LCL: Liga Control Language

U. Kastens
K.J. Prott
H.D. Siewert

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

$Id: lcl.tnf,v 4.6 1996/03/15 12:44:25 mjung Exp $
$Revision: 4.6 $

Table of Contents

1 Introduction

The analysis and translation of an attribute grammar by the LIGA system may be influenced in various ways. For that purpose the user specifies certain options in the LIGA Control Language LCL. You can introduce options to the ELI system in a file ending in `.ctl`. A default is assumed for each option if it is not set or if no LCL specification is given. The denotations of the options are headed by a prefix indicating one of the liga phases the option mainly refers to.

LCL has a simple, keyword-oriented syntax. Each keyword consists solely of upper-case letters and underscores. Comments are enclosed in '/*' and '*/', and must not be nested.

```
Options  ::=   ( Option ';' )*
Option   ::=   'EXPAND'  ':' ExpandOpts /
               'ORDER'   ':' OrderOpts /
               'OPTIM'   ':' OptimOpts

ExpandOpts  ::=  ExpandOpt  ',' ExpandOpts / ExpandOpt
OrderOpts   ::=  OrderOpt   ',' OrderOpts / OrderOpt
OptimOpts   ::=  OptimOpt   ',' OptimOpts / OptimOpt

Expandopt   ::= 'INFO' / 'INCLUDINGS_SEPARATE' /
                'INCLUDING' 'ON' / 'INCLUDING' 'OFF'

OrderOpt    ::= 'PARTITION' Strategy / 'TOPOLOGICAL' Strategy /
                'TREE' treeClass / 'GRAPH' Type ( ident ) * /
                'ARRANGE' arrangePart
Strategy    ::= 'EARLY' / 'LATE'
treeClass   ::= 'COMPLETE' / 'BOTTOM_UP'
Type        ::= 'DIRECT_SYMBOL' / 'TRANSITIVE_SYMBOL' / 'INDUCED_SYMBOL' /
                'DIRECT_RULE' / 'TRANSITIVE_RULE' / 'INDUCED_RULE' /
                'PARTITIONED_RULE' / 'PARTITION' / 'VISIT_SEQUENCE'
arrangePart ::= 'AUTOMATICALLY' / 'FAST' /
                'FOR' 'SYMB' ident 'EVAL' ident 'BEFORE' ident /
                'IN' 'RULE' ident 'EVAL' ident '[' intval ']' '.' ident
                'BEFORE' ident '[' intval ']' '.' ident

OptimOpt    ::= 'OFF' / 'INFO' / 'MORE_GLOBALS' /
                'NO_VARIABLES' / 'NO_STACKS' / 'GROUPING' 'VARIABLE' /
                'ATTRSPEZ' Type ( symbname '[' ( ident ) + ']' ) *
symbname    ::= ident / 'ANYSYMBOL'
Type        ::= 'GLOBAL VAR' / 'GLOBAL STACK' / 'TREE_NODE' /
                'GROUP VAR' / 'GROUP STACK'
```

Figure 1: Context-free grammar for LCL

The next four sections of this manual describe the detailed syntax and effects of the options.

2 Expand Options

Options introduced by "Expandopt" influence the expansion of used shorthands.

```
Expandopt  ::= 'INFO' / 'INCLUDINGS_SEPARATE' /
               'INCLUDING' 'ON' / 'INCLUDING' 'OFF'
```

'INFO'
If this option is set, additional information on the expansion results is included in the listing. The default is not to produce the additional information. This option affects REPLINCL and EXPAND. This option can be included automatically in Eli by using the derivation `:ExpInfo`.

'INCLUDINGS_SEPARATE'
In the default case, identical `INCLUDING` constructs are expanded by one single set of generated attributes. The number of new attributes is thereby reduced, and in some situations OAG-conflicts are avoided. This option is recommended for debugging purposes. If it is set, each `INCLUDING`-construct is expanded separately.

'INCLUDING' 'OFF'
In the default case attributes introduced by `INCLUDING` will be converted to global variables. You will need this option only if the default handling changes.

'INCLUDING' 'ON'
With this option you can specify that the attributes generated by `INCLUDING` statements are handled like all other attributes. This normally means that they are put into local variables and passed through as parameters to the places where they are needed. This option also affects many oher phases of LIGA.

