LPGlab manual
1.0

Contents

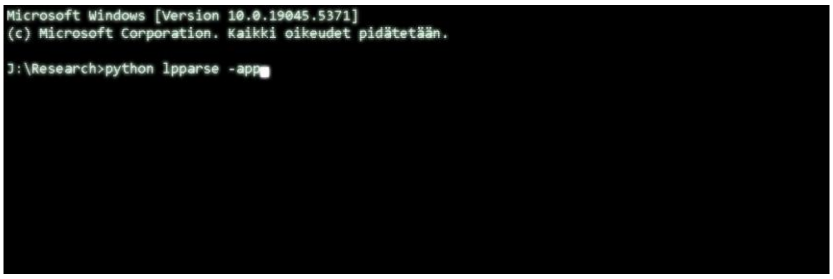
1 Purpose ... 2
2 Installation ... 2
3 Executing a study ... 2
4 Drawing phrase structure trees from scratch ... 3
4.1 Saving and cropping images ... 5
4.2 Moving and manipulating nodes ... 6
4.3 Zooming in and out ... 6
4.4 Compressing the phrase structure ... 6
4.5 Changing the label of a primitive or complex node ... 8
4.6 Adding words, glosses and features ... 10
4.7 Dependencies ... 12
4.8 Special shapes on the image ... 15
4.9 How to use special symbols for text ... 16
4.10 Some further tips ... 17
5 Examining the output of the linguistic model ... 18
5.1 Introduction ... 18
5.2 Visualizing a single phrase structure ... 18
5.3 Inspecting properties of the phrase structure ... 20
5.4 Examining the whole derivation ... 21
5.5 Creating raw output image data while running a whole study ... 22

1 Purpose

The LPGlab (”linear phrase grammar lab”) is a graphical interface component originally created to simplify certain recurrent tasks when working with LPG, such as inspection and creation of phrase structure tree images, inspecting whole derivations in a visual form, selecting single items in the dataset for analysis, and others. This document provides a user manual for version 1.0.[1] Only the basic features are documented here, and some functions that appear in the menu are experimental, do not yet work or have not been properly tested; they are not documented. The software is currently improved and debugged only on the basis of actual use, and is not maintained in any systematic way.

2 Installation

The application comes bundled with the LPG system itself. Installing LPG installs the GUI component. Once installed, it can be launched by writing **python lpparse -app** into the command prompt. The user must have Python (3x) installed and configured in the local machine. No external third-party libraries are required, since the software was written by using Python’s own **tkinter** library.

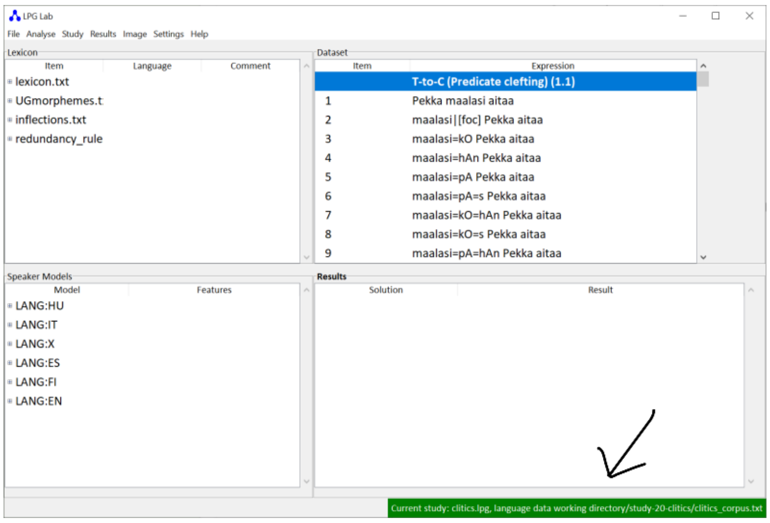


```
Microsoft Windows [Version 10.0.19045.5371]
(c) Microsoft Corporation. Kaikki oikeudet pidätetään.

J:\Research>python lpparse -app
```

3 Executing a study

It is possible to execute one complete study from the LPGlab menu **Study > Run Study**. Notice that the user interface will currently become unresponsive when the study if executed. The study that will be executed is the one defined by the $**app_settings.txt** configuration file in the root application directory pointing to the actual study configuration file which then defines the location of the dataset, lexicons and other auxiliary files. This information is visible at the bottom right corner of the application window:

[1] The application currently comes bundled with the LPG theory itself and is written to read output from the LPG. The two should eventually be separated into different applications, and then a standard phrase structure protocol (notation) should be created that enables communication between the two.



Item	Language	Comment
lexicon.txt		
UGmorphemes.t		
inflections.txt		
redundancy_rule		

Dataset

Item	Expression
	T-to-C (Predicate clefting) (1.1)
1	Pekka maalasi aitaa
2	maalasi|[foc] Pekka aitaa
3	maalasi=kO Pekka aitaa
4	maalasi=hAn Pekka aitaa
5	maalasi=pA Pekka aitaa
6	maalasi=pA=s Pekka aitaa
7	maalasi=kO=hAn Pekka aitaa
8	maalasi=kO=s Pekka aitaa
9	maalasi=pA=hAn Pekka aitaa

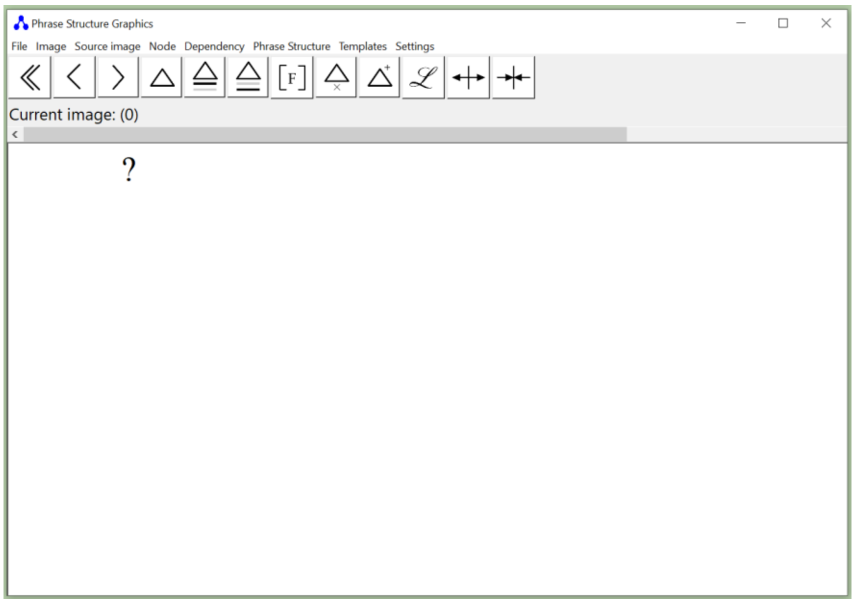
Model	Features
LANG:HU	
LANG:IT	
LANG:X	
LANG:ES	
LANG:FI	
LANG:EN	

