

Giorgio Gonnella

RGFA library - API documentation

Version 1.3.1

Table of Contents

Table of Contents	2
Documentation by YARD 0.8.7.6	14
Top Level Namespace	15
Defined Under Namespace	15
Module: RGFATools	16
Overview	16
Defined Under Namespace	16
Constant Summary	16
Constant Summary	16
Constants included from Multiplication	16
Method Summary	16
Methods included from PBubbles	16
Methods included from LinearPaths	16
Methods included from SuperfluousLinks	16
Methods included from Multiplication	17
Methods included from InvertibleSegments	17
Methods included from CopyNumber	17
Methods included from Artifacts	17
Module: RGFA::Paths	18
Overview	18
Instance Method Summary (collapse)	18
Instance Method Details	18
- (RGFA) delete_path(pt)	18
- (RGFA::Line::Path?) path(pt)	18
- (RGFA::Line::Path) path!(pt)	18
- (Array<RGFA::Line::Path>) paths	19
- (Array<RGFA::Line::Path>) paths_with(s)	19
Module: RGFA::Lines	20
Overview	20
Instance Method Summary (collapse)	20
Instance Method Details	20
- (RGFA) <<(gfa_line_string) - (RGFA) <<(gfa_line)	20
- (RGFA) rename(old_name, new_name)	20
- (RGFA) rm(segment) - (RGFA) rm(path) - (RGFA) rm(link) - (RGFA) rm(containment) - (RGFA) rm(:headers) - (RGFA) rm(array) - (RGFA) rm(method_name, *args)	21
Module: RGFA::Links	23
Overview	23
Instance Method Summary (collapse)	23
Instance Method Details	24
- (RGFA) delete_link(l)	24
- (RGFA) delete_other_links(segment_end, other_end, conserve_components: false)	24
- (RGFA::Line::Link?) link(segment_end1, segment_end2)	24
- (RGFA::Line::Link) link!(segment_end1, segment_end2)	24
- (RGFA::Line::Link?) link_from_to(oriented_segment1, oriented_segment2, cigar = [], equivalent = true)	24
- (RGFA::Line::Link) link_from_to!(oriented_segment1, oriented_segment2, cigar = [], equivalent = true)	25
- (Array<RGFA::Line::Link>) links	25
- (Array<RGFA::Line::Link>) links_between(segment_end1, segment_end2)	25
- (Array<RGFA::Line::Link>) links_from(oriented_segment, equivalent = true)	25
- (Array<RGFA::Line::Link>) links_from_to(oriented_segment1, oriented_segment2, cigar = [], equivalent = true)	26
- (Array<RGFA::Line::Link>) links_of(segment_end)	26
- (Array<RGFA::Line::Link>) links_to(oriented_segment, equivalent = true)	26
- (Array<RGFA::SegmentEnd>) neighbours(segment_end)	27
Module: RGFA::LoggerSupport	28
Overview	28

Instance Method Summary (collapse)	28
Instance Method Details	28
- (RGFA) enable_progress_logging(part: 0.1, channel: STDERR)	28
- (RGFA) progress_log(symbol, progress = 1, **keyargs)	28
- (RGFA) progress_log_end(symbol, **keyargs)	28
- (RGFA) progress_log_init(symbol, units, total, initmsg = nil)	29
Module: RGFA::Headers	30
Overview	30
Multiple header lines defining the same tag	30
Instance Method Summary (collapse)	30
Instance Method Details	30
- (RGFA) delete_headers	30
- (RGFA::Line::Header) header	31
- (Array<RGFA::Line::Header>) headers	31
Module: RGFA::Segments	32
Overview	32
Instance Method Summary (collapse)	32
Instance Method Details	32
- (Array<String>) connected_segments(segment)	32
- (RGFA) delete_segment(s, cascade = true)	32
- (RGFA::Line::Segment?) segment(s)	32
- (RGFA::Line::Segment) segment!(s)	33
- (Array<RGFA::Line::Segment>) segments	33
- (RGFA) unconnect_segments(segment1, segment2)	33
Module: RGFA::Sequence	34
Overview	34
Constant Summary	34
Instance Method Summary (collapse)	34
Instance Method Details	34
- (String) rc(tolerant: false, masequence: false)	34
Module: RGFA::LinearPaths	36
Overview	36
Instance Method Summary (collapse)	36
Instance Method Details	36
- (Array<RGFA::SegmentEnd>) linear_path(s, exclude = Set.new)	36
- (Array<Array<RGFA::SegmentEnd>>) linear_paths	36
- (RGFA) merge_linear_path(segpath, **options)	36
- (RGFA) merge_linear_paths(**options)	37
Module: RGFA::FieldWriter Private	38
Overview	38
Instance Method Summary (collapse)	38
Instance Method Details	38
- (RGFA::Line::FIELD_DATATYPE) default_gfa_datatype	38
- (String) to_gfa_field(datatype: nil)	38
- (Object) to_gfa_optfield(fieldname, datatype: default_gfa_datatype)	39
Module: RGFA::Containments	40
Overview	40
Instance Method Summary (collapse)	40
Instance Method Details	40
- (Array<RGFA::Line::Containment>) contained_in(s)	40
- (Array<RGFA::Line::Containment>) containing(s)	40
- (RGFA::Line::Containment?) containment(container, contained)	41
- (RGFA::Line::Containment) containment!(container, contained)	41
- (Array<RGFA::Line::Containment>) containments	41
- (Array<RGFA::Line::Containment>) containments_between(container, contained)	41
- (RGFA) delete_containment(c)	41

Module: RGFA::Connectivity	43
Overview	43
Instance Method Summary (collapse)	43
Instance Method Details	43
- (Array<Array<String>>) connected_components	43
- (Array<conn_symbol,conn_symbol>) connectivity(segment)	43
- (Boolean) cut_link?(link)	44
- (Boolean) cut_segment?(segment)	44
- (Array<String>) segment_connected_component(segment, visited = Set.new)	44
- (Array<RGFA>) split_connected_components	44
Module: RGFA::FieldParser Private	45
Overview	45
Defined Under Namespace	45
Instance Method Summary (collapse)	45
Instance Method Details	45
- (Object) parse_gfa_field(datatype: nil, validate_strings: true, fieldname: nil, frozen: false)	45
- (Array(Symbol, RGFA::Line::FIELD_DATATYPE, String)) parse_gfa_optfield	45
Module: RGFATools::PBubbles	47
Overview	47
Instance Method Summary (collapse)	47
Instance Method Details	47
- (RGFA) remove_p_bubble(segment_end1, segment_end2, count_tag: @default[:count_tag], unit_length: @default[:unit_length])	47
- (RGFA) remove_p_bubbles	47
Module: RGFATools::Artifacts	48
Overview	48
Instance Method Summary (collapse)	48
Instance Method Details	48
- (RGFA) remove_dead_ends(minlen)	48
- (RGFA) remove_small_components(minlen)	48
Module: RGFA::Multiplication	49
Overview	49
Instance Method Summary (collapse)	49
Instance Method Details	49
- (RGFA) multiply(segment, factor, copy_names: :lowercase, conserve_components: true)	49
Automatic computation of the copy names	49
Module: RGFA::FieldValidator Private	50
Overview	50
Constant Summary	50
Instance Method Summary (collapse)	50
Instance Method Details	50
- (void) validate_gfa_field!(datatype, fieldname = nil)	50
- (void) validate_segment_name!	51
Module: RGFATools::CopyNumber	52
Overview	52
Instance Method Summary (collapse)	52
Instance Method Details	52
- (RGFA) apply_copy_number(segment, count_tag: :cn, distribute: :auto, copy_names_suffix: :lowercase, origin_tag: :or, conserve_components: true)	52
- (RGFA) apply_copy_numbers(count_tag: :cn, distribute: :auto, copy_names_suffix: :lowercase, origin_tag: :or, conserve_components: true)	53
- (RGFA) compute_copy_numbers(single_copy_coverage, mincov: single_copy_coverage * 0.25, count_tag: @default[:count_tag], cn_tag: :cn, unit_length: @default[:unit_length])	53
- (RGFA) delete_low_coverage_segments(mincov, count_tag: @default[:count_tag], unit_length: @default[:unit_length])	53
- (RGFA) set_count_unit_length(unit_length)	53

- (RGFA) set_default_count_tag(tag)	54
Module: RGFATools::LinearPaths	55
Overview	55
Constant Summary	55
Instance Method Summary (collapse)	55
Instance Method Details	55
- (RGFA) merge_linear_path(segpath, **options)	55
Module: RGFATools::Multiplication	57
Overview	57
Constant Summary	57
Instance Method Summary (collapse)	57
Instance Method Details	57
- (RGFA) multiply_extended(segment, factor, copy_names: :lowercase, distribute: :auto, conserve_components: true, origin_tag: :or)	57
- (RGFA) multiply(segment, factor, copy_names: :lowercase, distribute: :auto, conserve_components: true, origin_tag: :or)	58
Module: RGFATools::SuperfluousLinks	59
Overview	59
Instance Method Summary (collapse)	59
Instance Method Details	59
- (RGFA) enforce_all_mandatory_links(conserve_components: true)	59
- (RGFA) enforce_segment_mandatory_links(segment, conserve_components: true)	59
- (RGFA) remove_self_link(segment)	59
- (RGFA) remove_self_links	60
Module: RGFATools::InvertibleSegments	61
Overview	61
Instance Method Summary (collapse)	61
Instance Method Details	61
- (RGFA) randomly_orient_invertible(segment)	61
- (RGFA) randomly_orient_invertibles	61
Class: RGFA	62
Overview	62
Interacting with the graph	62
Defined Under Namespace	63
Constant Summary	63
Constant Summary	63
Constants included from RGFATools::Multiplication	63
Instance Attribute Summary (collapse)	63
Class Method Summary (collapse)	63
Instance Method Summary (collapse)	63
Methods included from RGFATools::PBubbles	64
Methods included from RGFATools::LinearPaths	64
Methods included from RGFATools::SuperfluousLinks	64
Methods included from RGFATools::Multiplication	64
Methods included from RGFATools::InvertibleSegments	64
Methods included from RGFATools::CopyNumber	64
Methods included from RGFATools::Artifacts	64
Methods included from LoggerSupport	64
Methods included from Multiplication	64
Methods included from Connectivity	64
Methods included from LinearPaths	64
Methods included from Paths	64
Methods included from Containments	65
Methods included from Links	65
Methods included from Segments	65
Methods included from Headers	65
Methods included from Lines	65

Constructor Details	65
- (RGFA) initialize(validate: 2)	65
Instance Attribute Details	65
- (Object) validate	65
Class Method Details	65
+ (RGFA) from_file(filename, validate: 2)	65
Instance Method Details	66
- (Boolean) ==(other)	66
- (RGFA) clone	66
- (void) disable_extensions	66
- (void) enable_extensions	66
- (String) info(short = false)	66
- (Integer) n_dead_ends	66
- (Array<Symbol>) path_names	67
- (self) read_file(filename)	67
- (void) require_segments_first_order	67
- (Array<Symbol>) segment_names	67
- (void) to_file(filename)	67
- (self) to_rgfa	68
- (String) to_s	68
- (void) turn_off_validations	68
- (void) validate!	68
Class: String	69
Overview	69
Constant Summary	69
Constant Summary	69
Constants included from RGFA::FieldValidator	69
Constants included from RGFA::Sequence	69
Instance Method Summary (collapse)	69
Methods included from RGFA::FieldValidator	69
Methods included from RGFA::FieldParser	69
Methods included from RGFA::Sequence	69
Instance Method Details	69
- (RGFA::ByteArray) to_byte_array	69
- (RGFA::CIGAR) to_cigar	70
- (RGFA::NumericArray) to_numeric_array(validate: true)	70
- (RGFA) to_rgfa(validate: 2)	70
- (subclass of RGFA::Line) to_rgfa_line(validate: 2)	70
Class: Array	72
Overview	72
Direct Known Subclasses	72
Instance Method Summary (collapse)	72
Instance Method Details	72
- (RGFA::Line::FIELD_DATATYPE) default_gfa_datatype	73
- (Boolean) rgfa_field_array?	73
- (RGFA::ByteArray) to_byte_array	73
- (RGFA::CIGAR) to_cigar	73
- (RGFA::CIGAR::Operation) to_cigar_operation	73
- (String) to_gfa_field(datatype: default_gfa_datatype)	73
- (RGFA::NumericArray) to_numeric_array(validate: true)	74
- (RGFA::OrientedSegment) to_oriented_segment	74
- (RGFA) to_rgfa(validate: 2)	74
- (Object) to_rgfa_field_array(datatype = nil)	74
- (subclass of RGFA::Line) to_rgfa_line(validate: 2)	74
- (RGFA::SegmentEnd) to_segment_end	75
- (void) validate_gfa_field!(datatype, fieldname = nil)	75
Class: RGFA::Line	76
Overview	76
Direct Known Subclasses	76

Defined Under Namespace	76
Constant Summary	76
Class Method Summary (collapse)	77
Instance Method Summary (collapse)	77
Constructor Details	78
- (RGFA::Line) initialize(data, validate: 2, virtual: false)	78
Dynamic Method Handling	79
- (Object) method_missing(m, *args, &block)	79
Class Method Details	80
+ (Class) subclass(record_type)	80
Instance Method Details	80
- (Boolean) ==(o)	80
- (RGFA::Line) clone	80
- (Object?) delete(fieldname)	80
- (String) field_to_s(fieldname, optfield: false)	81
- (Array<Symbol>) fieldnames	81
- (Object?) get(fieldname, frozen: false)	81
- (Object?) get!(fieldname)	81
- (RGFA::Line::FIELD_DATATYPE) get_datatype(fieldname)	81
- (Array<Symbol>) optional_fieldnames	82
- (Object) real!(real_line)	82
- (Symbol) record_type	82
- (Array<Symbol>) required_fieldnames	82
- (Boolean) respond_to?(m, include_all = false)	82
- (Object) set(fieldname, value)	82
- (RGFA::Line::FIELD_DATATYPE) set_datatype(fieldname, datatype)	83
- (Array<[Symbol, Symbol, Object]>) tags	83
- (Array<String>) to_a	83
- (Object) to_rgfa_line(validate: nil)	83
- (String) to_s	84
- (void) validate!	84
- (void) validate_field!(fieldname)	84
- (Boolean) virtual?	84
Class: RGFA::CIGAR	85
Overview	85
Defined Under Namespace	85
Class Method Summary (collapse)	85
Instance Method Summary (collapse)	85
Methods inherited from Array	85
Class Method Details	85
+ (RGFA::CIGAR) from_string(str)	85
Instance Method Details	86
- (RGFA::CIGAR) clone	86
- (RGFA::CIGAR) complement	86
- (RGFA::CIGAR) to_cigar	86
- (String) to_s	86
- (void) validate!	86
- (void) validate_gfa_field!(datatype, fieldname = nil)	87
Exception: RGFA::CIGAR::ValueError	88
Overview	88
Class: RGFA::CIGAR::Operation	89
Overview	89
Constant Summary	89
Instance Attribute Summary (collapse)	89
Instance Method Summary (collapse)	89
Constructor Details	89
- (Operation) initialize(len, code)	89
Instance Attribute Details	89

