Giorgio Gonnella

RGFA library - API documentation

Version 1.1

Table of Contents

Documentation by YARD 0.8.7.6

The Graphical Fragment Assembly (GFA) is a proposed format which allow to describe the product of sequence assembly. This gem implements the proposed specifications for the GFA format described under github.com/pmelsted/GFA-spec/blob/master/GFA-spec.md as close as possible.

The library allows to create a GFA object from a file in the GFA format or from scratch, to enumerate the graph elements (segments, links, containments, paths and header lines), to traverse the graph (by traversing all links outgoing from or incoming to a segment), to search for elements (e.g. which links connect two segments) and to manipulate the graph (e.g. to eliminate a link or a segment or to duplicate a segment distributing the read counts evenly on the copies).

Top Level Namespace

Defined Under Namespace

Classes: Array, Float, Hash, Integer, Object, String

Constant Summary

RGFA =

© 2016, Giorgio Gonnella, ZBH, Uni-Hamburg <gonnella@zbh.uni-hamburg.de>

```
Class.new
```

Module: RGFA::Edit

Included in:	RGFA
Defined in:	lib/rgfa/edit.rb

Overview

Methods for the RGFA class, which allow to modify the content of the graph without requiring complex graph traversal.

See Also:

- Traverse

Instance Method Summary

(collapse)

- (RGFA) **delete_alignments**
  Eliminate all CIGAR from L/C/P lines, changing them to "*".
- (RGFA) **delete_sequences**
  Eliminate all sequences from S lines, changing them to a "*".
- (RGFA) **multiply**(segment, factor, copy_names: :lowcase, conserve_components: true)
  Create multiple copies of a segment.
- (RGFA) **rename**(old_name, new_name)
  Rename a segment or a path.

Instance Method Details

```
- (RGFA) delete_alignments
```

Eliminate all CIGAR from L/C/P lines, changing them to "*"

Returns:

- (RGFA) — self

```
- (RGFA) delete_sequences
```

Eliminate all sequences from S lines, changing them to a "*"

Returns:

- (RGFA) — self

```
- (RGFA) multiply(segment, factor, copy_names: :lowcase, conserve_components:
true)
```

Create multiple copies of a segment.

Automatic computation of the copy names:

- Can be overridden, by providing an array of copy names.
- First, itis checked if the name of the original segment ends with a relevant string,

i.e. a lower case letter (for `:lowcase`), an upper case letter (for `:upcase`), a digit (for `:number`), or the string "`_copy`" plus one or more optional digits (for `:copy`).
- If so, it is assumed, it was already a copy, and it is not altered.
- If not, then `a` (for `:lowcase`), `A` (for `:upcase`), `1` (for `:number`), `_copy` (for `:copy`) is appended to the string.
- Then, in all cases, next (*) is called on the string, until a valid, non-existant name is found for each of the segment copies
- (*) = except for `:copy`, where for the first copy no digit is present, but for the following is, i.e. the segment names will be `:copy`, `:copy2`, `:copy3`, etc.

Parameters:

- **`factor`** (`Integer`) — multiplication factor; if 0, delete the segment; if 1; do nothing; if > 1; number of copies to create
- **`segment`** (`String`, `RGFA::Line::Segment`) — segment name or instance
- **`copy_names`** (`:lowcase`, `:upcase`, `:number`, `:copy`, `Array<String>`) — *(Defaults to: `:lowcase`)* Array of names for the copies of the segment, or a symbol, which defines a system to compute the names from the name of the original segment. See "automatic computation of the copy names".
- **`conserve_components`** (`Boolean`) — *(Defaults to: `true`)* If factor == 0 (i.e. deletion), delete segment only if Traverse#cut_segment?(segment) is `false`.

Returns:

- (`RGFA`) — self

```
- (RGFA) rename(old_name, new_name)
```

Rename a segment or a path

@raise

```
if +new_name+ is already a segment or path name
```

Parameters:

- **`old_name`** (`String`) — the name of the segment or path to rename
- **`new_name`** (`String`) — the new name for the segment or path

Returns:

- (`RGFA`) — self

Module: RGFA::CIGAR

Included in:	String
Defined in:	lib/rgfa/cigar.rb

Overview

Extensions of the String class to handle CIGAR strings

Defined Under Namespace

Classes: ValueError

Instance Method Summary

(collapse)

- ("*", Array<RGFA::CigarOperation>) **cigar_operations**
  Parses a CIGAR string into an array of cigar operations, each represented by a tuple of operation length and operation symbol (one of MIDNSHPX=).
- ("*", String) **reverse_cigar**
  Parses a CIGAR string representing an overlap and reverses it, i.e.
- ("*", Array<RGFA::CigarOperation>) **reverse_cigar_operations**
  Parses a CIGAR string representing an overlap and reverses it, i.e.

Instance Method Details

- ("*", Array<RGFA::CigarOperation>) cigar_operations

Parses a CIGAR string into an array of cigar operations, each represented by a tuple of operation length and operation symbol (one of MIDNSHPX=).

Returns:

- ("*") — if self == "*"
- (Array<RGFA::CigarOperation>) — otherwise

Raises:

- (TypeError) — if the string is not a valid CIGAR string

- ("*", String) reverse_cigar

Parses a CIGAR string representing an overlap and reverses it, i.e. computes the CIGAR for the segments in reverse direction.

Examples:

```
"2M1D3M".reverse_cigar # => "3M1I2M"

# S1 + S2 + 2M1D3M
#
# S1+  ACGACTGTGA
# S2+      CT-TGACGG
```

```
#
# S2-  CCGTCA-AG
# S1-     TCACAGTCGT
#
# S2 - S1 - 3M1I2M
```

Returns:

- ("*") — if self == "*"
- (String) — the reverse CIGAR, otherwise

Raises:

- (TypeError) — if the string is not a valid CIGAR string

See Also:

- #reverse_cigar_operations

```
- ("*", Array<RGFA::CigarOperation>) reverse_cigar_operations
```

Parses a CIGAR string representing an overlap and reverses it, i.e. computes the CIGAR for the segments in reverse direction. Returns an array of CIGAR operations.

Returns:

- ("*") — if self == "*"
- (Array<RGFA::CigarOperation>) — otherwise

Raises:

- (TypeError) — if the string is not a valid CIGAR string

See Also:

- #reverse_cigar

Module: RGFA::LoggerSupport

Included in:	RGFA
Defined in:	lib/rgfa/logger.rb

Overview

Progress logging related-methods for RGFA class

Instance Method Summary

(collapse)

- (RGFA) **enable_progress_logging**(part: 0.1, channel: STDERR)
  Activate logging of progress.
- (RGFA) **progress_log**(symbol, progress = 1, **keyargs)
  Updates progress logging for a computation.
- (RGFA) **progress_log_end**(symbol, **keyargs)
  Completes progress logging for a computation.
- (RGFA) **progress_log_init**(symbol, units, total, initmsg = nil)
  Initialize progress logging for a computation.

Instance Method Details

- (RGFA) **enable_progress_logging**(part: 0.1, channel: STDERR)

Activate logging of progress

Returns:

- (RGFA) — self

- (RGFA) **progress_log**(symbol, progress = 1, **keyargs)

Updates progress logging for a computation

Parameters:

- **symbol** (Symbol) — the symbol assigned to the computation at init time
- **keyargs** (Hash) — additional units to display, with their current value (e.g. segments_processed: 10000)
- **progress** (Integer) *(defaults to: 1)* — how many units were processed

Returns:

- (RGFA) — self

- (RGFA) **progress_log_end**(symbol, **keyargs)

Completes progress logging for a computation

Parameters:

- **symbol** (Symbol) — the symbol assigned to the computation at init time
- **keyargs** (Hash) — additional units to display, with their current value (e.g.

segments_processed: 10000)

Returns:

- (RGFA) — self

```
- (RGFA) progress_log_init(symbol, units, total, initmsg = nil)
```

Initialize progress logging for a computation

Parameters:

- **symbol** (Symbol) — a symbol assigned to the computation
- **units** (String) — a string with the name of the units, in plural
- **total** (Integer) — total number of units
- **initmsg** (String) *(defaults to: nil)* — an optional message to output at the beginning

Returns:

- (RGFA) — self

Module: RGFA::Sequence

Included in:	String
Defined in:	lib/rgfa/sequence.rb

Overview

Extensions of the String class to handle nucleotidic sequences

Constant Summary

WCC =

Watson-Crick Complements

```
{"a"=>"t","t"=>"a","A"=>"T","T"=>"A",
"c"=>"g","g"=>"c","C"=>"G","G"=>"C",
"b"=>"v","B"=>"V","v"=>"b","V"=>"B",
"h"=>"d","H"=>"D","d"=>"h","D"=>"H",
"R"=>"Y","Y"=>"R","r"=>"y","y"=>"r",
"K"=>"M","M"=>"K","k"=>"m","m"=>"k",
"S"=>"S","s"=>"s","w"=>"w","W"=>"W",
"n"=>"n","N"=>"N","u"=>"a","U"=>"A",
"-"=>"-",".">".","="=>"=",
" "=>"","\n"=>""}
```

Instance Method Summary

(collapse)

- (String) **rc**(tolerant: false, rnasequence: false)

  Computes the reverse complement of a nucleotidic sequence.

Instance Method Details

```
- (String) rc(tolerant: false, rnasequence: false)
```

Computes the reverse complement of a nucleotidic sequence

Examples:

```
"ACTG".rc  # => "CAGT"
"acGT".rc  # => "ACgt"
```

Undefined sequence is represented by "*":

```
"*".rc     # => "*"
```

Extended IUPAC Alphabet:

```
"ARBN".rc  # => "NVYT"
```

Usage with RNA sequences:

```
"ACUG".rc                    # => "CAGU"
"ACG".rc(rnasequence: true)  # => "CGU"
"ACUT".rc                    # (raises RuntimeError, both U and T)
```

Parameters:

- **`tolerant`** (`Boolean`) — *(defaults to: `false`)* if true, anything non-sequence is complemented to itself
- **`rnasequence`** (`Boolean`) — *(defaults to: `false`)* if true, any A and a is complemented into u and U; otherwise it is so, only if an U is found; otherwise DNA is assumed

Returns:

- (`String`) — reverse complement, without newlines and spaces
- (`String`) — "*" if string is "*"

Raises:

- (`RuntimeError`) — if not `tolerant` and chars are found for which no Watson-Crick complement is defined
- (`RuntimeError`) — if sequence contains both U and T

Module: RGFA::Traverse

Included in:	RGFA
Defined in:	lib/rgfa/traverse.rb

Overview

Methods for the RGFA class, which involve a traversal of the graph following links

Instance Method Summary

(collapse)

- (Array<Array<String>>) **connected_components**
  Find the connected components of the graph.
- (Array<conn_symbol,conn_symbol>) **connectivity**(segment)
  Computes the connectivity of a segment from its number of links.
- (Boolean) **cut_link?**(link)
  Does the removal of the link alone divide a component of the graph into two?.
- (Boolean) **cut_segment?**(segment)
  Does the removal of the segment and its links divide a component of the graph into two?.
- (Array<RGFA::SegmentEnd>) **linear_path**(segment, exclude = Set.new)
  Find an eventual path without branches which includes `segment` and excludes segments in `exclude`.
- (Array<Array<RGFA::SegmentEnd>>) **linear_paths**
  Find all unbranched paths of segments connected by links in the graph.
- (RGFA) **merge_linear_path**(segpath, **options)
  Merge a linear path, i.e.
- (RGFA) **merge_linear_paths**(**options)
  Merge all linear paths in the graph, i.e.
- (Array<String>) **segment_connected_component**(segment, visited = Set.new)
  Find the connected component of the graph in which a segment is included.
- (Array<RGFA>) **split_connected_components**
  Split connected components of the graph into single-component RGFAs.

Instance Method Details

- (Array<Array<String>>) **connected_components**

Find the connected components of the graph

Returns:

- (Array<Array<String>>) — array of components, each an array of segment names

---

- (Array<conn_symbol,conn_symbol>) **connectivity**(segment)

Computes the connectivity of a segment from its number of links.

Connectivity symbol: (conn_symbol)

- Let *n* be the number of links to an end (`:B` or `:E`) of a segment. Then the connectivity symbol is `:M` if *n* > *1*, otherwise *n*.

Parameters:

- **segment** (`String|RGFA::Line::Segment`) — segment name or instance

Returns:

- (`Array<conn_symbol,conn_symbol>`) — conn. symbols respectively of the :B and :E ends of `segment`.

```
- (Boolean) cut_link?(link)
```

Returns does the removal of the link alone divide a component of the graph into two?

Parameters:

- **link** (`RGFA::Line::Link`) — a link

Returns:

- (`Boolean`) — does the removal of the link alone divide a component of the graph into two?

```
- (Boolean) cut_segment?(segment)
```

Returns does the removal of the segment and its links divide a component of the graph into two?

