Detano: Library for Automata-based Detection of Network Anomalies

Generated by Doxygen 1.9.2

1 Namespace Index

1.1 Packages

Here are the packages with brief descriptions (if available):

2 Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

5 Namespace Documentation

5.1 alergia Namespace Reference

Alergia algorithm.

Functions

- def alergia (freq_aut, alpha, t0)

  PA learning using the Alergia algorithm.
- def choose_blue_state (freq_aut, blue_set, t0)

  Chose a blue state from a set of blue states.
- def choose_red_state (freq_aut, red_set, blue, alpha)

  Chose a red state from a set of red states.

5.1.1 Detailed Description

Alergia algorithm.

Alergia algorithm for learning deterministic probabilistic automata for the context of network communication.

Author

Vojtěch Havlena

Copyright

5.1.2 Function Documentation

5.1.2.1 alergia() `def alergia.alergia ( freq_aut, alpha, t0 )`

PA learning using the Alergia algorithm.

Parameters

freq_aut	A frequency automaton constructed from the input sample
alpha	Merging parameter
t0	The minimum number of strings for merging a state

Returns

Compact frequency automaton (no normalization applied)

5.1.2.2 choose_blue_state() `def alergia.choose_blue_state ( freq_aut, blue_set, t0 )`

Chose a blue state from a set of blue states.

Parameters

freq_aut	Frequency automaton
blue_set	Set of blue states
t0	The minimum number of strings for merging a state

Returns

Chosen blue state

5.1.2.3 choose_red_state() `def alergia.choose_red_state ( freq_aut, red_set, blue, alpha )`

Chose a red state from a set of red states.

Parameters

freq_aut	Frequency automaton
red_set	Set of red states
blue	Blue state
alpha	Merging parameter

Returns

Chosen red state

5.2 anom_detect_base Namespace Reference

Anomaly detection base class.

Classes

- class AnomDetectBase

  Base class providing an interface for concrete detections.

5.2.1 Detailed Description

Anomaly detection base class.

Base class giving an interface for methods used for concrete analyses.

Author

Vojtěch Havlena

Copyright

5.3 dffa Namespace Reference

Class for deterministic frequency automata.

Classes

- class DFFA

  Deterministic frequency automaton class.

5.3.1 Detailed Description

Class for deterministic frequency automata.

Class providing operations for deterministic frequency automata.

Author

Vojtěch Havlena

Copyright

5.4 distance Namespace Reference

Class for removing similar automata in a set.

Classes

- class Distance

  Class removing items from a set causing the minimum error.

5.4.1 Detailed Description

Class for removing similar automata in a set.

Implementation of a greedy approach for removing items from a given set that causes a smallest error (the minimum distance from a removed item to a remaining item).

Author

Vojtěch Havlena

Copyright

5.5 distr_comparison Namespace Reference

Distribution-based anomaly detection.

Classes

- class AnomDistrComparison

  Anomaly detection based on comparing distributions.

Variables

- bool SPARSE = False

  Use sparse matrices to comput the Euclid distance.

5.5.1 Detailed Description

Distribution-based anomaly detection.

This file contains support for anomaly detection based on comparing distributions, which works as follows. In the first step, we learn a PA from an input traffic window. Consequently, we compare the difference between a model PA and the PA representing input window.

Author

Vojtěch Havlena

Copyright

5.5.2 Variable Documentation

5.5.2.1 SPARSE `bool distr_comparison.SPARSE = False`

Use sparse matrices to comput the Euclid distance.

5.6 ffa Namespace Reference

Class for general frequency automata.

Classes

- class FFA

  General frequency automata (FFA)
- class FFATrans

  Class representing a transtion of the FFA.

5.6.1 Detailed Description

Class for general frequency automata.

Class providing operations for general (nondeterministic) frequency automata.

Author

Vojtěch Havlena

Copyright

5.7 fpt Namespace Reference

Class for frequency prefix tree automataa.

Classes

- class FPT

  Frequency prefix tree (FPT)

5.7.1 Detailed Description

Class for frequency prefix tree automataa.

Class providing operations for frequency prefix tree automata

Author

Vojtěch Havlena

Copyright

5.8 member Namespace Reference

Member-based anomaly detection.

Classes

- class AnomMember

  Anomaly detection based on a single message reasoning.

5.8.1 Detailed Description

Member-based anomaly detection.

Anomaly detection based on a single message reasoning. Given PAs representing a valid network traffic, we check if input messages in a window are in the language of a model.

Author

Vojtěch Havlena

Copyright

5.9 packet_loss Namespace Reference

Packet-loss detection.

Classes

- class PacketLoss

  Language-based approach for a detection of packet losses.

5.9.1 Detailed Description

Packet-loss detection.

Language-based approach for a detection of packet losses. It computes edit distance (assuming only the delete operation) between two strings.

Author

Vojtěch Havlena

Copyright

5.10 parser Namespace Reference

Namespaces

- namespace conversation_parser_base

  Dividing list of messages into conversations – base class.
- namespace IEC104_conv_parser

  Parsing files with already divided conversations.
- namespace IEC104_parser

  Dividing list of messages into conversations.

5.11 parser.conversation_parser_base Namespace Reference

Dividing list of messages into conversations – base class.

