
SEMSTR Documentation

Release 1.2.2

Daniel Hershcovich

Jun 27, 2019

Contents:

1	semstr.constraints Module	3
1.1	Functions	3
1.2	Classes	4
1.3	Class Inheritance Diagram	6
2	semstr.convert Module	7
2.1	Functions	7
3	semstr.evaluate Module	13
3.1	Functions	13
3.2	Classes	14
3.3	Class Inheritance Diagram	15
4	semstr.validation Module	17
4.1	Functions	17
5	Indices and tables	21
	Python Module Index	23
	Index	25

For more information about how to use this library, see the *API Documentation*.

1.1 Functions

<i>contains</i> (s, tag)
<i>incoming_tags</i> (node, except_edge)
<i>outgoing_tags</i> (node, except_edge)
<i>set_prod</i> (set1[, set2])
<i>tags</i> (node, except_edge, direction)

1.1.1 contains

`semstr.constraints.contains(s, tag)`

1.1.2 incoming_tags

`semstr.constraints.incoming_tags(node, except_edge)`

1.1.3 outgoing_tags

`semstr.constraints.outgoing_tags(node, except_edge)`

1.1.4 set_prod

`semstr.constraints.set_prod(set1, set2=None)`

1.1.5 tags

`semstr.constraints.tags(node, except_edge, direction)`

1.2 Classes

<i>Constraints</i> ([multigraph, ...])	
<i>Direction</i>	An enumeration.
Enum	Generic enumeration.
<i>TagRule</i> (trigger[, allowed, disallowed])	
<i>Valid</i> ([valid, message])	

1.2.1 Constraints

```
class semstr.constraints.Constraints (multigraph=False,    require_implicit_childless=True,
                                     allow_orphan_terminals=False,        al-
                                     low_root_terminal_children=False,
                                     top_level_allowed=None,    top_level_only=None,
                                     possible_multiple_incoming=(),    child-
                                     less_incoming_trigger=(),    child-
                                     less_outgoing_allowed=(),    unique_incoming=(),
                                     unique_outgoing=(), mutually_exclusive_incoming=(),
                                     mutually_exclusive_outgoing=(),    exclu-
                                     sive_outgoing=(),    required_outgoing=(),    im-
                                     plicit=False, **kwargs)
```

Bases: `object`

Methods Summary

<i>allow_action</i> (action, history)
<i>allow_child</i> (node, tag)
<i>allow_edge</i> (edge)
<i>allow_label</i> (node, label)
<i>allow_parent</i> (node, tag)

Methods Documentation

allow_action (action, history)

allow_child (node, tag)

allow_edge (edge)

allow_label (node, label)

allow_parent (node, tag)

1.2.2 Direction

```
class semstr.constraints.Direction
```

Bases: `enum.Enum`

An enumeration.

Attributes Summary

<code>incoming</code>
<code>outgoing</code>

Attributes Documentation

`incoming` = 0
`outgoing` = 1

1.2.3 TagRule

class `semstr.constraints.TagRule` (*trigger, allowed=None, disallowed=None*)
Bases: `object`

Methods Summary

<code>violation</code> (<i>node, tag, direction[, message]</i>)

Methods Documentation

violation (*node, tag, direction, message=False*)

1.2.4 Valid

class `semstr.constraints.Valid` (*valid=True, message=""*)
Bases: `object`

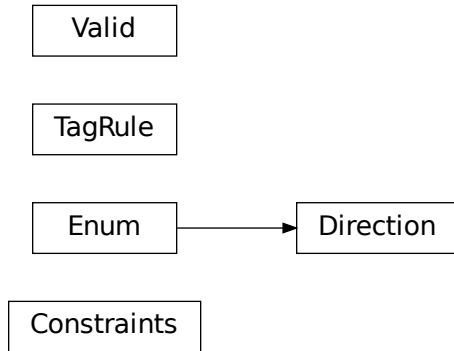
Methods Summary

<code>__call__</code> (<i>valid[, message]</i>)	Call self as a function.
---	--------------------------

