

# Local Action Diagrams

**Super awesome package for local action  
diagrams.**

0.1

29 January 2023

**Marcus Chijoff**

**Marcus Chijoff**

Email: [marcus.chijoff@uon.edu.au](mailto:marcus.chijoff@uon.edu.au)

Homepage: <https://newcastle.edu.au>

Address: No

# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Creating Local Action Diagrams</b>	<b>4</b>
2.1	Creating Local Action Diagrams . . . . .	4
<b>3</b>	<b>Local Action Diagram Attributes and Operations</b>	<b>6</b>
3.1	Local Action Diagram Attributes . . . . .	6
3.2	Local Action Diagram Operations . . . . .	7
<b>4</b>	<b>Input/Output and Visualisation</b>	<b>8</b>
4.1	Reading and Writing Local Action Diagrams To Files . . . . .	8
4.2	Visualising Local Action Diagrams and $\Delta$ -trees. . . . .	9
	<b>Index</b>	<b>10</b>

# Chapter 1

## Introduction

A local action diagram  $\Delta = (\Gamma, (G(v)), (X_a))$  consists of:

- a directed graph  $\Gamma$ ,
- a closed permutation group  $G(v)$  for each  $v \in V(\Gamma)$ , and
- a set  $X_a$  for each  $a \in A(\Gamma)$  such that each  $X_a$  is disjoint and  $X_a$  is an orbit of the action of  $G(o(a))$  on  $X_v := \bigsqcup_{a \in o^{-1}(v)} X_a$ .

The definition of digraph used is different to that of the `Digraphs` package. For our purposes, a digraph  $\Gamma$  consists of:

- a vertex set  $V$ ,
- an arc set  $A$ ,
- a map  $o : A \rightarrow V$  assigning each arc to an *origin* vertex, and
- a bijection  $r : A \rightarrow A$  (denoted by  $a \mapsto \bar{a}$ ) such that  $r^2 = \text{id}$ .

The bijection  $r$  defines a reverse arc for each arc of the graph. This is more specific than the definition in the `digraphs` package which does not require a reverse mapping.

The local action diagrams package provides a category for local action diagrams. It is built in the `IsDigraph` category from the `Digraphs` package.

## Chapter 2

# Creating Local Action Diagrams

### 2.1 Creating Local Action Diagrams

#### 2.1.1 IsLocalActionDiagram (for IsDigraph)

▷ `IsLocalActionDiagram(lad)` (filter)

**Returns:** true if `lad` is of the category `IsLocalActionDiagram` and false otherwise.

Every local action diagram belongs to the `IsLocalActionDiagram` category. Every local action diagram is immutable.

#### 2.1.2 Constructing From Data

▷ `LocalActionDiagramFromData(D, vertex_labels, edge_labels, rev)` (operation)

▷ `LocalActionDiagramFromDataNC(D, vertex_labels, edge_labels, rev)` (operation)

**Returns:** A local action diagram.

Constructs a local action diagram, checking that the arguments given are a valid local action diagram. The argument `D` is a digraph and `rev` must be a compatible involution on the edges of `D`. The argument `vertex_labels` is a list of vertex labels such that `vertex_labels[i]` is the group labelling vertex `i` of `D`.

The argument `edge_labels` is a list of edge labels. The edges of `D` are stored in lexicographical order and `edge_labels[i]` is the set labelling edge `i` of `D` (when sorted in lexicographical order).

The NC variant of the operation does not check that the arguments given are a valid local action diagram.

#### 2.1.3 LocalActionDiagramUniversalGroup (for IsPermGroup)

▷ `LocalActionDiagramUniversalGroup(F)` (operation)

**Returns:** A local action diagram.

Constructs a local action diagram corresponding to the Burger-Mozes group  $U(F)$  where  $F$  is a permutation group. This diagram has a single vertex labelled by the group  $F$  and a self-reverse loop for each orbit of the action of  $F$ . The argument  $F$  can instead be an object of the category `IsLocalAction` from the package `UGALY`.