Classes

- class ConvParserBase

  Base class for parsing conversations.

5.11.1 Detailed Description

Dividing list of messages into conversations – base class.

Base class providing interface for conversation parsers (from the input list of messages).

Author

Vojtěch Havlena

Copyright

5.12 parser.IEC104_conv_parser Namespace Reference

Parsing files with already divided conversations.

Classes

- class IEC104ConvParser

  Class for parsing IEC104 conversations from already divided messages.

5.12.1 Detailed Description

Parsing files with already divided conversations.

Parsing IEC104 conversations from a file. Allowing to split according to communication pairs and time windows.

Author

Vojtěch Havlena

Copyright

5.13 parser.IEC104_parser Namespace Reference

Dividing list of messages into conversations.

Classes

- class ConvType

  Type of a conversation.
- class IEC104Parser

  Class for parsing IEC104 conversations.

Functions

- def get_messages (fd)

  Get all messages from a csv file.

5.13.1 Detailed Description

Dividing list of messages into conversations.

Parsing IEC104 conversations from a list of messages (each message is a dictionary). Allowing to split according to communication pairs and time windows.

Author

Vojtěch Havlena

Copyright

5.13.2 Function Documentation

5.13.2.1 get_messages() `def parser.IEC104_parser.get_messages ( fd )`

Get all messages from a csv file.

Parameters

fd	File descriptor

Returns

Messages from the csv file fd

5.14 wfa Namespace Reference

Namespaces

- namespace aux_functions

  Auxiliary functions for WFAs.
- namespace core_wfa

  Core class for working with WFAs.
- namespace core_wfa_export

  Class for exporting WFAs in a textual format.
- namespace matrix_wfa

  Class for working with a computation of language weights.
- namespace wfa_exceptions

  Exception class for specifying errors when working with WFAs.

5.15 wfa.aux_functions Namespace Reference

Auxiliary functions for WFAs.

Functions

- def convert_to_pritable (dec, dot=False)

  Convert string containing also non-printable characters to printable hexa number.

5.15.1 Detailed Description

Auxiliary functions for WFAs.

Auxiliary functions for printing WFAs. Taken and modified from https://github.com/vhavlena/appreal

Author

Vojtěch Havlena

Copyright

5.15.2 Function Documentation

5.15.2.1 convert_to_pritable() `def wfa.aux_functions.convert_to_pritable ( dec, dot = False )`

Convert string containing also non-printable characters to printable hexa number.

Inspired by the Netbench tool.

Parameters

dec	Input string.
dot	Use the result for converting to dot format.

Returns

Input string with replaced nonprintable symbols with their hexa numbers.

5.16 wfa.core_wfa Namespace Reference

Core class for working with WFAs.

Classes

- class CoreWFA

  Basic class for representation of WFA.
- class Transition

  Class for the represention of a WFA transition.

5.16.1 Detailed Description

Core class for working with WFAs.

Class providing basic support for working with WFA. Implements various usefull algorithms, such as, product, trim, ... Taken and modified from https://github.com/vhavlena/appreal

Author

Vojtěch Havlena

Copyright

5.17 wfa.core_wfa_export Namespace Reference

Class for exporting WFAs in a textual format.

Classes

- class CoreWFAExport

  Class for exporting WFAs to a text format.

Variables

- int PRECISE = 3

  Precise of float numbers (for output)
- int SYMBOLS = 25

  Max number of symbols on transition (DOT format)

5.17.1 Detailed Description

Class for exporting WFAs in a textual format.

Class providing exporting a WFA into FA or DOT format. Taken and modified from https://github.com/vhavlena/appreal

Author

Vojtěch Havlena

Copyright

5.17.2 Variable Documentation

5.17.2.1 PRECISE `int wfa.core_wfa_export.PRECISE = 3`

Precise of float numbers (for output)

5.17.2.2 SYMBOLS `int wfa.core_wfa_export.SYMBOLS = 25`

Max number of symbols on transition (DOT format)

5.18 wfa.matrix_wfa Namespace Reference

Class for working with a computation of language weights.

Classes

- class ClosureMode

  Ignore a particular warning.
- class MatrixWFA

  Class for matrix operations with WFAs involving matrix operations.
- class MatrixWFAOperationException

  Exception for invalid operations and errors during the closure computing.

Variables

- float THRESHOLD = 0.0

  Threshold for sparse matrices.

5.18.1 Detailed Description

Class for working with a computation of language weights.

Class providing support for a computation of weight of the language (specified by the WFA). Inmplements various methods and approaches for transition closure computation. Taken and modified from https://github.com/vhavlena/appreal

Author

Vojtěch Havlena

Copyright

5.18.2 Variable Documentation

5.18.2.1 THRESHOLD `float wfa.matrix_wfa.THRESHOLD = 0.0`

Threshold for sparse matrices.

5.19 wfa.wfa_exceptions Namespace Reference

Exception class for specifying errors when working with WFAs.

Classes

- class WFAErrorType
  Error types for WFAs.
- class WFAOperationException
  Exception used when an error during parsing is occured.

5.19.1 Detailed Description

Exception class for specifying errors when working with WFAs.