Methods Documentation

__call__ (*valid, message=None*)
Call self as a function.

1.3 Class Inheritance Diagram



2.1 Functions

<code>add_boolean_option(argparser, name, description)</code>	
<code>add_convert_args(p)</code>	
<code>add_verbose_arg(argparser, **kwargs)</code>	
<code>from_amr(lines[, passage_id, ...])</code>	Converts from parsed text in AMR PENMAN format to a Passage object.
<code>from_conll(lines, passage_id[, ...])</code>	Converts from parsed text in CoNLL format to a Passage object.
<code>from_conllu(lines, passage_id[, ...])</code>	Converts from parsed text in Universal Dependencies format to a Passage object.
<code>from_export(lines, passage_id, return_original)</code>	Converts from parsed text in NeGra export format to a Passage object.
<code>from_sdp(lines, passage_id[, mark_aux, ...])</code>	Converts from parsed text in SemEval 2015 SDP format to a Passage object.
<code>glob(pathname, *[, recursive])</code>	Return a list of paths matching a pathname pattern.
<code>iter_files(patterns)</code>	
<code>iter_passages(patterns[, desc, ...])</code>	
<code>main(args)</code>	
<code>map_labels(passage, label_map_file)</code>	
<code>print_errors(errors, passage_id[, id_len])</code>	
<code>to_amr(passage[, metadata, wikification, ...])</code>	Convert from a Passage object to a string in AMR PENMAN format (export)
<code>to_conll(passage[, test, tree, preprocess])</code>	Convert from a Passage object to a string in CoNLL-X format (conll)
<code>to_conllu(passage[, test, enhanced, preprocess])</code>	Convert from a Passage object to a string in Universal Dependencies format (conllu)

Continued on next page

Table 1 – continued from previous page

<code>to_export(passage[, test, tree])</code>	Convert from a Passage object to a string in NeGra export format (export)
<code>to_sdp(passage[, test, tree, mark_aux, ...])</code>	Convert from a Passage object to a string in SemEval 2015 SDP format (sdp)
<code>validate(passage[, normalization, ...])</code>	
<code>write_passage(passage[, out_dir, ...])</code>	

2.1.1 add_convert_args

`semstr.convert.add_convert_args(p)`

2.1.2 from_amr

`semstr.convert.from_amr` (*lines*, *passage_id=None*, *return_original=False*, *save_original=True*, *wikification=False*, *placeholders=True*, ***kwargs*)

Converts from parsed text in AMR PENMAN format to a Passage object.

Parameters

- **lines** – iterable of lines in AMR PENMAN format, describing a single passage.
- **passage_id** – ID to set for passage, overriding the ID from the file
- **save_original** – whether to save original AMR text in `passage.extra`
- **return_original** – return triple of (UCCA passage, AMR string, AMR ID)
- **wikification** – whether to use wikification for replacing node labels with placeholders based on tokens
- **placeholders** – introduce placeholders into node labels when they include the terminal's text?

:return generator of Passage objects

2.1.3 from_conll

`semstr.convert.from_conll` (*lines*, *passage_id*, *return_original=False*, *dep=False*, *preprocess=True*, ***kwargs*)

Converts from parsed text in CoNLL format to a Passage object.

Parameters

- **lines** – iterable of lines in CoNLL format, describing a single passage.
- **passage_id** – ID to set for passage
- **return_original** – return triple of (UCCA passage, CoNLL string, sentence ID)
- **dep** – return dependency graph rather than converted UCCA passage
- **preprocess** – preprocess the dependency graph before converting to UCCA (or returning it)?

:return generator of Passage objects

2.1.4 from_conllu

`semstr.convert.from_conllu` (*lines*, *passage_id=None*, *return_original=False*, *annotate=False*, *terminals_only=False*, *dep=False*, *enhanced=True*, *preprocess=True*, ***kwargs*)

Converts from parsed text in Universal Dependencies format to a Passage object.