### 2.1.4 LocalActionDiagramToUniversalGroup (for IsLocalActionDiagram)

▷ LocalActionDiagramToUniversalGroup(*lad*) (operation)

**Returns:** A local action.

Given a local action diagram *lad* corresponding to a Burger-Mozes group this returns the corresponding group as an object of the category IsLocalAction from the package UGALY. The local action diagram *lad* must have a single vertex and have only self-reverse edges. It must also have at least one edge.

## Chapter 3

# Local Action Diagram Attributes and Operations

### 3.1 Local Action Diagram Attributes

#### 3.1.1 LocalActionDiagramVertexLabels (for IsLocalActionDiagram)

▷ LocalActionDiagramVertexLabels(*lad*) (attribute)

**Returns:** A list of groups.

Returns the groups labelling the vertices of *lad*. Entry *i* of the list corresponds to the group labelling vertex *i* of the digraph.

#### 3.1.2 LocalActionDiagramEdges (for IsLocalActionDiagram)

▷ LocalActionDiagramEdges(*lad*) (attribute)

**Returns:** A list of lists.

Returns a list of edges of *lad*. Each edge is stored as a list [*i*, *j*] where *i* is the origin vertex and *j* is the terminus vertex. The list is stored in lexicographical order.

#### 3.1.3 LocalActionDiagramEdgeLabels (for IsLocalActionDiagram)

▷ LocalActionDiagramEdgeLabels(*lad*) (attribute)

**Returns:** A list of lists.

Returns the edge labels of *lad*. Entry *i* of the list corresponds to the label of edge *i* of the digraph.

#### 3.1.4 LocalActionDiagramEdgeReversal (for IsLocalActionDiagram)

▷ LocalActionDiagramEdgeReversal(*lad*) (attribute)

**Returns:** A permutation.

Returns the reversal mapping of the local action digram *lad*.

## 3.2 Local Action Diagram Operations

### 3.2.1 LocalActionDiagramScopos (for IsLocalActionDiagram)

▷ LocalActionDiagramScopos(*lad*) (attribute)

**Returns:** A list of lists.

Returns a list of all scopos of *lad*. Each entry of the list is a list of edges in the scopo. This list will always contain the empty scopo [].

### 3.2.2 LocalActionDiagramGroupType (for IsLocalActionDiagram)

▷ LocalActionDiagramGroupType(*lad*) (attribute)

**Returns:** A string.

Returns the group type the local action diagram corresponds to. This is either "Fixed Vertex", "Edge Inversion", "Lineal", "Focal", or "General". Note that all "Horocyclic" local action diagrams have infinitely many vertices and so can never be the type returned by this function.

### 3.2.3 LocalActionDiagramIsDiscrete (for IsLocalActionDiagram)

▷ LocalActionDiagramIsDiscrete(*lad*) (attribute)

**Returns:** true or false.

Returns true if *lad* corresponds to a discrete group and false otherwise.

### 3.2.4 LocalActionDiagramIsUniscalar (for IsLocalActionDiagram)

▷ LocalActionDiagramIsUniscalar(*lad*) (attribute)

**Returns:** true if the corresponding group is uniscalar and false otherwise.

If *lad* is of type "Fixed Vertex", "Edge Inversion", or "Lineal" then it is always uniscalar and so the function returns true. If it is of type "General" then it is uniscalar if and only if all vertices in the unique minimal cotree of *lad* are semi-regular with their action restricted to the cotree.

Is it better to describe the maths here or just say returns true or false. Maybe give a theorem citation in the paper?

## Chapter 4

# Input/Output and Visualisation

### 4.1 Reading and Writing Local Action Diagrams To Files

Local action diagrams made in GAP can be written to a file. They can also be read from the file into GAP. The format of these files are as follows:

Example

```
<Digraph String>
<Vertex Labels>
<Edge Labels>
<Edge Reversal Map>
<List of Other Attributes>
```

Multiple local action diagrams can be stored in the same file.

