a menu from which you can choose a label set.

   From one of these menus, you will be given the opportunity to toggle individual labels on or off.

2. The easier way, however, is to use the label panel at the left of the window:



The label panel is only active when you have selected one node, and one node only. If you select more than one node, or if no nodes are selected, then the label panel is inactive ("greyed out").

You can toggle as many labels as you want for any given node.

4.3.16 Keyboard shortcuts

You can use the keyboard shortcuts defined in the following table:

Key	Command
P	Make a phrase
Y	Make an syntax node
X	Make an X-bar node
N	Make an RRG LSNP node
S	Make an RRG LSC node
Del	Delete the current selection
Ctrl-D	Clear the selection
U	Move one upwards

5 Appendix A: SFM files

SFM files are an old SIL standard for encoding textual information. The acronym stands for "Standard Format Marker". It is a line-oriented, text-based file format. Each line that bears information begins with a "field marker", which consists of a backslash ("\") followed by one or more letters. After the field marker is a space, followed by the content of the field. The field continues until the end of the line. If the field value is empty, there need not be a space after the field marker.

Records consist of one or more fields, and can optionally end with a "record end" marker. This is usually signalled with the field "\re" without any value.

Blank lines and lines which do not start with a field marker are ignored.

In LTC, the .gen file format is an SFM file format, as are the library files described in Appendix B.

6 Appendix B: library files

This appendix describes the various "library files" that are used with LTC.

6.1 Location

The library files are looked for by LTC in two places. First, the program looks in a configuration directory (specified below) which the user can control. If the file is found there, the program looks no further. If the file is not found there, however, the program looks in a directory (specified below) which the program controls.

- The user-controlled directory is located as follows:
  - **On Windows XP:** In the directory "C:\Documents and Settings*user name*\Application Data\LingTreeConstructor"
  - **On Windows Vista:** In the directory "C:\Users*user name*\Application Data\LingTreeConstructor"
  - **On Mac OS X:** In the directory~/Library/Application Support/LingTreeConstructor

  - **On Unix/Linux:** In the directory ~/.ltc

- The program-controlled directory is located as follows:
  - **On Windows:** In the lib\ directory underneath the installation path. Typically, it will be "C:\Program Files\LingTreeConstructor\lib"
  - **On Mac OS X:** In the "Contents/Lib" directory underneath the "LingTreeConstructor.app" folder. Typically, it will be "/Applications/LingTreeConstructor.app/Contents/Lib".
  - **On Unix/Linux:** In the /usr/local/share/ltc/lib directory.

The user may copy the files from the program-controlled directory to the user-controlled directory, and edit them further there, or start new ones from scratch.

All of the files are "SFM files" (please refer to Appendix A for a general introduction to SFM files).

The program will never overwrite the files in the user-controlled directory, whereas the files in the program-controlled directory are overwritten each time the program is reinstalled.

6.2 phrase_types.txt

The "phrase_types.txt" file is an "SFM File", where each line signfies either one phrase type or the tool tip of a phrase type. Each line signifying a phrase type starts with "\pt" and a space ("pt" for "phrase type"). Then comes the name of the phrase type. The name must be encoded in the ASCII character set. Each line signifying a tool tip must start with "\ptt" and a space, followed by some free-form text describing the phrase type on the previous line. The tooltips will show up in the program when one hovers the mouse over the button with that phrase type.

Example:

```
\pt VP
\ptt Verb phrase
\pt NP
\ptt Noun phrase
\pt AP
\ptt Adjective phrase
\pt PP
\ptt Prepositional phrase
.
.
.
```

The phrase type names must be "C identifiers". This means that they must start with one of the letters A-Z, a-z, or an underscore, and the rest of the characters in the name must be one of these characters, or the characters 0-9.

For example, "VP" is a valid "C identifier", as is "VP2", whereas "1V" is not, since it does not start with one of the letters A-Z, a-z, or an underscore.

6.3 syntax_nodes.txt

The "syntax_nodes.txt" file is an "SFM File", with two adjacent lines describing one syntax node level.

The first line in each pair must have a backslash, followed by "ls" (for "short label"), followed by a space, followed by a short label to be used in the tree for the syntax node level.

The second line in each pair must have a backslash, followed by "ll" (for "long label"), followed by a space, followed by an arbitrary label to be used in the side-panel for buttons with which one can make the syntax node level.

Example:

```
\ls S
\ll Sentence
\ls Par
\ll Paragraph
.
.
.
```

All syntax node namdes must be "C identifiers". This means that they must start with one of the letters A-Z, a-z, or an underscore, and the rest of the characters in the name must be one of these characters, or the characters 0-9.

For example, "Par" is a valid "C identifier", as is "S2", whereas "1S" is not, since it does not start with one of the letters A-Z, a-z, or an underscore.

6.4 labels.txt

The labels, which are accessible by shift-right-clicking, are stored in the "labels.txt" file. The format can be exemplified as in this example:

```
\lsn Function
\l Pred
\lt Predicate
\l Subj
\lt Subject
\l Objc
\lt Object
\l Cmpl
\l Adju
\l Rela
\l Time
\l Loca
\re
```

The file is an SFM file with records that start with an \lsn field (label set name), followed by one or more label fields (\l), each optionally followed by a

corresponding tool tip field (\lt) belonging to the previous \l field, and ended by a "record end" field (\re). See Appendix A for a description of the SFM file format.

Only the following characters may be used in a label or label set name:

- The letters A-Z
- The letters a-z
- The underscore
- The numbers 0-9

Both labels and label set names **must** begin with a letter or an underscore.