Parameters

- **lines** – iterable of lines in Universal Dependencies format, describing a single passage.
- **passage_id** – ID to set for passage
- **return_original** – return triple of (UCCA passage, Universal Dependencies string, sentence ID)
- **annotate** – whether to save dependency annotations in “extra” dict of layer 0
- **terminals_only** – create only terminals (with any annotation if specified), no non-terminals
- **dep** – return dependency graph rather than converted UCCA passage
- **enhanced** – whether to include enhanced edges
- **preprocess** – preprocess the dependency graph before converting to UCCA (or returning it)?

:return generator of Passage objects

2.1.5 from_export

`semstr.convert.from_export` (*lines*, *passage_id=None*, *return_original=False*, ***kwargs*)

Converts from parsed text in NeGra export format to a Passage object.

Parameters

- **lines** – iterable of lines in NeGra export format, describing a single passage.
- **passage_id** – ID to set for passage, overriding the ID from the file
- **return_original** – return triple of (UCCA passage, Export string, sentence ID)

:return generator of Passage objects

2.1.6 from_sdp

`semstr.convert.from_sdp` (*lines*, *passage_id*, *mark_aux=False*, *return_original=False*, *dep=False*, *preprocess=True*, ***kwargs*)

Converts from parsed text in SemEval 2015 SDP format to a Passage object.

Parameters

- **lines** – iterable of lines in SDP format, describing a single passage.
- **passage_id** – ID to set for passage
- **mark_aux** – add a preceding # for labels of auxiliary edges added
- **return_original** – return triple of (UCCA passage, SDP string, sentence ID)
- **dep** – return dependency graph rather than converted UCCA passage

- **preprocess** – preprocess the dependency graph before converting to UCCA (or returning it)?

:return generator of Passage objects

2.1.7 iter_files

`semstr.convert.iter_files(patterns)`

2.1.8 iter_passages

`semstr.convert.iter_passages(patterns, desc=None, input_format=None, prefix="", label_map=None, output_format=None, **kwargs)`

2.1.9 main

`semstr.convert.main(args)`

2.1.10 map_labels

`semstr.convert.map_labels(passage, label_map_file)`

2.1.11 to_amr

`semstr.convert.to_amr(passage, metadata=True, wikification=True, use_original=True, verbose=False, default_label=None, **kwargs)`

Convert from a Passage object to a string in AMR PENMAN format (export)

Parameters

- **passage** – the Passage object to convert
- **metadata** – whether to print ::id and ::tok lines
- **wikification** – whether to wikify named concepts, adding a :wiki triple
- **use_original** – whether to use original AMR text from passage.extra
- **verbose** – whether to print extra information
- **default_label** – label to use in case node has no label attribute

:return list of lines representing an AMR in PENMAN format, constructed from the passage

2.1.12 to_conll

`semstr.convert.to_conll(passage, test=False, tree=False, preprocess=True, **kwargs)`

Convert from a Passage object to a string in CoNLL-X format (conll)

Parameters

- **passage** – the Passage object to convert
- **test** – whether to omit the head and deprel columns. Defaults to False
- **tree** – whether to omit rows for non-primary parents. Defaults to False

- **preprocess** – preprocess the converted dependency graph before returning it?

:return list of lines representing the dependencies in the passage

2.1.13 to_conllu

`semstr.convert.to_conllu (passage, test=False, enhanced=True, preprocess=True, **kwargs)`

Convert from a Passage object to a string in Universal Dependencies format (conllu)

Parameters

- **passage** – the Passage object to convert
- **test** – whether to omit the head and deprel columns. Defaults to False
- **enhanced** – whether to include enhanced edges
- **preprocess** – preprocess the converted dependency graph before returning it?