Solution	Result

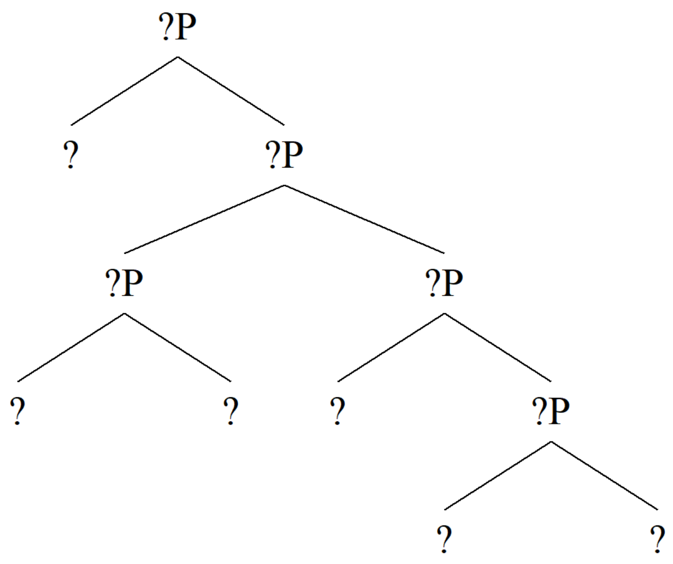
Current study: clitics.lpg, language data working directory/study-20-clitics/clitics_corpus.txt

4 Drawing phrase structure trees from scratch

One function of LPGlab is to allow the user to draw standardized phrase structure tree images. The function can be activated from the main menu **Image > New Image.** This launches a separate phrase structure image generation windows with one **?** drawn initially on the canvas.

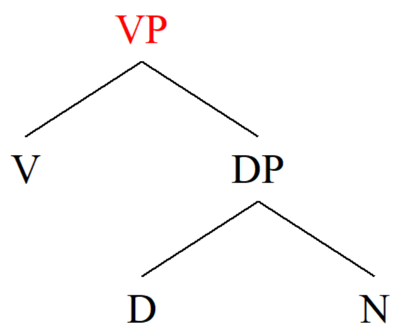

Phrase Structure Graphics
File Image Source image Node Dependency Phrase Structure Templates Settings
[F]
Current image: (0)
?

We can then create a new phrase structure by expanding from the ?. To do this, click ? or any other node on the canvas and press **E** to “expand” each node (or select **Phrase Structure > Expand > Phrase** from the menu). This creates two daughter nodes. By repeating the same procedure it is possible to create complex phrase structure images, like so:

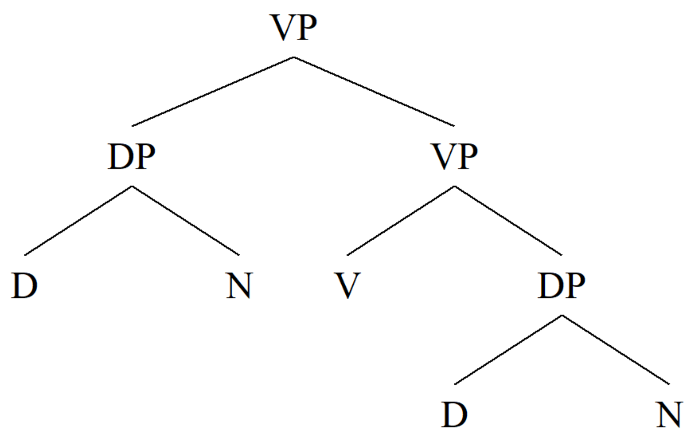


It is possible to edit the image by using either shortcuts or by the menu items above the canvas itself. These operations are explained in the sections that follow.

The program is able to generate several standard template structures for easier editing and a convenient starting point. Select **Templates** from the menu and then the desired item (currently **Basic XP**, **VP**, **vP**, **TP**, **CP**). Whatever else exists at the canvas will be deleted. It is also possible generate standard complex constituents directly into an existing structure by selecting a node and then **Phrase Structure > Add Left…** and then select the appropriate item from the list. For example, selecting DP from this list creates the following effect. Suppose the original phrase structure is



with VP selected. If we then select **Phrase Structure > Add Left… > DP**, the result will be



where the new DP was added to the left of the selected VP. It is not currently possible to add elements directly "to the right" by using this method. Instead, first add the structure to the left and then flip it by **Phrase Structure > Flip (structure)**.

Phrase structure image created in this way out of nothing does not have a real phrase structure representation underlying it, rather it is a pure phrase structure image, thus an illustration. On the other hand, if the image is created from the output of the model, as explained later in this document, then the graphical image should be considered as "extended projection" of the original underlying phrase structure. There will be, in effect, two phrase structure objects behind the image: the original linguistic phrase structure and the image. In this case all editing will be done "over" the original phrase structure, and it is also possible to revert back to the original (**Phrase Structure > Recover Original**).[2]

4.1 Saving and cropping images

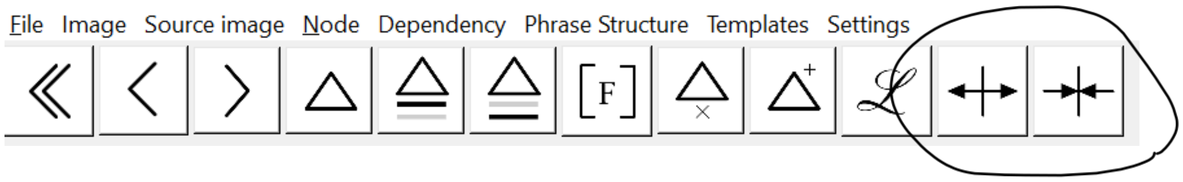
The phrase structure image can be saved by **File > Save** or **File > Save As…** This saves the image in its internal format for later editing and not as an image. The underlying Python library limits image saving to postscript format which is not very useful for most users (**File > Save postscript image…**).[3] The easiest way to export the image from the canvas (in Windows) is to crop it by pressing **Shift + ⊞ (windows key) + S** and selecting the relevant area from the screen by using the mouse. The image is saved to the clipboard, and can be pasted to an external program such as Word.

[2] The original purpose of this program was to generate phrase structure images from the output of the algorithm and then fine-tune them for publication.

[3] The font size in the postscript image does not match with font size displayed on the screen. I have not been able to correct this.

4.2 Moving and manipulating nodes

The tree drawing algorithm tries to generate a balanced phrase structure image from the original source structure without too much cluttering or overlap. In some cases, especially if the phrase structure is complex, the algorithm produces nonoptimal results. It is possible to move the nodes around in several ways. First, we can expand and shrink the daughters (i.e. moving them either farer apart or closer together) by using the buttons shown here:



The first button will expand them, the latter will shrink. The same operations can also be triggered from the menu (**Node > Constituent shape > Squeeze/Widen/Make symmetric**). A selected node can be moved around by using the arrow keys (**left**, **right**, **up**, **down**), by dragging with mouse, or by using the menu (**Phrase Structure > Up/Down/Left/Right**). Menu item **Image > Recalculate layout** calculates the geometrical relationships anew for the whole phrase structure, thus it erases all changes.

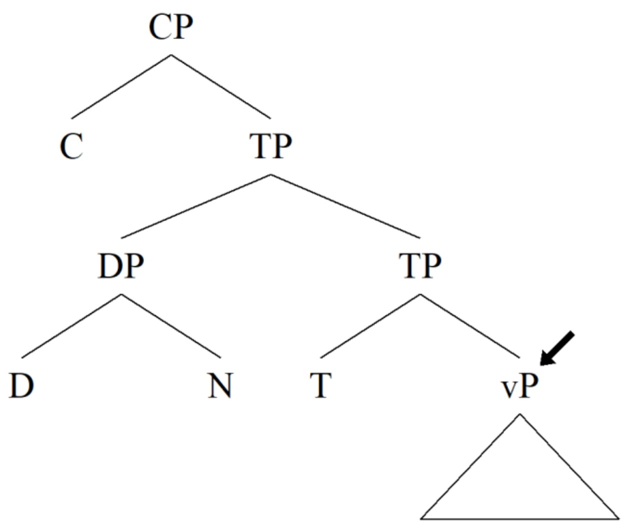
We can also flip the left and right constituent (**Phrase Structure > Flip(structure)/Flip(presentation)**). When we flip the *structure*, the left and right constituents simply change place in the underlying phrase structure representation. If we flip for *presentation*, the left and right constituents retain their asymmetry at the level of the phrase structure and linguistic theory (e.g., for labeling), but they are drawn in reverse order for illustration. The latter feature allows one to compress the images on its x-axis if they expand too much to some one direction.