Exception class for specifying errors when working with WFAs. Taken and modified from https://github.com/vhavlena/appreal

Author

Vojtěch Havlena

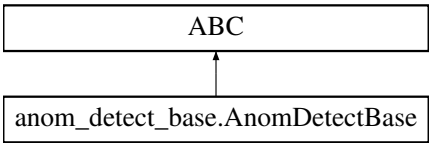
Copyright

6 Class Documentation

6.1 anom_detect_base.AnomDetectBase Class Reference

Base class providing an interface for concrete detections.

Inheritance diagram for anom_detect_base.AnomDetectBase:



Public Member Functions

- def apply_detection (self, aut, window, compair)

  Abstract apply detection on a given window.
- def detect (self, window, compair)

  Abstract anomaly detection.
- def dpa_selection (self, window, compair)

  Abstract DPA selection.

6.1.1 Detailed Description

Base class providing an interface for concrete detections.

6.1.2 Member Function Documentation

6.1.2.1 apply_detection() `def anom_detect_base.AnomDetectBase.apply_detection (`

self,
aut,
window,
compair)

Abstract apply detection on a given window.

Parameters

aut	Golden PA (representing a normal behavior)
window	List of messages corresponding to a single window to be checked
compair	Pair of communicating devices

Returns

abstact detection values

6.1.2.2 detect() `def anom_detect_base.AnomDetectBase.detect (`

self,
window,
compair)

Abstract anomaly detection.

Parameters

window	List of messages corresponding to a single window to be checked
compair	Pair of communicating devices

Returns

abstact detection values

6.1.2.3 dpa_selection() `def anom_detect_base.AnomDetectBase.dpa_selection (`

```
self,
window,
compair )
```

Abstract DPA selection.

Parameters

window	List of messages corresponding to a single window
compair	Pair of communicating devices

Returns

Selected DPA

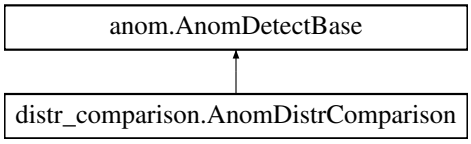
The documentation for this class was generated from the following file:

- anom_detect_base.py

6.2 distr_comparison.AnomDistrComparison Class Reference

Anomaly detection based on comparing distributions.

Inheritance diagram for distr_comparison.AnomDistrComparison:



Public Member Functions

- def __init__ (self, aut_map, learning_procedure)

  Constructor.
- def apply_detection (self, aut, window, compair)

  Apply distribution-comparison-based anomaly detection.
- def detect (self, window, compair)

  Detect if anomaly occurrs in the given window.
- def dpa_selection (self, window, compair)

  Select appropriate DPA according to a communication window and a communication pair.
- def remove_euclid_similar (self, max_error)

  Remove Euclid similar automata from the golden map (with the error bounded by max_error).
- def remove_identical (self)

  Remove identical automata from the golden map.

Static Public Member Functions

- def euclid_distance (aut1, aut2)

  Compute Euclid distance between two automata.

Public Attributes

- golden_map

  Mapping of communication pairs to automata representing normal behavior.
- learning_proc

  Procedure used to obtain a PA from a list of messages.

6.2.1 Detailed Description

Anomaly detection based on comparing distributions.

6.2.2 Constructor & Destructor Documentation

6.2.2.1 __init__() `def distr_comparison.AnomDistrComparison.__init__ ( self, aut_map, learning_procedure )`

Constructor.

Parameters

aut_map	Mapping of communication pairs to automata representing normal behavior
learning_procedure	procedure used to obtain a PA from a list of messages

6.2.3 Member Function Documentation

6.2.3.1 apply_detection() `def distr_comparison.AnomDistrComparison.apply_detection ( self, aut, window, compair )`

Apply distribution-comparison-based anomaly detection.

Parameters

aut	Golden automaton
window	List of messages to be inspected
compair	Pair of communicating devices

Returns

Number representing similarity of aut and window

6.2.3.2 detect() def distr_comparison.AnomDistrComparison.detect (self, window, compair)

Detect if anomaly occurrs in the given window.

Parameters

window	List of messages corresponding to a single window to be checked
compair	Pair of communicating devices

Returns

List of floats representing distance between golden automata and a window

6.2.3.3 dpa_selection() def distr_comparison.AnomDistrComparison.dpa_selection (self, window, compair)

Select appropriate DPA according to a communication window and a communication pair.

Parameters

window	List of messages corresponding to a single window
compair	Pair of communicating devices

Returns

Selected DPA

6.2.3.4 euclid_distance() def distr_comparison.AnomDistrComparison.euclid_distance (aut1, aut2) [static]

Compute Euclid distance between two automata.

Parameters

aut1	First PA
aut2	Second PA

Returns

Euclid distance of aut1 and aut2

6.2.3.5 remove_euclid_similar() `def distr_comparison.AnomDistrComparison.remove_euclid_similar ( self, max_error )`

Remove Euclid similar automata from the golden map (with the error bounded by max_error).

Parameters

max_error	Maximum error bound

6.2.3.6 remove_identical() `def distr_comparison.AnomDistrComparison.remove_identical ( self )`

Remove identical automata from the golden map.