:return list of lines representing the semantic dependencies in the passage

2.1.14 to_export

`semstr.convert.to_export (passage, test=False, tree=False, **kwargs)`

Convert from a Passage object to a string in NeGra export format (export)

Parameters

- **passage** – the Passage object to convert
- **test** – whether to omit the edge and parent columns. Defaults to False
- **tree** – whether to omit columns for non-primary parents. Defaults to False

:return list of lines representing a (discontinuous) tree structure constructed from the passage

2.1.15 to_sdp

`semstr.convert.to_sdp (passage, test=False, tree=False, mark_aux=False, preprocess=True, **kwargs)`

Convert from a Passage object to a string in SemEval 2015 SDP format (sdp)

Parameters

- **passage** – the Passage object to convert
- **test** – whether to omit the top, head, frame, etc. columns. Defaults to False
- **tree** – whether to omit columns for non-primary parents. Defaults to False
- **mark_aux** – omit edges with labels with a preceding #
- **preprocess** – preprocess the converted dependency graph before returning it?

:return list of lines representing the semantic dependencies in the passage

2.1.16 write_passage

`semstr.convert.write_passage (passage, out_dir='', output_format=None, binary=False, verbose=False, label_map=False, split=False, join=None, **kwargs)`

3.1 Functions

<code>add_boolean_option(argparser, name, description)</code>	
<code>add_verbose_arg(argparser, **kwargs)</code>	
<code>align_fields(fields, titles, title2index)</code>	Make sure score fields for individual passage are aligned with summary result fields by inserting empties
<code>evaluate_all(evaluate, files[, name, ...])</code>	
<code>evaluate_amr(*args, **kwargs)</code>	
<code>evaluate_conllu(*args, **kwargs)</code>	
<code>evaluate_sdp(*args, **kwargs)</code>	
<code>main(args)</code>	
<code>passage_format(filename)</code>	
<code>read_files(files[, verbose, force_basename])</code>	
<code>summarize(scores[, errors])</code>	
<code>write_csv(filename, rows)</code>	

3.1.1 align_fields

`semstr.evaluate.align_fields(fields, titles, title2index)`
 Make sure score fields for individual passage are aligned with summary result fields by inserting empties

3.1.2 evaluate_all

`semstr.evaluate.evaluate_all(evaluate, files, name=None, verbose=0, quiet=False, base-name=False, matching_ids=False, units=False, unlabeled=False, **kwargs)`

3.1.3 evaluate_amr

```
semstr.evaluate.evaluate_amr(*args, **kwargs)
```

3.1.4 evaluate_conllu

```
semstr.evaluate.evaluate_conllu(*args, **kwargs)
```

3.1.5 evaluate_sdp

```
semstr.evaluate.evaluate_sdp(*args, **kwargs)
```

3.1.6 main

```
semstr.evaluate.main(args)
```

3.1.7 passage_format

```
semstr.evaluate.passage_format(filename)
```

3.1.8 read_files

```
semstr.evaluate.read_files(files, verbose=0, force_basename=False, **kw)
```

3.1.9 summarize

```
semstr.evaluate.summarize(scores, errors=False)
```

3.1.10 write_csv

```
semstr.evaluate.write_csv(filename, rows)
```

3.2 Classes

<code>ConvertedPassage(converted[, original, ...])</code>	
<code>Scores(scores)</code>	Keeps score objects from multiple formats and/or languages
<code>groupby(iterable[, key])</code>	keys and groups from the iterable.
<code>repeat(object [,times])</code>	for the specified number of times.

3.2.1 ConvertedPassage

```
class semstr.evaluate.ConvertedPassage (converted, original=None, passage_id=None,
                                         converted_format=None, in_converter=None,
                                         out_converter=None)
```

Bases: `object`

3.2.2 Scores

```
class semstr.evaluate.Scores (scores)
```

Bases: `object`

Keeps score objects from multiple formats and/or languages

Methods Summary

`aggregate`(*scores*)

`average_f1`(**args*, ***kwargs*)

`details`(*average_f1*)

`fields`(**args*, ***kwargs*)

`print`(**args*, ***kwargs*)

`titles`(**args*[, *prefix*])