#### 4.1.1 Writing Local Action Diagrams

- ▷ `WriteLocalActionDiagram(lad, filename[, directory])` (operation)
- ▷ `WriteLocalActionDiagram(lad_list, filename[, directory])` (operation)

**Returns:** true if writing to the file was successful and false otherwise.

Writes the local action diagram *lad* or every local action diagram in the list *lad\_list* to the file *filename.lad*. The list *lad\_list* must be a dense list of local action diagrams.

If the optional argument *directory* is specified then the file is written to that directory. Otherwise the file is written to the current directory of the GAP session --- i.e. `DirectoryCurrent()`. The argument *directory* can either be in the category `IsDirectory` or `IsString`.

If the file *filename.lad* does not exist in the directory then it is create; otherwise the file is appended to.

#### 4.1.2 Reading Local Action Diagrams

- ▷ `ReadLocalActionDiagram(filename[, directory])` (operation)

**Returns:** A list of local action diagrams if reading the file was successful or raises an error and returns fail otherwise.

Reads the file *filename*. If the optional argument *directory* is specified then the file is read from that directory. Otherwise the file is read from the current directory of the GAP session --- i.e.



`DirectoryCurrent()`. The argument *directory* can either be in the category `IsDirectory` or `IsString`.

It is expected that the file was written with the `WriteLocalActionDiagram` function. Some basic error checking of the file is implemented as it is read but this will not catch all errors in the file.

## 4.2 Visualising Local Action Diagrams and $\Delta$ -trees.

Functions are provided to visualise local action diagrams and  $\Delta$ -trees. Insert some more text here?

### 4.2.1 GenerateTikZCode (for `IsLocalActionDiagram`, `IsString`)

▷ `GenerateTikZCode(lad, filename[, highlight_edges, directory])` (operation)

**Returns:** `true` if the code was generated successfully and `false` otherwise.

Generates TikZ code that draws the local action diagram *lad*. Currently this function only supports local action diagrams with one or two vertices. The TikZ code is written to *filename.tex*. This file must be compiled separately.

The optional argument *highlight\_edges* must be a list of edges --- i.e. a subset of `LocalActionDiagramEdges(lad)`. If given, the edges specified in this list are highlighted. This could, for example, highlight a particular *scopo* in *lad*.

By default the output file is written in the current working directory --- i.e. `DirectoryCurrent()`. The optional argument *directory* can either be a string or directory object. If it is given then the file will be output to the specified directory.

# Index

`GenerateTikZCode`  
for `IsLocalActionDiagram`, `IsString`, 9

`IsLocalActionDiagram`  
for `IsDigraph`, 4

`LocalActionDiagramEdgeLabels`  
for `IsLocalActionDiagram`, 6

`LocalActionDiagramEdgeReversal`  
for `IsLocalActionDiagram`, 6

`LocalActionDiagramEdges`  
for `IsLocalActionDiagram`, 6

`LocalActionDiagramFromData`  
for `IsDigraph`, `IsList`, `IsList`, `IsPerm`, 4

`LocalActionDiagramFromDataNC`  
for `IsDigraph`, `IsList`, `IsList`, `IsPerm`, 4

`LocalActionDiagramGroupType`  
for `IsLocalActionDiagram`, 7

`LocalActionDiagramIsDiscrete`  
for `IsLocalActionDiagram`, 7

`LocalActionDiagramIsUniscalar`  
for `IsLocalActionDiagram`, 7

`LocalActionDiagramScopos`  
for `IsLocalActionDiagram`, 7

`LocalActionDiagramToUniversalGroup`  
for `IsLocalActionDiagram`, 5

`LocalActionDiagramUniversalGroup`  
for `IsPermGroup`, 4

`LocalActionDiagramVertexLabels`  
for `IsLocalActionDiagram`, 6

`ReadLocalActionDiagram`  
for `IsString`, 8

`WriteLocalActionDiagram`  
for `IsList`, `IsString`, 8  
for `IsLocalActionDiagram`, `IsString`, 8