4.3 Zooming in and out

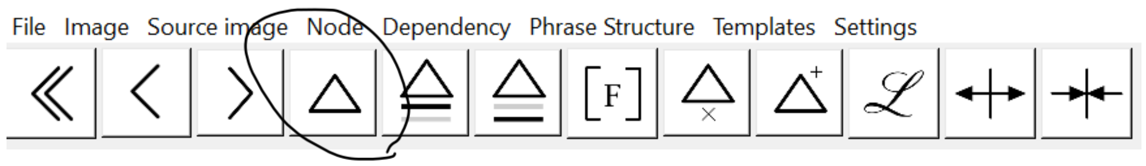
It is possible to scale (zoom) the image by **Ctrl + Mouse wheel**. This is useful if the image is so large that it does not fit into one page, though in such cases it is usually much better to compress some parts of the image (4.4).

4.4 Compressing the phrase structure

When the complexity of the image increases, it usually becomes harder to read. In addition, part of it might get hidden outside the boundaries of the canvas (it is possible to see such parts by using the scroll bars). There are several ways of compressing the image into a smaller and more readable format. The primary method is to replace a complex constituent with a compressed triangle notation as shown here:



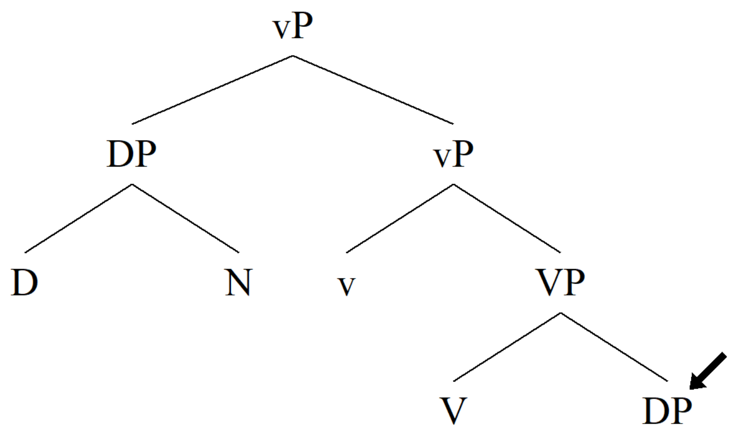
This was created by selecting the node (here vP, shown by the arrow) and selecting **Node > Compress (triangle)** from the menu or by pressing the following button:


File Image Source image Node Dependency Phrase Structure Templates Settings
[F]

The operation can be cancelled by selecting **Node > Decompress**, which restores the original decompressed format. Pressing **C** ("c̲ompress") shifts between compressed and decompressed formats.

Another option for compression is deletion. Select the node to be deleted and use **Phrase Structure > Delete** to delete it. The operation is permanent and cannot be reversed without creating the deleted constituents from scratch.

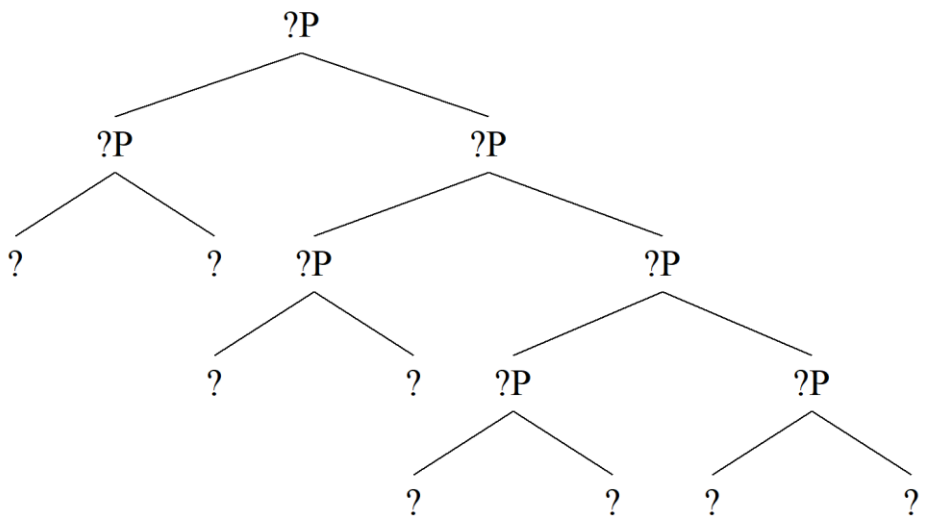
In some cases we want to represent a single complex node by using only its label, like so:



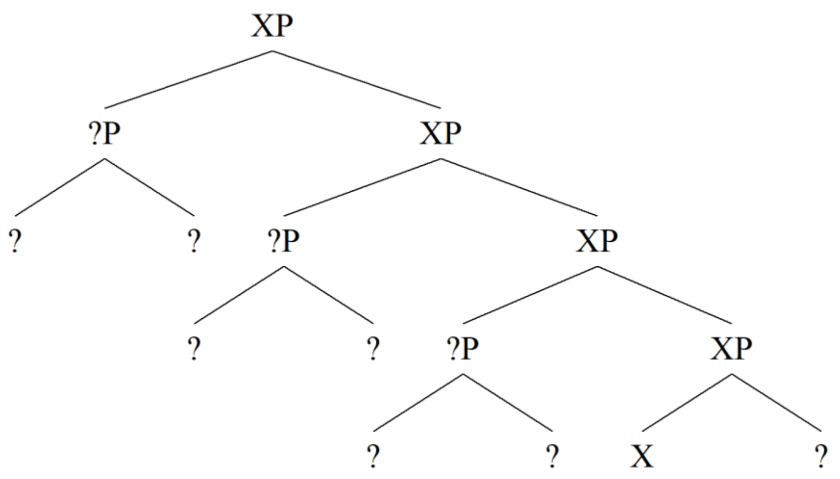
One way to do this is to shrink the original complex constituent (**Phrase Structure > Shrink**) into a primitive constituent and then use a custom label, here **DP**. Generation of custom labels is explained in 4.5, but the menu item is **Node > Custom label…** This will treat the node as a primitive head, however, so in some cases also the labels of local complex nodes must be changed accordingly. Shrinking is a permanent operation that destroys the complex target node and leaves only the primitive node.