6.2.4 Member Data Documentation

6.2.4.1 golden_map `distr_comparison.AnomDistrComparison.golden_map`

Mapping of communication pairs to automata representing normal behavior.

6.2.4.2 learning_proc `distr_comparison.AnomDistrComparison.learning_proc`

Procedure used to obtain a PA from a list of messages.

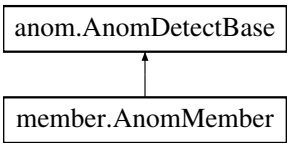
The documentation for this class was generated from the following file:

- distr_comparison.py

6.3 member.AnomMember Class Reference

Anomaly detection based on a single message reasoning.

Inheritance diagram for member.AnomMember:



Public Member Functions

- def __init__ (self, aut_map, learning_procedure)
  Constructor.
- def apply_detection (self, aut, window, compair)
  Apply member-based anomaly detection.
- def detect (self, window, compair)
  Detect if anomaly occurrs in the given window.
- def dpa_selection (self, window, compair)
  Select appropriate DPA according to a communication window and a communication pair.

Public Attributes

- golden_map
  Mapping of communication pairs to automata representing normal behavior.
- learning_proc
  Procedure used to obtain a PA from a list of messages.

6.3.1 Detailed Description

Anomaly detection based on a single message reasoning.

6.3.2 Constructor & Destructor Documentation

6.3.2.1 __init__() `def member.AnomMember.__init__ ( self, aut_map, learning_procedure )`

Constructor.

Parameters

aut_map	Mapping of communication pairs to automata representing normal behavior
learning_procedure	procedure used to obtain a PA from a list of messages

6.3.3 Member Function Documentation

6.3.3.1 apply_detection() def member.AnomMember.apply_detection (self, aut, window, compair)

Apply member-based anomaly detection.

Returns list of conversations that are not accepted by aut.

Parameters

aut	Golden automaton
window	List of messages to be inspected
compair	Pair of communicating devices

Returns

List of not accepted messages

6.3.3.2 detect() def member.AnomMember.detect (self, window, compair)

Detect if anomaly occurrs in the given window.

Parameters

window	List of messages to be inspected
compair	Pair of communicating devices

Returns

List of detection result for each model

6.3.3.3 dpa_selection() def member.AnomMember.dpa_selection (self, window, compair)

Select appropriate DPA according to a communication window and a communication pair.

Parameters

window	List of messages to be inspected
compair	Pair of communicating devices

Returns

Selected DPA

6.3.4 Member Data Documentation

6.3.4.1 golden_map `member.AnomMember.golden_map`

Mapping of communication pairs to automata representing normal behavior.

6.3.4.2 learning_proc `member.AnomMember.learning_proc`

Procedure used to obtain a PA from a list of messages.

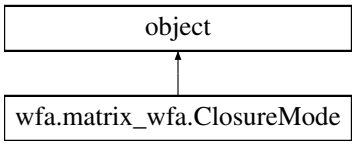
The documentation for this class was generated from the following file:

- member.py

6.4 wfa.matrix_wfa.ClosureMode Class Reference

Ignore a particular warning.

Inheritance diagram for wfa.matrix_wfa.ClosureMode:



Static Public Attributes

- int hotelling_bodewig = 3

  Hotteling-Bodeqig algorithm.
- int inverse = 1

  Use matrix inversion.
- int iterations = 2

  Iterative matrix multiplication.

6.4.1 Detailed Description

Ignore a particular warning.

Implemented methods for computing the closure.

6.4.2 Member Data Documentation

6.4.2.1 hotelling_bodewig `int wfa.matrix_wfa.ClosureMode.hotelling_bodewig = 3 [static]`

Hotteling-Bodeqig algorithm.

6.4.2.2 inverse `int wfa.matrix_wfa.ClosureMode.inverse = 1 [static]`

Use matrix inversion.

6.4.2.3 iterations `int wfa.matrix_wfa.ClosureMode.iterations = 2 [static]`

Iterative matrix multiplication.

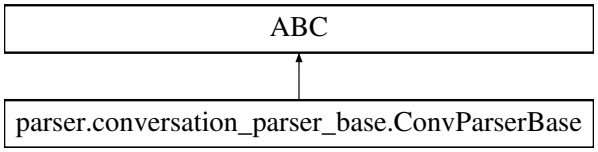
The documentation for this class was generated from the following file:

- matrix_wfa.py

6.5 parser.conversation_parser_base.ConvParserBase Class Reference

Base class for parsing conversations.

Inheritance diagram for parser.conversation_parser_base.ConvParserBase:



Public Member Functions

- def get_all_conversations (self, proj=None)

  Get all conversations (possibly projected by abstraction)
- def get_conversation (self)

  Get a following conversation from a list of messages.
- def parse_conversations (self)

  Parse and store all conversations.
- def split_communication_pairs (self)

  Split input according to the communication pairs.
- def split_to_windows (self, dur)

  Split input according to time windows.

6.5.1 Detailed Description

Base class for parsing conversations.

6.5.2 Member Function Documentation

6.5.2.1 get_all_conversations() `def parser.conversation_parser_base.ConvParserBase.get_all_↩ conversations (`

```
                self,
                proj = None )
```

Get all conversations (possibly projected by abstraction)

Parameters

proj	Projection applied on data

Returns

List of all conversations

6.5.2.2 get_conversation() `def parser.conversation_parser_base.ConvParserBase.get_conversation (`

```
                self )
```

Get a following conversation from a list of messages.