3 Order Options

Options introduced by "Orderopt" control the general strategies applied by Order.

```
OrderOpt    ::= 'PARTITION' Strategy / 'TOPOLOGICAL' Strategy /
                'GRAPH' Type ( ident ) * / 'TREE' treeClass /
                'ARRANGE' arrangePart
Strategy    ::= 'EARLY' / 'LATE'
Type        ::= 'DIRECT_SYMBOL' / 'TRANSITIVE_SYMBOL' / 'INDUCED_SYMBOL' /
                'DIRECT_RULE' / 'TRANSITIVE_RULE' / 'INDUCED_RULE' /
                'PARTITIONED_RULE' / 'PARTITION' / 'VISIT_SEQUENCE'
treeClass   ::= 'COMPLETE' / 'BOTTOM_UP'
arrangePart ::= 'AUTOMATICALLY' / 'FAST' /
                'FOR' 'SYMB' ident 'EVAL' ident 'BEFORE' ident /
                'IN' 'RULE' ident 'EVAL' ident '[' intval ']' '.' ident
                'BEFORE' ident '[' intval ']' '.' ident
```

`'PARTITION' Strategy`

This option is used to specify the strategy for computation of attribute partitions.

`'TOPOLOGICAL' Strategy`

This option is used to specify the strategy for topological sorting of visitsequences.

`'GRAPH' Type ( ident ) *`

This option is used to obtain additional information about specific attribute dependencies. It is most useful for tracking down dependency cycles. The dependency graphs for the given symbols and rules are listed. If the `Identlist` is empty, then the graphs for all symbols or rules are listed. The default for this option is to provide no listings of any dependency graphs.

`'TREE' treeClass`

This option is used to specify whether the first attribution traversal should be combined with tree construction. If `COMPLETE` is specified, attribution begins after the complete structure tree has been built. If `BOTTOM_UP` is specified, the first computations are combined with the `BOTTOM_UP` tree construction. The default for this option is `BOTTOM_UP`.

See also Section 'Computations' in LIDO - Reference Manual.

`'ARRANGE' arrangePart`

This option is used to fix the evaluation order of attributes of a single symbol or of a single rule. If an attribute grammar is not ordered, but may be well-defined –i.e. cycles do not occur before computation of the partitioned rule graphs – these specifications may be used to constrain the admissible partitions.

`AUTOMATICALLY` invokes an additional algorithm that tries to avoid cycles while computing the visit-sequences. `FAST` avoids to invoke that algorithm, speeding up the analysis and increasing the risk not to find suitale visit-sequences in certain cases. The default is `AUTOMATICALLY`. By specifying another arrange option a certain evaluation sequence is explicitly requested.

4 Optim Options

Options introduced by "Optimopt" control the strategy for attribute optimization applied by Optim.

```
OptimOpt    ::= 'OFF' / 'INFO' / 'MORE_GLOBALS' /
                'NO_VARIABLES' / 'NO_STACKS' / 'GROUPING VARIABLE'
                'ATTRSPEZ' Type ( symbname '[' ( ident ) + ']' ) *
symbname    ::= ident / 'ANYSYMBOL'
Type        ::= 'GLOBAL VAR' / 'GLOBAL STACK' / 'TREE_NODE' /
                'GROUP VAR' / 'GROUP STACK'
```

`'OFF'`
This option supresses the optimization entirely.

`'INFO'`
This option produces a listing of the results of the optimization that gives all attribute storage decisions and the visit-sequences as modified by stack operations. Attribute evaluations that are omitted due to optimization are marked by the string `IS DELETED`.

This option can be included automatically in Eli by using the derivation `:OptimInfo`.

`'MORE_GLOBALS'`
If this option is set, a second optimization algorithm is started. This algorithm takes the attributes which are classified as tree nodes by the first optimization algorithm and tries to globalize them too.

`'NO_VARIABLES'`
No attributes are implemented as global variable. Instead all attributes which can be globalized are implemented as global stack.

`'NO_STACKS'`
No attributes are implemented as global stack. Instead all attributes which could be implemented by global stack are allocated in the tree.

`'GROUPING VARIABLE'`
If this option is set, different attributes which can be implemented by a global variable each are possibly grouped to one global variable.

`'ATTRSPEZ' Type ( symbname '[' ( ident )+ ']' )*`
This option requires a particular storage decision for the specified attributes.

Type `'GLOBAL VAR'` requires global variables.

Type `'GLOBAL STACK'` requires global stacks.

Type `'TREE_NODE'` requires allocation in the tree.

Type `'GROUP VAR'` requires allocation to a single global variable.

Type `'GROUP STACK'` requires allocation to a single global stack.

`symbname` specifies a symbolname and `ident` an attributname. If `'ANY_SYMBOL'` is used instead, all attributes with corresponding attribute name are taken.

Optim will try to fulfill the above given specifications. It will ignore those that would result in incorrect computation. A specified group will not be enlarged by an attribute not belonging to this specification; an attribute specified but not in a group will not be put in any group; and a group specification consisting of n attributes will result in a set of groups within these n attributes, which is the best found by the algorithm.

Any attributes which are not appearing in any `ATTRSPEZ` specification will be optimized as usual.

To specify the `ATTRSPEZ`-specifications, the tool Gorto can be used, see Graphical Dependancy Analyzer.