Parameters:

- **segment** (`String`, `RGFA::Line::Segment`) — a segment name or instance

Returns:

- (`Boolean`) — does the removal of the segment and its links divide a component of the graph into two?

```
- (Array<RGFA::SegmentEnd>) linear_path(segment, exclude = Set.new)
```

Find an eventual path without branches which

```
includes +segment+ and excludes segments in +exclude+.
```

Any segment used in the returned path will be added to `exclude`

Parameters:

- **segment** (`String|RGFA::Line::Segment`) — a segment name or instance
- **exclude** (`Set<String>`) *(defaults to: `Set.new`)* — a set of segment names to exclude from the path

Returns:

- (`Array<RGFA::SegmentEnd>`)

```
- (Array<Array<RGFA::SegmentEnd>>) linear_paths
```

Find all unbranched paths of segments connected by links in the graph.

Returns:

- (`Array<Array<RGFA::SegmentEnd>>`)

- (RGFA) **merge_linear_path**(segpath, **options)

Merge a linear path, i.e. a path of segments without extra-branches Limitations: all containments und paths involving merged segments are deleted.

Parameters:

- **segpath** (Array<RGFA::SegmentEnd>) — a linear path, such as that retrieved by #linear_path
- **options** (Hash) — optional keyword arguments

Options Hash (options):**

- **:merged_name** (String, :short, nil) — default: nil — if nil, the merged_name is automatically computed; if :short, a name is computed starting with "merged1" and calling next until an available name is founf; if String, the name to use
- **:cut_counts** (Boolean) — default: false — if true, total count in merged segment m, composed of segments s of set S is multiplied by the factor Sum(|s in S|)/|m|

Returns:

- (RGFA) — self

See Also:

- #merge_linear_paths

- (RGFA) **merge_linear_paths**(**options)

Merge all linear paths in the graph, i.e. paths of segments without extra-branches Limitations: all containments und paths involving merged segments are deleted.

Parameters:

- **options** (Hash) — optional keyword arguments

Options Hash (options):**

- **:merged_name** (String, :short, nil) — default: nil — if nil, the merged_name is automatically computed; if :short, a name is computed starting with "merged1" and calling next until an available name is founf; if String, the name to use
- **:cut_counts** (Boolean) — default: false — if true, total count in merged segment m, composed of segments s of set S is multiplied by the factor Sum(|s in S|)/|m|

Returns:

- (RGFA) — self

- (Array<String>) **segment_connected_component**(segment, visited = Set.new)

Find the connected component of the graph in which a segment is included

Parameters:

- **segment** (String, RGFA::Line::Segment) — a segment name or instance
- **visited** (Set<String>) *(defaults to: Set.new)* — a set of segments to ignore during graph traversal; all segments in the found component will be added to it

Returns:

- (Array<String>) — array of segment names

- (Array<RGFA>) **split_connected_components**

Split connected components of the graph into single-component RGFAs

Returns:

- (Array<RGFA>)

Module: RGFA::FieldParser

Included in:	String
Defined in:	lib/rgfa/field_parser.rb

Overview

Methods to parse the string representations of the GFA fields

Defined Under Namespace

Classes: FormatError, UnknownDatatypeError

Instance Method Summary

(collapse)

- (Object) **parse_datastring**(datatype)
  Parse a string representation of a GFA field value; it is assumed that the string is valid with respect to the specified `datatype`.
- (Array(Symbol, RGFA::Line::FIELD_DATATYPE, String)) **parse_optfield**
  Parses an optional field in the form tagname:datatype:value and parses the value according to the datatype.

Instance Method Details

- (Object) **parse_datastring**(datatype)

Parse a string representation of a GFA field value; it is assumed that the string is valid with respect to the specified `datatype`.

Parameters:

- **datatype** (RGFA::Line::FIELD_DATATYPE)

- (Array(Symbol, RGFA::Line::FIELD_DATATYPE, String)) **parse_optfield**

Parses an optional field in the form tagname:datatype:value and parses the value according to the datatype

Returns:

- (Array(Symbol, RGFA::Line::FIELD_DATATYPE, String)) — the parsed content of the field

Raises:

- (RGFA::FieldParser::FormatError) — if the string does not represent an optional field

Module: RGFA::FieldWriter

Included in:	Object
Defined in:	lib/rgfa/field_writer.rb

Overview

Methods to convert ruby objects to the GFA string representations

The default conversion is implemented in this module, which is included in Object;
single classes may overwrite the following methods, if necessary:

- #gfa_datatype, which returns the symbol of the optional field GFA datatype to use,
  if none is specified (See RGFA::Line::FIELD_DATATYPE); the default is :Z
- #to_gfa_datastring(datatype) should return a GFA string representation, eventually
  depending on the specified datatype; no validation is done; the default is #to_s

Instance Method Summary

(collapse)

- (RGFA::Line::FIELD_DATATYPE) **gfa_datatype**

  Optional field GFA datatype to use, if none is provided.

- (String) **to_gfa_datastring**(datatype)

  Representation of the data for GFA fields; this method does not automatically validate the string.

- (String) **to_gfa_field**(datatype: nil, validate: true, fieldname: nil, optfield: false)

  Representation of an object as GFA field.

Instance Method Details

```
- (RGFA::Line::FIELD_DATATYPE) gfa_datatype
```

Optional field GFA datatype to use, if none is provided

Returns:

- (RGFA::Line::FIELD_DATATYPE)

```
- (String) to_gfa_datastring(datatype)
```

Representation of the data for GFA fields; this method does not automatically validate the string.

Parameters:

- **datatype** (RGFA::Line::FIELD_DATATYPE)

Returns:

- (String)

```
- (String) to_gfa_field(datatype: nil, validate: true, fieldname: nil,
optfield: false)
```

Representation of an object as GFA field

Parameters:

- **`datatype`** (`RGFA::Line::FIELD_DATATYPE`, `nil`) — <i>(default: `nil`) one of the provided GFA datatypes; if `nil`, it is determined using the #gfa_datatype method
- **`validate`** (`Boolean`) — *(default: `true`)* validate the data string using the predefined regular expression, depending on the datatype (RGFA::FieldValidator::DATASTRING_VALIDATION_REGEXP)
- **`fieldname`** (`Symbol`, `nil`) — *(default: `nil`)* fieldname to use for error messages and for the output if `optfield` is true
- **`optfield`** (`Boolean`) — *(default: `false`)* if true, the output will contain field name, datatype symbol and datastring joined by `:`; otherwise only the datastring is returned

Returns:

- (`String`) — the GFA field content

Module: RGFA::LineGetters

Included in:	RGFA
Defined in:	lib/rgfa/line_getters.rb

Overview

Methods for the RGFA class, which allow to retrieve specific lines.

Instance Method Summary

(collapse)

- (Array<String>) **connected_segments**(segment)
  List of names of segments connected to `segment` by links or containments.
- (Array<RGFA::Line::Containment>) **contained_in**(segment)
  Find containment lines whose `from` segment name is `segment_name`.
- (Array<RGFA::Line::Containment>) **containing**(segment)
  Find containment lines whose `to` segment name is `segment_name`.
- (RGFA::Line::Containment?) **containment**(container, contained)
  Searches a containment of `contained` in `container`.
- (RGFA::Line::Containment) **containment!**(container, contained)
  Searches a containment of `contained` in `container`.
- (Array<RGFA::Line::Containment>) **containments**
  All containments of the graph.
- (Array<RGFA::Line::Containment>) **containments_between**(container, contained)
  Searches all containments of `contained` in `container`.
- (Object) **each_containment** {|RGFA::Line::Containment| ... }
  Iterate over all containments of the graph.
- (Object) **each_header** {|RGFA::Line::Header| ... }
  Iterate over all header lines of the graph.
- (Object) **each_line** {|RGFA::Line| ... }
  Iterate over all lines of the graph.
- (Object) **each_link** {|RGFA::Line::Link| ... }
  Iterate over all links of the graph.
- (Object) **each_path** {|RGFA::Line::Path| ... }
  Iterate over all path lines of the graph.
- (Object) **each_segment** {|RGFA::Line::Segment| ... }
  Iterate over all segments of the graph.
- (Array<RGFA::Line::Header>) **headers**
  All header lines of the graph.
- (Hash{Symbol:Object}) **headers_data**
  Data contained in the header fields; the special key :multiple_values contains an array of fields for which multiple values were defined in multiple lines; in this case the values are summarized in an array.
- (RGFA::Line::Link?) **link**(segment_end1, segment_end2)
  Searches a link between `segment_end1` and `segment_end2`.
- (RGFA::Line::Link) **link!**(segment_end1, segment_end2)
  Searches a link between `segment_end1` and `segment_end2`.

- (Array<RGFA::Line::Link>) **links**

  All links of the graph.

- (Array<RGFA::Line::Link>) **links_between**(segment_end1, segment_end2)

  Searches all links between `segment_end1` and `segment_end2`.

- (Array<RGFA::Line::Link>) **links_of**(segment_end)

  Finds links of the specified end of segment.

- (Array<RGFA::SegmentEnd>) **neighbours**(segment_end)

  Finds segment ends connected to the specified segment end.

- (RGFA::SegmentEnd) **other_segment_end**(segment_end)

  The other end of a segment.

- (RGFA::Line::Path?) **path**(path_name)

  Searches the path with name equal to `path_name`.

- (RGFA::Line::Path) **path!**(path_name)

  Searches the path with name equal to `path_name`.

- (Array<RGFA::Line::Path>) **paths**

  All path lines of the graph.

- (Array<RGFA::Line::Path>) **paths_with**(segment)

  Paths whose `segment_names` include the specified segment.

- (RGFA::Line::Segment?) **segment**(segment_name)

  Searches the segment with name equal to `segment_name`.

- (RGFA::Line::Segment) **segment!**(segment_name)

  Searches the segment with name equal to `segment_name`.

- (Array<RGFA::Line::Segment>) **segments**

  All segments of the graph.

Instance Method Details

- (Array<String>) **connected_segments**(segment)

Returns list of names of segments connected to `segment` by links or containments

Returns:

- (Array<String>) — list of names of segments connected to `segment` by links or containments

---

- (Array<RGFA::Line::Containment>) **contained_in**(segment)

Find containment lines whose `from` segment name is `segment_name`

Parameters:

- **segment** (RGFA::Line::Segment, String) — a segment instance or name

Returns:

- (Array<RGFA::Line::Containment>)

- (Array<RGFA::Line::Containment>) **containing**(segment)

Find containment lines whose `to` segment name is `segment_name`

Parameters:

- **segment** (RGFA::Line::Segment, String) — a segment instance or name

Returns:

- (Array<RGFA::Line::Containment>)

```
- (RGFA::Line::Containment?) containment(container, contained)
```

Searches a containment of `contained` in `container`. Returns the first containment found or nil if none found.

Parameters:

- **container** (RGFA::Line::Segment, String) — a segment instance or name
- **contained** (RGFA::Line::Segment, String) — a segment instance or name

Returns:

- (RGFA::Line::Containment, nil)

```
- (RGFA::Line::Containment) containment!(container, contained)
```

Searches a containment of `contained` in `container`. Raises a `RuntimeError` if no containment was found.

Parameters:

- **container** (RGFA::Line::Segment, String) — a segment instance or name
- **contained** (RGFA::Line::Segment, String) — a segment instance or name

Returns:

- (RGFA::Line::Containment)

Raises:

- (RGFA::LineMissingError) — if no such containment found

```
- (Array<RGFA::Line::Containment>) containments
```

All containments of the graph

Returns:

- (Array<RGFA::Line::Containment>)

```
- (Array<RGFA::Line::Containment>) containments_between(container, contained)
```

Searches all containments of `contained` in `container`. Returns a possibly empty array of containments.

Parameters:

- **container** (RGFA::Line::Segment, String) — a segment instance or name
- **contained** (RGFA::Line::Segment, String) — a segment instance or name

Returns:

- (Array<RGFA::Line::Containment>)

```
- (Object) each_containment {|RGFA::Line::Containment| ... }
```

Iterate over all containments of the graph

Yields:

- (RGFA::Line::Containment)

- (Object) **each_header** {|RGFA::Line::Header| ... }

Iterate over all header lines of the graph

Yields:

- (RGFA::Line::Header)

- (Object) **each_line** {|RGFA::Line| ... }

Iterate over all lines of the graph

Yields:

- (RGFA::Line)

- (Object) **each_link** {|RGFA::Line::Link| ... }

Iterate over all links of the graph

Yields:

- (RGFA::Line::Link)

- (Object) **each_path** {|RGFA::Line::Path| ... }

Iterate over all path lines of the graph

Yields:

- (RGFA::Line::Path)

- (Object) **each_segment** {|RGFA::Line::Segment| ... }

Iterate over all segments of the graph

Yields:

- (RGFA::Line::Segment)

- (Array<RGFA::Line::Header>) **headers**

All header lines of the graph

Returns:

- (Array<RGFA::Line::Header>)

- (Hash{Symbol:Object}) **headers_data**

Returns data contained in the header fields; the special key :multiple_values contains

an array of fields for which multiple values were defined in multiple lines; in this case the values are summarized in an array

Returns:

- (`Hash{Symbol:Object}`) — data contained in the header fields; the special key :multiple_values contains an array of fields for which multiple values were defined in multiple lines; in this case the values are summarized in an array

---

```
- (RGFA::Line::Link?) link(segment_end1, segment_end2)
```

Searches a link between `segment_end1` and `segment_end2`

Parameters:

- **segment_end1** (`RGFA::SegmentEnd`) — a segment end
- **segment_end2** (`RGFA::SegmentEnd`) — a segment end

Returns:

- (`RGFA::Line::Link`) — the first link found
- (`nil`) — if no link is found.

```
- (RGFA::Line::Link) link!(segment_end1, segment_end2)
```

Searches a link between `segment_end1` and `segment_end2`

Parameters:

- **segment_end1** (`RGFA::SegmentEnd`) — a segment end
- **segment_end2** (`RGFA::SegmentEnd`) — a segment end

Returns:

- (`RGFA::Line::Link`) — the first link found

Raises:

- (`RGFA::LineMissingError`) — if no link is found.

```
- (Array<RGFA::Line::Link>) links
```

All links of the graph

Returns:

- (`Array<RGFA::Line::Link>`)

```
- (Array<RGFA::Line::Link>) links_between(segment_end1, segment_end2)
```

Searches all links between `segment_end1` and `segment_end2`

Parameters:

- **segment_end1** (`RGFA::SegmentEnd`) — a segment end
- **segment_end2** (`RGFA::SegmentEnd`) — a segment end

Returns:

- (`Array<RGFA::Line::Link>`) — (possibly empty)

- (Array<RGFA::Line::Link>) **links_of**(segment_end)

Note: to add or remove links, use the appropriate methods; adding or removing links from the returned array will not work

Finds links of the specified end of segment.

