typeset

Automatic typesetting framework for common GAP objects, with LaTeX generation

1.2.2

1 March 2024

Zachariah Newbery

Zachariah Newbery
Email: me@zachnewbery.com
Homepage: https://zachnewbery.com

Contents

Chapter 1

Introduction

1.1 Core Framework Functions

typeset is a package that implements a typesetting framework that can be implemented for numerous typesetting languages as a standardised way to generate renderable strings.

At it's core, it implements the function `Typeset` (1.1.2) which makes use of typesetting language-specific functions to generate format strings. These strings are then populated with a list of the semantic features of the GAP objects they represent, which is obtained from the operation `GenArgs` (1.2.1).

An example implementation of this framework is also provided by this package for LaTeX typesetting within chapter 2.

Guidelines for extending the framework to support more types, or for implementing the framework for another typesetting language can be found within the contributing guidelines in the GitHub repository.

1.1.1 InfoTypeset

▷ `InfoTypeset` (info class)

Info class for the typeset package. Set this to the following levels for different levels of information:

- 0 - No messages
- 1 - Problems only: messages describing what went wrong, with no messages if an operation is successful
- 2 - Required preamble packages: displays informations about any required LaTeX packages that need to be added to the preamble to be rendered.
- 3 - Progress: also shows step-by-step progress of operations

Set this using, for example `SetInfoLevel(InfoTypeset, 1)`. Default value is 2.

1.1.2 Typeset

▷ `Typeset(`*`obj[, options]`*`)` (function)

Returns: A String, if `ReturnStr` option is set to `true`

Generates a mark-up string representing the object *obj* in the given mark-up language. GAP options can also be added to modify the result:

- `ReturnStr` : Whether the method should return a string (`true`), or simply print the result (`false`). (default - `false`)
- `LDelim` : Left Delimiter for matrices. (default - `"("`)
- `RDelim` : Right Delimiter for matrices. (default - `")"`)
- `Lang` : Markup language of output, currently only `"latex"` is supported. (default - `"latex"`)
- `DigraphOut` : Typesetting method for Digraphs, one of `"dot"` to use raw dot within TeX, or `"dot2tex"` to convert the dot to native TeX. (default - `"dot"`)
- `SubCallOpts` : Alternate GAP options for nested sub-objects, via a record with the same options as the parent (but different values), or `false` if all options are to stay the same between sub-calls. Options merging is handled by `MergeSubOptions` (1.3.1). (default - `false`) either by specifying each options as an individual GAP options like below:

Example

```
gap> Typeset([[1, 2], [2, 1]] : LDelim := "[", ReturnStr := true);
"\\left[\\begin{array}{rr}\n1 & 2 \\\\\n2 & 1 \\\\\n\\end{array}\\right)\n"
```

or wrapping them in a record under an `options` GAP option, like:

Example

```
gap> Typeset([[1, 2], [2, 1]] : options := rec(LDelim := "[", ReturnStr := true));
"\\left[\\begin{array}{rr}\n1 & 2 \\\\\n2 & 1 \\\\\n\\end{array}\\right)\n"
```

or even simply passing a record object as the optional second argument:

Example

```
gap> Typeset([[1, 2], [2, 1]], rec(LDelim := "[", ReturnStr := true));
"\\left[\\begin{array}{rr}\n1 & 2 \\\\\n2 & 1 \\\\\n\\end{array}\\right)\n"
```

1.1.3 TypesetInternal

▷ `TypesetInternal(`*`obj`*`)` (function)

Returns: A String

Generates a string representation of a passed GAP object *obj* that can be rendered by a typesetter. Called from the top-level method `Typeset` (1.1.2), which also passes a constructed options record as the `options` GAP option.

1.2 Core Operations

1.2.1 GenArgs (for IsObject)

▷ `GenArgs(`*`obj`*`)` (operation)

Returns: A List of Strings

Generates the arguments describing the semantic definition of the passed GAP object *obj*. This returns a list that can be used to populate a format string in any mark-up language. If no method is installed for a type, it will fallback to returning the list [ViewString(obj)].

1.3 Constants and Utility Functions

1.3.1 MergeSubOptions

▷ `MergeSubOptions(`*`opts`*`)` (function)

Returns: A Record

Merges the passed options record *opts* to change any values that are set in the GAP option `SubCallOpts`. If this option is not false (default), it can contain a record of any GAP options that can be passed to `Typeset` (1.1.2) which should differ for sub-calls.