- (Object) code	90
- (Object) len	90
Instance Method Details	90
- (Boolean) ==(other)	90
- (RGFA::CIGAR::Operation) to_cigar_operation	90
- (String) to_s	90
- (void) validate!	90
Exception: RGFA::Error	91
Overview	91
Direct Known Subclasses	91
Exception: RGFA::DuplicatedLabelError	92
Overview	92
Exception: RGFA::LineMissingError	93
Overview	93
Class: RGFA::Logger Private	94
Overview	94
Defined Under Namespace	94
Instance Method Summary (collapse)	94
Constructor Details	94
- (RGFA::Logger) initialize(verbose_level: 1, channel: STDERR, prefix: "#")	94
Instance Method Details	95
- (void) disable_progress	95
- (void) enable_progress(part: 0.1)	95
- (void) log(msg, min_verbose_level = 1)	95
- (void) progress_end(symbol, **keyargs)	95
- (void) progress_init(symbol, units, total, initmsg = nil)	96
- (void) progress_log(symbol, progress = 1, **keyargs)	96
Class: RGFA::Logger::ProgressData Private	97
Overview	97
Instance Attribute Summary (collapse)	97
Instance Attribute Details	97
- (Object) counter	97
- (Object) lastpart	97
- (Object) partsize	97
- (Object) starttime	98
- (Object) strlen	98
- (Object) total	98
- (Object) units	98
Class: RGFA::Line::Link	99
Overview	99
Constant Summary	99
Constants inherited from RGFA::Line	99
Instance Method Summary (collapse)	99
Methods inherited from RGFA::Line	100
Constructor Details	101
Dynamic Method Handling	101
Instance Method Details	101
- (Boolean) canonical?	101
Definition of canonical link	101
Details	101
- (RGFA::Line::Link) canonize!	101
- (Boolean) circular?	101
- (Boolean) circular_same_end?	102
- (Boolean) compatible?(other_oriented_from, other_oriented_to, other_overlap = [], equivalent = true)	102
- (Boolean) compatible_complement?(other_oriented_from, other_oriented_to, other_overlap = [])	102
- (Boolean) compatible_direct?(other_oriented_from, other_oriented_to, other_overlap = [])	102
- (RGFA::Line::Link) complement	103

- (RGFA::Line::Link) complement!	103
- (Boolean) complement?(other)	103
- (RGFA::CIGAR) complement_overlap	104
- (Boolean) eql?(other)	104
- (Boolean) eql_optional?(other)	104
- (RGFA::SegmentEnd) from_end	104
- (Symbol) from_name	104
- (Object) hash	105
- (RGFA::OrientedSegment) oriented_from	105
- (RGFA::OrientedSegment) oriented_to	105
- (Symbol) other(segment)	105
- (RGFA::SegmentEnd) other_end(segment_end)	105
- (Array<Array<(RGFA::Line::Path, Boolean)>>) paths	106
- (Boolean) same?(other)	106
- (Object) segment_ends_s	106
- (RGFA::SegmentEnd) to_end	106
- (Symbol) to_name	106
Class: RGFA::Line::Path	107
Overview	107
Defined Under Namespace	107
Constant Summary	107
Constants inherited from RGFA::Line	107
Instance Method Summary (collapse)	107
Methods inherited from RGFA::Line	108
Constructor Details	108
Dynamic Method Handling	108
Instance Method Details	108
- (Boolean) circular?	108
- (Boolean) linear?	108
- (Array<RGFA::Line::Link, Boolean>) links	108
- (Array<[RGFA::OrientedSegment, RGFA::OrientedSegment, RGFA::Cigar]>) required_links	108
- (Symbol) to_sym	109
- (Boolean) undef_overlaps?	109
Exception: RGFA::Line::Path::ListLengthsError	110
Overview	110
Class: RGFA::ByteArray	111
Overview	111
Defined Under Namespace	111
Instance Method Summary (collapse)	111
Methods inherited from Array	111
Instance Method Details	111
- (RGFA::Line::FIELD_DATATYPE) default_gfa_datatype	111
- (RGFA::ByteArray) to_byte_array	111
- (String) to_s	112
- (void) validate!	112
- (void) validate_gfa_field!(datatype, fieldname = nil)	112
Exception: RGFA::ByteArray::ValueError	113
Overview	113
Exception: RGFA::ByteArray::FormatError	114
Overview	114
Class: RGFA::FieldArray	115
Overview	115
Defined Under Namespace	115
Instance Attribute Summary (collapse)	115
Instance Method Summary (collapse)	115
Methods inherited from Array	115
Constructor Details	115

- (FieldArray) initialize(datatype, data = [])	115
Instance Attribute Details	115
- (Object) datatype (readonly)	116
Instance Method Details	116
- (Object) default_gfa_datatype	116
- (Object) push_with_validation(value, type, fieldname = nil)	116
- (Object) to_gfa_field(datatype: nil)	116
- (Object) validate_gfa_field!(datatype, fieldname = nil)	116
Exception: RGFA::FieldArray::Error	117
Overview	117
Exception: RGFA::FieldArray::TypeMismatchError	118
Overview	118
Class: RGFA::Line::Header	119
Overview	119
Constant Summary	119
Constants inherited from RGFA::Line	119
Instance Method Summary (collapse)	119
Methods inherited from RGFA::Line	119
Constructor Details	120
Dynamic Method Handling	120
Instance Method Details	120
- (Object) add(fieldname, value, datatype = nil)	120
- (self) merge(gfa_line)	120
- (Array<RGFA::Line::Header>) split	120
- (Array<(Symbol, Symbol, Object)>) tags	121
Class: RGFA::SegmentInfo Private	122
Overview	122
Direct Known Subclasses	122
Defined Under Namespace	122
Class Method Summary (collapse)	122
Instance Method Summary (collapse)	122
Methods inherited from Array	123
Class Method Details	123
+ (Symbol) invert(attribute)	123
Instance Method Details	123
- (Boolean) <=>(other)	123
- (Boolean) ==(other)	123
- (Symbol) attribute	124
- (Symbol) attribute=(value)	124
- (Symbol) attribute_inverted	124
- (RGFA::SegmentInfo) invert_attribute	124
- (Symbol) name	124
- (Symbol, RGFA::Line::Segment) segment	125
- (Object) segment=(value)	125
- (String) to_s	125
- (Symbol) to_sym	125
- (void) validate!	126
Exception: RGFA::SegmentInfo::InvalidSizeError Private	127
Overview	127
Exception: RGFA::SegmentInfo::InvalidAttributeError Private	128
Overview	128
Class: RGFA::SegmentEnd	129
Overview	129
Constant Summary	129
Method Summary	129
Methods inherited from SegmentInfo	129

Methods inherited from Array	129
Class: RGFA::OrientedSegment	130
Overview	130
Constant Summary	130
Method Summary	130
Methods inherited from SegmentInfo	130
Methods inherited from Array	130
Class: Object	131
Instance Method Summary (collapse)	131
Methods included from RGFA::FieldWriter	131
Instance Method Details	131
- (void) validate_gfa_field!(datatype, fieldname = nil)	131
Class: Fixnum	132
Instance Method Summary (collapse)	132
Instance Method Details	132
- (RGFA::Line::FIELD_DATATYPE) default_gfa_datatype	132
- (void) validate_gfa_field!(datatype, fieldname = nil)	132
Class: Float	133
Instance Method Summary (collapse)	133
Instance Method Details	133
- (RGFA::Line::FIELD_DATATYPE) default_gfa_datatype	133
- (void) validate_gfa_field!(datatype, fieldname = nil)	133
Class: Hash	134
Instance Method Summary (collapse)	134
Instance Method Details	134
- (RGFA::Line::FIELD_DATATYPE) default_gfa_datatype	134
- (String) to_gfa_field(datatype: nil)	134
- (void) validate_gfa_field!(datatype, fieldname = nil)	134
Class: RGFA::NumericArray	136
Overview	136
Defined Under Namespace	136
Constant Summary	136
Class Method Summary (collapse)	137
Instance Method Summary (collapse)	137
Methods inherited from Array	137
Class Method Details	137
+ (RGFA::NumericArray::INT_SUBTYPE) integer_type(range)	137
Instance Method Details	137
- (RGFA::NumericArray::SUBTYPE) compute_subtype	137
- (RGFA::Line::FIELD_DATATYPE) default_gfa_datatype	138
- (RGFA::NumericArray) to_numeric_array(validate: false)	138
- (String) to_s	138
- (Object) validate!	138
- (void) validate_gfa_field!(datatype, fieldname = nil)	139
Class: RGFA::Line::Segment	140
Overview	140
Defined Under Namespace	140
Constant Summary	140
Constants inherited from RGFA::Line	140
Instance Attribute Summary (collapse)	140
Instance Method Summary (collapse)	141
Methods inherited from RGFA::Line	141
Constructor Details	141
Dynamic Method Handling	141
Instance Attribute Details	141

- (Hash{RGFA::Line::DIRECTION => Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Containment>}})	141
containments	141
- (Hash{RGFA::Line::DIRECTION => Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Link>}}) links	142
- (Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Path>}) paths	142
Instance Method Details	142
- (Object) all_connections	142
- (Object) all_containments	142
- (Object) all_links	143
- (Object) all_paths	143
- (Object) all_references	143
- (Integer?) coverage(count_tag: :RC, unit_length: 1)	143
- (Integer) coverage!(count_tag: :RC, unit_length: 1)	143
- (Integer?) length	144
- (Integer) length!	144
- (String) to_gfa_field(datatype: nil)	144
- (Object) to_s(without_sequence: false)	144
- (Symbol) to_sym	145
- (void) validate_gfa_field!(datatype, fieldname = nil)	145
- (Object) validate_length!	145
Class: RGFA::Line::Comment	146
Overview	146
Constant Summary	146
Constants inherited from RGFA::Line	146
Method Summary	146
Methods inherited from RGFA::Line	146
Constructor Details	146
Dynamic Method Handling	146
Exception: RGFA::Line::Segment::UndefinedLengthError	147
Overview	147
Exception: RGFA::Line::Segment::InconsistentLengthError	148
Overview	148
Exception: RGFA::FieldParser::FormatError Private	149
Overview	149
Exception: RGFA::FieldParser::UnknownDatatypeError Private	150
Overview	150
Exception: RGFA::NumericArray::ValueError	151
Overview	151
Exception: RGFA::NumericArray::TypeError	152
Overview	152
Class: Symbol	153
Instance Method Summary (collapse)	153
Instance Method Details	153
- (void) validate_gfa_field!(datatype, fieldname = nil)	153
Class: RGFA::Line::Containment	154
Overview	154
Constant Summary	154
Constants inherited from RGFA::Line	154
Instance Method Summary (collapse)	154
Methods inherited from RGFA::Line	155
Constructor Details	155
Dynamic Method Handling	155
Instance Method Details	155
- (Boolean) canonical?	155
Definition of canonical containment	155
Details	155

- (Symbol) from_name	155
- (RGFA::OrientedSegment) oriented_from	156
- (RGFA::OrientedSegment) oriented_to	156
- (Integer?) rpos	156
- (Symbol) to_name	156
Class: RGFA::SegmentEndsPath	157
Overview	157
Instance Method Summary (collapse)	157
Methods inherited from Array	157
Instance Method Details	157
- (RGFA::SegmentEndsPath) reverse	157
Exception: RGFA::Line::UnknownRecordTypeError	158
Overview	158
Exception: RGFA::Line::UnknownDatatype	159
Overview	159
Exception: RGFA::Line::FieldnameError	160
Overview	160
Exception: RGFA::Line::TagMissingError	161
Overview	161
Exception: RGFA::Line::RequiredFieldMissingError	162
Overview	162
Exception: RGFA::Line::CustomOptfieldnameError	163
Overview	163
Exception: RGFA::Line::DuplicatedOptfieldnameError	164
Overview	164
Exception: RGFA::Line::PredefinedOptfieldTypeError	165
Overview	165

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/GFA-spec/GFA-spec/blob/master/GFA-spec.md as close as possible.

The library allows to create a RGFA 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).

Installation

The latest release of the gem can be installed from the rubygems repository using: ```gem install rgfa```

Alternatively this git repository can be cloned or the source code installed from a release archive, and then the gem created and installed using: ```rake install```

Usage

To use the library in your Ruby scripts, just require it as follows: ```require "rgfa"```

Additional functionality, which requires custom tags and additional conventions, is included in a separate part of the code named [RGFATools](#) and can be accessed with: ```require "rgfatools"```

Documentation

A cheatsheet is available as pdf under github.com/ggonnella/rgfa/blob/master/cheatsheet/rgfa-cheatsheet-1.3.pdf

The full API documentation is available as pdf under github.com/ggonnella/rgfa/blob/master/pdfdoc/rgfa-api-1.3.pdf or in HTML format (www.rubydoc.info/github/ggonnella/rgfa/master/RGFA).

The main class of the library is [RGFA](#), which is a good starting point when reading the documentation.

References

The manuscript describing the library has been presented at the German Conference on Bioinformatics 2016. Currently it is under review and available as a Peer Journal preprint:

Gonnella G, Kurtz S. (2016) RGFA: powerful and convenient handling of assembly graphs. PeerJ Preprints 4:e2381v1 doi.org/10.7287/peerj.preprints.2381v1

Top Level Namespace

Defined Under Namespace

Modules: [RGFATools](#) **Classes:** [Array](#), [Fixnum](#), [Float](#), [Hash](#), [Object](#), [RGFA](#), [String](#), [Symbol](#)

Module: RGFATools

Includes:	Artifacts , CopyNumber , InvertibleSegments , LinearPaths , Multiplication , PBubbles , SuperfluousLinks
Included in:	RGFA
Defined in:	lib/rgfatools.rb

Overview

Module defining additional methods for the RGFA class.

RGFATools is an extension to the RGFA library, which allow to perform further operations. Thereby additional conventions are required, with respect to the GFA specification, which are compatible with it.

The methods defined here allow, e.g., to randomly orient a segment which has the same connections on both sides, to compute copy numbers and multiply or delete segments according to them, to distribute the links of copies after multiplying a segment, or to eliminate edges in the graph which are incompatible with an hamiltonian path.

Custom optional fields are defined, such as "cn" for the copy number of a segment, "or" for the original segment(s) of a duplicated or merged segment, "mp" for the starting position of original segments in a merged segment, "rp" for the position of possible inversions due to arbitrary orientation of some segments by the program.

Furthermore a convention for the naming of the segments is introduced, which gives a special meaning to the characters "_^()".

Defined Under Namespace

Modules: [Artifacts](#), [CopyNumber](#), [InvertibleSegments](#), [LinearPaths](#), [Multiplication](#), [PBubbles](#), [SuperfluousLinks](#)

Constant Summary

Constant Summary

Constants included from [Multiplication](#)

[Multiplication::LINKS_DISTRIBUTION_POLICY](#)

Method Summary

Methods included from [PBubbles](#)

[#remove_p_bubble](#), [#remove_p_bubbles](#)

Methods included from [LinearPaths](#)

[#merge_linear_path](#)

Methods included from [SuperfluousLinks](#)

[#enforce_all_mandatory_links](#), [#enforce_segment_mandatory_links](#),
[#remove_self_link](#), [#remove_self_links](#)

Methods included from [Multiplication](#)

`#multiply_extended, #multiply_with_rgfatools`

Methods included from [InvertibleSegments](#)

`#randomly_orient_invertible, #randomly_orient_invertibles`

Methods included from [CopyNumber](#)

`#apply_copy_number, #apply_copy_numbers, #compute_copy_numbers,
#delete_low_coverage_segments, #set_count_unit_length, #set_default_count_tag`

Methods included from [Artifacts](#)

`#remove_dead_ends, #remove_small_components`

Module: RGFA::Paths

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

Overview

Methods for the RGFA class, which allow to handle paths in the graph.