It implements just a couple of cases (definitely not all of them)

Returns

Next conversation

6.5.2.3 parse_conversations() def parser.conversation_parser_base.ConvParserBase.parse_conversations (

```
self )
```

Parse and store all conversations.

6.5.2.4 split_communication_pairs() def parser.conversation_parser_base.ConvParserBase.split_↩ communication_pairs (

```
self )
```

Split input according to the communication pairs.

Returns

List of ConvParserBase (or derived)

6.5.2.5 split_to_windows() def parser.conversation_parser_base.ConvParserBase.split_to_windows (

```
self,
dur )
```

Split input according to time windows.

Parameters

dur	Time duration

Returns

List of ConvParserBase (or derived)

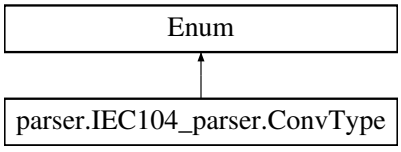
The documentation for this class was generated from the following file:

- conversation_parser_base.py

6.6 parser.IEC104_parser.ConvType Class Reference

Type of a conversation.

Inheritance diagram for parser.IEC104_parser.ConvType:



Static Public Attributes

- int FILETRANSFER = 0

  File transfer.
- int GENERAL = 1

  General interrogation.
- int GENERAL_ACT = 2

  General acknowledgement.
- int SPONTANEOUS = 3

  Spontaneous conversation.
- int UNKNOWN = 99

  Unknowt type.

6.6.1 Detailed Description

Type of a conversation.

6.6.2 Member Data Documentation

6.6.2.1 FILETRANSFER `int parser.IEC104_parser.ConvType.FILETRANSFER = 0 [static]`

File transfer.

6.6.2.2 GENERAL `int parser.IEC104_parser.ConvType.GENERAL = 1 [static]`

General interrogation.

6.6.2.3 GENERAL_ACT `int parser.IEC104_parser.ConvType.GENERAL_ACT = 2 [static]`

General acknowledgement.

6.6.2.4 SPONTANEOUS `int parser.IEC104_parser.ConvType.SPONTANEOUS = 3 [static]`

Spontaneous conversation.

6.6.2.5 UNKNOWN `int parser.IEC104_parser.ConvType.UNKNOWN = 99 [static]`

Unknowt type.

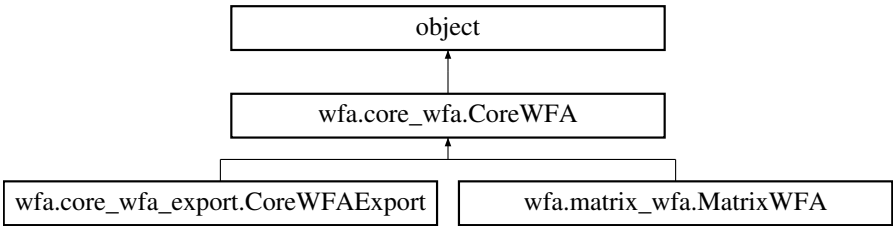
The documentation for this class was generated from the following file:

- IEC104_parser.py

6.7 wfa.core_wfa.CoreWFA Class Reference

Basic class for representation of WFA.

Inheritance diagram for wfa.core_wfa.CoreWFA:



Public Member Functions

- def __eq__ (self, other)
  Equality of two WFAs.
- def __hash__ (self)
  Hash method.
- def __init__ (self, transitions=None, finals=None, start=None, alphabet=None)
  Constructor.
- def breadth_first_search (self, state, visited, tr_dict)
  BFS in the automaton graph.
- def get_accessible_states (self, tr_dict=None)
  Get accessible states of the WFA.
- def get_alphabet (self)
  Get alphabet used by the WFA.
- def get_automata_restriction (self, states)
  Get WFA restriction to only states in states.
- def get_coaccessible_states (self, tr_dict=None)
  Get coaccessible states of the WFA.
- def get_dictionary_transitions (self)
  Get transitions in the form of dictionary (for each state there is a list of transitions leading from this state).
- def get_finals (self)
  Get all final states of the WFA.
- def get_predecessors (self, state)
  Operation that finds predessors of the state state.
- def get_predecessors_transitions (self)
  Get predecessors of all states of the WFA.
- def get_rename_dict (self)
  Get the dictionary containing original state labels and renamed state labels.

- def get_rev_transitions_aut (self)

  Get automaton with reversed directions of transitios.
- def get_single_dictionary_transitions (self)

  Get the transitions (ommiting transitions that differ only on the symbol) in the form of dictinary (for each state there is a list of transitions leading from this state).
- def get_starts (self)

  Get the start state (only one start state is allowed).
- def get_state_symbol_dict (self)

  Get transitions in the form of dictionary (for each state there is a dictionary assigning to symbols a set of transitions)
- def get_states (self)

  Get all states of the WFA (the list of states is computed from the transitions).
- def get_transitions (self)

  Get all transitions of the WFA.
- def get_trim_automaton (self)

  Get trimed WFA.
- def is_deterministic (self)

  Is the WFA deterministic.
- def map_symbols (self, fnc)

  Apply the function fnc on the symbols of all transitions.
- def product (self, aut)

  Perform the product of two WFAs.
- def rename_alphabet (self, dct)

  Rename alphabet of the automaton (in place).
- def rename_states (self)

  Rename states of the WFA.
- def set_all_finals (self)

  Set all states to be final (all having the accepting weight 1.0)
- def set_finals (self, finals)

  Set final states of the WFA.
- def set_starts (self, start)

  Set the initial state.
- def string_prob_deterministic (self, word)

  Compute the probability of the word word.