For example, to alter the delimiters for nested objects so that the outer object is delimited by square braces and the inner object by parentheses, the following can be set:

Example

```
gap> MergeSubOptions(rec(ReturnStr := false, Lang := "latex", DigraphOut := "dot", RDelim := "]",
rec(ReturnStr := false, Lang := "latex", DigraphOut := "dot", RDelim := ")", LDelim := "(", SubCa
```

It should be noted that `SubCallOpts` only changes the options for one level of recursion (i.e. it is set back to the default of `false` once this function is called). To change options for more recursion levels, the `SubCallOpts` option can be nested as many times as necessary.

1.3.2 DEFAULT_TYPESET_OPTIONS

▷ `DEFAULT_TYPESET_OPTIONS` (global variable)

Default options record passed to `Typeset` (1.1.2). Merged with user-provided options to ensure correct construction of options for sub-calls, whilst also allowing option-less calls to the method.

Chapter 2

LaTeX Generation

2.1 LaTeX Generation Functions for GAP Objects

The introduction in 1.1 describes a powerful framework created in this package allowing for a standardised methodology to generate typesetting strings via the semantic features of objects.

This section describes the implementation of the framework for LaTeX, providing both the invaluable functionality to typeset a subset of GAP objects as LaTeX strings whilst also serving as an example of how the framework can be used for other typesetting languages.

It also provides some insight into the kinds of functions that would be expected from an implementation for a different language. A bare-boned example implementation for MathML is provided in 4.

Currently, the following types have explicit methods installed for LaTeX generation:

- Rationals (Integers and Fractions)
- Infinity and Negative Infinity
- Internal Finite-Field Elements
- Permutations
- Lists
- Matrices
- Polynomials and Non-Polynomial Rational Functions
- Character Tables
- Generators for FP, PC, Matrix and Permutation Groups
- Associative Words in Letter Representation

It should also be noted that `Typeset` (1.1.2) does fallback to the core library function `ViewString` (**Reference: ViewString**), which can be used for any types which do not require LaTeX-specific formatting to be renderable in math mode (e.g. floats).

2.1.1 GenLatexTmpl (for IsObject)

▷ `GenLatexTmpl(`*`obj`*`)` (operation)

Returns: An Unpopulated LaTeX Format String

Generates a format string that represents the structural definition of the given GAP object *obj* in LaTeX. It contains no parameter values, and will need to be populated with the arguments representing the semantic values of the object, generated via `GenArgs` (1.2.1), before it can be rendered in a LaTeX environment.

2.1.2 CtblEntryLatex

▷ `CtblEntryLatex(`*`s`*`)` (function)

Returns: A String

Formats a string representation of an entry *s* returned by the undocumented function `CharacterTableDisplayStringEntryDefault` to include the LaTeX-specific bar environment for complex conjugates.

2.1.3 CtblLegendLatex

▷ `CtblLegendLatex(`*`data`*`)` (function)

Returns: A String

Generates a string representation of the mathematical substitutions *data*, generated by the undocumented function `CharacterTableDisplayStringEntryDataDefault` for entries within a character table.

2.1.4 GenNameAssocLetterLatex

▷ `GenNameAssocLetterLatex(`*`s`*`)` (function)

Returns: A String

Generates a string representation of the provided letter string *s* correctly subscripted with a LaTeX math-mode subscript environment.

2.1.5 FactoriseAssocWordLatex

▷ `FactoriseAssocWordLatex(`*`l, names, tseed`*`)` (function)

Returns: A Factorised String

Factorises the string representation of an assoc word in letter representation *l*, based on the return value from the undocumented function `FindSubstringPowers`, using the passed list of letters *names*, and a list of reserved numbers *tseed* (typically empty for initial calls).

This method is essentially a LaTeX-specific implementation of the the undocumented function `NiceStringAssocWord`.

2.2 Rendering LaTeX Strings

To aid users using `Typeset` (1.1.2), being able to view the results quickly in a variety of widely-used formats would help streamline the usage of the package. As such, a number of functions described below have been written to enable users to render the output LaTeX-renderable snippets in different fashions.

2.2.1 RenderLatex

▷ RenderLatex(*str[, options]*) (function)

Renders a given LaTeX string *str* in a LaTeX environment, providing a visual example of what the string would look like in a paper. By default, this involved creating a HTML file that includes the MathJax script, but the GAP option *output* can be passed to change the rendering method.

Currently implemented rendering methods are:

- "mathjax", calling MathJax (2.2.3)
- "pdflatex", calling PDFLatex (2.2.2)
- "overleaf", calling Overleaf (2.2.4)

These functions can be called independently of RenderLatex, which serves only to be a more centralised rendering method for users.

2.2.2 PDFLatex

▷ PDFLatex(*str*) (function)

Renders a given LaTeX string *str* in a new PDF file, specifically via the pdflatex bash tool.

2.2.3 MathJax

▷ MathJax(*str*) (function)

Renders a given LaTeX string *str* in a HTML file, making use of the MathJax and TikzJax scripts.