Instance Method Summary

(collapse)

- (RGFA) **delete_path**(pt)
Delete a path from the RGFA graph.
- (RGFA::Line::Path?) **path**(pt)
Searches the path with name equal to pt.
- (RGFA::Line::Path) **path!**(pt)
Searches the path with name equal to pt.
- (Array<RGFA::Line::Path>) **paths**
All path lines of the graph.
- (Array<RGFA::Line::Path>) **paths_with**(s)
Paths whose segment_names include the specified segment.

Instance Method Details

- (RGFA) **delete_path**(pt)

Delete a path from the RGFA graph

Parameters:

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

Returns:

- (RGFA) — self

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

Searches the path with name equal to pt.

Parameters:

- pt (String, RGFA::Line::Path) — a path or 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!**(pt)

Searches the path with name equal to pt.

Parameters:

- `pt (String, RGFA::Line::Path)` — a path or 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(s)
```

Returns paths whose `segment_names` include the specified segment.

Parameters:

- `s (RGFA::Line::Segment, Symbol)` — a segment instance or name

Returns:

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

Module: RGFA::Lines

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

Overview

Methods for the RGFA class, which allow to handle lines of multiple types.

Instance Method Summary

(collapse)

- (RGFA) <<(gfa_line)

Add a line to a RGFA.

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

Rename a segment or a path.

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

Delete elements from the RGFA graph.

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) **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

```
- (RGFA) rm(segment)
- (RGFA) rm(path)
- (RGFA) rm(link)
- (RGFA) rm(containment)
- (RGFA) rm(:headers)
- (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(link)
```

Parameters:

- `link (RGFA::Line::Link)` — link

```
- (RGFA) rm(containment)
```

Parameters:

- `link (RGFA::Line::Containment)` — containment

```
- (RGFA) rm(:headers)
```

Remove all headers

```
- (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

Module: RGFA::Links

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

Overview

Methods for the RGFA class, which allow to handle links in the graph.

Instance Method Summary

(collapse)

- (RGFA) **delete_link**(l)

Deletes a link and all paths depending on it.

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

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

- (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.

- (RGFA::Line::Link?) **link_from_to**(oriented_segment1, oriented_segment2, cigar = [], equivalent = true)

Search the link from a segment S1 in a given orientation to another segment S2 in a given, or the equivalent link from S2 to S1 with inverted orientations.

- (RGFA::Line::Link) **link_from_to!**(oriented_segment1, oriented_segment2, cigar = [], equivalent = true)

Search the link from a segment S1 in a given orientation to another segment S2 in a given, or the equivalent link from S2 to S1 with inverted orientations.

- (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_from**(oriented_segment, equivalent = true)

Find links from the segment in the specified orientation (or the equivalent links, i.e. to the segment in opposite orientation).

- (Array<RGFA::Line::Link>) **links_from_to**(oriented_segment1, oriented_segment2, cigar = [], equivalent = true)

Search all links from a segment S1 in a given orientation to another segment S2 in a given, or the equivalent links from S2 to S1 with inverted orientations.

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

Finds links of the specified end of segment.

- (Array<RGFA::Line::Link>) **links_to**(oriented_segment, equivalent = true)

Find links to the segment in the specified orientation (or the equivalent links, i.e. from the segment in opposite orientation).

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

Finds segment ends connected to the specified segment end.

Instance Method Details

```
- (RGFA) delete_link(1)
```

Deletes a link and all paths depending on it

Parameters:

- 1 (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 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::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.
-

```
- (RGFA::Line::Link?) link_from_to(oriented_segment1, oriented_segment2, cigar
```



```
= [], equivalent = true)
```

Search the link from a segment S1 in a given orientation to another segment S2 in a given, or the equivalent link from S2 to S1 with inverted orientations.

Parameters:

- **oriented_segment1** ([RGFA::OrientedSegment](#)) — a segment with orientation
- **oriented_segment2** ([RGFA::OrientedSegment](#)) — a segment with orientation
- **cigar** ([RGFA::CIGAR](#)) (*defaults to: []*) — shall match if not empty/undef
- **equivalent** ([Boolean](#)) (*defaults to: true*) — return also equivalent links.

Returns:

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

```
- (RGFA::Line::Link) link_from_to!(oriented_segment1, oriented_segment2, cigar  
= [], equivalent = true)
```

Search the link from a segment S1 in a given orientation to another segment S2 in a given, or the equivalent link from S2 to S1 with inverted orientations.

Parameters:

- **oriented_segment1** ([RGFA::OrientedSegment](#)) — a segment with orientation
- **oriented_segment2** ([RGFA::OrientedSegment](#)) — a segment with orientation
- **cigar** ([RGFA::CIGAR](#)) (*defaults to: []*) — shall match if not empty/undef
- **equivalent** ([Boolean](#)) (*defaults to: true*) — return also equivalent links.

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_from(oriented_segment, equivalent = true)
```

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

Find links from the segment in the specified orientation (or the equivalent links, i.e. to the segment in opposite orientation).

Parameters:

- **oriented_segment** ([RGFA::OrientedSegment](#)) — a segment with orientation
- **equivalent** ([Boolean](#)) (*defaults to: true*) — return also equivalent links.

Returns:

- ([Array<RGFA::Line::Link>](#))

```
- (Array<RGFA::Line::Link>) links_from_to(oriented_segment1, oriented_segment2,
cigar = [], equivalent = true)
```

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

Search all links from a segment S1 in a given orientation to another segment S2 in a given, or the equivalent links from S2 to S1 with inverted orientations.

Parameters:

- **oriented_segment1** ([RGFA::OrientedSegment](#)) — a segment with orientation
- **oriented_segment2** ([RGFA::OrientedSegment](#)) — a segment with orientation
- **cigar** ([RGFA::CIGAR](#)) (*defaults to: []*) — shall match if not empty/undef
- **equivalent** ([Boolean](#)) (*defaults to: true*) — return also equivalent links.

Returns:

- ([Array<RGFA::Line::Link>](#))

```
- (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::Line::Link>) links_to(oriented_segment, equivalent = true)
```

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

Find links to the segment in the specified orientation (or the equivalent links, i.e. from

the segment in opposite orientation).

Parameters:

- `oriented_segment` (`RGFA::OrientedSegment`) — a segment with orientation
- `equivalent` (`Boolean`) (*defaults to: true*) — return also equivalent links.

Returns:

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

```
- (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`

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)

private

Updates progress logging for a computation.

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

private

Completes progress logging for a computation.

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

private

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)

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.

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)

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.

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)
```

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.

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::Headers

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

Overview

Methods for accessing the GFA header information.

The GFA header is accessed using `RGFA#header`, which returns a `Line::Header` object.

Multiple header lines defining the same tag

The specification does not explicitly forbid to have the same tag on different lines. To represent this case, a "field array" (`RGFA::FieldArray`) is used, which is an array of instances of a tag, from different lines of the header.

Examples:

Accessing the header information

```
rgfa.header.VN # => "1.0"
rgfa.header.co = "This the header comment"
rgfa.header.ni = 100
rgfa.header.field_to_s(:ni) # => "ni:i:100"
```

Header with tags repeated on different lines (see `FieldArray`)

```
rgfa.header.ni # => RGFA::FieldArray<[100,200] @datatype: :i>
rgfa.header.ni[0] # 100
rgfa.header.ni << 200 # "200" is also OK
rgfa.header.ni.map!{|i|i-10}
rgfa.header.ni = [100,200,300].to_rgfa_field_array
```

Adding instances of a tag (will go on different header lines)

```
rgfa.header.add(:xx, 100) # => 100 # single i tag, if .xx did not exist yet
rgfa.header.add(:xx, 100) # => RGFA::FieldArray<[100,100] @datatype: :i>
rgfa.header.add(:xx, 100) # => RGFA::FieldArray<[100,100,100] @datatype :i>
```

Instance Method Summary

(collapse)

- (RGFA) `delete_headers` private

Remove all information from the header.

- (RGFA::Line::Header) `header`

An header line representing the entire header information; if multiple header line were present, and they contain the same tag, the tag value is represented by a `FieldArray`.

- (Array<RGFA::Line::Header>) `headers` private

Header information in single-tag-lines.

Instance Method Details

- (RGFA) `delete_headers`

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.

Remove all information from the header.

Returns:

- ([RGFA](#)) — self

- ([RGFA::Line::Header](#)) **header**

Returns an header line representing the entire header information; if multiple header line were present, and they contain the same tag, the tag value is represented by a [FieldArray](#)

Returns:

- ([RGFA::Line::Header](#)) — an header line representing the entire header information; if multiple header line were present, and they contain the same tag, the tag value is represented by a [FieldArray](#)

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

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: Read-only! The returned array contains copies of the original values, i.e. changes in the lines will not affect the RGFA object; to update the values in the RGFA use the `#header` method.

Header information in single-tag-lines.

Returns an array of [RGFA::Line::Header](#) objects, each containing a single field of the header.

Returns:

- ([Array<RGFA::Line::Header>](#))

Module: RGFA::Segments

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

Overview

Methods for the RGFA class, which allow to handle segments in the graph.

Instance Method Summary

(collapse)

- (Array<String>) **connected_segments**(segment)
List of names of segments connected to `segment` by links or containments.
- (RGFA) **delete_segment**(s, cascade = true)
Delete a segment from the RGFA graph.
- (RGFA::Line::Segment?) **segment**(s)
Searches the segment with name equal to `segment_name`.
- (RGFA::Line::Segment) **segment!**(s)
Searches the segment with name equal to `segment_name`.
- (Array<RGFA::Line::Segment>) **segments**
All segment lines of the graph.
- (RGFA) **unconnect_segments**(segment1, segment2)
Delete all links/containments involving two segments.

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

- (RGFA) **delete_segment**(s, cascade = true)

Delete a segment from the RGFA graph

Parameters:

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

Returns:

- (RGFA) — self

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

Searches the segment with name equal to `segment_name`.

Parameters:

- `s (String, RGFA::Line::Segment)` — a segment or 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!(s)
```

Searches the segment with name equal to `segment_name`.

Parameters:

- `s (String, RGFA::Line::Segment)` — a segment or 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 segment lines of the graph

Returns:

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

```
- (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::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::LinearPaths

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

Overview

Methods for the RGFA class, which allow to find and merge linear paths.

Instance Method Summary

(collapse)

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

Find a path without branches.

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

Find all unbranched paths in the graph.

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

Merge a linear path, i.e.

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

Merge all linear paths in the graph, i.e.

Instance Method Details

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

Find a path without branches.

The path must include `segment` and excludes segments in `exclude`. Any segment used in the returned path will be added to `exclude`

Parameters:

- **s** (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 in the graph.

Returns:

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

- (RGFA) **merge_linear_path**(segbath, **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:

- `segbath` ([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 found; 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 $\text{Sum}(|s|)/|m|$

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 and 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 found; 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 $\text{Sum}(|s|)/|m|$

Returns:

- ([RGFA](#)) — self

Module: RGFA::FieldWriter Private

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

Overview

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

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:

- `#default_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_field` 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`) `default_gfa_datatype`

private

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

- (`String`) `to_gfa_field`(datatype: nil)

private

Representation of the data for GFA fields; this method does not (in general) validate the string.

- (`Object`) `to_gfa_optfield`(fieldname, datatype: default_gfa_datatype)

private

Representation of the data as an optional field.

Instance Method Details

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

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.

Optional field GFA datatype to use, if none is provided

Returns:

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

- (`String`) `to_gfa_field`(datatype: 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.

Representation of the data for GFA fields; this method does not (in general) validate the string. The method can be overwritten for a given class, and may take the `#default_gfa_datatype` into consideration.

Returns:

- (String)

```
- (Object) to_gfa_optfield(fieldname, datatype: default_gfa_datatype)
```

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.

Representation of the data as an optional field

Parameters:

- **fieldname** (Symbol) — the tag name
- **datatype** (RGFA::Line::OPTFIELD_DATATYPE) — *(defaults to: the value returned by #default_gfa_datatype)*

Module: RGFA::Containments

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

Overview

Methods for the RGFA class, which allow to handle containments in the graph.

Instance Method Summary

(collapse)

- (Array<RGFA::Line::Containment>) **contained_in**(s)
Find containment lines whose from segment name is segment_name.
- (Array<RGFA::Line::Containment>) **containing**(s)
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 in the graph.
- (Array<RGFA::Line::Containment>) **containments_between**(container, contained)
Searches all containments of contained in container.
- (RGFA) **delete_containment**(c)
Delete a containment.

Instance Method Details

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

Find containment lines whose from segment name is segment_name

Parameters:

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

Returns:

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

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

Find containment lines whose to segment name is segment_name

Parameters:

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

Returns:

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

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

Searches a containment of `contained` in `container`. Raises an exception if no such containment was found.

Parameters:

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

Returns:

- (RGFA::Line::Containment)

Raises:

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

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

All containments in 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, Symbol) — a segment instance or name
- **contained** (RGFA::Line::Segment, Symbol) — a segment instance or name

Returns:

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

```
- (RGFA) delete_containment(c)
```

Delete a containment

Parameters:

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

Returns:

- (RGFA) — self

Module: RGFA::Connectivity

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

Overview

Methods which analyse the connectivity of the graph.

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<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 RGFA's.

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)
```

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

Parameters:

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

Returns:

- (Boolean)
-

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

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)
-

```
- (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 RGFA's

Returns:

- (`Array<RGFA>`)

Module: RGFA::FieldParser Private

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

Overview

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

Methods to parse the string representations of the GFA fields

Defined Under Namespace

Classes: [FormatError](#), [UnknownDatatypeError](#)

Instance Method Summary

(collapse)

```
- (Object) parse_gfa_field(datatype: nil, validate_strings: true, fieldname: nil, frozen: false) private
```

Parse a string representation of a GFA field value.

```
- (Array(Symbol, RGFA::Line::FIELD_DATATYPE, String)) parse_gfa_optfield private
```

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

Instance Method Details

```
- (Object) parse_gfa_field(datatype: nil, validate_strings: true, fieldname: nil, frozen: false)
```

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.

Parse a string representation of a GFA field value

Parameters:

- **datatype** ([RGFA::Line::FIELD_DATATYPE](#))

Raises:

- ([RGFA::Error](#)) — if the value is not valid

```
- (Array(Symbol, RGFA::Line::FIELD_DATATYPE, String)) parse_gfa_optfield
```

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.

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: RGFATools::PBubbles

Included in:	RGFATools
Defined in:	lib/rgfatools/p_bubbles.rb

Overview

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

Instance Method Summary

(collapse)

```
- (RGFA) remove_p_bubble(segment_end1, segment_end2, count_tag:  
@default[:count_tag], unit_length: @default[:unit_length])
```

Removes a p-bubble between segment_end1 and segment_end2.