Parameters:

- **segment_end** (RGFA::SegmentEnd) — a segment end

Returns:

- (Array<RGFA::Line::Link>) — if segment_end == :E, links from sn with from_orient + and to sn with to_orient -
- (Array<RGFA::Line::Link>) — if segment_end == :B, links to sn with to_orient + and from sn with from_orient -

---

- (Array<RGFA::SegmentEnd>) **neighbours**(segment_end)

Finds segment ends connected to the specified segment end.

Parameters:

- **segment_end** (RGFA::SegmentEnd) — a segment end

Returns:

- (Array<RGFA::SegmentEnd>) —] segment ends connected by links to `segment_end`

- (RGFA::SegmentEnd) **other_segment_end**(segment_end)

Returns the other end of a segment

Parameters:

- **segment_end** (RGFA::SegmentEnd) — a segment end

Returns:

- (RGFA::SegmentEnd) — the other end of a segment

- (RGFA::Line::Path?) **path**(path_name)

Searches the path with name equal to `path_name`.

Parameters:

- **path_name** (String) — a path name

Returns:

- (RGFA::Line::Path) — if a path is found
- (nil) — if no such path exists in the RGFA instance

- (RGFA::Line::Path) **path!**(path_name)

Searches the path with name equal to `path_name`.

Parameters:

- **path_name** (String) — a path name

Returns:

- (RGFA::Line::Path) — if a path is found

Raises:

- (RGFA::LineMissingError) — if no such path exists in the RGFA instance

---

- (Array<RGFA::Line::Path>) **paths**

All path lines of the graph

Returns:

- (Array<RGFA::Line::Path>)

- (Array<RGFA::Line::Path>) **paths_with**(segment)

Returns paths whose `segment_names` include the specified segment.

Parameters:

- **segment** (RGFA::Line::Segment, String) — a segment instance or name

Returns:

- (Array<RGFA::Line::Path>) — paths whose `segment_names` include the specified segment.

- (RGFA::Line::Segment?) **segment**(segment_name)

Searches the segment with name equal to `segment_name`.

Parameters:

- **segment_name** (String) — a segment name

Returns:

- (RGFA::Line::Segment) — if a segment is found
- (nil) — if no such segment exists in the RGFA instance

- (RGFA::Line::Segment) **segment!**(segment_name)

Searches the segment with name equal to `segment_name`.

Parameters:

- **segment_name** (String) — a segment name

Returns:

- (RGFA::Line::Segment) — if a segment is found

Raises:

- (RGFA::LineMissingError) — if no such segment exists

- (Array<RGFA::Line::Segment>) **segments**

All segments of the graph

Returns:

- (Array<RGFA::Line::Segment>)

Module: RGFA::LineCreators

Included in:	RGFA
Defined in:	lib/rgfa/line_creators.rb

Overview

Methods for the RGFA class, which allow to add lines.

Instance Method Summary

(collapse)

- (RGFA) **<<**(gfa_line)
  Add a line to a RGFA.
- (RGFA) **set_header_field**(field, value, existing: :ignore)
  Sets the value of a field in the header.
- (RGFA) **set_headers**(headers_data)
  Sets the header data Multiple definitions of optional fields The specification does not forbid that different header line contain the same optional field.

Instance Method Details

```
- (RGFA) <<(gfa_line_string)
- (RGFA) <<(gfa_line)
```

Add a line to a RGFA

Overloads:

```
- (RGFA) <<(gfa_line_string)
```

Parameters:

- **gfa_line_string** (String) — representation of a RGFA line

```
- (RGFA) <<(gfa_line)
```

Parameters:

- **gfa_line** (RGFA::Line) — instance of a subclass of RGFA::Line

Returns:

- (RGFA) — self

Raises:

- (RGFA::DuplicatedLabelError) — if multiple segment or path lines with the same name are added

---

```
- (RGFA) set_header_field(field, value, existing: :ignore)
```

Sets the value of a field in the header

Parameters:

- **existing** (Symbol) — *(Default: :ignore)* what shall be done if a field already exist; :replace: the previous value is replaced by value; :duplicate: if the previous value is an array, value is added to it, otherwise the field is set to [previous value, value]; :ignore (and anything else): the new value is ignored.

Returns:

- (RGFA) — self

```
- (RGFA) set_headers(headers_data)
```

Sets the header data **Multiple definitions of optional fields**

```
The specification does not forbid that different header line contain
the same optional field. If this is desired, then the +headers_data+
for the optional field shall be set to an array, and
+headers_data[:multiple_values]+ shall be an array containing the tag
name.
```

Parameters:

- **headers_data** (Hash{Symbol:Object}) — data contained in the header fields; the key :multiple_values has a special meaning, see below.

Returns:

- (RGFA) — self

Module: RGFA::FieldValidator

Included in:	String
Defined in:	lib/rgfa/field_validator.rb

Overview

Methods to validate the string representations of the GFA fields data

Constant Summary

DATASTRING_VALIDATION_REGEXP =

Validation regular expressions, derived from the GFA specification

```
{
  :A => /^[!-~]$/,          # Printable character
  :i => /^[-+]?[0-9]+$/,    # Signed integer
  :f => /^[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?$/,
                            # Single-precision floating number
  :Z => /^[ !-~]+$/,        # Printable string, including space
  :J => /^[ !-~]+$/,        # JSON, excluding new-line and tab characters
  :H => /^[0-9A-F]+$/,      # Byte array in the Hex format
  :B => /^[cCsSiIf](,[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?)+$/,
                            # Integer or numeric array
  :lbl => /^[!-)+-<>-~][!-~]*$/,       # segment/path label
  :orn => /^\+|-$/,                    # segment orientation
  :lbs => /^[!-)+-<>-~][!-~]*[+-](,[!-)+-<>-~][!-~]*[+-])*$/,
                            # multiple labels with orientations, comma-sep
  :seq => /^\*$|^[A-Za-z=.]+$/,          # nucleotide sequence
  :pos => /^[0-9]*$/,                    # positive integer
  :cig => /^\*$|^([0-9]+[MIDNSHPX=])+$/, # CIGAR string
  :cgs => /^\*$|^([0-9]+[MIDNSHPX=])+((,[0-9]+[MIDNSHPX=])+)*$/,
                                  # multiple CIGARs, comma-sep
}
```

Instance Method Summary

(collapse)

- (void) **validate_datastring**(datatype, fieldname: nil)

  Validates the string according to the provided datatype.

Instance Method Details

- (void) **validate_datastring**(datatype, fieldname: nil)

This method returns an undefined value.

Validates the string according to the provided datatype

Parameters:

- **datatype** (RGFA::Line::FIELD_DATATYPE)
- **fieldname** (#to_s) — Fieldname to use in the error msg

Raises:

- (RGFA::FieldParser::FormatError) — if the string does not match the regexp for the provided datatype

Module: RGFA::LineDestructors

Included in:	RGFA
Defined in:	lib/rgfa/line_destructors.rb

Overview

Methods for the RGFA class, which allow to delete lines.

Instance Method Summary

(collapse)

- (RGFA) **delete_containment**(from, from_orient = nil, to = nil, to_orient = nil, pos = nil)
  Delete a containment of a segment in another, with given orientations and starting position.
- (RGFA) **delete_headers**
  Remove all headers.
- (RGFA) **delete_link**(from, from_orient = nil, to = nil, to_orient = nil)
  Delete a link from a segment to another.
- (RGFA) **delete_other_links**(segment_end, other_end, conserve_components: false)
  Remove all links of a segment end end except that to the other specified segment end.
- (RGFA) **delete_path**(path)
  Delete a path from the RGFA graph.
- (RGFA) **delete_segment**(segment, cascade = true)
  Delete a segment from the RGFA graph.
- (RGFA) **rm**(x, *args)
  Delete elements from the RGFA graph.
- (RGFA) **unconnect_segments**(segment1, segment2)
  Delete all links/containments involving two segments.

Instance Method Details

```
- (RGFA) delete_containment(from, from_orient, to, to_orient, pos)
- (RGFA) delete_containment(containment)
```

Delete a containment of a segment in another, with given orientations and starting position

Overloads:

- (RGFA) **delete_containment**(from, from_orient, to, to_orient, pos)

Parameters:

- **from** (String, RGFA::Line::Segment) — segment name or instance
- **from_orient** (nil, RGFA::Line::Segment::ORIENTATION) — orientation of from segment (use nil for both orientations)
- **to** (String, RGFA::Line::Segment) — segment name or instance
- **to_orient** (nil, RGFA::Line::Segment::ORIENTATION) — orientation of to segment (use nil for both orientations)
- **pos** (Integer, nil) — starting position (any if nil)

```
- (RGFA) delete_containment(containment)
```

Parameters:

- **containment** (RGFA::Line::Containment) — containment instance

Returns:

- (RGFA) — self

```
- (RGFA) delete_headers
```

Remove all headers

Returns:

- (RGFA) — self

```
- (RGFA) delete_link(from, from_orient, to, to_orient)
- (RGFA) delete_link(link)
```

Delete a link from a segment to another

Overloads:

```
- (RGFA) delete_link(from, from_orient, to, to_orient)
```

Parameters:

- **from** (String, RGFA::Line::Segment) — segment name or instance
- **from_orient** (nil, RGFA::Line::Segment::ORIENTATION) — orientation of from segment (use nil for both orientations)
- **to** (String, RGFA::Line::Segment) — segment name or instance
- **to_orient** (nil, RGFA::Line::Segment::ORIENTATION) — orientation of to segment (use nil for both orientations)

```
- (RGFA) delete_link(link)
```

Parameters:

- **link** (RGFA::Line::Link) — link instance

Returns:

- (RGFA) — self

```
- (RGFA) delete_other_links(segment_end, other_end, conserve_components: false)
```

Remove all links of a segment end end except that to the other specified segment end.

Parameters:

- **segment_end** (RGFA::SegmentEnd) — the segment end
- **other_end** (RGFA::SegmentEnd) — the other segment end
- **conserve_components** (Boolean) — *(defaults to: `false`)* Do not remove links if removing them breaks the graph into unconnected components.

Returns:

- (RGFA) — self

```
- (RGFA) delete_path(path)
```

Delete a path from the RGFA graph

Parameters:

- **path** (String, RGFA::Line::Path) — path name or instance

Returns:

- (RGFA) — self

Raises:

- (ArgumentError)

```
- (RGFA) delete_segment(segment, cascade = true)
```

Delete a segment from the RGFA graph

Parameters:

- **segment** (String, RGFA::Line::Segment) — segment name or instance

Returns:

- (RGFA) — self

```
- (RGFA) rm(segment)
- (RGFA) rm(path)
- (RGFA) rm(segment1, segment1_orient, segment2, segment2_orient)
- (RGFA) rm(:sequences)
- (RGFA) rm(:headers)
- (RGFA) rm(:alignments)
- (RGFA) rm(array)
- (RGFA) rm(method_name, *args)
```

Delete elements from the RGFA graph

Overloads:

```
- (RGFA) rm(segment)
```

Parameters:

- **segment** (String, RGFA::Line::Segment) — segment name or instance

```
- (RGFA) rm(path)
```

Parameters:

- **path** (String, RGFA::Line::Segment) — path name or instance

```
- (RGFA) rm(segment1, segment1_orient, segment2, segment2_orient)
```

Remove all links/containments where segment1 is the "From segment and segment2 is the "To" segment

Parameters:

- **segment1** (String, RGFA::Line::Segment) — segment 1 name or instance
- **segment1_orient** (RGFA::Line::Segment::ORIENTATION) — orientation of segment 1
- **segment2** (String, RGFA::Line::Segment) — segment 2 name or instance
- **segment2_orient** (RGFA::Line::Segment::ORIENTATION) — orientation of segment 2

- (RGFA) **rm**(:sequences)

Replace all sequences with "*"

- (RGFA) **rm**(:headers)

Remove all headers

- (RGFA) **rm**(:alignments)

Replace all CIGAR strings with "*"

- (RGFA) **rm**(array)

Calls `#rm` using each element of the array as argument

Parameters:

- **array** (`Array`)

- (RGFA) **rm**(method_name, *args)

Call a method of RGFA instance, then `#rm` for each returned value

Parameters:

- **method_name** (`Symbol`) — method to call
- **args** — arguments of the method

Returns:

- (`RGFA`) — self

- (RGFA) **unconnect_segments**(segment1, segment2)

Delete all links/containments involving two segments

Parameters:

- **segment1** (`String`, `RGFA::Line::Segment`) — segment 1 name or instance
- **segment2** (`String`, `RGFA::Line::Segment`) — segment 2 name or instance

Returns:

- (`RGFA`) — self

Module: RGFA::SegmentReferences

Included in:	Line::Containment, Line::Link
Defined in:	lib/rgfa/segment_references.rb

Overview

Methods common to links and containments in their references to segments

Instance Method Summary

(collapse)

- (Boolean) **circular?**
  Is the from segment of the link/containment the same as the to segment?.
- (String) **other**(segment)
  The other segment of a link/containment.

