GORTO

Graphical Dependency Analyzer User Manual

V. Niepel
K.J. Prott
U. Kastens

University of Paderborn
D-4790 Paderborn
F.R.G

$Revision: 4.3 $

Table of Contents

1 Introduction

This is a user manual for the tool GORTO. GORTO is a graphical tool for analysis and modification of dependencies in attribute grammars. It is part of the LIGA system. The graphical representation requires execution under the X-Window system. GORTO offers the following support for attribute grammar development.

Graphical representation and analysis of the dependency graphs for productions and symbols.

Graphical representation and modification of partitions.

Graphical representation and modification of visit-sequences.

GORTO has a graphical user interface in which the mentioned information is represented. Interactive modification by the user are possible. The layout of the presentation can be influenced by the user.

After a short description of the user interface, we first describe the usage of the tool. the description is related to the standard configuration. Afterwards we discuss the possibilities of modifying the user interface for individual partiality.

2 Using GORTO within ELI

It is recommended to use GORTO in order to trace cyclic attribute dependencies which are indicated by the ORDER pass of LIGA (message "CYCLE IN INDUCED GRAPHS"). Also in the case that ORDER fails to compute an evaluation order (message "CYCLE IN PARTITIONED GRAPHS") GORTO can be used to analyze the reason and to arrange the dependencies differently. The results of such modifications are described by ARRANGE options and made available in a file GORTO.ctl. GORTO may be called for a correct attribute grammar as well, in order to study the dependencies. If the attribute grammar is incomplete, GORTO does not produce any information additional to that given by ORDER.

GORTO is started by the following derivation

```
<file>.specs:gorto
```

After completion of the interactive GORTO session, the results (if any) are stored in the file GORTO.ctl in the current working directory of the user. If there already exists a file with that name, it is saved with the name GORTO.bak. If GORTO.ctl is not yet part of the specifications, its name should be added in a .specs file in order to use the results of the GORTO session. Eli recognizes if GORTO.ctl has been modified by a GORTO session, and restarts the necessary derivation steps. GORTO itself can be restarted even if no specification has been modified. .ctl options for ORDER are recognized by GORTO too. They are passed on into GORTO.ctl.

An interactive GORTO session requires that Eli is started under X-Windows (X11). The environment variable DISPLAY must be set.

3 Usage

3.1 The windows of GORTO

The Graphical User Interface

The user interface of GORTO is composed of several distinct windows, which can be manipulated in the same manner as any other window. A view over the different windows is given in Figure 1. (While reading this documentation we suggest to call GORTO under ELI to practice what is described in the following.) The following types of windows are offered:

Main window

The main window of GORTO is divided in several parts and includes the possibility of activating several menus and opening symbol and production windows. The actual status of the tool is announced in the main window.

Symbols Each Symbol-window contains the dependency graphs for one symbol. The dependencies between attributes are described by arrows and the partitions are implicitly described by the graphical layout arrangement of the attributes

Productions

Production windows contain the dependency graph for one production each. The dependencies between attributes are also described by arrows.

Visit-Sequences

Visit-Sequence windows include the visit-sequence for one production each. In one visit-sequence the lifetime of those attribute instances are described which appear defining in the corresponding production.

LIDO-Source text The source text in LIDO for this AG.

3.2 The Main Window

The main window is horizontally divided in several parts and allows mainly the opening of production and symbol windows. Furthermore different menus can be activated in the main window.

Main Menu and Status Display

The upper part of the main window contains the main menu and an overall status indication. Depending on the situation the status indication informs either about the actual state of a running dependency analysis or about the success of the last dependency analysis. The main menu allows the following operations:

'**Lido window**' Opening and closing source text windows.

'**Close all windows**' Close all opened windows.

'**New computation**' Start a new dependency analysis.

'**Hide attribute**' Hiding attributes.

'**Unhide attribute**' Show hidden attributes.

'**Show Partitioned Dependencies**' Show partitioned dependencies.

'**Leave GORTO**' Leaving GORTO.

Selection of Productions and Symbols

Another part of the main window allows the selection of productions and symbols. Gorto indicates two lists of the names of productions and of symbols. The selection of a name by pressing the mouse-button opens the according window. The amount of listed productions and symbols can be influenced with the switchboard arranged above. The switches have the following significance:

Cyclic Productions or symbols with cyclic dependency graphs.

Arranged Productions or symbols with dependencies added manually.

Changed Productions and symbols changed after the last computation.

Visit-Sequence
Productions a visit sequence has been computed for.