```
- (RGFA) remove_p_bubbles
```

Removes all p-bubbles in the graph.

Instance Method Details

```
- (RGFA) remove_p_bubble(segment_end1, segment_end2, count_tag:  
@default[:count_tag], unit_length: @default[:unit_length])
```

Removes a p-bubble between segment_end1 and segment_end2

Parameters:

- **segment_end1** (RGFA::SegmentEnd) — a segment end
- **segment_end2** (RGFA::SegmentEnd) — another segment end
- **count_tag** (Symbol) — (defaults to: :RC or the value set by [CopyNumber#set_default_count_tag](#)) the count tag to use for coverage computation
- **unit_length** (Integer) — (defaults to: 1 or the value set by [CopyNumber#set_count_unit_length](#)) the unit length to use for coverage computation

Returns:

- (RGFA) — self

```
- (RGFA) remove_p_bubbles
```

Removes all p-bubbles in the graph

Returns:

- (RGFA) — self

Module: RGFATools::Artifacts

Included in:	RGFATools
Defined in:	lib/rgfatools/artifacts.rb

Overview

Methods which edit the graph components without traversal

Instance Method Summary

(collapse)

- (RGFA) **remove_dead_ends**(minlen)

Remove end segments, whose sequence length is under a specified value.

- (RGFA) **remove_small_components**(minlen)

Remove connected components whose sum of lengths of the segments is under a specified value.

Instance Method Details

- (RGFA) **remove_dead_ends**(minlen)

Remove end segments, whose sequence length is under a specified value.

Parameters:

- **minlen** (Integer) — the minimum length

Returns:

- (RGFA) — self
-

- (RGFA) **remove_small_components**(minlen)

Remove connected components whose sum of lengths of the segments is under a specified value.

Parameters:

- **minlen** (Integer) — the minimum length

Returns:

- (RGFA) — self

Module: RGFA::Multiplication

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

Overview

Method for the RGFA class, which allow to split a segment into multiple copies.

Instance Method Summary

(collapse)

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

Create multiple copies of a segment.

Instance Method Details

```
- (RGFA) multiply(segment, factor, copy_names: :lowercase, 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, it is checked if the name of the original segment ends with a relevant string, i.e. a lower case letter (for :lowercase), an upper case letter (for :uppercase), 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 :lowercase), A (for :uppercase), 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-existent 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** (:lowercase, :uppercase, :number, :copy, Array<String>) — (Defaults to: :lowercase) 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 Connectivity#cut_segment?(segment) is false.

Returns:

- (RGFA) — self

Module: RGFA::FieldValidator Private

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

Overview

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

Methods to validate the string representations of the GFA fields data

Constant Summary

DATASTRING_VALIDATION_REGEX =

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

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
  :cmt => /.*/ , # content of comment line, everything is allowed
}
```

Instance Method Summary

(collapse)

- (void) **validate_gfa_field!**(datatype, fieldname = nil)

private

Validates the string according to the provided datatype.

- (void) **validate_segment_name!**

private

Validates segment names, to check that they do not contain + or - followed by comma.

Instance Method Details

- (void) **validate_gfa_field!**(datatype, fieldname = 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.

Validates the string according to the provided datatype

Parameters:

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

Raises:

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

- (void) **validate_segment_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.

This method returns an undefined value.

Validates segment names, to check that they do not contain + or - followed by comma

Raises:

- (`RGFA::FieldParser::FormatError`) — if the segment name is invalid

Module: RGFATools::CopyNumber

Included in:	RGFATools
Defined in:	lib/rgfatools/copy_number.rb

Overview

Methods which edit the graph components without traversal

Instance Method Summary

(collapse)

```
- (RGFA) apply_copy_number(segment, count_tag: :cn, distribute: :auto, copy_names_suffix: :lowercase, origin_tag: :or, conserve_components: true)
```

Applies the computed copy number to a segment.

```
- (RGFA) apply_copy_numbers(count_tag: :cn, distribute: :auto, copy_names_suffix: :lowercase, origin_tag: :or, conserve_components: true)
```

Applies the computed copy number to all segments.

```
- (RGFA) compute_copy_numbers(single_copy_coverage, mincov: single_copy_coverage * 0.25, count_tag: @default[:count_tag], cn_tag: :cn, unit_length: @default[:unit_length])
```

Self.

```
- (RGFA) delete_low_coverage_segments(mincov, count_tag: @default[:count_tag], unit_length: @default[:unit_length])
```

Delete segments which have a coverage under a specified value.

```
- (RGFA) set_count_unit_length(unit_length)
```

Sets the unit length (k-mer size, average read length or average fragment length) to use for coverage computation (defaults to: 1).

```
- (RGFA) set_default_count_tag(tag)
```

Sets the count tag to use as default by coverage computations (defaults to: :RC).

Instance Method Details

```
- (RGFA) apply_copy_number(segment, count_tag: :cn, distribute: :auto, copy_names_suffix: :lowercase, origin_tag: :or, conserve_components: true)
```

Applies the computed copy number to a segment

Parameters:

- **copy_names_suffix** (:lowercase, :uppercase, :number, :copy) — (Defaults to: :lowercase)
Symbol representing a system to compute the names from the name of the original segment. See "Automatic computation of the copy names" in #multiply_extended.
- **count_tag** (Symbol) — tag to use for storing the copy number (default: cn)
- **distribute** (Symbol) — distribution policy, see #multiply_extended
- **origin_tag** (Symbol) — tag to use for storing the origin (default: or)
- **conserve_components** (Boolean) — when factor is 0, do not remove segments if doing so increases the number of components in the graph (default: true)
- **segment** (RGFA::Line::Segment, Symbol) — segment or segment name

Returns:

- (RGFA) — self

```
- (RGFA) apply_copy_numbers(count_tag: :cn, distribute: :auto,  
copy_names_suffix: :lowercase, origin_tag: :or, conserve_components: true)
```

Applies the computed copy number to all segments

Parameters:

- **copy_names_suffix** (*Symbol*) — *(Defaults to: :lowercase)* Symbol representing a system to compute the names from the name of the original segment. See "Automatic computation of the copy names" in #multiply_extended.
- **count_tag** (*Symbol*) — tag to use for storing the copy number (default: cn)
- **distribute** (*Symbol*) — distribution policy, see #multiply_extended
- **origin_tag** (*Symbol*) — tag to use for storing the origin (default: or)
- **conserve_components** (*Boolean*) — when factor is 0, do not remove segments if doing so increases the number of components in the graph (default: true)

Returns:

- (RGFA) — self

```
- (RGFA) compute_copy_numbers(single_copy_coverage, mincov:  
single_copy_coverage * 0.25, count_tag: @default[:count_tag], cn_tag: :cn,  
unit_length: @default[:unit_length])
```

Returns self

Parameters:

- **mincov** (*Integer*) — *(defaults to: 1/4 of single_copy_coverage)* the minimum coverage, cn for segments under this value is set to 0
- **single_copy_coverage** (*Integer*) — the coverage that shall be considered to be single copy
- **cn_tag** (*Symbol*) — *(defaults to: :cn)* the tag to use for storing the copy number
- **count_tag** (*Symbol*) — *(defaults to: :RC or the value set by #set_default_count_tag)* the count tag to use for coverage computation
- **unit_length** (*Integer*) — *(defaults to: 1 or the value set by #set_count_unit_length)* the unit length to use for coverage computation

Returns:

- (RGFA) — self

```
- (RGFA) delete_low_coverage_segments(mincov, count_tag: @default[:count_tag],  
unit_length: @default[:unit_length])
```

Delete segments which have a coverage under a specified value.

Parameters:

- **mincov** (*Integer*) — the minimum coverage
- **count_tag** (*Symbol*) — *(defaults to: :RC or the value set by #set_default_count_tag)* the count tag to use for coverage computation
- **unit_length** (*Integer*) — *(defaults to: 1 or the value set by #set_count_unit_length)* the unit length to use for coverage computation

Returns:

- (RGFA) — self

```
- (RGFA) set_count_unit_length(unit_length)
```

Sets the unit length (k-mer size, average read length or average fragment length) to use for coverage computation (*defaults to: 1*).

Parameters:

- `unit_length` (`Integer`) — the unit length to use

Returns:

- (`RGFA`) — self
-

```
- (RGFA) set_default_count_tag(tag)
```

Sets the count tag to use as default by coverage computations (*defaults to: :RC*).

Parameters:

- `tag` (`Symbol`) — the tag to use

Returns:

- (`RGFA`) — self

Module: RGFATools::LinearPaths

Included in:	RGFATools
Defined in:	lib/rgfatools/linear_paths.rb

Overview

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

Constant Summary

Instance Method Summary

(collapse)

– (RGFA) `merge_linear_path(segpath, **options)`

Merge a linear path, i.e.

Instance Method Details

– (RGFA) `merge_linear_path(segpath, **options)`

Merge a linear path, i.e. a path of segments without extra-branches. Extends the RGFA method, with additional functionality:

- `name`: the name of the merged segment is set to the name of the single segments joined by underscore (`_`). If a name already contained an underscore, it is splitted before merging. Whenever a segment is reversed complemented, its name (or the name of all its components) is suffixed with a `^`; if the last letter was already `^`, it is removed; if it contained `_` the name is splitted, the elements reversed and joined back using `_`; round parentheses are removed from the name before processing and added back after it.
- `:or`: keeps track of the origin of the merged segment; the origin tag is set to an array of `:or` or name (if no `:or` available) tags of the segment which have been merged; the character `^` is assigned the same meaning as in `name`
- `:rn`: tag used to store possible inversion positions and it is updated by this method; i.e. it is passed from the single segments to the merged segment, and the coordinates updated
- `:mp`: tag used to store the position of the single segments in the merged segment; it is created or updated by this method

Note that the extensions to the original method will only be run if either `#enable_extensions` has been called on RGFA object or the `enable_tracking` parameter is set.. After calling `#enable_extensions`, you may still obtain the original behaviour by setting the `disable_tracking` parameter.

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` (see RGFA API documentation)
- `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 found; 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 $\text{Sum}(|s| \text{ in } S) / |m|$
- `:enable_tracking` (Boolean) — default: `false` — if true, the extended method with `RGFATools` is called, no matter if `RGFA#enable_extensions` was called.
- `:disable_tracking` (Boolean) — default: `false` — if true, the original method of `RGFA` without `RGFATools` is called, no matter if `RGFA#enable_extensions` was called.

Returns:

- (`RGFA`) — self

See Also:

- `#merge_linear_paths`

Module: RGFATools::Multiplication

Included in:	RGFATools
Defined in:	lib/rgfatools/multiplication.rb

Overview

Methods which edit the graph components without traversal

Constant Summary

LINKS_DISTRIBUTION_POLICY =

Allowed values for the links_distribution_policy option

```
[ :off, :auto, :equal, :E, :B ]
```

Instance Method Summary

[\(collapse\)](#)

```
- (RGFA) multiply_extended(segment, factor, copy_names: :lowercase, distribute: :auto, conserve_components: true, origin_tag: :or)
```

Create multiple copies of a segment.

```
- (RGFA) multiply(segment, factor, copy_names::lowercase, distribute::auto, conserve_components:true, origin_tag::or)
```

Create multiple copies of a segment.

Instance Method Details

```
- (RGFA) multiply_extended(segment, factor, copy_names: :lowercase, distribute: :auto, conserve_components: true, origin_tag: :or)
```

Create multiple copies of a segment.

Complements the multiply method of gfatools with additional functionality. To always run the additional functionality when multiply is called, use `RGFA#enable_extensions`.

Automatic computation of the copy names:

- First, it is checked if the name of the original segment ends with a relevant string, i.e. a lower case letter (for `:lowercase`), an upper case letter (for `:uppercase`), 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 `:lowercase`), A (for `:uppercase`), 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-existent 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.
- Can be overridden, by providing an array of copy names.

Links distribution policy

Depending on the value of the option `distribute`, an end is eventually selected for distribution of the links.

- `:off`: no distribution performed

- `:E`: links of the E end are distributed
- `:B`: links of the B end are distributed
- `:equal`: select an end for which the number of links is equal to `factor`, if any; if both, then the E end is selected
- `:auto`: automatically select E or B, trying to maximize the number of links which can be deleted

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** (`:lowercase`, `:uppercase`, `:number`, `:copy`, `Array<String>`) — (*Defaults to: :lowercase*) 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 `#cut_segment?(segment)` is `false` (see RGFA API).
- **distribute** (`RGFATools::Multiplication::LINKS_DISTRIBUTION_POLICY`) — (*Defaults to: :auto*) Determines if and for which end of the segment, links are distributed among the copies. See "Links distribution policy".
- **origin_tag** (`Symbol`) — (*Defaults to: :or*) Name of the custom tag to use for storing origin information.

Returns:

- (`RGFA`) — self

```
- (RGFA) multiply(segment, factor, copy_names::lowercase, distribute::auto,
conserve_components:true, origin_tag::or)
```

Create multiple copies of a segment.

Complements the `multiply` method of `gfatools` with additional functionality. These extensions are used only after `#enable_extensions` is called on the `RGFA` object. After that, you may still call the original method using `#multiply_without_rgfatools`.

For more information on the additional functionality, see `#multiply_extended`.

Returns:

- (`RGFA`) — self

Module: RGFATools::SuperfluousLinks

Included in:	RGFATools
Defined in:	lib/rgfatools/superfluous_links.rb

Overview

Methods which edit the graph components without traversal

Instance Method Summary

(collapse)

- (RGFA) **enforce_all_mandatory_links**(conserve_components: true)
Remove superfluous links in the presence of mandatory links in the entire graph.
- (RGFA) **enforce_segment_mandatory_links**(segment, conserve_components: true)
Remove superfluous links in the presence of mandatory links for a single segment.
- (RGFA) **remove_self_link**(segment)
Remove links of segment to itself.
- (RGFA) **remove_self_links**
Remove all links of segments to themselves.

Instance Method Details

- (RGFA) **enforce_all_mandatory_links**(conserve_components: true)

Remove superfluous links in the presence of mandatory links in the entire graph

Parameters:

- **conserve_components** (Boolean) — (Defaults to: true) delete links only if #cut_link?(link) is false (see RGFA API).

Returns:

- (RGFA) — self

- (RGFA) **enforce_segment_mandatory_links**(segment, conserve_components: true)

Remove superfluous links in the presence of mandatory links for a single segment

Parameters:

- **segment** (String, RGFA::Line::Segment) — segment name or instance
- **conserve_components** (Boolean) — (Defaults to: true) delete links only if #cut_link?(link) is false (see RGFA API).

Returns:

- (RGFA) — self

- (RGFA) **remove_self_link**(segment)

Remove links of segment to itself

Parameters:

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

Returns:

- (`RGFA`) — self
-