4.5 Changing the label of a primitive or complex node

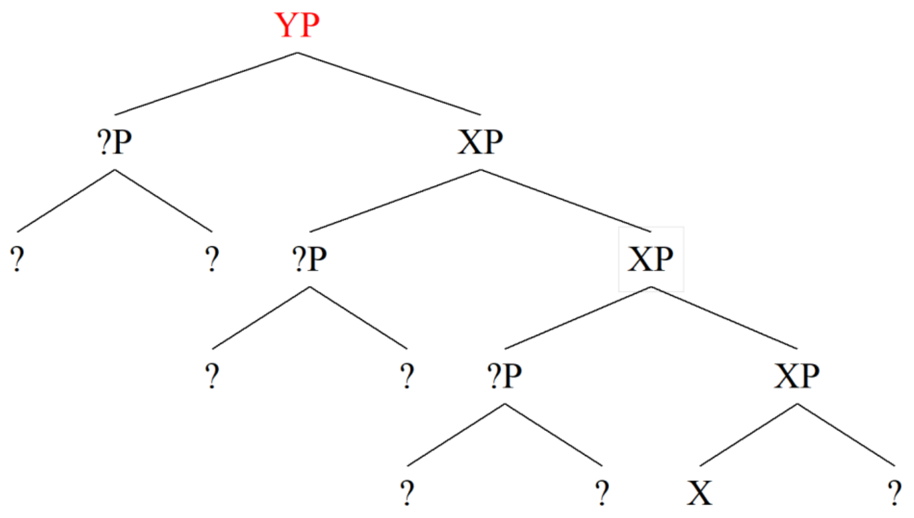
There are two ways to change the label of a node in the image. We can either change the original label that is part of the underlying phrase structure, or we can generate a "surface" custom label that overrides the linguistic theory and all label calculations. Thus, if we change original, linguistically relevant label of a constituent, the label algorithm will calculate labels for all complex nodes that are affected by the change. Suppose we have created the following initial phrase structure sketch:



If we select the lowest left node and change its original label into X, for example by clicking on the node and then selecting **Node > Change original label…** and writing **XP** to the dialog that opens us, the result will be

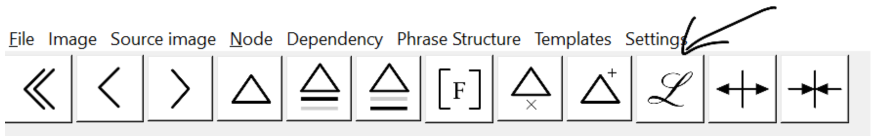


where the label gets inherited by all dominating nodes due to the interaction between the underlying linguistic labeling algorithm and the geometry of the phrase structure itself. The same effect can be created by pressing **Ctrl+L**, which is a shortcut for the menu selection just described. We can also change the label manually by creating a custom label that overrides the linguistic labeling. For example, if we click on the highest XP, select **Node > Custom label > New** and then provide **YP** into the dialog box, the result will be



where the custom label YP overrides the XP. It is of course possible to provide all labels by using custom labels, which will override anything calculated on the basis of the linguistic theory. We can revert back to the original label by **Node > Custom label >**

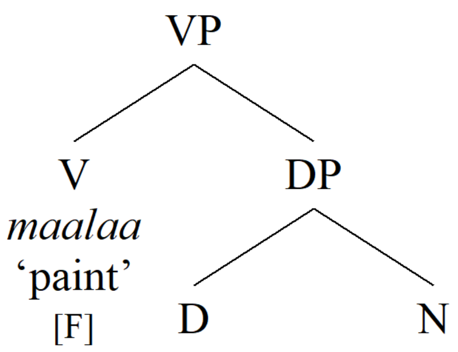
Original. Custom labels can be generated by pressing **Ctrl+L** or by using the L-button in the menu bar:



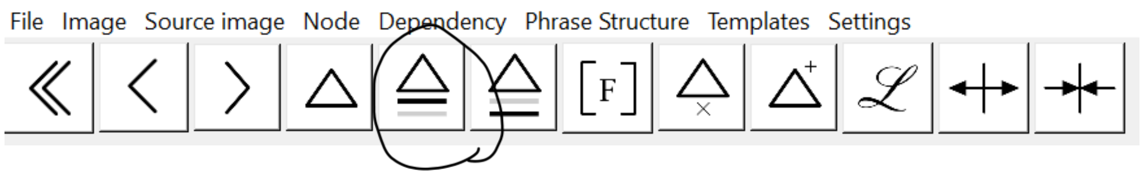
If the initial phrase structure image comes from the output of the model, as explained in another section, then the labels are provided according to how they are provided in the output and calculated on the basis of the labeling algorithm. What we see, then, is what the empirical theory predicts. The user can then use custom labels to override the linguistic labels or even to change the original labels, as explained above, though the latter is not recommended since it in essence means that we would be tampering with the raw output data. Custom labels can be anything: complex nodes, for example, do not have to have "P" applied to them. The label can also be empty (**Node > Custom label > Empty**).

4.6 Adding words, glosses and features

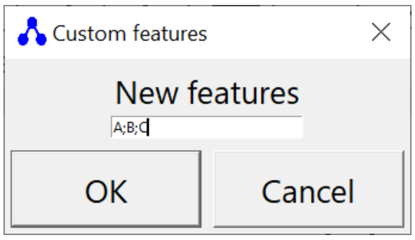
Primitive nodes can be equipped with words (phonological material), gloss and features. The user can select the items presented, but they will always occur in the same order word – gloss – features. For example:



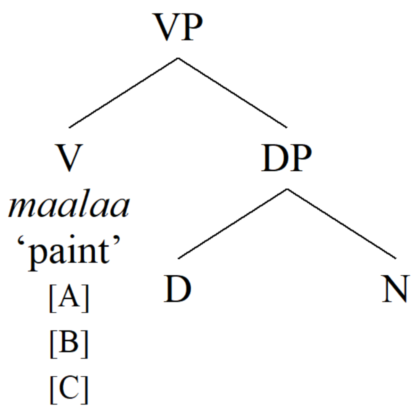
To add a word, select the node and then **Node > Phonology > New** or select the button



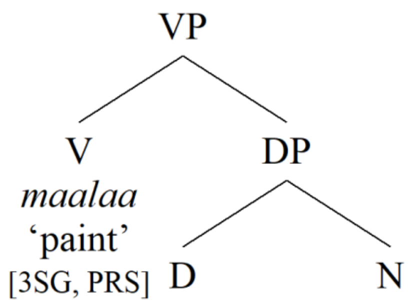
Both will open a dialog where the user can write the word to be displayed, such as *maalaa* in the above example. Glosses and features can be added in the same way by using the two buttons to the right of the above button (gloss, features) or by selecting the corresponding items from the menu. If you want to display several features, they must be separated by semicolon in the dialog. Thus, writing



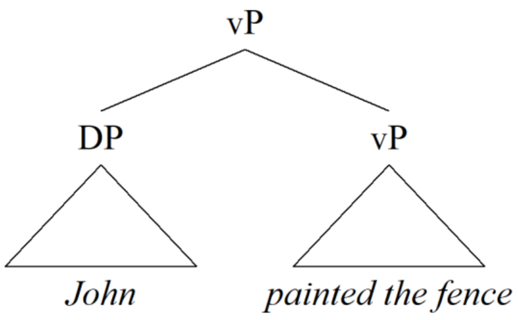
will generate



To generate several features one the same line, use for example regular comma to separate the features. This will create examples such as