Partitioned
Symbols for which a partition has been computed

The total list of productions and symbols is determined as a union of all groups activated with the switches.

Strategy Menus

For computation of visit-sequences and partitions Order offers different strategies, which can be activated with the according menu in the main window. Therefore two menus exist:

Topological Strategy
Strategy to compute visit sequences.

Partition Strategy
Strategy to compute partitions.

The menu-points allow the selection of a specific strategy. After selecting such a menu-point GORTO starts automatically a new dependency analysis.

Hiding of attributes

It is possible to hide several attributes in the representation of production graphs via the main menu. Hidden attributes are not visable any more in the production graphs to save space on the screen and get a better overview. They still are considered in all internal computations.

After selection of the menu point '**Hide attribute**' a dialog window appears, in which the desired attribute name has to be entered. Regular terms can be used in the defining form of regexp(3). The input can be confirmed with '**Confirm**' or canceled with '**Cancel**'. When confirming all selected attributes are hidden no matter to which symbol they belong. Additionally with the commands '**Including, Constituent**' and '**Chain**' it is possible to hide the attributes generated by LIGA when expanding the corresponding LIDO notations.

With the menu point '**Unhide attribute**' attributes can be reinserted with the same regulations. The structure of the dialog window to insert attributes is the same as it is for hiding attributes.

3.3 Description of Symbols and Dependencies

First the general concepts of descriptions of symbols and dependencies shall be explained.

Description of symbols

Symbols are described by their names and their attributes. The attributes are arranged under the symbol name and sorted in the sequence of the computed partitions. Additionally the synthesized attributes are moved right with respect to the inherited attributes, so that the layout of all attributes implicitly describe the computed partition.

If in case of cyclic dependencies no partitions can be computed, the sequence of the description of course does not represent a partition. In this case the inherited attributes are located before the synthesized attributes.

Description of dependencies

Dependencies are described by an arrow, which is located between both involved attributes pointing the dependent attribute. To mark different origins of dependencies the following different lines are used:

straight line
direct dependency

dashed line
caused by induction of dependencies

doted line caused by partitioning

3.4 Productions

A production window shows the dependency graph for a production as well as the information concerning the status of the production. The described dependencies can be traced interactively.

Status indication

The upper part of the production window contains information about the actual status of this production. Every single entry has the same meaning as the corresponding button in the main window.

Layout of symbols and dependencies

Additionally to the different types of lines described above the different types of dependencies are furthermore distinguished by their graphical location. The dependency arrows are located either outside at a single symbol or inside between the symbols. The meaning of this layout is as follows:

Outside Dependencies which originate from another context

Inside Dependencies which originate from this production

Tracing Dependencies

tracing dependencies

The origin of the shown dependencies can be traced interactively by selecting an arrow with the mouse-button. The origin of this dependency is shown as a directed path or it is shown directly in the source-text. In case of the directed path a new window will be opened in which the dependencies are marked on their path with thick lines. The selected dependency is then marked thick and dashed. Both ways of indication are described in Figure 2. There are three variants of starting a dependency trace which are described in the following with the corresponding mouse-buttons in brackets:

FOLLOW <Shift-Button1>

The selected dependency is traced without consideration of dependencies which might be marked.

FOLLOW-MARKED (Button 1)

If the dependency had been marked by a previous trace, the selected dependency will be traced now. The trace of non-marked dependencies is therefore omitted.

FOLLOW-AND-CLOSE (Ctrl-Button 1)

The selected dependency will be traced in any case as with the variant FOLLOW. Additionally all windows are automatically closed which are not involved in this trace.

The Production Menu

The production menu can be activated by pressing Button3 and allows the following operations:

Close this production

Close this production window

Close all productions

Close all production windows

Follow all marks

Automatic tracing of marked dependencies

Show visit-sequence

Open the corresponding visit sequence window

3.5 Symbols

A symbol-window shows the dependency graph of a symbol together with the affiliated status information. The shown dependency graphs can be traced and the computed partition of the symbol can be modified.

Status Notification

The upper part of a symbol-window contains information about the actual status of the symbol. The entries have the same meaning as the corresponding buttons in the main window.

Trace of Dependencies

The dependencies can be trace in symbol-windows in the same way as in production-windows. Again there are the three variants FOLLOW, FOLLOW-MARKED and FOLLOW-AND-CLOSE, which show the derivation of the selected dependencies as a directed path and open a new window, if necessary.