```
- (RGFA) remove_self_links
```

Remove all links of segments to themselves

Returns:

- (`RGFA`) — self

Module: RGFATools::InvertibleSegments

Included in:	RGFATools
Defined in:	lib/rgfatools/invertible_segments.rb

Overview

Methods which edit the graph components without traversal

Instance Method Summary

(collapse)

- (RGFA) `randomly_orient_invertible(segment)`

Selects a random orientation for an invertible segment.

- (RGFA) `randomly_orient_invertibles`

Selects a random orientation for all invertible segments.

Instance Method Details

- (RGFA) `randomly_orient_invertible(segment)`

Selects a random orientation for an invertible segment

Parameters:

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

Returns:

- (RGFA) — self
-

- (RGFA) `randomly_orient_invertibles`

Selects a random orientation for all invertible segments

Returns:

- (RGFA) — self

Class: RGFA

Inherits:	Object show all
Includes:	Connectivity , Containments , Headers , LinearPaths , Lines , Links , LoggerSupport , Multiplication , Paths , RGL , Segments , RGFATools
Defined in:	lib/rgfa.rb

Overview

Main class of the RGFA library.

RGFA provides a representation of a GFA graph. It supports creating a graph from scratch, input and output from/to file or strings, as well as several operations on the graph. The examples below show how to create a RGFA object from scratch or from a GFA file, write the RGFA to file, output the string representation or a statistics report, and control the validation level.

Interacting with the graph

- [Lines](#): module with methods for finding, editing, iterating over, removing lines belonging to a RGFA instance. Specialized modules exist for each kind of line:
 - [Headers](#): accessing and creating header information is done using a single header line object ([RGFA#header](#))
 - [Segments](#)
 - [Links](#)
 - [Containments](#)
 - [Paths](#)
- [Line](#): most interaction with the GFA involve interacting with its record, i.e. instances of a subclass of this class. Subclasses:
 - [Line::Header](#)
 - [Line::Segment](#)
 - [Line::Link](#)
 - [Line::Containment](#)
 - [Line::Path](#)
- Further modules contain methods useful for interacting with the graph
 - [Connectivity](#) analysis of the connectivity of the graph
 - [LinearPaths](#) finding and merging of linear paths
 - [Multiplication](#) separation of the implicit instances of a repeat
- Additional functionality is provided by [RGFATools](#)

Examples:

Creating an empty RGFA object

```
gfa = RGFA.new
```

Parsing and writing GFA format

```
gfa = RGFA.from_file(filename) # parse GFA file
gfa.to_file(filename) # write to GFA file
puts gfa # show GFA representation of RGFA object
```

Basic statistics report

```
puts gfa.info # print report
puts gfa.info(short = true) # compact format, in one line
```

Validation

```
gfa = RGFA.from_file(filename, validate: 1) # default level is 2
gfa.validate = 3 # change validation level
gfa.turn_off_validations # equivalent to gfa.validate = 0
gfa.validate! # run post-validations (e.g. check segment names in links)
```

Defined Under Namespace

Modules: [Connectivity](#), [Containments](#), [FieldParser](#), [FieldValidator](#), [FieldWriter](#), [Headers](#), [LinearPaths](#), [Lines](#), [Links](#), [LoggerSupport](#), [Multiplication](#), [Paths](#), [Segments](#), [Sequence](#)
Classes: [ByteArray](#), [CIGAR](#), [DuplicatedLabelError](#), [Error](#), [FieldArray](#), [Line](#), [LineMissingError](#), [Logger](#), [NumericArray](#), [OrientedSegment](#), [SegmentEnd](#), [SegmentEndsPath](#), [SegmentInfo](#)

Constant Summary

Constant Summary

Constants included from [RGFATools::Multiplication](#)

[RGFATools::Multiplication::LINKS_DISTRIBUTION_POLICY](#)

Instance Attribute Summary

(collapse)

- (Object) **validate**

Returns the value of attribute validate.

Class Method Summary

(collapse)

+ (RGFA) **from_file**(filename, validate: 2)

Creates a RGFA instance parsing the file with specified filename.

Instance Method Summary

(collapse)

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

Compare two RGFA instances.

- (RGFA) **clone**

Create a copy of the RGFA instance.

- (void) **disable_extensions**

Disable [RGFATools](#) extensions of RGFA methods.

- (void) **enable_extensions**

Enable [RGFATools](#) extensions of RGFA methods.

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

Output basic statistics about the graph's sequence and topology information.

- (RGFA) **initialize**(validate: 2)

A new instance of RGFA.

constructor

- (Integer) **n_dead_ends**

Counts the dead ends.

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

List all names of path lines in the graph.

- (self) **read_file**(filename)

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<Symbol>) **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**

Set the validation level to 0.

- (void) **validate!**

Post-validation of the RGFA.

Methods included from [RGFATools::PBubbles](#)

[#remove_p_bubble](#), [#remove_p_bubbles](#)

Methods included from [RGFATools::LinearPaths](#)

[#merge_linear_path](#)

Methods included from [RGFATools::SuperfluousLinks](#)

[#enforce_all_mandatory_links](#), [#enforce_segment_mandatory_links](#),
[#remove_self_link](#), [#remove_self_links](#)

Methods included from [RGFATools::Multiplication](#)

[#multiply_extended](#), [#multiply_with_rgfatools](#)

Methods included from [RGFATools::InvertibleSegments](#)

[#randomly_orient_invertible](#), [#randomly_orient_invertibles](#)

Methods included from [RGFATools::CopyNumber](#)

[#apply_copy_number](#), [#apply_copy_numbers](#), [#compute_copy_numbers](#),
[#delete_low_coverage_segments](#), [#set_count_unit_length](#), [#set_default_count_tag](#)

Methods included from [RGFATools::Artifacts](#)

[#remove_dead_ends](#), [#remove_small_components](#)

Methods included from [LoggerSupport](#)

[#enable_progress_logging](#), [#progress_log](#), [#progress_log_end](#), [#progress_log_init](#)

Methods included from [Multiplication](#)

[#multiply](#)

Methods included from [Connectivity](#)

[#connected_components](#), [#connectivity](#), [#cut_link?](#), [#cut_segment?](#),
[#segment_connected_component](#), [#split_connected_components](#)

Methods included from [LinearPaths](#)

[#linear_path](#), [#linear_paths](#), [#merge_linear_path](#), [#merge_linear_paths](#)

Methods included from [Paths](#)

[#delete_path](#), [#path](#), [#path!](#), [#paths](#), [#paths_with](#)

Methods included from [Containments](#)

```
#contained_in, #containing, #containment, #containment!, #containments,  
#containments_between, #delete_containment
```

Methods included from [Links](#)

```
#delete_link, #delete_other_links, #link, #link!, #link_from_to,  
#link_from_to!, #links, #links_between, #links_from, #links_from_to, #links_of,  
#links_to, #neighbours
```

Methods included from [Segments](#)

```
#connected_segments, #delete_segment, #segment, #segment!, #segments,  
#unconnect_segments
```

Methods included from [Headers](#)

```
#delete_headers, #header, #headers
```

Methods included from [Lines](#)

```
#<<, #rename, #rm
```

Constructor Details

```
- (RGFA) initialize(validate: 2)
```

Returns a new instance of RGFA

Parameters:

- **validate** (Integer) — (defaults to: 2) the validation level; see “Validation level” under [RGFA::Line#initialize](#).

Instance Attribute Details

```
- (Object) validate
```

Returns the value of attribute validate

Class Method Details

```
+ (RGFA) from_file(filename, validate: 2)
```

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

Parameters:

- **filename** (String)
- **validate** (Integer) — (defaults to: 2) the validation level; see “Validation level” under [RGFA::Line#initialize](#).

Returns:

- (RGFA)

Raises:

- if file cannot be opened for reading

Instance Method Details

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

Compare two RGFA instances.

Returns:

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

```
- (RGFA) clone
```

Create a copy of the RGFA instance.

Returns:

- (RGFA)
-

```
- (void) disable_extensions
```

This method returns an undefined value.

Disable [RGFATools](#) extensions of RGFA methods

```
- (void) enable_extensions
```

This method returns an undefined value.

Enable [RGFATools](#) extensions of RGFA methods

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

Output basic statistics about the graph's sequence and topology information.

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 some additional one: the length of the largest component, as well as 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.

Dead ends are here defined as segment ends without connections.

Returns:

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

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

List all names of path lines in the graph

Returns:

- (Array<Symbol>)
-

```
- (self) read_file(filename)
```

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

Parameters:

- `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<Symbol>) segment_names
```

List all names of segments in the graph

Returns:

- (Array<Symbol>)
-

```
- (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.

Set the validation level to 0. See "Validation level" under [RGFA::Line#initialize](#).

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

This method returns an undefined value.

Post-validation of the RGFA

Raises:

- if validation fails

Class: String

Inherits:	Object	show all
Includes:	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::CIGAR) `to_cigar`

Parse CIGAR string and return an array of CIGAR operations.

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

Create a numeric array from a string.

- (RGFA) `to_rgfa`(validate: 2)

Converts a String into a RGFA instance.

- (subclass of RGFA::Line) `to_rgfa_line`(validate: 2)

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_gfa_field!`, `#validate_segment_name!`

Methods included from *RGFA::FieldParser*

`#parse_gfa_field`, `#parse_gfa_optfield`

Methods included from *RGFA::Sequence*

`#rc`

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

Raises:

- (`RGFA::ByteArray::FormatError`) — if the string size is not > 0 and even
-

```
- (RGFA::CIGAR) to_cigar
```

Parse CIGAR string and return an array of CIGAR operations

Returns:

- (`RGFA::CIGAR`) — CIGAR operations (empty if string is "")

Raises:

- (`RGFA::CIGAR::ValueError`) — if the string is not a valid CIGAR string
-

```
- (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: 2)
```

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

Parameters:

- **validate** (`Integer`) — (*defaults to: 2*) the validation level; see "Validation level" under `RGFA::Line#initialize`.

Returns:

- (`RGFA`)
-

```
- (subclass of RGFA::Line) to_rgfa_line(validate: 2)
```

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

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

Parameters:

- **validate** (`Integer`) — (*defaults to: 2*) see `RGFA::Line#initialize`

Returns:

- (subclass of `RGFA::Line`)

Raises:

- (`RGFA::Error`) — if the fields do 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::CIGAR](#), [RGFA::FieldArray](#), [RGFA::NumericArray](#), [RGFA::SegmentEndsPath](#), [RGFA::SegmentInfo](#)

Instance Method Summary

(collapse)

- (RGFA::Line::FIELD_DATATYPE) **default_gfa_datatype** private
Optional field GFA datatype to use, if none is provided.
- (Boolean) **rgfa_field_array?**
Is this possibly a [RGFA::FieldArray](#) instance?.
- (RGFA::ByteArray) **to_byte_array**
Create a [RGFA::ByteArray](#) from an Array instance.
- (RGFA::CIGAR) **to_cigar**
Create a [RGFA::CIGAR](#) instance from the content of the array.
- (RGFA::CIGAR::Operation) **to_cigar_operation**
Create a [RGFA::CIGAR::Operation](#) instance from the content of the array.
- (String) **to_gfa_field**(datatype: [default_gfa_datatype](#)) private
Representation of the data for GFA fields; this method does not (in general) 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: 2)
Converts an Array of strings or [RGFA::Line](#) instances into a [RGFA](#) instance.
- (Object) **to_rgfa_field_array**(datatype = nil)
Create a [RGFA::FieldArray](#) from an array.
- (subclass of RGFA::Line) **to_rgfa_line**(validate: 2)
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.
- (void) **validate_gfa_field!**(datatype, fieldname = nil) private
Validates the object according to the provided datatype.

Instance Method Details


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

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.

Optional field GFA datatype to use, if none is provided

Returns:

- (RGFA::Line::FIELD_DATATYPE)

```
- (Boolean) rgfa_field_array?
```

Is this possibly a `RGFA::FieldArray` instance?

(i.e. are the two last elements a datatype symbol and a zero byte?)

Returns:

- (Boolean)

```
- (RGFA::ByteArray) to_byte_array
```

Create a `RGFA::ByteArray` from an Array instance

Returns:

- (RGFA::ByteArray) — the byte array

```
- (RGFA::CIGAR) to_cigar
```

Create a `RGFA::CIGAR` instance from the content of the array.

Returns:

- (RGFA::CIGAR)

```
- (RGFA::CIGAR::Operation) to_cigar_operation
```

Create a `RGFA::CIGAR::Operation` instance from the content of the array.

Returns:

- (RGFA::CIGAR::Operation)

```
- (String) to_gfa_field(datatype: default_gfa_datatype)
```

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.

Representation of the data for GFA fields; this method does not (in general) validate the string. The method can be overwritten for a given class, and may take the `#default_gfa_datatype` into consideration.

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: 2)
```

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

Parameters:

- **validate** (Integer) — (*defaults to: 2*) the validation level; see "Validation level" under [RGFA::Line#initialize](#).

Returns:

- (RGFA)

```
- (Object) to_rgfa_field_array(datatype = nil)
```

Create a RGFA::FieldArray from an array

Parameters:

- **datatype** (RGFA::Line::OPTFIELD_DATATYPE, nil) (*defaults to: nil*) — the datatype to use

```
- (subclass of RGFA::Line) to_rgfa_line(validate: 2)
```

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` (Integer) — (defaults to: 2) see `RGFA::Line#initialize`

Returns:

- (subclass of `RGFA::Line`)

Raises:

- (`RGFA::Error`) — 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

```
- (void) validate_gfa_field!(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (`RGFA::FieldParser::FormatError`) — if the object type or content is not compatible to the provided datatype

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

[Comment](#), [Containment](#), [Header](#), [Link](#), [Path](#), [Segment](#)

Defined Under Namespace

Classes: [Comment](#), [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 =

List of data types which are parsed only on access; all other are parsed when read.

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

DIRECTION =

Direction of a segment for links/containments

`[:from, :to]`

ORIENTATION =

Orientation of segments in paths/links/containments

`[:+, :-]`

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 a RGFA::Line instance.

- (Object?) **delete**(fieldname)

Remove an optional field from the line, if it exists; do nothing if it does not.

- (String) **field_to_s**(fieldname, optfield: false)

Compute the string representation of a field.

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

Fields defined for this instance.

- (Object?) **get**(fieldname, frozen: false)

Get the value of a field.

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

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

- (RGFA::Line::FIELD_DATATYPE) **get_datatype**(fieldname)

Returns a symbol, which specifies the datatype of a field.

- (RGFA::Line) **initialize**(data, validate: 2, virtual: false)

Constants defined by subclasses .

constructor

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

Methods are dynamically created for non-existing but valid optional field names.

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

Name of the optional fields.

- (Object) **real!**(real_line)

private

Make a virtual line real.

- (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.

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

Set the value of a field.

- (RGFA::Line::FIELD_DATATYPE) **set_datatype**(fieldname, datatype)

Set the datatype of a field.

- (Array<[Symbol, Symbol, Object]>) **tags**

Returns the optional fields as an array of [fieldname, datatype, value] arrays.

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

An array of string representations of the fields.

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

Self.

- (String) **to_s**

A string representation of self.

- (void) **validate!**

Validate the RGFA::Line instance.

- (void) **validate_field!**(fieldname)

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

- (Boolean) **virtual?**

private

Is the line virtual?.

Constructor Details

- (RGFA::Line) **initialize**(data, validate: 2, virtual: false)

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

Validation levels

The default is 2, i.e. if a field content is changed, the user is responsible to call #validate_field!, if necessary.

- 0: no validation
- 1: the number of required fields must be correct; optional fields

cannot be duplicated; custom optional field names must be correct; predefined optional fields must have the correct type; only some fields are validated on initialization or first-time access to the field content

- 2: 1 + all fields are validated on initialization or first-time

access to the field content

- 3: 2 + all fields are validated on initialization and record-specific

validations are run (e.g. compare segment LN tag and sequence lenght)

- 4: 3 + all fields are validated on writing to string
- 5: 4 + all fields are validated by get and set methods

Parameters:

- **data** (`Array<String>`) — the content of the line; if an array of strings, this is interpreted as the splitted content of a GFA file line; note: an hash is also allowed, but this is for internal usage and shall be considered private
- **validate** (`Integer`) — see paragraph Validation
- **virtual** (`Boolean`) — (*default: false*) mark the line as virtual, i.e. not yet found in the GFA file; e.g. a link is allowed to refer to a segment which is not yet created; in this case a segment marked as virtual is created, which is replaced by a non-virtual segment, when the segment line is later found

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)
```

Methods are dynamically created for non-existing but valid optional field names. Methods for predefined optional fields and required fields are created dynamically for each subclass; methods for existing optional fields are created on instance initialization.