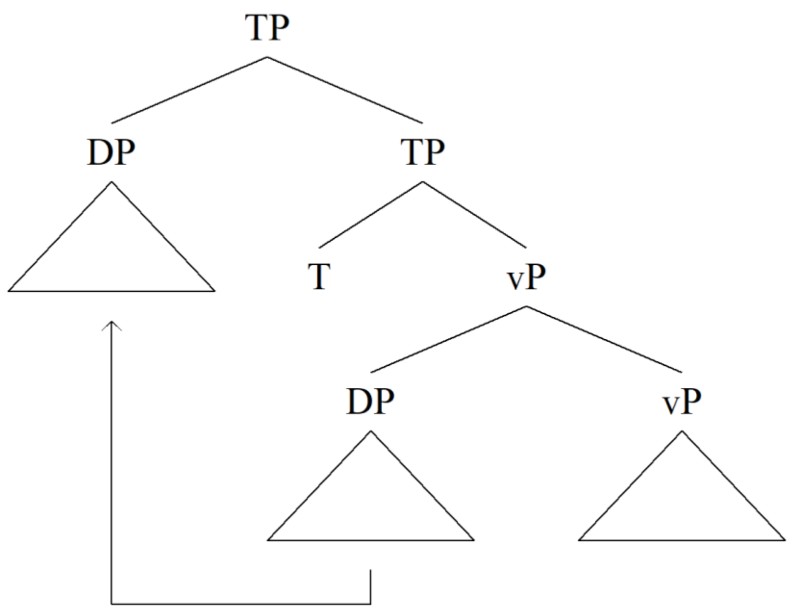
The positioning and the look of this information cannot be changed from the GUI (version 1.0), and must be done at the level of the source code or at the level of study configuration file.[4] It is not possible move the text fields around, they are always decoupled with the node. In the case some nonstandard information must be added to the image, it can be done in an external graphics program: crop the image to the clipboard (4.1) and paste it into an external graphics program. To use special symbols such as Greek alphabet, see 4.7. The same information can also be generated for compressed nodes. Select the compressed node and provide the required fields:



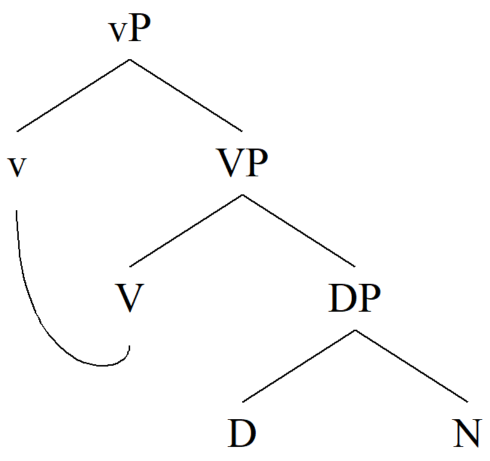
4.7 Dependencies

It is possible to draw dependencies between nodes (and generate them from the output of the underlying linguistic model). To draw a dependency between two nodes, select them both by pressing **Ctrl**. Both nodes should appear in red. The node selected first will be the *first node*, the node selected second the *second node*, so the order is not irrelevant. Next, select **Dependency** from the menu. The first five items shown in the submenu allow one to draw five types of dependencies between the selected nodes. For example, selecting **Forward Chain** will draw an arrow from the first node into the second node.

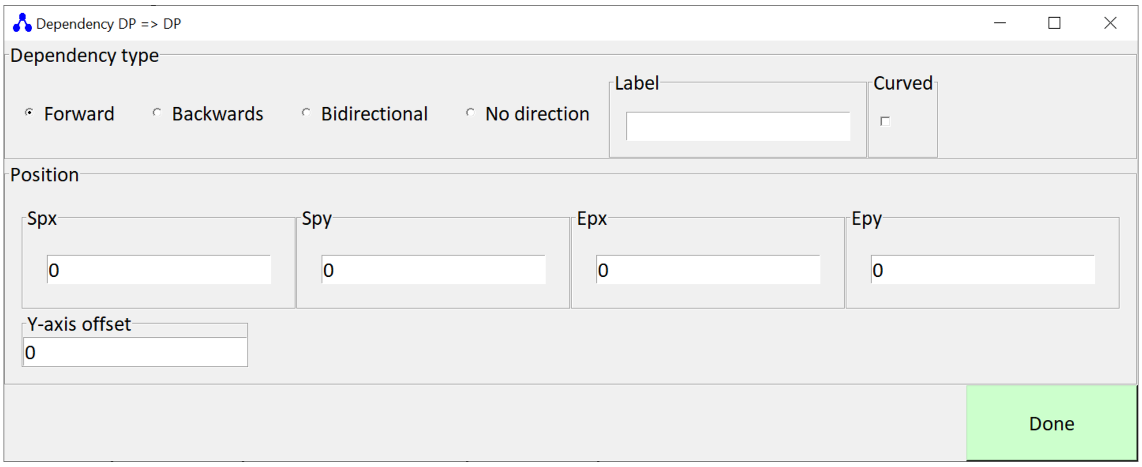
[4] The definitions are in the module **GUI_phrase_structure_canvas.py**, class **PhraseStructureCanvas** and variable **self.label_style** which currently defines fonts and sizes. The same values can be provided also in the study configuration file for parameters **image_parameter_font** and **image_parameter_tsize**, but these regulate all fonts and all the text sizes.



Other dependency types are **Backward Chain** (arrow from second to first), **Bidirectional Chain** (arrow into both directions), **Directionless Chain** (no arrow) and **Curved Chain** (changes the shape)**.** A curved dependency looks like this:



To delete a dependency, select it first and then **Dependency > Delete**. There is also the option **Dependency > Delete All**. We can add a label to the dependency by selecting the dependency of interest by mouse, then **Dependency > Modify…** which brings up a dialog where we can change the properties of the selected dependency:



Dependency DP => DP

Dependency type

[✓ Forward] Backwards Bidirectional No direction

Label:

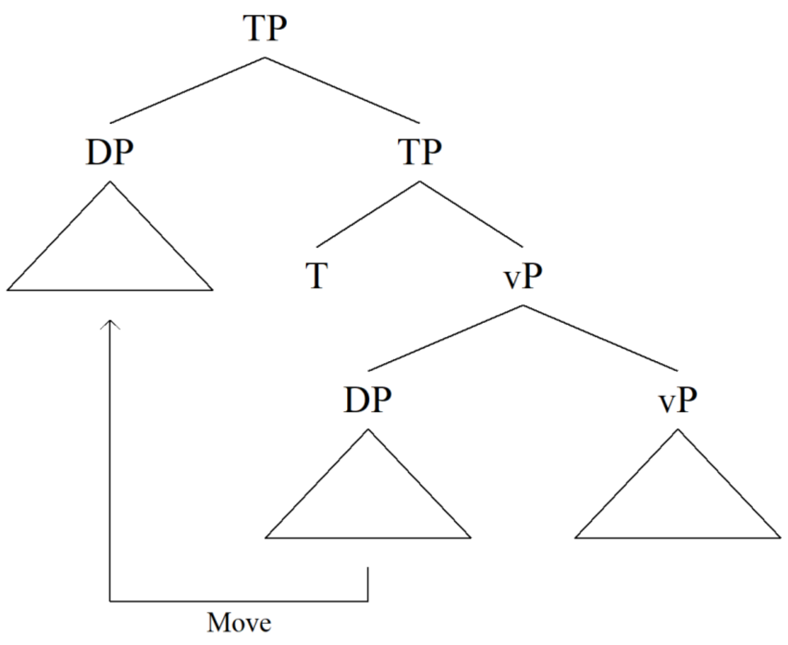
Curved: ☐

Position

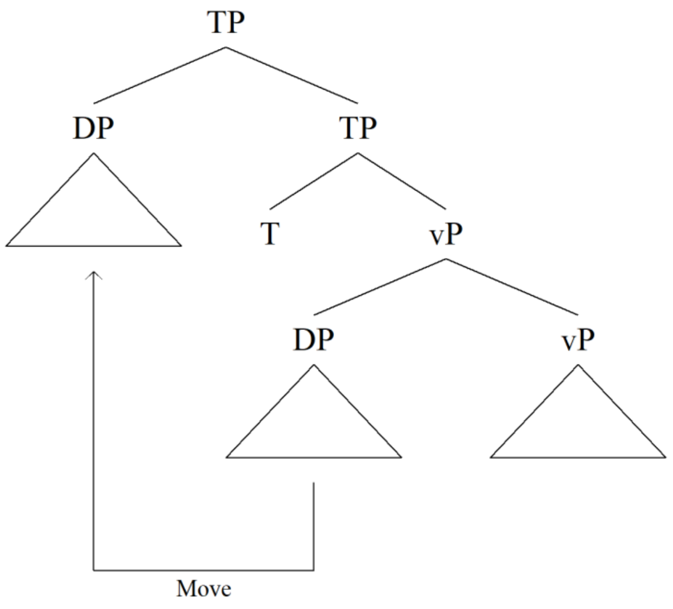
Spx	Spy	Epx	Epy
0	0	0	0

Y-axis offset: 0

We can change the dependency type and curvature, and provide a label, like so:



Spx, **Spy**, **Epx**, **Epy** change the starting and ending positions of the arrow, respectively, so we can control how the endpoints are positioned. Positive values move the x-positions to the right, negative values to the left (**Spx**, **Epx**). Positive values on the y-axis (**Spy**, **Epy**) move the positions downwards, negative values upwards. These values are calculated relative to the default positions, shown above, provided by the tree drawing algorithm. **Y-axis offset** determines how much the arrow is pulled downwards from the y-axis starting point. Providing a value +100 will generate the following variation of the above:

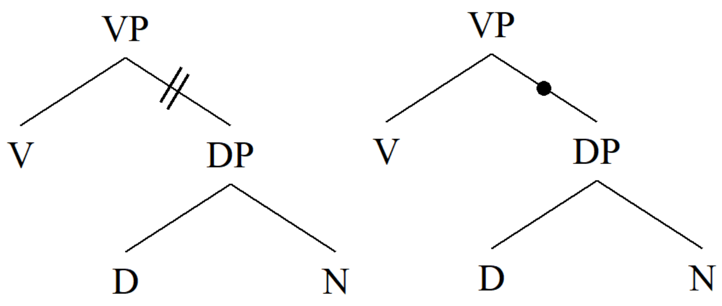


This allows the user to draw arrows that go around nodes if there is an overlap.

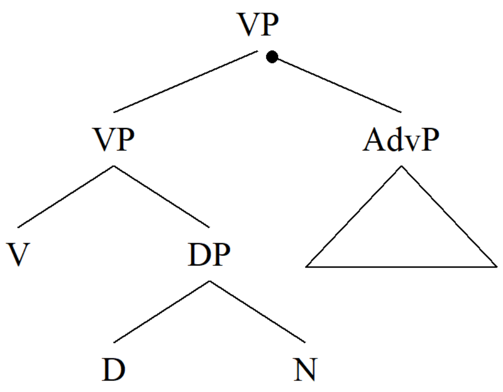
When the tools available in the current version of the LPGlab are not sufficient to generate special dependency illustrations, the best method is to crop the image (4.1) without any dependencies into an external graphics program and draw the special items there.

4.8 Special shapes on the image

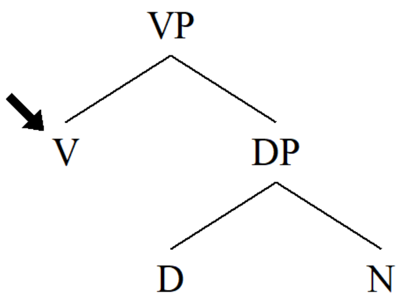
One way to add special shapes and markings to the image is to crop it (4.1) and export cropped image into an external graphics program. The program itself comes with some options, however. To create a special marking into the constituency line, select the mother node and select **Node > Special Constituent… > Cut…/Circle… > Left/Right**.



Adjuncts are represented by the following notation

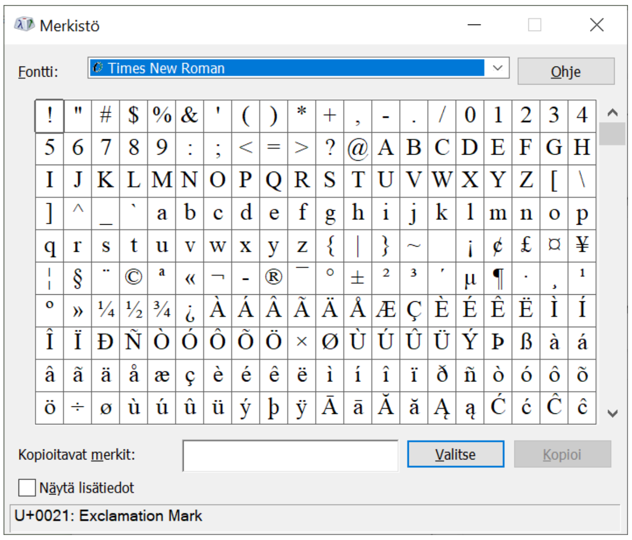


which can be created by selecting the adjunct (here, AdvP) and then **Phrase Structure > Make Adjunct/Regular**. This symbology can be used for any purpose, but notice that the underlying linguistic theory will interpret AdvP as an adjunct. We can highlight any node by **Node > Highlight** which adds the following symbol:



4.9 How to use special symbols for text

Dialogs that ask the user to provide text input, such as features for primitive nodes, take the input from the keyboard which usually does not have dedicated keys for special symbols such as Greek alphabet. The characters shown in the dialog are in reality numerical codes (depending on the font and encoding), so to generate special symbols we have to generate the corresponding numerical codes. One way is to generate them directly by pressing **Alt + #** where # is produced by pressing the corresponding numerical keys on the Numpad. An easier alternative is to pick up the symbols from a separate program such as Windows **characters**:


Merkistö

Fontti: Times New Roman

Ohje

!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4
5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\
]	^	_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
q	r	s	t	u	v	w	x	y	z	{	|	}	~		¡	¢	£	¤	¥
¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹
º	»	¼	½	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í
Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	à	á
â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ
ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	Ā	ā	Ă	ă	Ą	ą	Ć	ć	Ĉ	ĉ

Kopioitavat merkit:

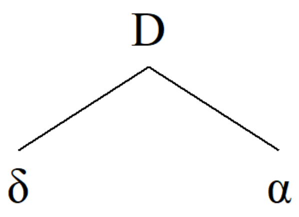
Valitse

Kopioi

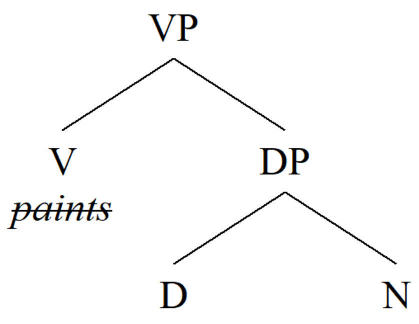
Näytä lisätiedot

U+0021: Exclamation Mark

Select the symbol from the table and copy paste it into the LPG dialog. We can generate illustrations such as the following:

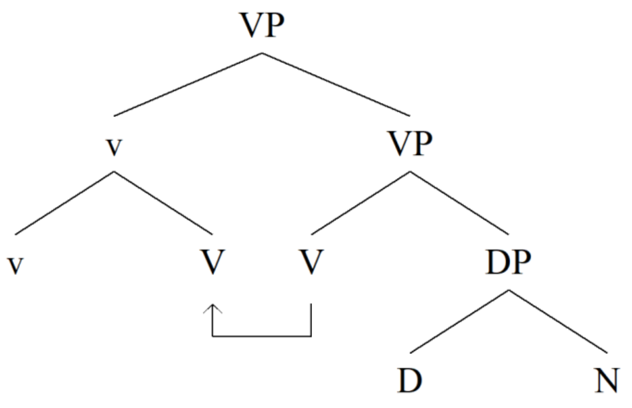


Elliptic text can be generated by selecting the node and **Node > Phonology… > Mark ellipsis**. The result will be



4.10 Some further tips

To create complex heads such as the one shown in the image below, simply create a complex node and use custom labels.