Modification of Partitions

GORTO divides the attributes of a symbol into two different classes: critical and non-critical attributes. Critical attributes are those attributes, which can't be moved to another partition without increasing the total number of partitions. Non-critical attributes can be moved, up to a certain extend, within the actual partitioning. The critical attributes are displayed in bold and italic, the non-critical attributes described in a normal font. GORTO offers three possibilities to modify a partition:

Moving of a non-critical attribute

First a non-critical attribute has to be selected with the mouse-button. This attribute can be moved by selecting another partition of attributes of the same class with button2. The attribute is then automatically fixed to the desired partition by insert onof two dependencies to critical attributes in the adjacent partitions.

Adding a new partition

Before or after the actual partitions a new partition can be added which contains a selected attribute. Again the attribute has to be selected with the mouse button. Afterwards the new partition can be added before or after all of the the existing partitions by selecting an attribute of the first or the last partition with the Shift-Button2. The selected attribute will be used to produce a new dependency, which forces the computation of a new partition with the attribute selected first.

Segmentation of existing partitions

An existing partition can be segmented into several partitions by 'squeezing' an attribute of another partition between the two attributes of the the original partition. To do that, click on the attribute which has to be 'squeezed' in. Within the partition which shall be segmented both surrounding attributes have to be selected. Both attributes are selected one by one by pressing the Ctrl-Button2, first the attribute which shall proceed, then the attribute which follows the new partition. If the respective partition contains only two attributes, then it is not necessary to select the second one. GORTO identifies it automatically.

A modification of the partition doesn't lead automatically to a completely new computation of all dependency graphs. The changes will be made only locally within the effected graphs. A completely new computation must be started manually using the main menu.

The Symbol Menu

The symbol-menu can be activated by pressing the Button3 and allows the following operations:

Close this symbol
: Close the symbol window

CLose all symbols
: CLose all symbol windows

Remove arranged dependencies
: Remove all the added dependencies

Force partition
: Freeze the partition

Select production
: Select production

Show last BnNF use

3.6 Visit-sequences

A visit-sequence window can be activated by an operation of the production menu. shows the visit-sequences of a production together with the lifetimes of the attribute instances within this production. The visit-sequence can be modified within the frame of the certain restrictions.

Description visit-sequences

A visit-sequence will be described by a sequence of operations which are located one below the other. The operations are described by a short text which contains several informations depending on the type of the operation. In case of more than one leave from root-symbol of the production each single visit is separated and terminated with a leave-operation.

The lifetime of all attribute instances which are computed in the described visit-sequence is shown in form of a dependency. This dependency starts with the computation and ends with the last use of the attribute value.

Modification of visit-sequences

As mentioned earlier the described visit-sequence can be modified within the frame of the remaining freedom. When selecting an operation by pressing the mouse-button, the area in which this operation can be moved without any conflicts is shown. The operation can be moved by pressing Button2 at the new position (similar to moving in partitions). Operations which can't be moved are printed in bold and italic, like the critical attributes in dependency graphs.

The Visit-sequence Menu

The visit-sequence-menu can be activated by pressing Button3 and allows the following operations:

Close this visit-sequence

Close all visit sequences

Don't show last attribute use

Fix position of this element

Remove arranged dependencies

4 Layout Adaptation

The following describes the possibilities of adaptation of GORTO to personal preferennces. The adaptation of GORTO is done by the resource mechanism of the X-Window-System, which can't be explained in detail here. In this paper only specific details of the implementation of GORTO is considered. For common explanations of resources see the original literature.

4.1 Resources of GORTO

GORTO uses several resources which are not related to any special widget. They rather have the status of globally used resources and are specified in the form of Gorto.Name. These are the following resources:

Common adjustments

```
Name              Class         Default    Meaning
----------------------------------------------------------------------------

sortProdList      SortList       False     sorting production lists
sortSymbolList    SortList       False     sorting symbol lists
leftSymbol        LeftSymbol     False     imaging symbol graphs
showPartDeps      ShowPartDeps   False     showing partitioned dependencies
makeTitleBars     MakeTitleBars  False     producing titelspaces
wmPushDown        WmPushDown     0         Pushdown of Window Manager
```

Filenames

```
Name        Class         Meaning
-----------------------------------------------
lidoFile    LidoFile      LIDO-Input-File
inputFile   InputFile     exp_idl-Input-File
outputFile  OutputFile    ord_idl-Output-File
lclFile     LCLFile       LCL-Output-File
```