6.7.1 Detailed Description

Basic class for representation of WFA.

6.7.2 Constructor & Destructor Documentation

6.7.2.1 __init__() def wfa.core_wfa.CoreWFA.__init__ (

```
self,
transitions = None,
finals = None,
start = None,
alphabet = None )
```

Constructor.

Parameters

transitions	Transitions
finals	Final states with weights
start	Initial state
alphabet	Alphabet

Reimplemented in wfa.core_wfa_export.CoreWFAExport, and wfa.matrix_wfa.MatrixWFA.

6.7.3 Member Function Documentation

6.7.3.1 __eq__() `def wfa.core_wfa.CoreWFA.__eq__ ( self, other )`

Equality of two WFAs.

Parameters

other	Other WFA

Returns

True – both WFAs are equal

6.7.3.2 __hash__() `def wfa.core_wfa.CoreWFA.__hash__ ( self )`

Hash method.

Returns

Hash

6.7.3.3 breadth_first_search() `def wfa.core_wfa.CoreWFA.breadth_first_search ( self, state, visited, tr_dict )`

BFS in the automaton graph.

Parameters

state	The start state of the BFS.
visited	The list of visited states (out parameter).
tr_dict	Transition dictionary.

Returns

Out parameter visited (the list of visited states).

6.7.3.4 get_accessible_states() `def wfa.core_wfa.CoreWFA.get_accessible_states ( self, tr_dict = None )`

Get accessible states of the WFA.

Parameters

tr_dict	Transition dictionary.

Returns

The list of accessible states.

6.7.3.5 get_alphabet() `def wfa.core_wfa.CoreWFA.get_alphabet ( self )`

Get alphabet used by the WFA.

If the alphabet is not explicitly given (in constructor), the alphabet is computed from the transitions.

Returns

List of symbols.

6.7.3.6 get_automata_restriction() `def wfa.core_wfa.CoreWFA.get_automata_restriction ( self, states )`

Get WFA restriction to only states in states.

Parameters

states	The list of states of the new WFA.

Returns

WFA (restriction to states in the list states)

6.7.3.7 get_coaccessible_states() def wfa.core_wfa.CoreWFA.get_coaccessible_states (self, tr_dict = None)

Get coaccessible states of the WFA.

Parameters

tr_dict	Transition dictionary.

Returns

The list of coaccessible states.

6.7.3.8 get_dictionary_transitions() def wfa.core_wfa.CoreWFA.get_dictionary_transitions (self)

Get transitions in the form of dictionary (for each state there is a list of transitions leading from this state).

Returns

Dictionary assigning State -> List(Transitions)

6.7.3.9 get_finals() def wfa.core_wfa.CoreWFA.get_finals (self)

Get all final states of the WFA.

Returns

Final states with accepting weights – Dictionary: Final state -> float (weight)

6.7.3.10 get_predecessors() def wfa.core_wfa.CoreWFA.get_predecessors (self, state)

Operation that finds predessors of the state state.

Parameters

state	The state whose predessors are found.

Returns

List of predecessors

6.7.3.11 get_predecessors_transitions() def wfa.core_wfa.CoreWFA.get_predecessors_transitions (self)

Get predecessors of all states of the WFA.

Returns

Dict: State -> Set(State)

6.7.3.12 get_rename_dict() def wfa.core_wfa.CoreWFA.get_rename_dict (self)

Get the dictionary containing original state labels and renamed state labels.

The dictionary is created after method rename_states is invoked.

Returns

Dictionary: State (original) -> State (renamed).

6.7.3.13 get_rev_transitions_aut() def wfa.core_wfa.CoreWFA.get_rev_transitions_aut (self)

Get automaton with reversed directions of transitios.

Returns

WFA with reversed transitions.

6.7.3.14 get_single_dictionary_transitions() def wfa.core_wfa.CoreWFA.get_single_dictionary_↩ transitions (self)

Get the transitions (ommiting transitions that differ only on the symbol) in the form of dictinary (for each state there is a list of transitions leading from this state).

Returns

Dictionary assigning State -> List(Transitions)

6.7.3.15 get_starts() def wfa.core_wfa.CoreWFA.get_starts (self)

Get the start state (only one start state is allowed).

Returns

Start state.

6.7.3.16 get_state_symbol_dict() def wfa.core_wfa.CoreWFA.get_state_symbol_dict (self)

Get transitions in the form of dictionary (for each state there is a dictionary assigning to symbols a set of transitions)

Returns

Dictionary assigning State -> (Dictionary: Symbol -> Set of transitions)

6.7.3.17 get_states() def wfa.core_wfa.CoreWFA.get_states (self)

Get all states of the WFA (the list of states is computed from the transitions).