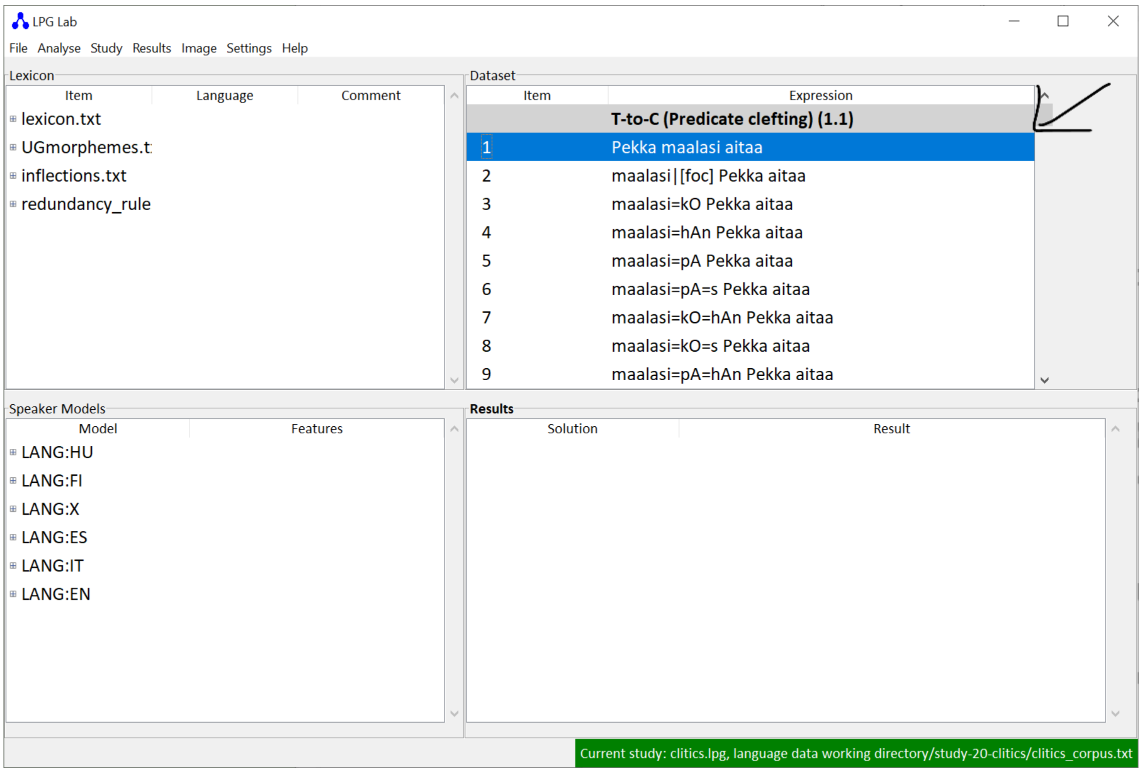
5 Examining the output of the linguistic model

5.1 Introduction

LPGlab was originally created to illustrate and examine the output of the underlying linguistic model (LPG). The linguistic model is documented in separate software documentation and several scientific publications, and will not be examined here.

5.2 Visualizing a single phrase structure

To examine the output of the underlying linguistic model, launch the GUI component (**python lpparse -app**) and then select the target sentence from the list of expressions in the upper right panel:



Item	Language	Comment
lexicon.txt		
UGmorphemes.t		
inflections.txt		
redundancy_rule		

Item	Expression
	T-to-C (Predicate clefting) (1.1)
1	Pekka maalasi aitaa
2	maalasi|[foc] Pekka aitaa
3	maalasi=kO Pekka aitaa
4	maalasi=hAn Pekka aitaa
5	maalasi=pA Pekka aitaa
6	maalasi=pA=s Pekka aitaa
7	maalasi=kO=hAn Pekka aitaa
8	maalasi=kO=s Pekka aitaa
9	maalasi=pA=hAn Pekka aitaa

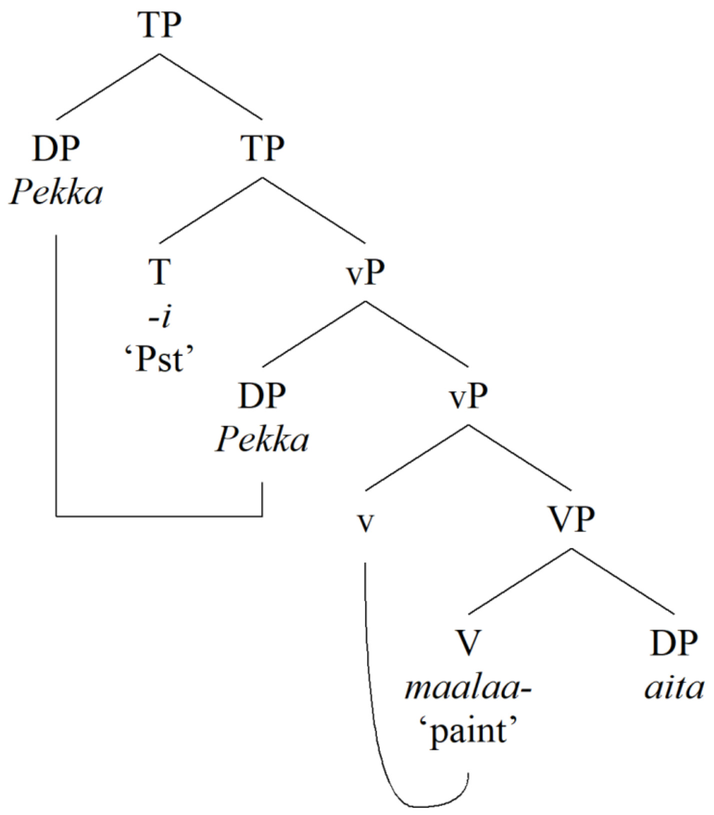
Model	Features
LANG:HU	
LANG:FI	
LANG:X	
LANG:ES	
LANG:IT	
LANG:EN	

Solution	Result

Current study: clitics.lpg, language data working directory/study-20-clitics/clitics_corpus.txt

This list itself is created on the basis of the *study* that has been selected for execution, and which always includes a dataset we see here. The study is defined in the **$app_settings.txt** file in the installation root directory which points to the study configuration file and its directory, which in turn points to the dataset and other files required to run one full study (i.e. $app_settings.txt → study configuration → dataset). This information is shown on the green bar at the bottom of the application window (see above image). If we want to derive a sentence that does not appear in this list, then either we can change the study and load a different dataset or we can add the sentence manually to the dataset file associated with the study. It is important to keep in mind, however, that the underlying linguistic model is not a general-purpose parser or framework but incorporates and implements a specific and detailed empirical hypothesis concerning some linguistic phenomenon. For example, the version of the software I am working here (version 20.0) was develop to capture properties of Finnish left peripheral clitics.