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

The parsed content of a field. See also `#get`.

Parameters:

Returns:

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

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

The parsed content of a field, raising an exception if not available. See also `#get!`.

Returns:

- (`String`, `Hash`, `Array`, `Integer`, `Float`) the parsed content of the field

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. See also `#set`, `#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
```

Deep copy of a `RGFA::Line` instance.

Returns:

- (`RGFA::Line`)
-

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

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

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

Parameters:

- **fieldname** (`Symbol`) — the tag name of the optfield to remove

Returns:

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

```
- (String) field_to_s(fieldname, optfield: false)
```

Compute the string representation of a field.

Parameters:

- **fieldname** (Symbol) — the tag name of the field
- **optfield** (Boolean) — (*defaults to: false*) return the tagname:datatype:value representation

Returns:

- (String) — the string representation

Raises:

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

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

Returns fields defined for this instance

Returns:

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

```
- (Object?) get(fieldname, frozen: false)
```

Get the value of a field

Parameters:

- **fieldname** (Symbol) — name of the field
- **frozen** (Boolean) — (*defaults to: false*) return a frozen value; this guarantees that a validation will not be necessary on output if the field value has not been changed using #set

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** (Symbol) — name of the field

Returns:

- (Object, nil) — value of the field

Raises:

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

```
- (RGFA::Line::FIELD_DATATYPE) get_datatype(fieldname)
```

Returns a symbol, which specifies the datatype of a field

Parameters:

- `fieldname` ([Symbol](#)) — the tag name of the field

Returns:

- ([RGFA::Line::FIELD_DATATYPE](#)) — the datatype symbol

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

Returns name of the optional fields

Returns:

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

```
- (Object) real!(real_line)
```

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.

Make a virtual line real. This is called when a line which is expected, and for which a virtual line has been created, is finally found. So the line is converted into a real line, by merging in the line information from the found line.

Parameters:

- `real_line` ([RGFA::Line](#)) — the real line fou

```
- (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](#)

```
- (Object) set(fieldname, value)
```

Set the value of a field.

If a datatype for a new custom optional field is not set, the default for the value assigned to the field will be used (e.g. J for Hashes, i for Integer, etc).

Parameters:

- **fieldname** ([Symbol](#)) — the name of the field to set (required field, predefined optional field (uppercase) or custom optional field name (lowercase))

Returns:

- ([Object](#)) — value

Raises:

- ([RGFA::Line::FieldnameError](#)) — if `fieldname` is not a valid predefined or custom optional name (and `validate`)

```
- (RGFA::Line::FIELD\_DATATYPE) set_datatype(fieldname, datatype)
```

Set the datatype of a field.

If an existing field datatype is changed, its content may become invalid (call `#validate_field!` if necessary).

If the method is used for a required field or a predefined field, the line will use the specified datatype instead of the predefined one, resulting in a potentially invalid line.

Parameters:

- **fieldname** ([Symbol](#)) — the field name (it is not required that the field exists already)
- **datatype** ([RGFA::Line::FIELD_DATATYPE](#)) — the datatype

Returns:

- ([RGFA::Line::FIELD_DATATYPE](#)) — the datatype

Raises:

- ([RGFA::Line::UnknownDatatype](#)) — if `datatype` is not a valid datatype for optional fields

```
- (Array<Symbol, Symbol, Object>) tags
```

Returns the optional fields as an array of [fieldname, datatype, value] arrays.

Returns:

- ([Array](#)<[Symbol](#), [Symbol](#), [Object](#)>)

```
- (Array<String>) to_a
```

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: nil)
```

Returns self

Parameters:

- **validate** ([Boolean](#)) — ignored (compatibility reasons)

Returns:

- `self`

```
- (String) to_s
```

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

```
- (void) validate_field!(fieldname)
```

This method returns an undefined value.

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

Parameters:

- `fieldname` (`Symbol`) — the tag name of the field to validate

Raises:

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

```
- (Boolean) virtual?
```

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.

Is the line virtual?

Is this `RGFA::Line` a virtual line representation (i.e. a placeholder for an expected but not encountered yet line)?

Returns:

- `(Boolean)`

Class: RGFA::CIGAR

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

Overview

Array of [CIGAR operations](#). Represents the contents of a CIGAR string.

Defined Under Namespace

Classes: [Operation](#), [ValueError](#)

Class Method Summary

(collapse)

+ (RGFA::CIGAR) [from_string](#)(str)

Parse a CIGAR string into an array of CIGAR operations.

Instance Method Summary

(collapse)

- (RGFA::CIGAR) [clone](#)

Create a copy.

- (RGFA::CIGAR) [complement](#)

Compute the CIGAR for the segments when these are switched.

- (RGFA::CIGAR) [to_cigar](#)

Self.

- (String) [to_s](#)

String representation of the CIGAR.

- (void) [validate!](#)

Validate the instance.

- (void) [validate_gfa_field!](#)(datatype, fieldname = nil)

private

Validates the object according to the provided datatype.

Methods inherited from [Array](#)

```
#default_gfa_datatype, #rgfa_field_array?, #to_byte_array, #to_cigar_operation,  
#to_gfa_field, #to_numeric_array, #to_oriented_segment, #to_rgfa,  
#to_rgfa_field_array, #to_rgfa_line, #to_segment_end
```

Class Method Details

+ (RGFA::CIGAR) [from_string](#)(str)

Parse a CIGAR string into an array of CIGAR operations.

Each operation is represented by a [Operation](#), i.e. a tuple of operation length and operation symbol (one of MIDNSHPX=).

Returns:

- (RGFA::CIGAR) — (empty if string is *)

Raises:

- (RGFA::CIGAR::ValueError) — if the string is not a valid CIGAR string

Instance Method Details

- (RGFA::CIGAR) **clone**

Create a copy

Returns:

- (RGFA::CIGAR)
-

- (RGFA::CIGAR) **complement**

Compute the CIGAR for the segments when these are switched.

Examples:

Computing the complement CIGAR

```
RGFA::CIGAR.from_string("2M1D3M").complement.to_s
# => "3M1I2M"

# S1 + S2 + 2M1D3M
#
# S1+  ACGACTGTGA
# S2+      CT-TGACGG
#
# S2-  CCGTCA-AG
# S1-      TCACAGTCGT
#
# S2 - S1 - 3M1I2M
```

Returns:

- (RGFA::CIGAR) — (empty if CIGAR string is *)
-

- (RGFA::CIGAR) **to_cigar**

Returns self

Returns:

- (RGFA::CIGAR) — self
-

- (String) **to_s**

String representation of the CIGAR

Returns:

- (String) — CIGAR string
-

- (void) **validate!**

This method returns an undefined value.

Validate the instance

Raises:

- if any component of the CIGAR array is invalid.
-

```
- (void) validate_gfa_field!(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (`RGFA::FieldParser::FormatError`) — if the object type or content is not compatible to the provided datatype

Exception: RGFA::CIGAR::ValueError

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

Overview

Exception raised by invalid CIGAR string content

Class: RGFA::CIGAR::Operation

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

Overview

An operation in a CIGAR string

Constant Summary

CODE =
CIGAR operation code

```
[ :M, :I, :D, :N, :S, :H, :P, :X, :=" ]
```

Instance Attribute Summary

(collapse)

- (Object) **code**
Returns the value of attribute code.
- (Object) **len**
Returns the value of attribute len.

Instance Method Summary

(collapse)

- (Boolean) **==(other)**
Compare two operations.
- (Operation) **initialize(len, code)** **constructor**
A new instance of Operation.
- (RGFA::CIGAR::Operation) **to_cigar_operation**
Self.
- (String) **to_s**
The string representation of the operation.
- (void) **validate!**
Validate the operation.

Constructor Details

```
- (Operation) initialize(len, code)
```

Returns a new instance of Operation

Parameters:

- **len** (Integer) — length of the operation
- **code** (RGFA::CIGAR::Operation::CODE) — code of the operation

Instance Attribute Details

- (Object) `code`

Returns the value of attribute `code`

- (Object) `len`

Returns the value of attribute `len`

Instance Method Details

- (Boolean) `==(other)`

Compare two operations

Returns:

- (Boolean)
-

- (RGFA::CIGAR::Operation) `to_cigar_operation`

Returns self

Returns:

- (RGFA::CIGAR::Operation) — self
-

- (String) `to_s`

The string representation of the operation

Returns:

- (String)
-

- (void) `validate!`

This method returns an undefined value.

Validate the operation

Raises:

- (RGFA::CIGAR::ValueError) — if the code is invalid or the length is not an integer larger than zero

Exception: RGFA::Error

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

Overview

Parent class for library-specific errors

Direct Known Subclasses

[ByteArray::FormatError](#), [ByteArray::ValueError](#), [CIGAR::ValueError](#), [DuplicatedLabelError](#), [FieldArray::Error](#), [FieldArray::TypeMismatchError](#), [FieldParser::FormatError](#), [FieldParser::UnknownDatatypeError](#), [Line::CustomOptfieldNameError](#), [Line::DuplicatedOptfieldNameError](#), [Line::FieldnameError](#), [Line::Path::ListLengthsError](#), [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](#)

Exception: RGFA::DuplicatedLabelError

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

Overview

Exception raised if a label for segment or path is duplicated

Exception: RGFA::LineMissingError

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

Overview

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

Class: RGFA::Logger Private

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

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

Constructor Details

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

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.

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 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.

Disable progress logging

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

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.

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 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.

Output a message

Parameters:

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

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

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.

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 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.

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 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.

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 Private

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

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::Link

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

Overview

A link connects two segments, or a segment to itself.

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](#), [DIRECTION](#), [FIELD_DATATYPE](#), [OPTFIELD_DATATYPE](#),
[ORIENTATION](#), [RECORD_TYPES](#), [RECORD_TYPE_LABELS](#), [REQFIELD_DATATYPE](#), [SEPARATOR](#)

Instance Method Summary

[\(collapse\)](#)

- (Boolean) **canonical?**

Returns true if the link is canonical, false otherwise.

- (RGFA::Line::Link) **canonize!**

Returns the unchanged link if the link is canonical, otherwise complements the link and returns it.

- (Boolean) **circular?**

Is the from and to segments are equal.

- (Boolean) **circular_same_end?**

Is the from and to segments are equal.

- (Boolean) **compatible?**(other_oriented_from, other_oriented_to, other_overlap
= [], equivalent = true)

Compares a link and optionally the complement link, with two oriented_segments and optionally an

overlap.

- (Boolean) **compatible_complement?**(other_oriented_from, other_oriented_to, other_overlap = [])

Compares the complement link with two oriented segments and optionally an overlap.

- (Boolean) **compatible_direct?**(other_oriented_from, other_oriented_to, other_overlap = [])

Compares a link with two oriented segments and optionally an overlap.

- (RGFA::Line::Link) **complement**

Creates the equivalent link with from/to inverted.

- (RGFA::Line::Link) **complement!**

Complements the link inplace, i.e.

- (Boolean) **complement?**(other)

Compares the link to the complement of another link and determine their equivalence.

- (RGFA::CIGAR) **complement_overlap**

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

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

Compares two links and determine their equivalence.

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

Compares the optional fields of two links.

- (RGFA::SegmentEnd) **from_end**

The segment end represented by the from/from_orient fields.

- (Symbol) **from_name**

The from segment name, in both cases where from is a segment name (Symbol) or a segment (RGFA::Line::Segment).

- (Object) **hash**

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

- (RGFA::OrientedSegment) **oriented_from**

The oriented segment represented by the from/from_orient fields.

- (RGFA::OrientedSegment) **oriented_to**

The oriented segment represented by the to/to_orient fields.

- (Symbol) **other**(segment)

The other segment of a link.

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

The other segment end.

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

Paths for which the link is required.

- (Boolean) **same?**(other)

Compares two links and determine their equivalence.

- (Object) **segment_ends_s** private

Signature of the segment ends, for debugging.

- (RGFA::SegmentEnd) **to_end**

The segment end represented by the to/to_orient fields.

- (Symbol) **to_name**

The to segment name, in both cases where to is a segment name (Symbol) or a segment (RGFA::Line::Segment).

Methods inherited from [RGFA::Line](#)

`#==, #clone, #delete, #field_to_s, #fieldnames, #get, #get!, #get_datatype,`

```
#initialize, #method_missing, #optional_fieldnames, #real!, #record_type,  
#required_fieldnames, #respond_to?, #set, #set_datatype, subclass, #tags,  
#to_a, #to_rgfa_line, #to_s, #validate!, #validate_field!, #virtual?
```

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) **canonical?**

Returns true if the link is canonical, false otherwise

Definition of canonical link

A link is canonical if:

- from != to and from < to (lexicographically); or
- from == to and at least one of from_orient or to_orient is +

Details

In the special case in which from == to (== s) we have the following equivalences:

```
s + s + == s - s -  
s - s - == s + s + (same as previous case)  
s + s - == s + s - (equivalent to itself)  
s - s + == s - s + (equivalent to itself)
```

Considering the values on the left, the first one can be taken as canonical, the second not, because it can be transformed in the first one; the other two values are canonical, as they are only equivalent to themselves.

Returns:

- (Boolean)
-

- ([RGFA::Line::Link](#)) **canonicalize!**

Note: The path references are not corrected by this method; therefore the method shall be used before the link is embedded in a graph.

Returns the unchanged link if the link is canonical, otherwise complements the link and returns it.

Returns:

- ([RGFA::Line::Link](#)) — self
-

- (Boolean) **circular?**

Returns is the from and to segments are equal

Returns:

- (Boolean) — is the from and to segments are equal

```
- (Boolean) circular_same_end?
```

Returns is the from and to segments are equal

Returns:

- (Boolean) — is the from and to segments are equal

```
- (Boolean) compatible?(other_oriented_from, other_oriented_to, other_overlap = [], equivalent = true)
```

Compares a link and optionally the complement link, with two oriented_segments and optionally an overlap.

Parameters:

- **other_oriented_from** (RGFA::OrientedSegment)
- **other_oriented_to** (RGFA::OrientedSegment)
- **equivalent** (Boolean) (*defaults to: true*) — shall the complement link also be considered?
- **other_overlap** (RGFA::CIGAR) (*defaults to: []*) — compared only if not empty

Returns:

- (Boolean) — does the link or, if *equivalent*, the complement link go from the first oriented segment to the second with an overlap equal to the provided one (if not empty)?

```
- (Boolean) compatible_complement?(other_oriented_from, other_oriented_to, other_overlap = [])
```

Compares the complement link with two oriented segments and optionally an overlap.