Methods Documentation

static aggregate (*scores*)

average_f1 (**args*, ***kwargs*)

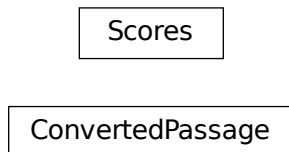
details (*average_f1*)

fields (**args*, ***kwargs*)

print (**args*, ***kwargs*)

titles (**args*, *prefix=True*, ***kwargs*)

3.3 Class Inheritance Diagram



4.1 Functions

<i>amr_constraints</i> (*args, **kwargs)
<i>check_implicit_children</i> (constraints, node)
<i>check_multigraph</i> (constraints, node)
<i>check_multiple_incoming</i> (constraints, node)
<i>check_orphan_terminals</i> (constraints, terminal)
<i>check_required_outgoing</i> (constraints, node)
<i>check_root_terminal_children</i> (constraints, ...)
<i>check_tag_rules</i> (constraints, node)
<i>check_top_level_allowed</i> (constraints, ll)
<i>check_top_level_only</i> (constraints, ll, node)
<i>conllu_constraints</i> (*args, **kwargs)
<i>detect_cycles</i> (passage)
<i>join</i> (edges)
<i>print_errors</i> (errors, passage_id[, id_len])
<i>sdp_constraints</i> (*args, **kwargs)
<i>ucca_constraints</i> (*args, **kwargs)
<i>validate</i> (passage[, normalization, ...])

4.1.1 amr_constraints

```
semstr.validation.amr_constraints(*args, **kwargs)
```

4.1.2 check_implicit_children

```
semstr.validation.check_implicit_children(constraints, node)
```

4.1.3 check_multigraph

`semstr.validation.check_multigraph (constraints, node)`

4.1.4 check_multiple_incoming

`semstr.validation.check_multiple_incoming (constraints, node)`

4.1.5 check_orphan_terminals

`semstr.validation.check_orphan_terminals (constraints, terminal)`

4.1.6 check_required_outgoing

`semstr.validation.check_required_outgoing (constraints, node)`

4.1.7 check_root_terminal_children

`semstr.validation.check_root_terminal_children (constraints, ll, terminal)`

4.1.8 check_tag_rules

`semstr.validation.check_tag_rules (constraints, node)`

4.1.9 check_top_level_allowed

`semstr.validation.check_top_level_allowed (constraints, ll)`

4.1.10 check_top_level_only

`semstr.validation.check_top_level_only (constraints, ll, node)`

4.1.11 conllu_constraints

`semstr.validation.conllu_constraints (*args, **kwargs)`

4.1.12 detect_cycles

`semstr.validation.detect_cycles (passage)`

4.1.13 join

`semstr.validation.join (edges)`

4.1.14 `print_errors`

`semstr.validation.print_errors(errors, passage_id, id_len=None)`

4.1.15 `sdp_constraints`

`semstr.validation.sdp_constraints(*args, **kwargs)`

4.1.16 `ucca_constraints`

`semstr.validation.ucca_constraints(*args, **kwargs)`

4.1.17 `validate`

`semstr.validation.validate(passage, normalization=False, extra_normalization=False,
ucca_validation=False, output_format=None, **kwargs)`

CHAPTER 5

Indices and tables

- `genindex`
- `modindex`
- `search`

S

`semstr.constraints`, [3](#)
`semstr.convert`, [7](#)
`semstr.evaluate`, [13](#)
`semstr.validation`, [17](#)

Symbols

`__call__()` (*semstr.constraints.Valid method*), 5

A

`add_convert_args()` (*in module semstr.convert*), 8
`aggregate()` (*semstr.evaluate.Scores static method*), 15