Layout of dependency graphs

Name	Class	Default	Meaning
boldFont	Font	fixed	font for symbol names
attributeFont	Font	fixed	font for non-critical attributes
criticalFont	Font	fixed	font for critical attributes
textPadding	Padding	2	free pixel around strings
synthesizedIndent	Indent	15	indentation of synthesized attributes
symbolvPadding	Padding	10	vertical space between symbols
symbolhPadding	Padding	5	horizontal space between symbols
arrowWidth	ArrowWidth	3	width of arrows
arrowHeight	ArrowHeight	9	hight of arrows
arrowLength	ArrowLength	30	minimal length of arrows
depColumnWidth	DepColumnWidth	3	width of column of dependencies
depFaseLength	DepFaseLength	5	length of bends in dependencies
selectWidth	SelectWidth	3	selection width of dependencies

Colours

Name	Used for
directDepColor	direct dependencies
inducedDepColor	induced dependencies
arrangedDepColor	added dependencies
arrangedInducedDepColor	induced added dependencies
partitionedDepColor	partitioned dependencies
inducedPartitionedDepColor	induced partitioned dependencies
attrColor	attributes
includingAttrColor	attributes produced by INCLUDING
constituentAttrColor	attributes produced by CONSTITUENT(S)
chainAttrColor	attribute produced by CHAIN
symbolColor	symbol names
visitColor	VISIT-Operations
conditionColor	CONDITION-Operations
lineColor	frame resp. cutlines

Strings

```
Name                Default
----------------------------------------------
emptyListString     (empty)
directRuleState     direct rules
directSymbolState   direct symbols
optionState         order options
transitiveState     transitive graphs
inducedState        induced graphs
arrangeOptionState  arrange option
arrangedState       arranged graphs
partitionState      partitions
partitionedState    partitioned graphs
visitSeqState       visit sequences
lifetimeState       lifetimes
infoState           display information
hideState           hide attribute
unhideState         unhide attribute
directDisplay       direct display
transitiveDisplay   transitive display
inducedDisplay      induced display
arrangedDisplay     arranged display
partDisplay         partition display
visitSeqDisplay     visit sequence display
constructIDLState   idl structure
ouputState          output files
noRecomputeString   not necessary
ruleString          rule
symbolString        symbol
transferString      transfer dependency
includingString     including dependency
constituentString   constituent dependency
chainString         chain dependency
unknownDepString    unknown dependency
cyclicString        cyclic
arrangedString      arranged
changedString       changed
partitionedString   partitioned
visitSeqString      visit-sequence
forcedString        forced
cycleMessage        would yield cycle
noBmNFMessage       no BmNF occurrence
```

4.2 Internal Interface Structure

To allow sensible specification of resources the following internal interface structure of GORTO is given. The widgets used by GORTO are Athena widgets of the X Version 11, Release 4. The main window of GORTO shows the following widget structure:

Structure of Rule- and Symbol windows:

Structure of the window to select the according rules in symbol windows:

Structure of the dialogue window to hide attributes:

Structure of visit-sequences, Lido-source-text and announcements:

4.3 Graph Widget

The graph widget is a widget adapted to GORTO. It is used to describe different dependency graphs. It uses the widget class Core and doesn't provide new resources for the adaptation of the graph widget. The adaptation of graph widgets is done by certain actions at certain events. This binding up is done with the so called binding translations. For their description see the original literature.

Two actions are available: menu-popup() and select(). The action menu-popup() is used to call the menus within the dependency graphs and expects the internal name of a menu as a parameter. The action select() is used for adaptation of mouse and keyboard handling within dependency graphs. It accepts the following parameters which stand for user-commands described above:

```
Name              Objects             possible contexts
--------------------------------------------------------------
FOLLOW            dependencies        symbol, rules
FOLLOW-AND-CLOSE  dependencies        symbol, rules
FOLLOW-MARKED     dependencies        symbol, rules
REMOVE            added  dependencies symbol, rules
HIDE              attributes          symbols, rules, visit-sequences
UNHIDE            attribute           symbols, visit-sequences
MOVE              attribute           symbols, visit-sequences
SPLIT-PARTITION   attributes          symbols
APPEND-PARTITION  attributes          symbols
```

The action select() is normally bound to the use of the mouse. Up to three parameters can be called to state the action to be executed depending to the selected object. If more than one parameter is given for the same object, only the last action will be executed the others are ignored. The default paramters for the graph-widget is the following:

```
<Btn3Down>: menu-popup(mainMenu) \n\
<BtnDown>:  select(FOLLOW) \n
```

Index