The \lt "tool tip" fields will show up when one hovers the mouse over a label in the panel containing all the labels. The contents of these fields can be free-form, but the content ends at the first line break.

7 Appendix C: *.gen file format

*.gen files must be word-per-record SFM files, and may only use the two encodings "UTF-8" and iso-latin-1 (also known as iso-8859-1). Each record must have a "record end" field. The file must contain a "header" such as the following, anywhere in the file:

```
\wordfield wd
\glossfield gl
\tagfield tg
\transliterationfield tr
\lemmafield lm
\recordend re
\foreignfont Galatia SIL
\glossfont Times New Roman
\transliterationfont Galatia SIL
```

This is explained in the following table:

Field	Meaning	Default value
\wordfield	The SFM field-marker used for the word	wd
\after_punctuationfield	The SFM field-marker used for adding punctuation to the word after the surface	after_punct
\glossfield	The SFM field-marker used for the gloss (if any)	gl
\tagfield	The SFM field-marker used for the tag (if any)	tg
\transliterationfield	The SFM field-marker used for the transliteration (if any)	tr
\lemmafield	The SFM field-marker used for the lemma (if any)	lm
\recordend	The SFM field marker used to signal "record end".	re
\foreignfont	The font of the word and the lemma.	Arial
\glossfont	The font of the gloss	Arial
\tagfont	The font of the tag	Arial
\transliterationfont	The font of the transliteration	Arial
\righttoleft	This file is to be displayed right to left	N/A
\lefttoright	This file is to be displayed left to right	N/A
\wordfieldisUTF8	The word field uses the UTF-8-encoding (if not defined, the field is assumed to be in iso-latin-1 encoding). If this is defined, then the lemma field will need to be UTF-8 as well, since the wordfield and the lemma field share the same font.	Not defined
\glossfieldisUTF8	The gloss field uses the UTF-8-encoding (if not defined, the field is assumed to be in iso-latin-1 encoding.)	Not defined
\transliterationfieldisUTF8	The transliteration field uses the UTF-8-encoding (if not defined, the field is assumed to be in iso-latin-1 encoding). ¡td¿Not defined	
\tagfieldisUTF8	The tag field uses the UTF-8-encoding (if not defined, the field is assumed to be in iso-latin-1 encoding.)	Not defined

None of the fields are mandatory. The missing fields will be given default values. For righttoleft and lefttoright, there is no field value; you just write "\righttoleft" or "\lefttoright" on a separate line to indicate which reading-direction you want. The default is "\lefttoright".

The "header" may appear anywhere within the file: At the top, at the bottom, interspersed with the language data. The file is read twice, first the header is extracted, and then, starting from the beginning, language data is extracted. Each time, fields that are unknown are ignored.

The "after_punctuationfield" field is for placing stuff into the word that belongs together with the word, but which is separate from the surface.

The language data may, for each record, have the fields in any order, but you must end the record with the field specified with "recordend" (e.g., \re).

A two-word database coming after the above example header might look like this:

```
\wd XXX
\gl The gloss of XXX
\tg pronoun
\tr xxx
\lm X
\re

\wd YYY
\gl The gloss of YYY
\tg verb
\tr yyy
\lm Y
\re
```

TIP: If you have a reference field, but not a transliteration field, tag field, or gloss field, you can put the reference field in one of those lines by setting, e.g., the \transliterationfield to the "ref" field marker.

7.1 Syntax in the .gen file format

There is more to the .gen file format: It supports limited import of objects above word level, i.e., syntax-level objects.

Consider the following table:

Field	**Meaning**	**Default value**
\beginfield	The SFM field-marker used for beginning an object	begin
\endfield	The SFM field-marker used for ending an object	end

Say the defaults are used, then the following .gen file is valid:

```
\wd I
\begin Sentence
\begin Clause
\re

\wd am,
\end Clause
\end Sentence
\re

\wd and
\re

\wd you
\begin Sentence
\begin Clause
\re

\wd are.
\end Clause
\end Sentence
\re
```

This imports a sentence, "I am, and you are.", with the following bracketing:
[S [Cl I am,] and [Cl you are.]]
The following rules apply:

- The name of the syntax level from which an object must be created must be given after the "begin" or "end" SFM marker. Example: "\begin Sentence".
- \begin starts an object of the specified kind on the left side of the current word (right side if we are doing right-to-left).
- \end ends an object of the specified kind on the right side of the current word (left side if we are doing right-to-left).
- If two or more objects end on the same boundary, their "\end" SFMs must be specified in the order that is opposite that in which they were created. In the example above, the Sentence is created first, then the Clause for the word "I"; therefore, the Clause must be "\end"ed first, then the "Sentence" can be ended; this is the order in which it is done on the word "are."
- If you specify "Clause" after "\begin" or "\end", a Clause object will be started or ended, respectively. The comparison is done case-IN-sensitively, so "clause" will also do.
- If you specify "Clause Cluster" or "ClauseCluster" after "\begin" or "\end", a Clause Cluster object will be started or ended, respectively. The comparison is done case-IN-sensitively, so "clause cluster" will also do.

- All other strings will be interpreted as "Syntax Nodes", and a corresponding entry must exist in the "syntax_nodes.txt" file (see Appendix B). The Long name must be specified.
- The string that is used to "\end" an object must be exactly the same as the string that was used to "\begin" the object.
- The name "Text" is special, and represents the Text node, so it must not be used after "\begin" or "\end".

Note that you cannot use this method to create discontiguous constituents. Those can be created later, once the input has been loaded into LTC.