Parameters:

- **other_oriented_from** (RGFA::OrientedSegment)
- **other_oriented_to** (RGFA::OrientedSegment)
- **other_overlap** (RGFA::CIGAR) (*defaults to: []*) — compared only if not empty

Returns:

- (Boolean) — does the complement link go from the first oriented segment to the second with an overlap equal to the provided one (if not empty)?

```
- (Boolean) compatible_direct?(other_oriented_from, other_oriented_to, other_overlap = [])
```

Compares a link with two oriented segments and optionally an overlap.

Parameters:

- **other_oriented_from** (RGFA::OrientedSegment)
- **other_oriented_to** (RGFA::OrientedSegment)
- **other_overlap** (RGFA::CIGAR) (*defaults to: []*) — compared only if not empty

Returns:

- (Boolean) — does the link go from the first oriented segment to the second with an overlap

equal to the provided one (if not empty)?

```
- (RGFA::Line::Link) complement
```

Note: The path references are not copied to the complement link.

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

Creates the equivalent link with from/to inverted.

The CIGAR operations (order/type) are inverted as well. Optional fields are left unchanged.

Returns:

- (RGFA::Line::Link) — the inverted link.

```
- (RGFA::Line::Link) complement!
```

Note: The path references are not complemented by this method; therefore the method shall be used before the link is embedded in a graph.

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

Complements the link inplace, i.e. sets:

```
from = to
from_orient = other_orient(to_orient)
to = from
to_orient = other_orient(from_orient)
overlap = complement_overlap.
```

The optional fields are left unchanged.

Returns:

- (RGFA::Line::Link) — self

```
- (Boolean) complement?(other)
```

Compares the link to the complement of another link and determine their equivalence. Thereby, optional fields are not considered.

Parameters:

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

Returns:

- (Boolean) — are self and the complement of other equivalent?

See Also:

- [#eql?](#)
 - [#same?](#)
 - [RGFA::Line#==](#)
-

- (RGFA::CIGAR) complement_overlap

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

Returns:

▪ (RGFA::CIGAR)

- (Boolean) eql?(other)

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

- [RGFA::Line#==](#)
 - [#same?](#)
 - [#complement?](#)
-

- (Boolean) eql_optional?(other)

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

Compares the optional fields of two links.

Parameters:

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

Returns:

▪ (Boolean) — are self and other equivalent?

See Also:

- [RGFA::Line#==](#)
-

- (RGFA::SegmentEnd) from_end

Returns the segment end represented by the from/from_orient fields

Returns:

▪ (RGFA::SegmentEnd) — the segment end represented by the from/from_orient fields

- (Symbol) from_name

The from segment name, in both cases where from is a segment name (Symbol) or a

segment (RGFA::Line::Segment)

Returns:

- (Symbol)

- (Object) hash

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

See Also:

- #eq?

- (RGFA::OrientedSegment) oriented_from

Returns the oriented segment represented by the from/from_orient fields

Returns:

- (RGFA::OrientedSegment) — the oriented segment represented by the from/from_orient fields

- (RGFA::OrientedSegment) oriented_to

Returns the oriented segment represented by the to/to_orient fields

Returns:

- (RGFA::OrientedSegment) — the oriented segment represented by the to/to_orient fields

- (Symbol) other(segment)

The other segment of a link

Parameters:

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

Returns:

- (Symbol) — the name of the other segment of the link if circular, then segment

Raises:

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

- (RGFA::SegmentEnd) other_end(segment_end)

Returns the other segment end

Parameters:

- segment_end (RGFA::SegmentEnd) — one of the two segment ends of the link

Returns:

- (RGFA::SegmentEnd) — 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

```
- (Array<Array<(RGFA::Line::Path, Boolean)>>) paths
```

Paths for which the link is required.

The return value is an empty array if the link is not embedded in a graph.

Otherwise, an array of tuples path/boolean is returned. The boolean value tells if the link is used (true) or its complement (false) in the path.

Returns:

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

```
- (Boolean) same?(other)
```

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:

- [#eq?](#)
- [#complement?](#)
- [RGFA::Line#==](#)

```
- (Object) segment_ends_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.

Signature of the segment ends, for debugging

```
- (RGFA::SegmentEnd) to_end
```

Returns the segment end represented by the to/to_orient fields

Returns:

- (RGFA::SegmentEnd) — the segment end represented by the to/to_orient fields

```
- (Symbol) to_name
```

The to segment name, in both cases where to is a segment name (Symbol) or a segment (RGFA::Line::Segment)

Returns:

- (Symbol)

Class: RGFA::Line::Path

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

Overview

A path line of a RGFA file

Defined Under Namespace

Classes: [ListLengthsError](#)

Constant Summary

RECORD_TYPE =

`:P`

REQFIELDS =

`[:path_name, :segment_names, :overlaps]`

PREDEFINED_OPTFIELDS =

`[]`

DATATYPE =

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

Constants inherited from [RGFA::Line](#)

[DELAYED_PARSING_DATATYPES](#), [DIRECTION](#), [FIELD_DATATYPE](#), [OPTFIELD_DATATYPE](#),
[ORIENTATION](#), [RECORD_TYPES](#), [RECORD_TYPE_LABELS](#), [REQFIELD_DATATYPE](#), [SEPARATOR](#)

Instance Method Summary

(collapse)

– (Boolean) [circular?](#)

Is the path circular? In this case the number of CIGARs must be equal to the number of segments.

– (Boolean) [linear?](#)

Is the path linear? This is the case when the number of CIGARs is equal to the number of segments minus 1, or the CIGARs are represented by a single “*”.

– (Array<RGFA::Line::Link, Boolean>) [links](#)

The links to which the path refers; it can be an empty array (e.g. from a line which is not embedded in a graph); the boolean is true if the equivalent reverse link is used.

– (Array<[RGFA::OrientedSegment, RGFA::OrientedSegment, RGFA::Cigar]>) [required_links](#)

computes the list of links which are required to support the path.

- (Symbol) **to_sym**

Name of the path as symbol.

- (Boolean) **undef_overlaps?**

Are the overlaps a single "*" ? This is a compact representation of a linear path where all CIGARs are "*" .

Methods inherited from *RGFA::Line*

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

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) **circular?**

Is the path circular? In this case the number of CIGARs must be equal to the number of segments.

Returns:

▪ (Boolean)

- (Boolean) **linear?**

Is the path linear? This is the case when the number of CIGARs is equal to the number of segments minus 1, or the CIGARs are represented by a single "*" .

Returns:

▪ (Boolean)

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

The links to which the path refers; it can be an empty array (e.g. from a line which is not embedded in a graph); the boolean is true if the equivalent reverse link is used.

Returns:

▪ (Array<*RGFA::Line::Link*, Boolean>)

- (Array<[*RGFA::OrientedSegment*, *RGFA::OrientedSegment*, *RGFA::Cigar*]>) **required_links**

computes the list of links which are required to support the path

Returns:

- (`Array<RGFA::OrientedSegment, RGFA::OrientedSegment, RGFA::Cigar>`) — an array, which elements are 3-tuples (from oriented segment, to oriented segment, cigar)
-

- (`Symbol`) `to_sym`

Returns name of the path as symbol

Returns:

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

- (`Boolean`) `undef_overlaps?`

Are the overlaps a single "*" ? This is a compact representation of a linear path where all CIGARs are "*"

Returns:

- (`Boolean`)

Exception: RGFA::Line::Path::ListLengthsError

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

Overview

Error raised if number of segments and overlaps are not consistent

Class: RGFA::ByteArray

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

Overview

Array of positive integers ≤ 255 ; representation of the data contained in an H field

Defined Under Namespace

Classes: [FormatError](#), [ValueError](#)

Instance Method Summary

(collapse)

- (RGFA::Line::FIELD_DATATYPE) **default_gfa_datatype** private

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

- (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.

- (void) **validate_gfa_field!**(datatype, fieldname = nil) private

Validates the object according to the provided datatype.

Methods inherited from [Array](#)

```
#rgfa_field_array?, #to_cigar, #to_cigar_operation, #to_gfa_field,  
#to_numeric_array, #to_oriented_segment, #to_rgfa, #to_rgfa_field_array,  
#to_rgfa_line, #to_segment_end
```

Instance Method Details

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

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.

Optional field GFA datatype to use, if none is provided

Returns:

- (RGFA::Line::FIELD_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

```
- (void) validate_gfa_field!(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (`RGFA::FieldParser::FormatError`) — if the object type or content is not compatible to the provided datatype

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

Exception: RGFA::ByteArray::FormatError

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

Overview

Exception raised if string is not a valid representation of byte array

Class: RGFA::FieldArray

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

Overview

Array representing multiple values of the same tag in different header lines

Defined Under Namespace

Classes: [Error](#), [TypeMismatchError](#)

Instance Attribute Summary

(collapse)

- (Object) **datatype** readonly
Returns the value of attribute datatype.

Instance Method Summary

(collapse)

- (Object) **default_gfa_datatype** private
Default datatype, in this case :J.
- (FieldArray) **initialize**(datatype, data = []) constructor
A new instance of FieldArray.
- (Object) **push_with_validation**(value, type, fieldname = nil)
Add a value to the array and validate.
- (Object) **to_gfa_field**(datatype: nil) private
Representation of the field array as JSON array, with two additional values: the datatype and a zero byte as "signature".
- (Object) **validate_gfa_field!**(datatype, fieldname = nil)
Run a datatype-specific validation on each element of the array.

Methods inherited from [Array](#)

```
#rgfa_field_array?, #to_byte_array, #to_cigar, #to_cigar_operation,  
#to_numeric_array, #to_oriented_segment, #to_rgfa, #to_rgfa_field_array,  
#to_rgfa_line, #to_segment_end
```

Constructor Details

- (FieldArray) **initialize**(datatype, data = [])

Returns a new instance of FieldArray

Parameters:

- **datatype** ([RGFA::Line::OPTFIELD_DATATYPE](#)) — the datatype to use

Instance Attribute Details

```
- (Object) datatype (readonly)
```

Returns the value of attribute datatype

Instance Method Details

```
- (Object) default_gfa_datatype
```

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.

Default datatype, in this case :J

```
- (Object) push_with_validation(value, type, fieldname = nil)
```

Add a value to the array and validate

Parameters:

- **value** (Object) — the value to add
- **type** (RGFA::Line::OPTFIELD_DATATYPE, nil) — the datatype to use; if not nil, it will be checked that the specified datatype is the same as for previous elements of the field array; if nil, the value will be validated, according to the datatype specified on field array creation
- **fieldname** (Symbol) (defaults to: nil) — the field name to use for error messages

Raises:

- (RGFA::FieldArray::TypeMismatchError) — if the type of the new value does not correspond to the type of existing values

```
- (Object) to_gfa_field(datatype: 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.

Representation of the field array as JSON array, with two additional values: the datatype and a zero byte as "signature".

Parameters:

- **datatype** (RGFA::Line::OPTFIELD_DATATYPE) — (ignored, J is always used)

```
- (Object) validate_gfa_field!(datatype, fieldname = nil)
```

Run a datatype-specific validation on each element of the array

Parameters:

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

Exception: RGFA::FieldArray::Error

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

Overview

Generic error associated with field arrays

Exception: RGFA::FieldArray::TypeMismatchError

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

Overview

Error raised when trying to add elements with a wrong datatype

Class: RGFA::Line::Header

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

Overview

A header line of a RGFA file

For examples on how to set the header data, see [Headers](#).

See Also:

- [RGFA::Line](#)

Constant Summary

RECORD_TYPE =

`:H`

REQFIELDS =

`[]`

PREDEFINED_OPTFIELDS =

`[:VN]`

DATATYPE =

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

Constants inherited from [RGFA::Line](#)

`DELAYED_PARSING_DATATYPES`, `DIRECTION`, `FIELD_DATATYPE`, `OPTFIELD_DATATYPE`,
`ORIENTATION`, `RECORD_TYPES`, `RECORD_TYPE_LABELS`, `REQFIELD_DATATYPE`, `SEPARATOR`

Instance Method Summary

[\(collapse\)](#)

- (Object) **add**(fieldname, value, datatype = nil)

Set a header value (multi-value compatible).

- (self) **merge**(gfa_line) private

Merge an additional [Header](#) line into this header line.

- (Array<RGFA::Line::Header>) **split** private

Split the header line into single-tag lines.

- (Array<(Symbol, Symbol, Object)>) **tags** private

Array of optional tags data.

Methods inherited from [RGFA::Line](#)

`#==`, `#clone`, `#delete`, `#field_to_s`, `#fieldnames`, `#get`, `#get!`, `#get_datatype`,
`#initialize`, `#method_missing`, `#optional_fieldnames`, `#real!`, `#record_type`,
`#required_fieldnames`, `#respond_to?`, `#set`, `#set_datatype`, `subclass`, `#to_a`,

`#to_rgfa_line, #to_s, #validate!, #validate_field!, #virtual?`

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

```
- (Object) add(fieldname, value, datatype = nil)
```

Set a header value (multi-value compatible).

If a field does not exist yet, set it to value. If it exists and it is a [FieldArray](#), add the value to the field array. If it exists and it is not a field array, create a field array with the previous value and the new one

Parameters:

- **fieldname** ([Symbol](#))
- **value** ([Object](#))
- **datatype** ([RGFA::Line::OPTFIELD_DATATYPE](#), nil) (*defaults to: nil*) — the datatype to use; the default is to determine the datatype according to the value or the previous values present in the field

```
- (self) merge(gfa_line)
```

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.

Merge an additional [RGFA::Line::Header](#) line into this header line.

Parameters:

- **gfa_line** ([RGFA::Line::Header](#)) — the header line to merge

Returns:

- (self)

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

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.

Split the header line into single-tag lines.

If a tag is a [FieldArray](#), this is splitted into multiple fields with the same fieldname.

Returns:

- ([Array<RGFA::Line::Header>](#))

- (`Array<(Symbol, Symbol, Object)>`) **tags**

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.

Array of optional tags data.

Returns the optional fields as an array of [fieldname, datatype, value] arrays. If a field is a `FieldArray`, this is splitted into multiple fields with the same fieldname.

Returns:

- (`Array<(Symbol, Symbol, Object)>`)

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)

+ (Symbol) **invert**(attribute) private
The other attribute value.

Instance Method Summary (collapse)

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

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

- (Symbol) **attribute** private
The attribute.

- (Symbol) **attribute=**(value) private
Set the attribute.

- (Symbol) **attribute_inverted** private
The other possible value of the attribute.

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

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

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

- (Object) **segment=**(value) private
Set the segment.

- (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*

```
#default_gfa_datatype, #rgfa_field_array?, #to_byte_array, #to_cigar,  
#to_cigar_operation, #to_gfa_field, #to_numeric_array, #to_oriented_segment,  
#to_rgfa, #to_rgfa_field_array, #to_rgfa_line, #to_segment_end,  
#validate_gfa_field!
```

Class Method Details

+ (Symbol) `invert`(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` (Symbol) — an attribute value

Returns:

- (Symbol) — 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)

- (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)`

- `(Symbol)` `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 attribute

Returns:

- `(Symbol)` — the attribute

- `(Symbol)` `attribute=(value)`

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.

Set the attribute

Parameters:

- `value (Symbol)` — the attribute

Returns:

- `(Symbol)` — value

- `(Symbol)` `attribute_inverted`

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 possible value of the attribute

Returns:

- `(Symbol)` — the other possible value of the attribute

- `(RGFA::SegmentInfo)` `invert_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 same segment, inverted attribute

Returns:

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

- `(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

```
- (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

```
- (Object) segment=(value)
```

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.