Instance Method Details

- (Boolean) **circular?**

Returns is the from segment of the link/containment the same as the to segment?

Returns:

- (Boolean) — is the from segment of the link/containment the same as the to segment?

- (String) **other**(segment)

The other segment of a link/containment

Parameters:

- **segment** (String, RGFA::Line::Segment) — segment name or instance

Returns:

- (String) — the name of the other segment of link/containment; if circular, then segment

Raises:

- (RGFA::LineMissingError) — if segment is not involved in link/containment

Class: RGFA

Inherits:	Object show all
Includes:	Edit, LineCreators, LineDestructors, LineGetters, LoggerSupport, RGL, Traverse
Defined in:	lib/rgfa.rb

Overview

This is the main class of the RGFA library. It provides a representation of the RGFA graph. Supports creating a graph from scratch, input and output from/to file or strings, as well as several operations on the graph.

Internals:

- The main structures are the @lines arrays, one for each record_type (e.g. header => @lines); these contain `RGFA::Line` objects of the corresponding subclass (e.g. `RGFA::Line::Header`)
- If an element is deleted, the position in @lines is set to `nil`, so that the links to all other positions still function
- The @segment_names and @path_names arrays contain the names of the segments and paths, in the same order as @lines and @lines; if a segment or path is added, its name is pushed on the @..._name array; if a segment or path is deleted, its position on the @..._name array is set to nil
- @c contains a RGFA::ConnectionInfo object, with hashes of indices of @lines which allow to directly find the links, containments and paths involving a given segment; @c is kept uptodate by the methods which allow to delete/rename or add links, containments or paths

Defined Under Namespace

Modules: CIGAR, Edit, FieldParser, FieldValidator, FieldWriter, LineCreators, LineDestructors, LineGetters, LoggerSupport, SegmentReferences, Sequence, Traverse
Classes: ByteArray, CigarOperation, ConnectionInfo, DuplicatedLabelError, Error, Line, LineMissingError, Logger, NumericArray, OrientedSegment, SegmentEnd, SegmentInfo

Class Method Summary (collapse)

`+ (RGFA) from_file(filename, validate: true)`
Creates a RGFA instance parsing the file with specified `filename`.

Instance Method Summary (collapse)

`- (Boolean) ==(other)`
Compares two RGFA instances.

`- (RGFA) clone`
Create a deep copy of the RGFA instance.

`- (String) info(short = false)`
Compact output has the following keys: - `ns`: number of segments - `nl`: number of links - `cc`: number of connected components - `de`: number of dead ends - `tl`: total length of segment sequences - `50`: N50 segment sequence length.

`- (RGFA) initialize` constructor

A new instance of RGFA.

- (Integer) **n_dead_ends**

  Counts the dead ends (i.e. segment ends without connections).

- (Array<String>) **path_names**

  List all names of path lines in the graph.

- (self) **read_file**(filename, validate: @validate)

  Populates a RGFA instance reading from file with specified `filename`.

- (void) **require_segments_first_order**

  Require that the links, containments and paths referring to a segment are added after the segment.

- (Array<String>) **segment_names**

  List all names of segments in the graph.

- (void) **to_file**(filename)

  Write RGFA to file with specified `filename`; overwrites it if it exists.

- (self) **to_rgfa**

  Return the gfa itself.

- (String) **to_s**

  Creates a string representation of RGFA conforming to the current specifications.

- (void) **turn_off_validations**

  Turns off validations.

- (void) **validate!**

  Post-validation of the RGFA; checks that L, C and P refer to existing S.

Methods included from LoggerSupport

#enable_progress_logging, #progress_log, #progress_log_end, #progress_log_init

Methods included from Traverse

#connected_components, #connectivity, #cut_link?, #cut_segment?, #linear_path, #linear_paths, #merge_linear_path, #merge_linear_paths, #segment_connected_component, #split_connected_components

Methods included from Edit

#delete_alignments, #delete_sequences, #multiply, #rename

Methods included from LineDestructors

#delete_containment, #delete_headers, #delete_link, #delete_other_links, #delete_path, #delete_segment, #rm, #unconnect_segments

Methods included from LineCreators

#<<, #set_header_field, #set_headers

Methods included from LineGetters

#connected_segments, #contained_in, #containing, #containment, #containment!, #containments, #containments_between, #each_containment, #each_header, #each_line, #each_link, #each_path, #each_segment, #headers, #headers_data, #link, #link!, #links, #links_between, #links_of, #neighbours, #other_segment_end, #path, #path!, #paths, #paths_with, #segment, #segment!, #segments

Constructor Details

- (RGFA) **initialize**

Returns a new instance of RGFA

Class Method Details

```
+ (RGFA) from_file(filename, validate: true)
```

Creates a RGFA instance parsing the file with specified `filename`

Parameters:

- **`validate`** (`Boolean`) — *(default: true)* calls #validate! after construction of the graph (note: setting it to false does not deactivate all validations; for this use #turn_off_validations)
- **`filename`** (`String`)

Returns:

- (`RGFA`)

Raises:

- if file cannot be opened for reading

Instance Method Details

```
- (Boolean) ==(other)
```

Compares two RGFA instances

Returns:

- (`Boolean`) — are the lines of the two instances equivalent?

```
- (RGFA) clone
```

Create a deep copy of the RGFA instance.

Returns:

- (`RGFA`)

```
- (String) info(short = false)
```

Compact output has the following keys:

- `ns`: number of segments
- `nl`: number of links
- `cc`: number of connected components
- `de`: number of dead ends
- `tl`: total length of segment sequences
- `50`: N50 segment sequence length

Normal output outputs a table with the same information, plus the largest component, the shortest and largest and 1st/2nd/3rd quartiles of segment sequence length.

Parameters:

- **`short`** (`boolean`) *(defaults to: `false`)* — compact output as a single text line

Returns:

- (`String`) — sequence and topology information collected from the graph.

- (Integer) **n_dead_ends**

Counts the dead ends (i.e. segment ends without connections)

Returns:

- (Integer) — number of dead ends in the graph

- (Array<String>) **path_names**

List all names of path lines in the graph

Returns:

- (Array<String>)

- (self) **read_file**(filename, validate: @validate)

Populates a RGFA instance reading from file with specified `filename`

Parameters:

- **validate** (boolean) — *(default: `true` if `#turn_off_validations` was never called, `false` otherwise)* calls #validate! after construction of the graph (note: setting it to false does not deactivate all validations; for this use #turn_off_validations)
- **filename** (String)

Returns:

- (self)

Raises:

- if file cannot be opened for reading

- (void) **require_segments_first_order**

This method returns an undefined value.

Require that the links, containments and paths referring to a segment are added after the segment. Default: do not require any particular ordering.

- (Array<String>) **segment_names**

List all names of segments in the graph

Returns:

- (Array<String>)

- (void) **to_file**(filename)

This method returns an undefined value.

Write RGFA to file with specified `filename`; overwrites it if it exists

Parameters:

- **`filename`** (`String`)

Raises:

- if file cannot be opened for writing

```
- (self) to_rgfa
```

Return the gfa itself

Returns:

- (`self`)

```
- (String) to_s
```

Creates a string representation of RGFA conforming to the current specifications

Returns:

- (`String`)

```
- (void) turn_off_validations
```

This method returns an undefined value.

Turns off validations. This increases the performance.

```
- (void) validate!
```

This method returns an undefined value.

Post-validation of the RGFA; checks that L, C and P refer to existing S.

Raises:

- if validation fails

Class: String

Inherits:	Object show all
Includes:	RGFA::CIGAR, RGFA::FieldParser, RGFA::FieldValidator, RGFA::Sequence
Defined in:	lib/rgfa.rb

Overview

Extensions to the String core class.

Constant Summary

Constant Summary

Constants included from RGFA::FieldValidator

RGFA::FieldValidator::DATASTRING_VALIDATION_REGEXP

Constants included from RGFA::Sequence

RGFA::Sequence::WCC

Instance Method Summary (collapse)

- (RGFA::ByteArray) **to_byte_array**

  Convert a GFA string representation of a byte array to a byte array.

- (RGFA::NumericArray) **to_numeric_array**(validate: true)

  Create a numeric array from a string.

- (RGFA) **to_rgfa**(validate: true)

  Converts a `String` into a `RGFA` instance.

- (subclass of RGFA::Line) **to_rgfa_line**(validate: true)

  Parses a line of a RGFA file and creates an object of the correct record type child class of `RGFA::Line`.

Methods included from RGFA::FieldValidator

#validate_datastring

Methods included from RGFA::FieldParser

#parse_datastring, #parse_optfield

Methods included from RGFA::Sequence

#rc

Methods included from RGFA::CIGAR

#cigar_operations, #reverse_cigar, #reverse_cigar_operations

Instance Method Details

- (RGFA::ByteArray) **to_byte_array**

Convert a GFA string representation of a byte array to a byte array

Returns:

- (`RGFA::ByteArray`) — the byte array

```
- (RGFA::NumericArray) to_numeric_array(validate: true)
```

Create a numeric array from a string

Parameters:

- **`validate`** (`Boolean`) — *(default: `true`)* if `true`, validate the range of the numeric values, according to the array subtype

Returns:

- (`RGFA::NumericArray`) — the numeric array

Raises:

- (`RGFA::NumericArray::ValueError`) — if validate is set and any value is not compatible with the subtype
- (`RGFA::NumericArray::TypeError`) — if the subtype code is invalid

```
- (RGFA) to_rgfa(validate: true)
```

Converts a `String` into a `RGFA` instance. Each line of the string is added separately to the gfa.

Parameters:

- **`validate`** (`Boolean`) — *(defaults to: `true`)* #validate! after construction of the graph (note: setting it to false does not deactivate all validations; for this use #turn_off_validations)

Returns:

- (`RGFA`)

```
- (subclass of RGFA::Line) to_rgfa_line(validate: true)
```

Parses a line of a RGFA file and creates an object of the correct

```
record type child class of {RGFA::Line}
```

Parameters:

- **`validate`** (`Boolean`) — *(defaults to: `true`)* if false, turn off validations

Returns:

- (`subclass of RGFA::Line`)

Raises:

- if the string does not comply to the RGFA specification

Class: Array

Inherits:	Object show all
Defined in:	lib/rgfa.rb

Overview

Extensions to the Array core class.

Direct Known Subclasses

RGFA::ByteArray, RGFA::CigarOperation, RGFA::NumericArray, RGFA::SegmentInfo

Instance Method Summary

(collapse)

- (Object) **gfa_datatype**
  !macro gfa_datatype.
- (RGFA::ByteArray) **to_byte_array**
  Create a RGFA::ByteArray from an Array instance.
- (RGFA::CigarOperation) **to_cigar_operation**
- (String) **to_gfa_datastring**(datatype)
  Representation of the data for GFA fields; this method does not automatically validate the string.
- (RGFA::NumericArray) **to_numeric_array**(validate: true)
  Create a numeric array from an Array instance.
- (RGFA::OrientedSegment) **to_oriented_segment**
  Create and validate a segment end from an array.
- (RGFA) **to_rgfa**(validate: true)
  Converts an `Array` of strings or RGFA::Line instances into a `RGFA` instance.
- (subclass of RGFA::Line) **to_rgfa_line**(validate: true)
  Parses an array containing the fields of a RGFA file line and creates an object of the correct record type child class of `RGFA::Line`.
- (RGFA::SegmentEnd) **to_segment_end**
  Create and validate a segment end from an array.

Instance Method Details

- (Object) **gfa_datatype**

!macro gfa_datatype

- (RGFA::ByteArray) **to_byte_array**

Create a RGFA::ByteArray from an Array instance

Returns:

- (`RGFA::ByteArray`) — the byte array

- (RGFA::CigarOperation) **to_cigar_operation**

Returns:

- (RGFA::CigarOperation)

- (String) **to_gfa_datastring**(datatype)

Representation of the data for GFA fields; this method does not automatically validate the string.