2.2.4 Overleaf

▷ Overleaf(*str*) (function)

Renders a given LaTeX string *str* in a new Overleaf project, specifically via a URL-encoded snippet.

2.2.5 URIEncodeComponent

▷ URIEncodeComponent(*raw*) (function)

Returns: A Percent-Encoded String

Replaces reserved characters within a URI component *raw* as per RFC-3986.

2.2.6 NeedsLatexMathMode

▷ NeedsLatexMathMode(*raw*) (function)

Returns: A Boolean

Determines if the LaTeX snippet *raw* requires LaTeX's math mode to be rendered correctly.

2.2.7 DEFAULT_LATEX_PREAMBLE

▷ DEFAULT_LATEX_PREAMBLE (global variable)

Default LaTeX preamble string used for creating compilable .tex files from LaTeX snippets.

2.2.8 DEFAULT_MATHJAX_TAGS

▷ DEFAULT_MATHJAX_TAGS (global variable)

Default HTML document and head tags used to create HTML files using MathJax to render LaTeX snippets.

2.2.9 ALWAYS_UNESCAPED_CHARS

▷ ALWAYS_UNESCAPED_CHARS (global variable)

String containing all of the characters that do not need to be percent-encoded within URI components, as per RFC-3986.

2.3 Digraphs Integration

digraphs is a powerful, widely-used packages, implementing helpful functionality to work with directed graphs amongst other objects. Due to it's popularity, and as a way to demonstrate how typeset can be integrated with external packages, the following functions have been implemented to allow directed graphs to be converted into LaTeX representations.

It should be noted that the conversion implemented here does use the output from DotDigraph (**Digraphs: DotDigraph**), which generates the DOT string representing a digraph. This is then used to either convert it to a tikz representation via the command-line tool dot2tex (using the GAP option digraphOut := "dot2tex"), or simply wrapping it up in a dot2tex environment provided by the LaTeX package dot2texi which will compile the wrapped DOT input into tikz during compilation of the LaTeX file itself.

While another method could be written to convert the internal representation of a directed graph directly into a tikzpicture, this would likely be incredibly convoluted and difficult, and may present numerous problems with edge positioning. Therefore, relying on dot2tex was chosen as the best approach.

2.3.1 DEFAULT_DOT2TEX_OPTIONS

▷ DEFAULT_DOT2TEX_OPTIONS (global variable)

Default command-line options passed to the dot2tex executable to convert dot strings.

2.3.2 Dot2Tex

▷ Dot2Tex(*obj*) (function)

Returns: A Tikz String

Executes dot2tex on the dot string representing a given digraph object *obj*.

Chapter 3

Structure Descriptions

3.1 Typesetting Structure Descriptions of Groups

3.1.1 TypesetStructureDescription

▷ `TypesetStructureDescription(`*`desc`*`)` (function)

Returns: A String

Generates a typesettable representation equivalent to a given structure description *`desc`* of a group. Said structure descriptions can be calculated via `StructureDescription` (**Reference: StructureDescription**).

3.1.2 LatexStructureDescription

▷ `LatexStructureDescription(`*`desc`*`)` (function)

Returns: A String

Generates a LaTeX representation equivalent to a given structure description *`desc`*. Called by `TypesetStructureDescription` (3.1.1) as the default markup language for structure description typesetting.

3.1.3 ConcatStructDescOperands

▷ `ConcatStructDescOperands(`*`desc, sep, newsep`*`)` (function)

Returns: A String

Concatenates the tokens parsed from splitting the string *`desc`* with the provided separator string *`sep`*. It will then process the tokens with `LatexStructureDescription` (3.1.2), and concatenate the results with the given new separator *`newsep`*.

While `StructureDescription` (**Reference: StructureDescription**) provides powerful functionality in calculating the structure description of an input group, the returned string is simply raw text, and does not look good when rendered in typical typesetting environment.

To improve this, the following functions have been written to help convert the structure description strings into typesettable representations, that use language-specific features to ensure that displaying structure descriptions aesthetically is easy and efficient.

Chapter 4

MathML Example Generation

4.1 MathML Generation Functions for GAP Objects

This section describes a bare-bones implementation of the framework for generating MathML representations. It is intended to be used as a reference for implementing the framework, and as a starting point for implementing the framework for other languages - not as a fully functional implementation of the framework.

4.1.1 GenMathmlTmpl (for IsObject)

▷ `GenMathmlTmpl(`*`obj`*`)` (operation)

Returns: An Unpopulated MathML Format String

Generates a format string that represents the structural definition of the given GAP object *obj* in MathML. It contains no parameter values, and will need to be populated with the arguments representing the semantic values of the object, generated via `GenArgs` (1.2.1), before it can be rendered in a MathML environment.

Index