Set the segment

Parameters:

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

Returns:

- `Symbol`, `RGFA::Line::Segment`] `value`

```
- (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, #attribute=, #attribute_inverted, invert,  
#invert_attribute, #name, #segment, #segment=, #to_s, #to_sym, #validate!
```

Methods inherited from [Array](#)

```
#default_gfa_datatype, #rgfa_field_array?, #to_byte_array, #to_cigar,  
#to_cigar_operation, #to_gfa_field, #to_numeric_array, #to_oriented_segment,  
#to_rgfa, #to_rgfa_field_array, #to_rgfa_line, #to_segment_end,  
#validate_gfa_field!
```

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, #attribute=, #attribute_inverted, invert,  
#invert_attribute, #name, #segment, #segment=, #to_s, #to_sym, #validate!
```

Methods inherited from [Array](#)

```
#default_gfa_datatype, #rgfa_field_array?, #to_byte_array, #to_cigar,  
#to_cigar_operation, #to_gfa_field, #to_numeric_array, #to_oriented_segment,  
#to_rgfa, #to_rgfa_field_array, #to_rgfa_line, #to_segment_end,  
#validate_gfa_field!
```

Class: Object

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

Instance Method Summary

(collapse)

- (void) **validate_gfa_field!**(datatype, fieldname = nil)

private

Validates the object according to the provided datatype.

Methods included from [RGFA::FieldWriter](#)

[#default_gfa_datatype](#), [#to_gfa_field](#), [#to_gfa_optfield](#)

Instance Method Details

- (void) **validate_gfa_field!**(datatype, fieldname = 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.

Validates the object according to the provided datatype

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

- Raises:**
- ([RGFA::FieldParser::FormatError](#)) — if the object type or content is not compatible to the provided datatype

Class: Fixnum

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

Instance Method Summary

(collapse)

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

private

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

- (void) **validate_gfa_field!**(datatype, fieldname = nil)

private

Validates the object according to the provided datatype.

Instance Method Details

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

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.

Optional field GFA datatype to use, if none is provided

Returns:

- (RGFA::Line::FIELD_DATATYPE)

- (void) **validate_gfa_field!**(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (RGFA::FieldParser::FormatError) — if the object type or content is not compatible to the provided datatype

Class: Float

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

Instance Method Summary

(collapse)

- (RGFA::Line::FIELD_DATATYPE) **default_gfa_datatype** private
Optional field GFA datatype to use, if none is provided.

- (void) **validate_gfa_field!**(datatype, fieldname = nil) private
Validates the object according to the provided datatype.

Instance Method Details

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

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.

Optional field GFA datatype to use, if none is provided

Returns:

- (RGFA::Line::FIELD_DATATYPE)

- (void) **validate_gfa_field!**(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (RGFA::FieldParser::FormatError) — if the object type or content is not compatible to the provided datatype

Class: Hash

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

Instance Method Summary

(collapse)

- (RGFA::Line::FIELD_DATATYPE) **default_gfa_datatype** private
Optional field GFA datatype to use, if none is provided.
- (String) **to_gfa_field**(datatype: nil) private
Representation of the data for GFA fields; this method does not (in general) validate the string.
- (void) **validate_gfa_field!**(datatype, fieldname = nil) private
Validates the object according to the provided datatype.

Instance Method Details

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

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.

Optional field GFA datatype to use, if none is provided

Returns:

- (RGFA::Line::FIELD_DATATYPE)

- (String) **to_gfa_field**(datatype: 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.

Representation of the data for GFA fields; this method does not (in general) validate the string. The method can be overwritten for a given class, and may take the `#default_gfa_datatype` into consideration.

Returns:

- (String)

- (void) **validate_gfa_field!**(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (`RGFA::FieldParser::FormatError`) — if the object type or content is not compatible to the provided datatype

Class: RGFA::NumericArray

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

Overview

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

Defined Under Namespace

Classes: [TypeError](#), [ValueError](#)

Constant Summary

SIGNED_INT_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

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

INT_SUBTYPE =
Subtypes for integers

```
UNSIGNED_INT_SUBTYPE + SIGNED_INT_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
end
```


Class Method Summary

[\(collapse\)](#)

+ (RGFA::NumericArray::INT_SUBTYPE) **integer_type**(range)

Computes the subtype for integers in a given range.

Instance Method Summary

[\(collapse\)](#)

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

Computes the subtype of the array from its content.

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

private

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

- (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.

- (void) **validate_gfa_field!**(datatype, fieldname = nil)

private

Validates the object according to the provided datatype.

Methods inherited from [Array](#)

#rgfa_field_array?, #to_byte_array, #to_cigar, #to_cigar_operation,
#to_gfa_field, #to_oriented_segment, #to_rgfa, #to_rgfa_field_array,
#to_rgfa_line, #to_segment_end

Class Method Details

+ (RGFA::NumericArray::INT_SUBTYPE) **integer_type**(range)

Computes the subtype for integers in a given range.

If all elements are non-negative, an unsigned subtype is selected, otherwise a signed subtype.

Parameters:

- **range** (Range) — the integer range

Returns:

- (RGFA::NumericArray::INT_SUBTYPE) — subtype code

Raises:

- (RGFA::NumericArray::ValueError) — if the integer range is outside all subtype ranges

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

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

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.

Optional field GFA datatype to use, if none is provided

Returns:

- (`RGFA::Line::FIELD_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

```
- (void) validate_gfa_field!(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (`RGFA::FieldParser::FormatError`) — if the object type or content is not compatible to the provided datatype

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, :SH, :UR]

DATATYPE =

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

Constants inherited from [RGFA::Line](#)

[DELAYED_PARSING_DATATYPES](#), [DIRECTION](#), [FIELD_DATATYPE](#), [OPTFIELD_DATATYPE](#),
[ORIENTATION](#), [RECORD_TYPES](#), [RECORD_TYPE_LABELS](#), [REQFIELD_DATATYPE](#), [SEPARATOR](#)

Instance Attribute Summary

(collapse)

- (Hash{RGFA::Line::DIRECTION => Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Containment>}}) **containments**

References to the containments in which the segment is involved.

- (Hash{RGFA::Line::DIRECTION => Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Link>}}) **links**

References to the links in which the segment is involved.

- (Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Path>}}) **paths**

References to the containments in which the segment is involved.

Instance Method Summary

(collapse)

- (Object) **all_connections**
All links and containments where the segment is involved.
- (Object) **all_containments**
All containments where a segment is involved.
- (Object) **all_links**
All links where the segment is involved.
- (Object) **all_paths**
All paths where the segment is involved.
- (Object) **all_references**
All paths, links and containments where the segment is involved.
- (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!**
- (String) **to_gfa_field**(datatype: nil) private
Representation of the data for GFA fields; this method does not (in general) validate the string.
- (Object) **to_s**(without_sequence: false)
String representation of the segment.
- (Symbol) **to_sym**
Name of the segment as symbol.
- (void) **validate_gfa_field!**(datatype, fieldname = nil) private
Validates the object according to the provided datatype.
- (Object) **validate_length!**

Methods inherited from [RGFA::Line](#)

```
#==, #clone, #delete, #field_to_s, #fieldnames, #get, #get!, #get_datatype,
#initialize, #method_missing, #optional_fieldnames, #real!, #record_type,
#required_fieldnames, #respond_to?, #set, #set_datatype, subclass, #tags,
#to_a, #to_rgfa_line, #validate!, #validate_field!, #virtual?
```

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 Attribute Details

```
- (Hash{RGFA::Line::DIRECTION => Hash{RGFA::Line::ORIENTATION =>
Array<RGFA::Line::Containment>}}) containments
```

References to the containments in which the segment is involved. The references are

in four arrays which are accessed from a nested hash table. The first key is the direction (from or to), the second is the orientation (+ or -).

Examples:

```
segment.containments[:from][:+]
```

Returns:

- (Hash{RGFA::Line::DIRECTION => Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Containment>}})

```
- (Hash{RGFA::Line::DIRECTION => Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Link>}}) links
```

References to the links in which the segment is involved.

The references are in four arrays which are accessed from a nested hash table. The first key is the direction (from or to), the second is the orientation (+ or -).

Examples:

```
segment.links[:from][:+]
```

Returns:

- (Hash{RGFA::Line::DIRECTION => Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Link>}})

```
- (Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Path>}) paths
```

References to the containments in which the segment is involved.

The references are in two arrays which are accessed from a hash table. The key is the orientation (+ or -).

Examples:

```
segment.paths[:+]
```

Returns:

- (Hash{RGFA::Line::ORIENTATION => Array<RGFA::Line::Path>})

Instance Method Details

```
- (Object) all_connections
```

Note: the list shall be considered read-only, as this is a copy of the original arrays of references, concatenated to each other.

All links and containments where the segment is involved.

```
- (Object) all_containments
```

Note: the list shall be considered read-only, as this is a copy of the original arrays of references, concatenated to each other.

All containments where a segment is involved.

- (Object) **all_links**

Note: the list shall be considered read-only, as this is a copy of the original arrays of references, concatenated to each other.

All links where the segment is involved.

- (Object) **all_paths**

Note: the list shall be considered read-only, as this is a copy of the original arrays of references, concatenated to each other.

All paths where the segment is involved.

- (Object) **all_references**

Note: the list shall be considered read-only, as this is a copy of the original arrays of references, concatenated to each other.

All paths, links and containments where the segment is involved.

- (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](#)

```
- (String) to_gfa_field(datatype: 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.

Representation of the data for GFA fields; this method does not (in general) validate the string. The method can be overwritten for a given class, and may take the `#default_gfa_datatype` into consideration.

Returns:

- `(String)`

```
- (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
-

```
- (void) validate_gfa_field!(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (RGFA::FieldParser::FormatError) — if the object type or content is not compatible to the provided datatype
-

```
- (Object) validate_length!
```

Raises:

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

Class: RGFA::Line::Comment

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

Overview

A comment line of a RGFA file

Constant Summary

```
RECORD_TYPE =  
  #  
  
REQFIELDS =  
  [:content]  
  
PREDEFINED_OPTFIELDS =  
  []  
  
DATATYPE =  
  {  
    :content => :cmt,  
  }
```

Constants inherited from [RGFA::Line](#)

```
DELAYED_PARSING_DATATYPES, DIRECTION, FIELD_DATATYPE, OPTFIELD_DATATYPE,  
ORIENTATION, RECORD_TYPES, RECORD_TYPE_LABELS, REQFIELD_DATATYPE, SEPARATOR
```

Method Summary

Methods inherited from [RGFA::Line](#)

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

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::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::FieldParser::FormatError

Private

Inherits:	Error	show all
Defined in:	lib/rgfa/field_parser.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 field content has an invalid format

Exception: RGFA::FieldParser::UnknownDatatypeError

Private

Inherits:	Error	show all
Defined in:	lib/rgfa/field_parser.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 datatype symbol is used

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

Class: Symbol

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

Instance Method Summary

[\(collapse\)](#)

- (void) **validate_gfa_field!**(datatype, fieldname = nil)

private

Validates the object according to the provided datatype.

Instance Method Details

- (void) **validate_gfa_field!**(datatype, fieldname = 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.

Validates the object according to the provided datatype

Parameters:

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

Raises:

- (`RGFA::FieldParser::FormatError`) — if the object type or content is not compatible to the provided datatype

Class: RGFA::Line::Containment

Inherits:	RGFA::Line	show all
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](#), [DIRECTION](#), [FIELD_DATATYPE](#), [OPTFIELD_DATATYPE](#),
[ORIENTATION](#), [RECORD_TYPES](#), [RECORD_TYPE_LABELS](#), [REQFIELD_DATATYPE](#), [SEPARATOR](#)

Instance Method Summary

(collapse)

- (Boolean) **canonical?**

Returns true if the containment is canonical, false otherwise.

- (Symbol) **from_name**

The from segment name, in both cases where from is a segment name (Symbol) or a segment (RGFA::Line::Segment).

- (RGFA::OrientedSegment) **oriented_from**

The oriented segment represented by the from/from_orient fields.

- (RGFA::OrientedSegment) **oriented_to**

The oriented segment represented by the to/to_orient fields.

- (Integer?) **rpos**

The rightmost 0-based coordinate of the contained sequence in the container; nil if the overlap is unspecified.

- (Symbol) **to_name**

The to segment name, in both cases where to is a segment name (Symbol) or a segment (RGFA::Line::Segment).

Methods inherited from [RGFA::Line](#)

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

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) **canonical?**

Returns true if the containment is canonical, false otherwise

Definition of canonical containment

A containment is canonical if the from orientation is +

Details

Each containment has an equivalent complement containment. Consider a containment of B (length:8) in A (length:100) at position 9 of A with a cigar 1M1I2M3D4M (i.e. rpos = 19).

```
A+ B+ 1M1I2M3D4M 9 == A- B- 4M3D2M1I1M 80
A+ B- 1M1I2M3D4M 9 == A- B+ 4M3D2M1I1M 80
A- B+ 1M1I2M3D4M 9 == A+ B- 4M3D2M1I1M 80
A- B- 1M1I2M3D4M 9 == A+ B+ 4M3D2M1I1M 80
```

Pos in the complement is equal to the length of A minus the right pos of B before reversing.

We require here that $A \neq B$ as $A == B$ makes no sense for containments. Thus it is always possible to express the containment using a positive from orientation.

For this reason the canon is simply defined as + from orientation.

Returns:

- (Boolean)

- (Symbol) **from_name**

The from segment name, in both cases where from is a segment name (Symbol) or a segment (RGFA::Line::Segment)

Returns:

- (Symbol)

- (RGFA::OrientedSegment) **oriented_from**

Returns the oriented segment represented by the from/from_orient fields

Returns:

- (RGFA::OrientedSegment) — the oriented segment represented by the from/from_orient fields

- (RGFA::OrientedSegment) **oriented_to**

Returns the oriented segment represented by the to/to_orient fields

Returns:

- (RGFA::OrientedSegment) — the oriented segment represented by the to/to_orient fields

- (Integer?) **rpos**

Returns the rightmost 0-based coordinate of the contained sequence in the container;
nil if the overlap is unspecified

Returns:

- (Integer, nil) — the rightmost 0-based coordinate of the contained sequence in the container; nil if the overlap is unspecified

- (Symbol) **to_name**

The to segment name, in both cases where to is a segment name (Symbol) or a segment (RGFA::Line::Segment)

Returns:

- (Symbol)

Class: RGFA::SegmentEndsPath

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

Overview

An array containing `SegmentEnd` elements, which defines a path in the graph

Instance Method Summary

(collapse)

- (RGFA::SegmentEndsPath) **reverse**

Create a reverse direction path.

Methods inherited from *Array*

```
#default_gfa_datatype, #rgfa_field_array?, #to_byte_array, #to_cigar,  
#to_cigar_operation, #to_gfa_field, #to_numeric_array, #to_oriented_segment,  
#to_rgfa, #to_rgfa_field_array, #to_rgfa_line, #to_segment_end,  
#validate_gfa_field!
```

Instance Method Details

- (RGFA::SegmentEndsPath) **reverse**

Create a reverse direction path

Returns:

- (RGFA::SegmentEndsPath)

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.

Generated on Mon Oct 10 13:49:41 2016 by [yard](#) 0.8.7.6 (ruby-2.0.0).