Parameters:

- **datatype** (RGFA::Line::FIELD_DATATYPE)

Returns:

- (String)

- (RGFA::NumericArray) **to_numeric_array**(validate: true)

Create a numeric array from an Array instance

Parameters:

- **validate** (Boolean) — *(default: true)* if true, validate the range of the numeric values, according to the array subtype

Returns:

- (RGFA::NumericArray) — the numeric array

Raises:

- (RGFA::NumericArray::ValueError) — if validate is set and any value is not compatible with the subtype

---

- (RGFA::OrientedSegment) **to_oriented_segment**

Create and validate a segment end from an array

Returns:

- (RGFA::OrientedSegment)

Raises:

- (RGFA::SegmentInfo::InvalidSizeError) — if size is not 2
- (RGFA::SegmentInfo::InvalidAttributeError) — if second element is not a valid info

---

- (RGFA) **to_rgfa**(validate: true)

Converts an Array of strings or RGFA::Line instances into a RGFA instance.

Parameters:

- **validate** (Boolean) — *(defaults to: true)* #validate! after construction of the graph (note: setting it to false does not deactivate all validations; for this use #turn_off_validations)

Returns:

- (RGFA)

- (subclass of RGFA::Line) **to_rgfa_line**(validate: true)

Note: This method modifies the content of the array; if you still need the array, you must create a copy before calling it

Parses an array containing the fields of a RGFA file line and creates an object of the correct record type child class of RGFA::Line

Parameters:

- **validate** (Boolean) — *(defaults to: true)* if false, turn off validations

Returns:

- (subclass of RGFA::Line)

Raises:

- if the fields do not comply to the RGFA specification

- (RGFA::SegmentEnd) **to_segment_end**

Create and validate a segment end from an array

Returns:

- (RGFA::SegmentEnd)

Raises:

- (RGFA::SegmentInfo::InvalidSizeError) — if size is not 2
- (RGFA::SegmentInfo::InvalidAttributeError) — if second element is not a valid info

Class: RGFA::Line

Inherits:	Object show all
Defined in:	lib/rgfa/line.rb

Overview

Note: This class is usually not meant to be directly initialized by the user; initialize instead one of its child classes, which define the concrete different record types.

Generic representation of a record of a RGFA file.

Direct Known Subclasses

Containment, Header, Link, Path, Segment

Defined Under Namespace

Classes: Containment, CustomOptfieldNameError, DuplicatedOptfieldNameError, FieldnameError, Header, Link, Path, PredefinedOptfieldTypeError, RequiredFieldMissingError, Segment, TagMissingError, UnknownDatatype, UnknownRecordTypeError

Constant Summary

SEPARATOR =
Separator in the string representation of RGFA lines

```
"\t"
```

RECORD_TYPES =
List of allowed record_type values

```
[ :H, :S, :L, :C, :P ]
```

RECORD_TYPE_LABELS =
Full name of the record types

```
{
  :H => "header",
  :S => "segment",
  :L => "link",
  :C => "containment",
  :P => "path",
}
```

OPTFIELD_DATATYPE =
A symbol representing a datatype for optional fields

```
[:A, :i, :f, :Z, :J, :H, :B]
```

REQFIELD_DATATYPE =
A symbol representing a datatype for required fields

```
[:lbl, :orn, :lbs, :seq, :pos, :cig, :cgs]
```

FIELD_DATATYPE =
A symbol representing a valid datatype

```
OPTFIELD_DATATYPE + REQFIELD_DATATYPE
```

DELAYED_PARSING_DATATYPES =
data types which are parsed only on access

```
[:cig, :cgs, :lbs, :H, :J, :B]
```

Class Method Summary

(collapse)

+ (Class) **subclass**(record_type)
Select a subclass based on the record type.

Instance Method Summary

(collapse)

- (Boolean) **==**(o)
Equivalence check.

- (RGFA::Line) **clone**
Deep copy of self (RGFA::Line subclass).

- (Object?) **delete**(fieldname)
Remove an optional field from the line, if it exists; do nothing if it does not.

- (Array<Symbol>) **fieldnames**
Fields defined for this instance.

- (Object?) **get**(fieldname)
Value of a field.

- (Object?) **get!**(fieldname)
Value of a field, raising an exception if it is not defined.

- (Symbol?) **get_datatype**(fieldname)
Returns a symbol, which specifies the datatype of a field.

- (String) **get_string**(fieldname, validate: true, optfield: false)
Returns the string representation of the content of a field.

- (String) **get_string!**(fieldname, validate: true, optfield: false)
Returns the string representation of the content of a field.

- (RGFA::Line) **initialize**(data, validate: true) constructor
Constants defined by subclasses .

- (Object) **method_missing**(m, *args, &block)
Three methods are dynamically created for each existing field name as well as for each non-existing but valid optional field name.

- (Array<Symbol>) **optional_fieldnames**
Name of the optional fields.

- (Symbol) **record_type**
Record type code.

- (Array<Symbol>) **required_fieldnames**
Name of the required fields.

- (Boolean) **respond_to?**(m, include_all = false)
Redefines respond_to? to correctly handle dynamical methods.

- (void) **set**(fieldname, value)
Set the value of a field.

- (Object) **set_datatype**(fieldname, datatype)

  Set or change the datatype of a custom optional field.

- (Array<String>) **to_a**(validate: true)

  An array of string representations of the fields.

- (Object) **to_rgfa_line**(validate: true)

  Self.

- (String) **to_s**(validate: true)

  A string representation of self.

- (void) **validate!**

  Validate the RGFA::Line instance.

- (nil) **validate_field!**(fieldname)

  Raises an error if the content of the field does not correspond to the field type.

Constructor Details

- (RGFA::Line) **initialize**(data, validate: true)

Note: This class is usually not meant to be directly initialized by the user; initialize instead one of its child classes, which define the concrete different record types.

Constants defined by subclasses

Subclasses of RGFA::Line *must* define the following constants:

- RECORD_TYPE [RGFA::Line::RECORD_TYPES]
- REQFIELDS [Array<Symbol>] required fields
- PREDEFINED_OPTFIELDS [Array<Symbol>] predefined optional fields
- DATATYPE [HashSymbol=>Symbol]: datatypes for the required fields and the predefined optional fields

Parameters:

- **data** (Array<String>) — the content of the line; the array content is not preserved by this method (the array will be empty after calling this method)
- **validate** (Boolean) — *(default: true)* validate the content of the fields using the regular expressions of the GFA specification

Raises:

- (RGFA::Line::RequiredFieldMissingError) — if too less required fields are specified
- (RGFA::Line::CustomOptfieldNameError) — if a non-predefined optional field uses upcase letters
- (RGFA::Line::DuplicatedOptfieldNameError) — if an optional field tag name is used more than once
- (RGFA::Line::PredefinedOptfieldTypeError) — if the type of a predefined optional field does not respect the specified type.

Dynamic Method Handling

This class handles dynamic methods through the method_missing method

- (Object) **method_missing**(m, *args, &block)

Three methods are dynamically created for each existing field name as well as for each non-existing but valid optional field name.

```
- (Object) <fieldname>(parse=true)
```

The value of a field.

Parameters:

- **parse** (Boolean) *(default: true)* parse the string and return the corresponding ruby object if `true`; return the GFA string representation of the data if `false`

Returns:

- (String, Hash, Array, Integer, Float) if the field exists
- (nil) if the field does not exist, but is a valid optional field name

---

```
- (Object) <fieldname>!(parse=true)
```

The valid of a field, raising an exception if not available.

Parameters:

- **parse** (Boolean) *(default: true)* parse the string and return the corresponding ruby object if `true`; return the GFA string representation of the data if `false`

Returns:

- (String, Hash, Array, Integer, Float) if the field exists

Raises:

- (RGFA::Line::TagMissingError) if the field does not exist

```
- (self) <fieldname>=(value)
```

Sets the value of a required or optional field, or creates a new optional field if the fieldname is non-existing but valid. No validation is performed by this method. See also #set, #validate_field!, #set_datatype.

Parameters:

- **value** (String|Hash|Array|Integer|Float) value to set

Class Method Details

```
+ (Class) subclass(record_type)
```

Select a subclass based on the record type

Returns:

- (`Class`) — a subclass of RGFA::Line

Raises:

- (`RGFA::Line::UnknownRecordTypeError`) — if the record_type is not valid

Instance Method Details

```
- (Boolean) ==(o)
```

Equivalence check

Returns:

- (Boolean) — does the line has the same record type, contains the same optional fields and all required and optional fields contain the same field values?

See Also:

- RGFA::Line::Link#==

```
- (RGFA::Line) clone
```

Returns deep copy of self (RGFA::Line subclass)

Returns:

- (RGFA::Line) — deep copy of self (RGFA::Line subclass)

```
- (Object?) delete(fieldname)
```

Remove an optional field from the line, if it exists;

```
do nothing if it does not
```

Parameters:

- **fieldname** (#to_sym) — the tag name of the optfield to remove

Returns:

- (Object, nil) — the deleted value or nil, if the field was not defined

---

```
- (Array<Symbol>) fieldnames
```

Returns fields defined for this instance

Returns:

- (Array<Symbol>) — fields defined for this instance

```
- (Object?) get(fieldname)
```

Value of a field

Parameters:

- **fieldname** (#to_sym) — name of the field

Returns:

- (Object, nil) — value of the field or nil if field is not defined

---

```
- (Object?) get!(fieldname)
```

Value of a field, raising an exception if it is not defined

Parameters:

- **fieldname** (#to_sym) — name of the field

Returns:

- (Object, nil) — value of the field

Raises:

- (RGFA::Line::TagMissingError) — if field is not defined

```
- (Symbol?) get_datatype(fieldname)
```

Returns a symbol, which specifies the datatype of a field

Returns:

- (Symbol, nil) — the datatype symbol or nil if the field does not exist and/or the datatype is not (yet) defined

---

```
- (String) get_string(fieldname, validate: true, optfield: false)
```

Returns the string representation of the content of a field. The datatype is either predefined (required fields, optional fields), manually set (see #set_datatype) or automatically computed.

Parameters:

- **validate** (Boolean) — *(defaults to: true)* perform a validation of the string, using the regular expression for the datatype
- **optfield** (Boolean) — *(defaults to: false)* return the tagname:datatype:datastring representation

Returns:

- (String) — the string representation (an empty string if the field does not exist)

```
- (String) get_string!(fieldname, validate: true, optfield: false)
```

Returns the string representation of the content of a field. The datatype is either predefined (required fields, optional fields), manually set (see #set_datatype) or automatically computed.

Parameters:

- **validate** (Boolean) — *(defaults to: true)* perform a validation of the string, using the regular expression for the datatype
- **optfield** (Boolean) — *(defaults to: false)* return the tagname:datatype:datastring representation

Returns:

- (String) — the string representation

Raises:

- (RGFA::Line::TagMissingError) — if field is not defined

```
- (Array<Symbol>) optional_fieldnames
```

Returns name of the optional fields

Returns:

- (Array<Symbol>) — name of the optional fields

```
- (Symbol) record_type
```

Returns record type code

Returns:

- (Symbol) — record type code

- (Array<Symbol>) **required_fieldnames**

Returns name of the required fields

Returns:

- (Array<Symbol>) — name of the required fields

- (Boolean) **respond_to?**(m, include_all = false)

Redefines respond_to? to correctly handle dynamical methods.

Returns:

- (Boolean)

See Also:

- #method_missing

- (void) **set**(fieldname, value)

This method returns an undefined value.

Set the value of a field. The field name must be a required field, a predefined optional field name (uppercase) or custom optional field name (lowercase). The field content is not validated by this method. To validate the content, use #validate_field!. Automatic validation is performed, when the fields are read from string or written to string. To explicitely set the datatype of a new custom optional fields use #set_datatype, otherwise the type will be automatically selected from the value.

Parameters:

- **fieldname** (#to_sym) — the name of the field to set

- (Object) **set_datatype**(fieldname, datatype)

Set or change the datatype of a custom optional field

Parameters:

- **fieldname** (#to_sym) — the field name
- **datatype** (#to_sym) — the datatype

Raises:

- (RGFA::Line::CustomOptfieldNameError) — if the field name is not a valid custom optional name
- (RGFA::Line::UnknownDatatype) — if datatype is not a valid datatype for optional fields

---

- (Array<String>) **to_a**(validate: true)

Returns an array of string representations of the fields

Returns:

- (Array<String>) — an array of string representations of the fields

---

```
- (Object) to_rgfa_line(validate: true)
```

Returns self

Parameters:

- **validate** (Boolean) — ignored (compatibility reasons)

Returns:

- self

```
- (String) to_s(validate: true)
```

Returns a string representation of self

Returns:

- (String) — a string representation of self

```
- (void) validate!
```

This method returns an undefined value.

Validate the RGFA::Line instance

Raises:

- (RGFA::FieldParser::FormatError) — if any field content is not valid

---

```
- (nil) validate_field!(fieldname)
```

Raises an error if the content of the field does not correspond to the field type

Parameters:

- **fieldname** (#to_sym) — the tag name of the field to validate

Returns:

- (nil)

Raises:

- (RGFA::FieldParser::FormatError) — if the content of the field is not valid, according to its required type

Exception: RGFA::CIGAR::ValueError

Inherits:	Error show all
Defined in:	lib/rgfa/cigar.rb

Overview

Exception raised by invalid cigar string content

Class: RGFA::CigarOperation

Inherits:	Array show all
Defined in:	lib/rgfa/cigar.rb

Overview

Class representing a CIGAR operation

Instance Method Summary (collapse)

- (String) **opcode**
  The operation code.
- (Integer) **oplen**
  The operation length.
- (String) **to_s**
  The string representation of the operation.