Returns

List of states.

6.7.3.18 get_transitions() def wfa.core_wfa.CoreWFA.get_transitions (self)

Get all transitions of the WFA.

Returns

List of all transitions

6.7.3.19 get_trim_automaton() def wfa.core_wfa.CoreWFA.get_trim_automaton (self)

Get trimed WFA.

Returns

Trimmed WFA.

6.7.3.20 is_deterministic() def wfa.core_wfa.CoreWFA.is_deterministic (self)

Is the WFA deterministic.

Returns

True deterministic, otherwise False

6.7.3.21 map_symbols() def wfa.core_wfa.CoreWFA.map_symbols (self, fnc)

Apply the function fnc on the symbols of all transitions.

Parameters

fnc	Function applied on symbols

6.7.3.22 product() def wfa.core_wfa.CoreWFA.product (self, aut)

Perform the product of two WFAs.

Parameters

aut	Second automaton for the product.

Returns

WFA representing the product of WFAs

6.7.3.23 rename_alphabet() `def wfa.core_wfa.CoreWFA.rename_alphabet ( self, dct )`

Rename alphabet of the automaton (in place).

Parameters

dct	Mapping of the new symbols

6.7.3.24 rename_states() `def wfa.core_wfa.CoreWFA.rename_states ( self )`

Rename states of the WFA.

Assign to the states numbers from 0 to n-1 (n is the number of states). The start state has number 0. The renamed and original states are stored in the states_dict dictionary.

6.7.3.25 set_all_finals() `def wfa.core_wfa.CoreWFA.set_all_finals ( self )`

Set all states to be final (all having the accepting weight 1.0)

6.7.3.26 set_finals() `def wfa.core_wfa.CoreWFA.set_finals ( self, finals )`

Set final states of the WFA.

Parameters

finals	Dictionary of final states and their weight of accepting.

6.7.3.27 set_starts() def wfa.core_wfa.CoreWFA.set_starts (self, start)

Set the initial state.

Parameters

start	New initial state

6.7.3.28 string_prob_deterministic() def wfa.core_wfa.CoreWFA.string_prob_deterministic (self, word)

Compute the probability of the word word.

Parameters

word	Word

Returns

Probability of word

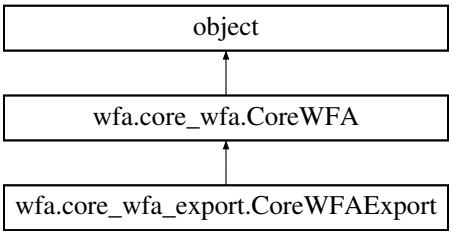
The documentation for this class was generated from the following file:

- core_wfa.py

6.8 wfa.core_wfa_export.CoreWFAExport Class Reference

Class for exporting WFAs to a text format.

Inheritance diagram for wfa.core_wfa_export.CoreWFAExport:



Public Member Functions

- def __init__ (self, transitions=None, finals=None, start=None, alphabet=None)
  Constructor.
- def get_aggregated_transitions (self)
  Get aggregated transitions (merging transitions which differs only on symbol into a transition labeled with the list of symbols).
- def to_dot (self, aggregate=True, state_label=None, legend=None)
  Convert the WFA to dot format (for graphical visualization).
- def to_fa_format (self, initial=False, alphabet=False)
  Converts automaton to FA format (WFA version).

6.8.1 Detailed Description

Class for exporting WFAs to a text format.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 __init__() def wfa.core_wfa_export.CoreWFAExport.__init__ (

```
self,
transitions = None,
finals = None,
start = None,
alphabet = None )
```

Constructor.

Parameters

transitions	Transitions
finals	Final states with weights
start	Initial state
alphabet	Alphabet

Reimplemented from wfa.core_wfa.CoreWFA.

6.8.3 Member Function Documentation

6.8.3.1 get_aggregated_transitions() def wfa.core_wfa_export.CoreWFAExport.get_aggregated_↩ transitions (

```
self )
```

Get aggregated transitions (merging transitions which differs only on symbol into a transition labeled with the list of symbols).

Returns

List of aggregated ransitions.

6.8.3.2 to_dot() def wfa.core_wfa_export.CoreWFAExport.to_dot (

```
self,
aggregate = True,
state_label = None,
legend = None )
```

Convert the WFA to dot format (for graphical visualization).

Use aggregation of transitions between same states.

Parameters

aggregate	Aggregate transitions between two states
state_label	label of each state (shown inside of the state)
legend	Optional legend to be part of the DOT automaton

Returns

String (DOT, Graphwiz format)

6.8.3.3 to_fa_format()

```
def wfa.core_wfa_export.CoreWFAExport.to_fa_format (
        self,
        initial = False,
        alphabet = False )
```

Converts automaton to FA format (WFA version).

Parameters

initial	Explicitly print the initial state
alphabet	Whether show explicitly symbols from alphabet.

Returns

String (WFA in the FA format)

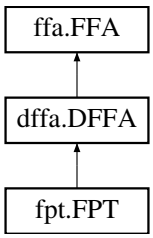
The documentation for this class was generated from the following file:

- core_wfa_export.py

6.9 dffa.DFFA Class Reference

Deterministic frequency automaton class.