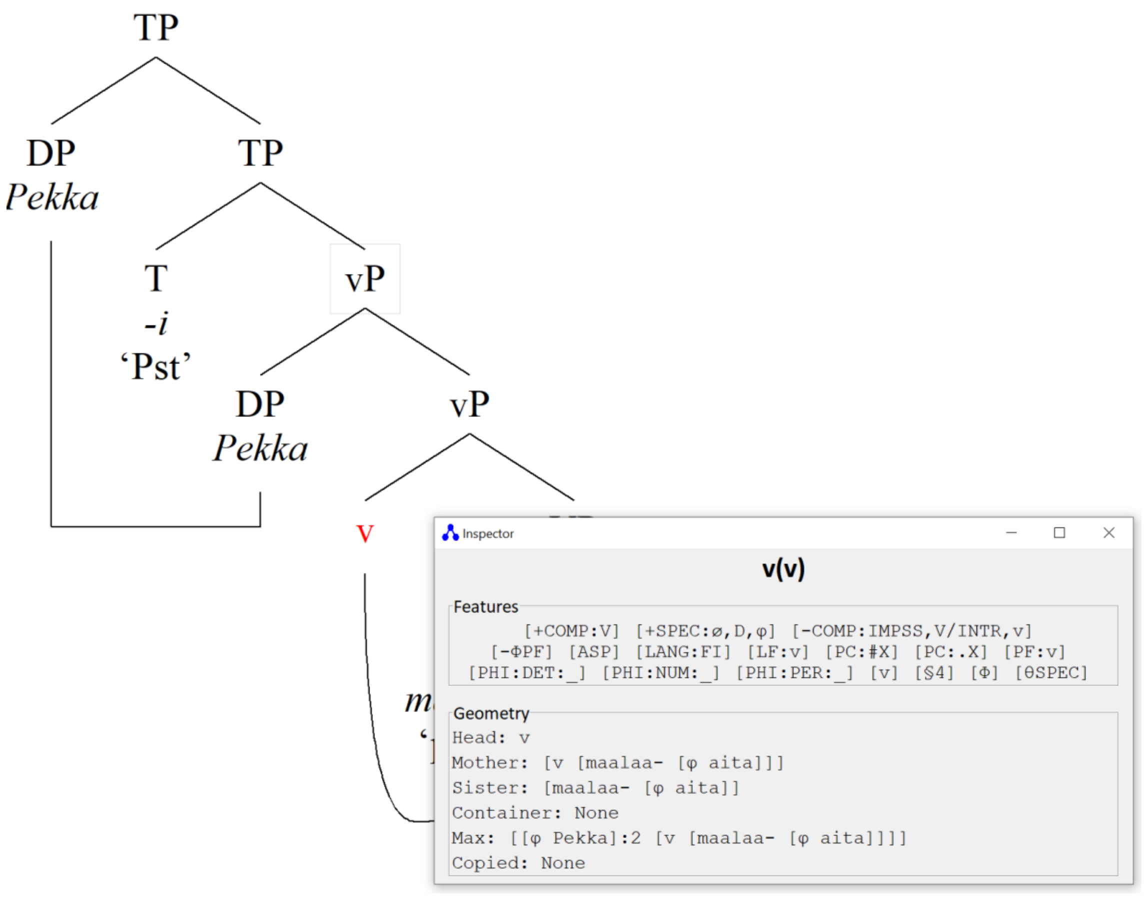
We can double click on any data item in the list, which will ask the underlying linguistic model to try to derive it. Thus, double clicking the first item in the dataset (grammatical transitive sentence in Finnish) will open the phrase structure image window with the following content:



This image was created on the basis of the *first acceptable LF-interface representation* generated for the target sentence. The user can manipulate it by using the tools discussed in Chapter 4, for example to prepare it for publication, but it is important to keep in mind that the image has now been created from an underlying linguistic phrase structure (output of the model). To visualize another expression/derivation, simply close the window and select another item from the dataset list. Currently, if the sentence is ungrammatical, no visual information will be produced (though this is clearly a limitation that will be removed in some later version).

5.3 Inspecting properties of the phrase structure

Clicking a node in the phrase structure and pressing **I** or selecting **Node > Inspect** opens up a window showing various properties of the node, such as its lexical features and other attributes calculated by the model:

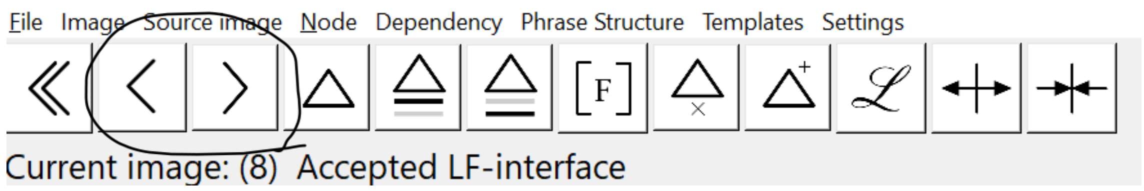


```
v(v)

Features
    [+COMP:V] [+SPEC:ø,D,φ] [-COMP:IMPSS,V/INTR,v]
  [-ΦPF] [ASP] [LANG:FI] [LF:v] [PC:#X] [PC:.X] [PF:v]
 [PHI:DET:_] [PHI:NUM:_] [PHI:PER:_] [v] [§4] [Φ] [θSPEC]

Geometry
Head: v
Mother: [v [maalaa- [φ aita]]]
Sister: [maalaa- [φ aita]]
Container: None
Max: [[φ Pekka]:2 [v [maalaa- [φ aita]]]]
Copied: None
```

The list of properties shown under the label of **Geometry** can be changed at the level of source code (class **InspectWindow** in module **GUI_phrase_structure_graphics.py** defines the contents of the above window).

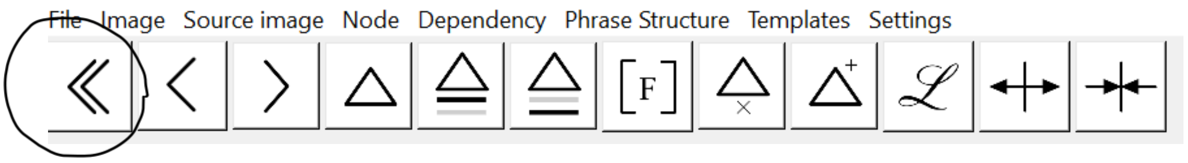
5.4 Examining the whole derivation

When the underlying model derives any given expression, it will usually generate several steps in which the grammatical operations are applied to the phrase structure in current syntactic working memory. Moreover, if the expression is ambiguous, there will be several legitimate LF-interface representations. What is shown by default is the first legitimate LF-interface representation. The user can examine the whole derivation, however. To scroll back and forward in the derivation, use the following buttons:


File Image Source image Node Dependency Phrase Structure Templates Settings
[F]

Current image: (8) Accepted LF-interface

Information concerning the image and the step in the derivation is visible in the status bar just below the buttons; see the above screenshot. To go to the first step, use the following button:



Another option is to use the menu items under **Source image**.

5.5 Creating raw output image data while running a whole study

It is possible to create a phrase structure image for each grammatical sentence when running the whole study. To do this, select **Study > Run Study and Generate Images**. The postscript (EPS) images for grammatical sentences are saved into the study folder. Properties of the resulting images are controlled from the study configuration file.