Methods inherited from Array

#gfa_datatype, #to_byte_array, #to_cigar_operation, #to_gfa_datastring, #to_numeric_array, #to_oriented_segment, #to_rgfa, #to_rgfa_line, #to_segment_end

Instance Method Details

- (String) opcode

The operation code

Returns:

- (String) — *(length: 1)* operation code

- (Integer) oplen

The operation length

Returns:

- (Integer) — operation length

- (String) to_s

The string representation of the operation

Returns:

- (String)

Exception: RGFA::Error

Inherits:	StandardError show all
Defined in:	lib/rgfa/error.rb

Overview

Parent class for library-specific errors

Direct Known Subclasses

ByteArray::ValueError, CIGAR::ValueError, ConnectionInfo::ValidationError, DuplicatedLabelError, FieldParser::FormatError, FieldParser::UnknownDatatypeError, Line::CustomOptfieldNameError, Line::DuplicatedOptfieldNameError, Line::FieldnameError, Line::PredefinedOptfieldTypeError, Line::RequiredFieldMissingError, Line::Segment::InconsistentLengthError, Line::Segment::UndefinedLengthError, Line::TagMissingError, Line::UnknownDatatype, Line::UnknownRecordTypeError, LineMissingError, NumericArray::TypeError, NumericArray::ValueError, SegmentInfo::InvalidAttributeError, SegmentInfo::InvalidSizeError

Class: RGFA::Logger

Inherits:	Object show all
Defined in:	lib/rgfa/logger.rb

Overview

This class allows to output a message to the log file or STDERR and to keep track of the progress of a method which takes long time to complete.

Defined Under Namespace

Classes: ProgressData

Instance Method Summary (collapse)

- (void) **disable_progress**
  Disable progress logging.
- (void) **enable_progress**(part: 0.1)
  Enable output from the Logger instance.
- (RGFA::Logger) **initialize**(verbose_level: 1, channel: STDERR, prefix: "#")
  constructor
  Create a Logger instance.
- (void) **log**(msg, min_verbose_level = 1)
  Output a message.
- (void) **progress_end**(symbol, **keyargs)
  Completes progress logging for a computation.
- (void) **progress_init**(symbol, units, total, initmsg = nil)
  Initialize progress logging for a computation.
- (void) **progress_log**(symbol, progress = 1, **keyargs)
  Updates progress logging for a computation.

Constructor Details

- (RGFA::Logger) **initialize**(verbose_level: 1, channel: STDERR, prefix: "#")

Create a Logger instance

Parameters:

- **channel** (#puts) — where to output (default: STDERR)
- **prefix** (String) — output prefix (default: "#")
- **verbose_level** (Integer) — 0: no logging; >0: the higher, the more logging

Instance Method Details

- (void) **disable_progress**

This method returns an undefined value.

Disable progress logging

```
- (void) enable_progress(part: 0.1)
```

This method returns an undefined value.

Enable output from the Logger instance

Parameters:

- **part** (Float) —
  - part = 0 => output at every call of progress_log
  - 0 < part < 1 => output once per part of the total progress

    (e.g. 0.001 = log every 0.1% progress)
  - part = 1 => output only total elapsed time

```
- (void) log(msg, min_verbose_level = 1)
```

This method returns an undefined value.

Output a message

Parameters:

- **msg** (String) — message to output
- **min_verbose_level** (Integer) *(defaults to: 1)*

```
- (void) progress_end(symbol, **keyargs)
```

This method returns an undefined value.

Completes progress logging for a computation

Parameters:

- **symbol** (Symbol) — the symbol assigned to the computation at init time
- **keyargs** (Hash) — additional units to display, with their current value (e.g. segments_processed: 10000)

```
- (void) progress_init(symbol, units, total, initmsg = nil)
```

This method returns an undefined value.

Initialize progress logging for a computation

Parameters:

- **symbol** (Symbol) — a symbol assigned to the computation
- **units** (String) — a string with the name of the units, in plural
- **total** (Integer) — total number of units
- **initmsg** (String) *(defaults to: nil)* — an optional message to output at the beginning

```
- (void) progress_log(symbol, progress = 1, **keyargs)
```

This method returns an undefined value.

Updates progress logging for a computation

Parameters:

- **`symbol`** (`Symbol`) — the symbol assigned to the computation at init time
- **`keyargs`** (`Hash`) — additional units to display, with their current value (e.g. segments_processed: 10000)
- **`progress`** (`Integer`) *(defaults to: `1`)* — how many units were processed

Class: RGFA::Logger::ProgressData

Inherits:	Struct show all
Defined in:	lib/rgfa/logger.rb

Overview

Information about the progress of a computation

Instance Attribute Summary (collapse)

- (Object) **counter**
  Returns the value of attribute counter.
- (Object) **lastpart**
  Returns the value of attribute lastpart.
- (Object) **partsize**
  Returns the value of attribute partsize.
- (Object) **starttime**
  Returns the value of attribute starttime.
- (Object) **strlen**
  Returns the value of attribute strlen.
- (Object) **total**
  Returns the value of attribute total.
- (Object) **units**
  Returns the value of attribute units.

Instance Attribute Details

- (Object) **counter**

Returns the value of attribute counter

Returns:

- (Object) — the current value of counter

- (Object) **lastpart**

Returns the value of attribute lastpart

Returns:

- (Object) — the current value of lastpart

- (Object) **partsize**

Returns the value of attribute partsize

Returns:

- (Object) — the current value of partsize

```
- (Object) starttime
```

Returns the value of attribute starttime

Returns:

- (Object) — the current value of starttime

```
- (Object) strlen
```

Returns the value of attribute strlen

Returns:

- (Object) — the current value of strlen

```
- (Object) total
```

Returns the value of attribute total

Returns:

- (Object) — the current value of total

```
- (Object) units
```

Returns the value of attribute units

Returns:

- (Object) — the current value of units

Class: RGFA::Line::Path

Inherits:	RGFA::Line show all
Defined in:	lib/rgfa/line/path.rb

Overview

A path line of a RGFA file

Constant Summary

RECORD_TYPE =

```
:P
```

REQFIELDS =

```
[:path_name, :segment_names, :cigars]
```

PREDEFINED_OPTFIELDS =

```
[]
```

DATATYPE =

```
{
  :path_name => :lbl,
  :segment_names => :lbs,
  :cigars => :cgs,
}
```

Constants inherited from RGFA::Line

DELAYED_PARSING_DATATYPES, FIELD_DATATYPE, OPTFIELD_DATATYPE, RECORD_TYPES, RECORD_TYPE_LABELS, REQFIELD_DATATYPE, SEPARATOR

Instance Method Summary (collapse)

- (Symbol) **to_sym**
  Name of the path as symbol.

Methods inherited from RGFA::Line

#==, #clone, #delete, #fieldnames, #get, #get!, #get_datatype, #get_string, #get_string!, #initialize, #method_missing, #optional_fieldnames, #record_type, #required_fieldnames, #respond_to?, #set, #set_datatype, subclass, #to_a, #to_rgfa_line, #to_s, #validate!, #validate_field!

Constructor Details

This class inherits a constructor from RGFA::Line

Dynamic Method Handling

This class handles dynamic methods through the `method_missing` method in the class RGFA::Line

Instance Method Details

```
- (Symbol) to_sym
```

Returns name of the path as symbol

Returns:

- (`Symbol`) — name of the path as symbol

Class: RGFA::Line::Link

Inherits:	RGFA::Line show all
Includes:	SegmentReferences
Defined in:	lib/rgfa/line/link.rb

Overview

A link line of a RGFA file

Constant Summary

RECORD_TYPE =

```
:L
```

REQFIELDS =

```
[:from, :from_orient, :to, :to_orient, :overlap]
```

PREDEFINED_OPTFIELDS =

```
[:MQ, :NM, :RC, :FC, :KC]
```

DATATYPE =

```
{
   :from => :lbl,
   :from_orient => :orn,
   :to => :lbl,
   :to_orient => :orn,
   :overlap => :cig,
   :MQ => :i,
   :NM => :i,
   :RC => :i,
   :FC => :i,
   :KC => :i,
}
```

Constants inherited from RGFA::Line

DELAYED_PARSING_DATATYPES, FIELD_DATATYPE, OPTFIELD_DATATYPE, RECORD_TYPES, RECORD_TYPE_LABELS, REQFIELD_DATATYPE, SEPARATOR

Instance Method Summary

(collapse)

- (Boolean) **==**(other)
  Compares two links and determine their equivalence.
- (Boolean) **eql?**(other)
  Compares two links and determine their equivalence.
- (Boolean) **eql_optional?**(other)
  Compares the optional fields of two links.
- (Object) **from_end**
  @return the segment end represented by the from/from_orient fields.
- (Object) **hash**

Computes an hash for including a link in an Hash tables, so that the hash of a link and its reverse is the same.

- (Object) **oriented_from**

  @return the oriented segment represented by the from/from_orient fields.

- (Object) **oriented_to**

  @return the oriented segment represented by the to/to_orient fields.

- (Object) **other_end**(segment_end)

  @param segment_end one of the two segment ends of the link @return the other segment end.

- (Object) **reverse**

  Creates a link with both strands of the sequences inverted.

- (String, Array<RGFA::CigarOperation>) **reverse_overlap**(cast = true)

  Compute the overlap when the strand of both sequences is inverted.

- (Object) **to_end**

  @return the segment end represented by the to/to_orient fields.

Methods included from SegmentReferences

#circular?, #other

Methods inherited from RGFA::Line

#clone, #delete, #fieldnames, #get, #get!, #get_datatype, #get_string, #get_string!, #initialize, #method_missing, #optional_fieldnames, #record_type, #required_fieldnames, #respond_to?, #set, #set_datatype, subclass, #to_a, #to_rgfa_line, #to_s, #validate!, #validate_field!

Constructor Details

This class inherits a constructor from RGFA::Line

Dynamic Method Handling

This class handles dynamic methods through the `method_missing` method in the class RGFA::Line

Instance Method Details

- (Boolean) **==**(other)

Note: Inverting the strand of both links and reversing the CIGAR operations (order/type), one obtains an equivalent link.

Compares two links and determine their equivalence. Optional fields must have the same content.

Parameters:

- **other** (RGFA::Line::Link) — a link

Returns:

- (Boolean) — are self and other equivalent?

See Also:

- #eql?
- #eql_optional?

- (Boolean) **eql?**(other)

Note: Inverting the strand of both links and reversing the CIGAR operations (order/type), one obtains a reverse but equivalent link.

Compares two links and determine their equivalence. Thereby, optional fields are not considered.

Parameters:

- **other** (RGFA::Line::Link) — a link

Returns:

- (Boolean) — are self and other equivalent?

See Also:

- #==

- (Boolean) **eql_optional?**(other)

Note: This method shall be overridden if custom optional fields are defined, which have a "reverse" operation which determines their value in the equivalent but reverse link.

Compares the optional fields of two links.

Parameters:

- **other** (RGFA::Line::Link) — a link

Returns:

- (Boolean) — are self and other equivalent?

See Also:

- #==

- (Object) **from_end**

@return the segment end represented by the

```
from/from_orient fields
```

- (Object) **hash**

Computes an hash for including a link in an Hash tables, so that the hash of a link and its reverse is the same. Thereby, optional fields are not considered.

See Also:

- #eql?

- (Object) **oriented_from**

@return the oriented segment represented by the

```
from/from_orient fields
```

- (Object) **oriented_to**

@return the oriented segment represented by the

```
to/to_orient fields
```

- (Object) **other_end**(segment_end)

@param segment_end one of the two segment ends

```
of the link
```

@return the other segment end

Raises:

- (ArgumentError) — if segment_end is not a valid segment end representation
- (RuntimeError) — if segment_end is not a segment end of the link

- (Object) **reverse**

Note: This method shall be overridden if custom optional fields are defined, which have a "reverse" operation which determines their value in the equivalent but reverse link.

Creates a link with both strands of the sequences inverted. The CIGAR operations (order/type) are inverted as well. Optional fields are left unchanged.

@return the inverted link.

- (String, Array<RGFA::CigarOperation>) **reverse_overlap**(cast = true)

Compute the overlap when the strand of both sequences is inverted.

Parameters:

- **cast** (Boolean) *(defaults to: true)* — cast value?

Returns:

- (String) — if cast is false
- (Array<RGFA::CigarOperation>) — if cast is true

- (Object) **to_end**

@return the segment end represented by the

```
to/to_orient fields
```

Class: RGFA::ByteArray

Inherits:	Array show all
Defined in:	lib/rgfa/byte_array.rb

Overview

Support of the conversion to GFA datastrings of type H

Defined Under Namespace

Classes: ValueError

Instance Method Summary (collapse)

- (Object) **gfa_datatype**
  !macro gfa_datatype.
- (RGFA::ByteArray) **to_byte_array**
  Returns self.
- (String) **to_s**
  GFA datatype H representation of the byte array.
- (void) **validate!**
  Validates the byte array content.