`align_fields()` (*in module semstr.evaluate*), 13

`allow_action()` (*semstr.constraints.Constraints method*), 4

`allow_child()` (*semstr.constraints.Constraints method*), 4

`allow_edge()` (*semstr.constraints.Constraints method*), 4

`allow_label()` (*semstr.constraints.Constraints method*), 4

`allow_parent()` (*semstr.constraints.Constraints method*), 4

`amr_constraints()` (*in module semstr.validation*), 17

`average_f1()` (*semstr.evaluate.Scores method*), 15

C

`check_implicit_children()` (*in module semstr.validation*), 17

`check_multigraph()` (*in module semstr.validation*), 18

`check_multiple_incoming()` (*in module semstr.validation*), 18

`check_orphan_terminals()` (*in module semstr.validation*), 18

`check_required_outgoing()` (*in module semstr.validation*), 18

`check_root_terminal_children()` (*in module semstr.validation*), 18

`check_tag_rules()` (*in module semstr.validation*), 18

`check_top_level_allowed()` (*in module semstr.validation*), 18

`check_top_level_only()` (*in module semstr.validation*), 18

`conllu_constraints()` (*in module semstr.validation*), 18

`Constraints` (*class in semstr.constraints*), 4

`contains()` (*in module semstr.constraints*), 3

`ConvertedPassage` (*class in semstr.evaluate*), 15

D

`details()` (*semstr.evaluate.Scores method*), 15

`detect_cycles()` (*in module semstr.validation*), 18

`Direction` (*class in semstr.constraints*), 4

E

`evaluate_all()` (*in module semstr.evaluate*), 13

`evaluate_amr()` (*in module semstr.evaluate*), 14

`evaluate_conllu()` (*in module semstr.evaluate*), 14

`evaluate_sdp()` (*in module semstr.evaluate*), 14

F

`fields()` (*semstr.evaluate.Scores method*), 15

`from_amr()` (*in module semstr.convert*), 8

`from_conll()` (*in module semstr.convert*), 8

`from_conllu()` (*in module semstr.convert*), 9

`from_export()` (*in module semstr.convert*), 9

`from_sdp()` (*in module semstr.convert*), 9

I

`incoming` (*semstr.constraints.Direction attribute*), 5

`incoming_tags()` (*in module semstr.constraints*), 3

`iter_files()` (*in module semstr.convert*), 10

`iter_passages()` (*in module semstr.convert*), 10

J

`join()` (*in module semstr.validation*), 18

M

`main()` (*in module semstr.convert*), 10

`main()` (*in module semstr.evaluate*), 14

`map_labels()` (in module *semstr.convert*), 10

O

`outgoing` (*semstr.constraints.Direction* attribute), 5

`outgoing_tags()` (in module *semstr.constraints*), 3

P

`passage_format()` (in module *semstr.evaluate*), 14

`print()` (*semstr.evaluate.Scores* method), 15

`print_errors()` (in module *semstr.validation*), 19

R

`read_files()` (in module *semstr.evaluate*), 14

S

`Scores` (class in *semstr.evaluate*), 15

`sdp_constraints()` (in module *semstr.validation*),
19

`semstr.constraints` (module), 3

`semstr.convert` (module), 7

`semstr.evaluate` (module), 13

`semstr.validation` (module), 17

`set_prod()` (in module *semstr.constraints*), 3

`summarize()` (in module *semstr.evaluate*), 14

T

`TagRule` (class in *semstr.constraints*), 5

`tags()` (in module *semstr.constraints*), 3

`titles()` (*semstr.evaluate.Scores* method), 15

`to_amr()` (in module *semstr.convert*), 10

`to_conll()` (in module *semstr.convert*), 10

`to_conllu()` (in module *semstr.convert*), 11

`to_export()` (in module *semstr.convert*), 11

`to_sdp()` (in module *semstr.convert*), 11

U

`ucca_constraints()` (in module *semstr.validation*),
19

V

`Valid` (class in *semstr.constraints*), 5

`validate()` (in module *semstr.validation*), 19

`violation()` (*semstr.constraints.TagRule* method), 5

W

`write_csv()` (in module *semstr.evaluate*), 14

`write_passage()` (in module *semstr.convert*), 11