Methods inherited from Array

#to_cigar_operation, #to_gfa_datastring, #to_numeric_array, #to_oriented_segment, #to_rgfa, #to_rgfa_line, #to_segment_end

Instance Method Details

- (Object) gfa_datatype

!macro gfa_datatype

- (RGFA::ByteArray) to_byte_array

Returns self

Returns:

- (RGFA::ByteArray) — self

- (String) to_s

GFA datatype H representation of the byte array

Returns:

- (String)

Raises:

- (RGFA::ByteArray::ValueError) — if the array is not a valid byte array

---

```
- (void) validate!
```

This method returns an undefined value.

Validates the byte array content

Raises:

- (RGFA::ByteArray::ValueError) — if any value is not a positive integer <= 255

Exception: RGFA::ByteArray::ValueError

Inherits:	Error show all
Defined in:	lib/rgfa/byte_array.rb

Overview

Exception raised if any value is not a positive integer <= 255

Class: RGFA::Line::Header

Inherits:	RGFA::Line show all
Defined in:	lib/rgfa/line/header.rb

Overview

A header line of a RGFA file

Constant Summary

RECORD_TYPE =

```
:H
```

REQFIELDS =

```
[]
```

PREDEFINED_OPTFIELDS =

```
[:VN]
```

DATATYPE =

```
{
  :VN => :Z
}
```

Constants inherited from RGFA::Line

DELAYED_PARSING_DATATYPES, FIELD_DATATYPE, OPTFIELD_DATATYPE, RECORD_TYPES, RECORD_TYPE_LABELS, REQFIELD_DATATYPE, SEPARATOR

Method Summary

Methods inherited from RGFA::Line

#==, #clone, #delete, #fieldnames, #get, #get!, #get_datatype, #get_string, #get_string!, #initialize, #method_missing, #optional_fieldnames, #record_type, #required_fieldnames, #respond_to?, #set, #set_datatype, subclass, #to_a, #to_rgfa_line, #to_s, #validate!, #validate_field!

Constructor Details

This class inherits a constructor from RGFA::Line

Dynamic Method Handling

This class handles dynamic methods through the `method_missing` method in the class RGFA::Line

Exception: RGFA::FieldParser::FormatError

Inherits:	Error show all
Defined in:	lib/rgfa/field_parser.rb

Overview

Error raised if the field content has an invalid format

Exception: RGFA::FieldParser::UnknownDatatypeError

Inherits:	Error show all
Defined in:	lib/rgfa/field_parser.rb

Overview

Error raised if an unknown datatype symbol is used

Class: RGFA::SegmentInfo Private

Inherits:	Array show all
Defined in:	lib/rgfa/segment_info.rb

Overview

This class is part of a private API. You should avoid using this class if possible, as it may be removed or be changed in the future.

A segment or segment name plus an additional boolean attribute

This class shall not be initialized directly.

Direct Known Subclasses

OrientedSegment, SegmentEnd

Defined Under Namespace

Classes: InvalidAttributeError, InvalidSizeError

Class Method Summary (collapse)

+ (Object) **other**(attribute) private
The other attribute value.

Instance Method Summary (collapse)

- (Boolean) **==**(other) private
Compare the segment names and attributes of two instances.

- (Object) **attribute** private

- (Symbol) **name** private
The segment name.

- (RGFA::SegmentInfo) **other** private
Same segment, inverted attribute.

- (Symbol, RGFA::Line::Segment) **segment** private
The segment instance or name.

- (String) **to_s** private
Name of the segment and attribute.

- (Symbol) **to_sym** private
Name of the segment and attribute.

- (void) **validate!** private
Check that the elements of the array are compatible with the definition.

Methods inherited from Array

#gfa_datatype, #to_byte_array, #to_cigar_operation, #to_gfa_datastring,

#to_numeric_array, #to_oriented_segment, #to_rgfa, #to_rgfa_line, #to_segment_end

Class Method Details

```
+ (Object) other(attribute)
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Returns the other attribute value

Parameters:

- **attribute** (Object) — an attribute value

Returns:

- (Object) — the other attribute value

Instance Method Details

```
- (Boolean) ==(other)
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Compare the segment names and attributes of two instances

Parameters:

- **other** (RGFA::SegmentInfo) — the other instance

Returns:

- (Boolean)

```
- (Object) attribute
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

```
- (Symbol) name
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Returns the segment name

Returns:

- (Symbol) — the segment name

```
- (RGFA::SegmentInfo) other
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Returns same segment, inverted attribute

Returns:

- (RGFA::SegmentInfo) — same segment, inverted attribute

- (Symbol, RGFA::Line::Segment) **segment**

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Returns the segment instance or name

Returns:

- (Symbol, RGFA::Line::Segment) — the segment instance or name

---

- (String) **to_s**

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Returns name of the segment and attribute

Returns:

- (String) — name of the segment and attribute

- (Symbol) **to_sym**

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Returns name of the segment and attribute

Returns:

- (Symbol) — name of the segment and attribute

- (void) **validate!**

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

This method returns an undefined value.

Check that the elements of the array are compatible with the definition.

Raises:

- (RGFA::SegmentInfo::InvalidSizeError) — if size is not 2
- (RGFA::SegmentInfo::InvalidAttributeError) — if second element is not a valid info

Exception: RGFA::SegmentInfo::InvalidSizeError Private

Inherits:	Error show all
Defined in:	lib/rgfa/segment_info.rb

Overview

This class is part of a private API. You should avoid using this class if possible, as it may be removed or be changed in the future.

Error raised if the size of the array is wrong

Exception: RGFA::SegmentInfo::InvalidAttributeError

Private

Inherits:	Error show all
Defined in:	lib/rgfa/segment_info.rb

Overview

This class is part of a private API. You should avoid using this class if possible, as it may be removed or be changed in the future.

Error raised if an unknown value for attribute is used

Class: RGFA::SegmentEnd

Inherits:	SegmentInfo show all
Defined in:	lib/rgfa/segment_info.rb

Overview

A representation of a segment end

Constant Summary

ATTR =
Segment end type (begin or end)

```
[ END_TYPE_BEGIN = :B, END_TYPE_END = :E ]
```

Method Summary

Methods inherited from SegmentInfo

#==, #attribute, #name, other, #other, #segment, #to_s, #to_sym, #validate!

Methods inherited from Array

#gfa_datatype, #to_byte_array, #to_cigar_operation, #to_gfa_datastring, #to_numeric_array, #to_oriented_segment, #to_rgfa, #to_rgfa_line, #to_segment_end

Class: RGFA::OrientedSegment

Inherits:	SegmentInfo show all
Defined in:	lib/rgfa/segment_info.rb

Overview

A segment plus orientation

Constant Summary

ATTR =
Segment orientation

```
[ ORIENT_FWD = :+, ORIENT_REV = :- ]
```

Method Summary

Methods inherited from SegmentInfo

#==, #attribute, #name, other, #other, #segment, #to_s, #to_sym, #validate!

Methods inherited from Array

#gfa_datatype, #to_byte_array, #to_cigar_operation, #to_gfa_datastring, #to_numeric_array, #to_oriented_segment, #to_rgfa, #to_rgfa_line, #to_segment_end

Class: Object

Inherits:	BasicObject
Includes:	RGFA::FieldWriter
Defined in:	lib/rgfa/field_writer.rb

Method Summary

Methods included from RGFA::FieldWriter

#gfa_datatype, #to_gfa_datastring, #to_gfa_field

Class: Integer

Inherits:	Object show all
Defined in:	lib/rgfa/field_writer.rb

Overview

Support of the conversion to GFA datastrings for Integer

Instance Method Summary (collapse)

- (Object) **gfa_datatype**
  !macro gfa_datatype.

Instance Method Details

- (Object) **gfa_datatype**

!macro gfa_datatype

Class: Float

Inherits:	Object show all
Defined in:	lib/rgfa/field_writer.rb

Overview

Support of the conversion to GFA datastrings for Float

Instance Method Summary (collapse)

- (Object) **gfa_datatype**
  !macro gfa_datatype.

Instance Method Details

- (Object) **gfa_datatype**

!macro gfa_datatype

Class: Hash

Inherits:	Object show all
Defined in:	lib/rgfa/field_writer.rb

Overview

Support of the conversion to GFA datastrings for Hash

Instance Method Summary (collapse)

- (Object) **gfa_datatype**
  !macro gfa_datatype.
- (String) **to_gfa_datastring**(datatype)
  Representation of the data for GFA fields; this method does not automatically validate the string.

Instance Method Details

- (Object) gfa_datatype

!macro gfa_datatype

- (String) to_gfa_datastring(datatype)

Representation of the data for GFA fields; this method does not automatically validate the string.

Parameters:

- **datatype** (RGFA::Line::FIELD_DATATYPE)

Returns:

- (String)

Class: RGFA::NumericArray

Inherits:	Array show all
Defined in:	lib/rgfa/field_writer.rb

Overview

A numeric array representable using the data type B of the GFA specification

Defined Under Namespace

Classes: TypeError, ValueError

Constant Summary

SIGNEDINT_SUBTYPE =
Subtypes for signed integers, from the smallest to the largest

```
c s i
```

UNSIGNED_INT_SUBTYPE =
Subtypes for unsigned integers, from the smallest to the largest

```
SIGNEDINT_SUBTYPE.map{|st|st.upcase}
```

INT_SUBTYPE =
Subtypes for integers

```
UNSIGNED_INT_SUBTYPE + SIGNEDINT_SUBTYPE
```

FLOAT_SUBTYPE =
Subtypes for floats

```
["f"]
```

SUBTYPE =
Subtypes

```
INT_SUBTYPE + FLOAT_SUBTYPE
```

SUBTYPE_BITS =
Number of bits of unsigned integer subtypes

```
{"c" => 8, "s" => 16, "i" => 32}
```

SUBTYPE_RANGE =
Range for integer subtypes

```
Hash[
  INT_SUBTYPE.map do |subtype|
    [
     subtype,
     if subtype == subtype.upcase
       0..((2**SUBTYPE_BITS[subtype.downcase])-1)
     else
       (-(2**(SUBTYPE_BITS[subtype]-1)))..((2**(SUBTYPE_BITS[subtype]-1))-1)
     end
    ]
  end
```

]

Instance Method Summary (collapse)

- (RGFA::NumericArray::SUBTYPE) **compute_subtype**
  Computes the subtype of the array from its content.
- (Object) **gfa_datatype**
  !macro gfa_datatype.
- (RGFA::NumericArray) **to_numeric_array**(validate: false)
  Return self.
- (String) **to_s**
  GFA datatype B representation of the numeric array.
- (Object) **validate!**
  Validate the numeric array.

Methods inherited from Array

#to_byte_array, #to_cigar_operation, #to_gfa_datastring, #to_oriented_segment, #to_rgfa, #to_rgfa_line, #to_segment_end

Instance Method Details

- (RGFA::NumericArray::SUBTYPE) **compute_subtype**

Computes the subtype of the array from its content.

If all elements are float, then the computed subtype is "f". If all elements are integer, the smallest possible numeric subtype is computed; thereby, if all elements are non-negative, an unsigned subtype is selected, otherwise a signed subtype. In all other cases an exception is raised.

Returns:

- (RGFA::NumericArray::SUBTYPE)

Raises:

- (RGFA::NumericArray::ValueError) — if the array is not a valid numeric array

---

- (Object) **gfa_datatype**

!macro gfa_datatype

- (RGFA::NumericArray) **to_numeric_array**(validate: false)

Return self

Parameters:

- **validate** (Boolean) — *(default: false)* if true, validate the range of the numeric values, according to the array subtype

Returns:

- (RGFA::NumericArray)

Raises:

- (RGFA::NumericArray::ValueError) — if validate is set and any value is not compatible with

the subtype

```
- (String) to_s
```

GFA datatype B representation of the numeric array

Returns:

- (String)

Raises:

- (RGFA::NumericArray::ValueError) — if the array if not a valid numeric array

---

```
- (Object) validate!
```

Validate the numeric array

Raises:

- (RGFA::NumericArray::ValueError) — if the array is not valid

Exception: RGFA::LineMissingError

Inherits:	Error show all
Defined in:	lib/rgfa/line_getters.rb

Overview

The error raised by banged line finders if no line respecting the criteria exist in the RGFA

Class: RGFA::Line::Segment

Inherits:	RGFA::Line show all
Defined in:	lib/rgfa/line/segment.rb

Overview

A segment line of a RGFA file

Defined Under Namespace

Classes: InconsistentLengthError, UndefinedLengthError

Constant Summary

RECORD_TYPE =

```
:S
```

REQFIELDS =

```
[:name, :sequence]
```

PREDEFINED_OPTFIELDS =

```
[:LN, :RC, :FC, :KC]
```

DATATYPE =

```
{
  :name => :lbl,
  :sequence => :seq,
  :LN => :i,
  :RC => :i,
  :FC => :i,
  :KC => :i
}
```

Constants inherited from RGFA::Line

DELAYED_PARSING_DATATYPES, FIELD_DATATYPE, OPTFIELD_DATATYPE, RECORD_TYPES, RECORD_TYPE_LABELS, REQFIELD_DATATYPE, SEPARATOR

Instance Method Summary (collapse)

- (Integer?) **coverage**(count_tag: :RC, unit_length: 1)
  The coverage computed from a count_tag.
- (Integer) **coverage!**(count_tag: :RC, unit_length: 1)
  The coverage computed from a count_tag.
- (Integer?) **length**
- (Integer) **length!**
- (Object) **to_s**(without_sequence: false)
  String representation of the segment.

- (Symbol) **to_sym**

  Name of the segment as symbol.

- (Object) **validate_length!**

Methods inherited from RGFA::Line

#==, #clone, #delete, #fieldnames, #get, #get!, #get_datatype, #get_string, #get_string!, #initialize, #method_missing, #optional_fieldnames, #record_type, #required_fieldnames, #respond_to?, #set, #set_datatype, subclass, #to_a, #to_rgfa_line, #validate!, #validate_field!

Constructor Details

This class inherits a constructor from RGFA::Line

Dynamic Method Handling

This class handles dynamic methods through the `method_missing` method in the class RGFA::Line

Instance Method Details

- (Integer?) **coverage**(count_tag: :RC, unit_length: 1)

The coverage computed from a count_tag. If unit_length is provided then: count/(length-unit_length+1), otherwise: count/length. The latter is a good approximation if length >>> unit_length.

Parameters:

- **count_tag** (Symbol) — *(defaults to :RC)* integer tag storing the count, usually :KC, :RC or :FC
- **unit_length** (Integer) — the (average) length of a read (for :RC), fragment (for :FC) or k-mer (for :KC)

Returns:

- (Integer) — coverage, if count_tag and length are defined
- (nil) — otherwise

See Also:

- #coverage!

- (Integer) **coverage!**(count_tag: :RC, unit_length: 1)

The coverage computed from a count_tag. If unit_length is provided then: count/(length-unit_length+1), otherwise: count/length. The latter is a good approximation if length >>> unit_length.

Parameters:

- **count_tag** (Symbol) — *(defaults to :RC)* integer tag storing the count, usually :KC, :RC or :FC
- **unit_length** (Integer) — the (average) length of a read (for :RC), fragment (for :FC) or k-mer (for :KC)

Returns:

- (Integer) — coverage, if count_tag and length are defined

Raises:

- (RGFA::Line::TagMissingError) — if segment does not have count_tag
- (RGFA::Line::Segment::UndefinedLengthError) — if not an LN tag and the sequence is "*"

See Also:

- #coverage

- (Integer?) **length**

Returns:

- (Integer) — value of LN tag, if segment has LN tag
- (Integer) — sequence length if no LN and sequence not "*"
- (nil) — if sequence is "*"

See Also:

- #length!

- (Integer) **length!**

Returns:

- (Integer) — value of LN tag, if segment has LN tag
- (Integer) — sequence length if no LN and sequence not "*"

Raises:

- (RGFA::Line::Segment::UndefinedLengthError) — if not an LN tag and the sequence is "*"

See Also:

- #length

- (Object) **to_s**(without_sequence: false)

Returns string representation of the segment

Parameters:

- **without_sequence** (Boolean) — if true, output "*" instead of sequence

Returns:

- string representation of the segment

- (Symbol) **to_sym**

Returns name of the segment as symbol

Returns:

- (Symbol) — name of the segment as symbol

- (Object) **validate_length!**

Raises:

- (RGFA::Line::Segment::InconsistentLengthError) — if sequence length and LN tag are not consistent.

Exception: RGFA::Line::Segment::UndefinedLengthError

Inherits:	Error show all
Defined in:	lib/rgfa/line/segment.rb

Overview

Error raised if length of segment cannot be computed

Exception: RGFA::Line::Segment::InconsistentLengthError

Inherits:	Error show all
Defined in:	lib/rgfa/line/segment.rb

Overview

Error raised if length of segment and LN are not consistent

Exception: RGFA::NumericArray::ValueError

Inherits:	Error show all
Defined in:	lib/rgfa/numeric_array.rb

Overview

Exception raised if a value in a numeric array is not compatible with the selected subtype

Exception: RGFA::NumericArray::TypeError

Inherits:	Error show all
Defined in:	lib/rgfa/numeric_array.rb

Overview

Exception raised if an invalid subtype code is found

Exception: RGFA::DuplicatedLabelError

Inherits:	Error show all
Defined in:	lib/rgfa/line_creators.rb

Overview

Exception raised if a label for segment or path is duplicated

Class: RGFA::ConnectionInfo Private

Inherits:	Object show all
Defined in:	lib/rgfa/connection_info.rb

Overview

This class is part of a private API. You should avoid using this class if possible, as it may be removed or be changed in the future.

Note: It is not required that a segment has already been added to the GFA using an :S line. This is necessary, as the order of the lines in the file during parsing is arbitrary.

Collection of hashes which allow fast retrieval of the lines of a GFA graph which refer to a given segment.

Defined Under Namespace

Classes: ValidationError

Instance Method Summary (collapse)

- (void) **add**(rt, value, sn, dir = nil, o = nil) private
  Add a reference to a link/containment or path to connection infos.
- (void) **delete**(rt, value, sn, dir = nil, o = nil) private
  Remove a link/containment/path reference from connection infos.
- (void) **delete_segment**(sn) private
  Delete all information about a segment in the connection info.
- (Array<Integer>) **find**(rt, sn, dir = nil, o = nil) private
  Find indices of RGFA lines referring to a segment.
- (RGFA::ConnectionInfo) **initialize**(lines) constructor private
- (Array<RGFA::Line>) **lines**(rt, sn, dir = nil, o = nil) private
  Find GFA lines referring to a segment.
- (void) **rename_segment**(sn, new_sn) private
  Rename a segment in the connection info.
- (void) **validate!** private
  Validate the information in connection info (useful for debugging).

Constructor Details

- (RGFA::ConnectionInfo) **initialize**(lines)

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Parameters:

- **lines** (Array) — reference to RGFA instance @lines array (required by the #validate! and the #lines methods)

Instance Method Details

```
- (void) add(rt, value, sn, dir = nil, o = nil)
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

This method returns an undefined value.

Add a reference to a link/containment or path to connection infos

Parameters:

- **rt** (:L, :C, :P) — the record type
- **sn** (String, RGFA::Line::Segment) — the segment name or instance
- **dir** (:from, :to, nil) *(defaults to: nil)* — is segment the from or the to segment of the link/containment?; use nil for paths
- **value** (Integer) — an index in @lines
- **o** (:+, :-, nil) *(defaults to: nil)* — the segment orientation (links/containments); use nil for # paths

---

```
- (void) delete(rt, value, sn, dir = nil, o = nil)
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

This method returns an undefined value.

Remove a link/containment/path reference from connection infos

Examples:

```
delete("P", value, sn)                          # => rm path ref
delete("C"|"L", value, sn, :from|:to, :+|:-) # => rm link/cont. ref
delete("C"|"L", value, sn, :from|:to, nil)
            # => rm link/cont. ref from sn in both :+ and :- orient
```

Parameters:

- **rt** (:L, :C, :P) — the record type
- **sn** (String, RGFA::Line::Segment) — the segment name or instance
- **dir** (:from, :to, nil) *(defaults to: nil)* — is segment the from or the to segment of the link/containment?; use nil for paths
- **o** (:+, :-, nil) *(defaults to: nil)* — orientation (for links/containments); set to nil for paths; if nil in links/containments: both orientations
- **value** (Integer) — index in @lines to remove

```
- (void) delete_segment(sn)
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

This method returns an undefined value.

Delete all information about a segment in the connection info.

Parameters:

- **sn** (RGFA::Line::Segment, String) — the segment instance or segment name

---

```
- (Array<Integer>) find(rt, sn, dir = nil, o = nil)
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Note: Do not modify the returned array; modifications must be done using #add and #delete.

Find indices of RGFA lines referring to a segment

Examples:

```
find("P", sn)                        # => find paths
find("C"|"L", sn, :from|:to)         # => both orientations
find("C"|"L", sn, :from|:to, :+|:-)  # => only specified orientation
```

Parameters:

- **rt** (:L, :C, :P) — the record type
- **sn** (String, RGFA::Line::Segment) — the segment name or instance
- **dir** (:from, :to, nil) *(defaults to: nil)* — is segment the from or the to segment of the link/containment?; use nil for paths
- **o** (:+, :-, nil) *(defaults to: nil)* — orientation (for links/containments); set to nil for paths; if nil in links/containments: both orientations

Returns:

- (Array<Integer>) — the indices of lines array of given record type for all lines referring to the segment and respecting the dir and o conditions

---

```
- (Array<RGFA::Line>) lines(rt, sn, dir = nil, o = nil)
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

Note: You can modify the line instances, but do not modify the returned array itself; modifications must be done using #add and #delete.

Find GFA lines referring to a segment

Parameters:

- **rt** (:L, :C, :P) — the record type
- **sn** (String, RGFA::Line::Segment) — the segment name or instance
- **dir** (:from, :to, nil) *(defaults to: nil)* — is segment the from or the to segment of the link/containment?; use nil for paths
- **o** (:+, :-, nil) *(defaults to: nil)* — orientation (for links/containments); set to nil for paths; if nil in links/containments: both orientations

Returns:

- (Array<RGFA::Line>) — the lines of given record type referring to the segment and respecting the `dir` and `o` conditions

---

```
- (void) rename_segment(sn, new_sn)
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

This method returns an undefined value.

Rename a segment in the connection info.

Parameters:

- **sn** (RGFA::Line::Segment, String) — the old segment instance or name
- **new_sn** (RGFA::Line::Segment, String) — the new segment instance or name

---

```
- (void) validate!
```

This method is part of a private API. You should avoid using this method if possible, as it may be removed or be changed in the future.

This method returns an undefined value.

Validate the information in connection info (useful for debugging).

Raises:

- (RGFA::ConnectionInfo::ValidationError) — if any path/link/containment was deleted from `@lines`
- (RGFA::ConnectionInfo::ValidationError) — if any link/containment field (from/from_orient/to/to_orient) is not consistent with the stored information
- (RGFA::ConnectionInfo::ValidationError) — if the paths segment name field is not consistent with the stored information

Exception: RGFA::ConnectionInfo::ValidationError Private

Inherits:	Error show all
Defined in:	lib/rgfa/connection_info.rb

Overview

This class is part of a private API. You should avoid using this class if possible, as it may be removed or be changed in the future.

Error raised if the validation of the connection info failed

Class: RGFA::Line::Containment

Inherits:	RGFA::Line show all
Includes:	SegmentReferences
Defined in:	lib/rgfa/line/containment.rb

Overview

A containment line of a RGFA file

Constant Summary

RECORD_TYPE =

```
:C
```

REQFIELDS =

```
[:from, :from_orient, :to, :to_orient, :pos, :overlap]
```

PREDEFINED_OPTFIELDS =

```
[:MQ, :NM]
```

DATATYPE =

```
{
   :from => :lbl,
   :from_orient => :orn,
   :to => :lbl,
   :to_orient => :orn,
   :pos => :pos,
   :overlap => :cig,
   :MQ => :i,
   :NM => :i,
}
```

Constants inherited from RGFA::Line

DELAYED_PARSING_DATATYPES, FIELD_DATATYPE, OPTFIELD_DATATYPE, RECORD_TYPES, RECORD_TYPE_LABELS, REQFIELD_DATATYPE, SEPARATOR

Method Summary

Methods included from SegmentReferences

#circular?, #other

Methods inherited from RGFA::Line

#==, #clone, #delete, #fieldnames, #get, #get!, #get_datatype, #get_string, #get_string!, #initialize, #method_missing, #optional_fieldnames, #record_type, #required_fieldnames, #respond_to?, #set, #set_datatype, subclass, #to_a, #to_rgfa_line, #to_s, #validate!, #validate_field!

Constructor Details

This class inherits a constructor from RGFA::Line

Dynamic Method Handling

This class handles dynamic methods through the `method_missing` method in the class RGFA::Line

Exception: RGFA::Line::UnknownRecordTypeError

Inherits:	Error show all
Defined in:	lib/rgfa/line.rb

Overview

Error raised if the record_type is not one of RGFA::Line::RECORD_TYPES

Exception: RGFA::Line::UnknownDatatype

Inherits:	Error show all
Defined in:	lib/rgfa/line.rb

Overview

Error raised if an invalid datatype symbol is found

Exception: RGFA::Line::FieldnameError

Inherits:	Error show all
Defined in:	lib/rgfa/line.rb

Overview

Error raised if an invalid fieldname symbol is found

Exception: RGFA::Line::TagMissingError

Inherits:	Error show all
Defined in:	lib/rgfa/line.rb

Overview

Error raised if optional tag is not present

Exception: RGFA::Line::RequiredFieldMissingError

Inherits:	Error show all
Defined in:	lib/rgfa/line.rb

Overview

Error raised if too less required fields are specified.

Exception: RGFA::Line::CustomOptfieldNameError

Inherits:	Error show all
Defined in:	lib/rgfa/line.rb

Overview

Error raised if a non-predefined optional field uses upcase letters.

Exception: RGFA::Line::DuplicatedOptfieldNameError

Inherits:	Error show all
Defined in:	lib/rgfa/line.rb

Overview

Error raised if an optional field tag name is used more than once.

Exception: RGFA::Line::PredefinedOptfieldTypeError

Inherits:	Error show all
Defined in:	lib/rgfa/line.rb

Overview

Error raised if the type of a predefined optional field does not respect the specified type.