Inheritance diagram for dffa.DFFA:



Public Member Functions

- def __init__ (self, states, trans, ini, fin)

  Constructor.
- def __init__ (self, states, trans, ini, fin, root)

  Constructor.
- def alergia_compatible (self, qa, qb, alpha)

  Determine whether two states are compatible for merging (wrt the parameter alpha).
- def get_root (self)

  Get the root (initial) state.
- def normalize (self)

  Normalize frequency automaton to obtain a probabilistic automaton (probabilities are in the range [0,1] with the sum-consistency condition).
- def state_freq (self, state)

  Compute frequency of a state (number of strings accepted at the state or leaving the state).
- def stochastic_fold (self, red, blue)

  Fold frequencies from subtree given by blue root into the automaton rooted at the red state.
- def stochastic_merge (self, red, blue)

  Merging two states red and blue (followed by folding frequencies from the merged subtree).

Static Public Member Functions

- def alergia_test (f1, n1, f2, n2, alpha)

  Alergia test for checking whether to merge two states.

6.9.1 Detailed Description

Deterministic frequency automaton class.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 __init__() [1/2]

```
def dffa.DFFA.__init__ (
        self,
        states,
        trans,
        ini,
        fin )
```

Constructor.

Parameters

states	States of the DFFA
trans	Transitions of the DFFA
ini	Initial states
fin	Final states

Reimplemented from ffa.FFA.

Reimplemented in fpt.FPT, and fpt.FPT.

6.9.2.2 __init__() [2/2] def dffa.DFFA.__init__ (self, states, trans, ini, fin, root)

Constructor.

Parameters

states	States of the DFFA
trans	Transitions of the DFFA
ini	Initial states
fin	Final states
root	The root state

Reimplemented from ffa.FFA.

Reimplemented in fpt.FPT, and fpt.FPT.

6.9.3 Member Function Documentation

6.9.3.1 alergia_compatible() def dffa.DFFA.alergia_compatible (self, qa, qb, alpha)

Determine whether two states are compatible for merging (wrt the parameter alpha).

Parameters

qa	The first state
qb	The second state
alpha	Merging parameter

Returns

Are two states compatible for merging

6.9.3.2 alergia_test() def dffa.DFFA.alergia_test (f1, n1, f2, n2, alpha) [static]

Alergia test for checking whether to merge two states.

Parameters

f1	Frequency of the first state
n1	Number of incomming strings of the first state
f2	Frequency of the second state
n2	Number of incomming strings of the second state
alpha	Merging parameter

Returns

Compatibility of two states/transitions (represented by freqencies)

6.9.3.3 get_root() def dffa.DFFA.get_root (self)

Get the root (initial) state.

Returns

Root (initial) state

6.9.3.4 normalize() def dffa.DFFA.normalize (self)

Normalize frequency automaton to obtain a probabilistic automaton (probabilities are in the range [0,1] with the sum-consistency condition).

Returns

Normalized automaton

6.9.3.5 state_freq() def dffa.DFFA.state_freq (self, state)

Compute frequency of a state (number of strings accepted at the state or leaving the state).

Parameters

state	Given state

Returns

Frequency of a state

6.9.3.6 stochastic_fold() `def dffa.DFFA.stochastic_fold (`

```
self,
red,
blue )
```

Fold frequencies from subtree given by blue root into the automaton rooted at the red state.

Parameters

red	Red state
blue	Blue state

6.9.3.7 stochastic_merge() `def dffa.DFFA.stochastic_merge (`

```
self,
red,
blue )
```

Merging two states red and blue (followed by folding frequencies from the merged subtree).

Parameters

red	Red state
blue	Blue state

The documentation for this class was generated from the following file:

- dffa.py

6.10 distance.Distance Class Reference

Class removing items from a set causing the minimum error.

Public Member Functions

- def __init__ (self, dists, pts)

  Constructor.
- def compute_subset_error (self, max_error)

  Get subset of items that meets the max_error bound.

Public Attributes

- dist
- points

6.10.1 Detailed Description

Class removing items from a set causing the minimum error.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 __init__() `def distance.Distance.__init__ (`

```
self,
dists,
pts )
```

Constructor.

Parameters

dists	Distances between items
pts	Items in the set

6.10.3 Member Function Documentation

6.10.3.1 compute_subset_error() `def distance.Distance.compute_subset_error (`

```
self,
max_error )
```

Get subset of items that meets the max_error bound.

Parameters

max_error	Maximum allowed error

Returns

: Subset of items causing error less that max_error

6.10.4 Member Data Documentation

6.10.4.1 dist `distance.Distance.dist`

6.10.4.2 points `distance.Distance.points`

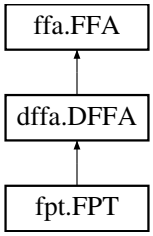
The documentation for this class was generated from the following file:

- distance.py

6.11 ffa.FFA Class Reference

General frequency automata (FFA)

Inheritance diagram for ffa.FFA:



Public Member Functions

- def __init__ (self, states, trans, ini, fin)

  Constructor.
- def get_finals (self)

  Get final states.
- def get_states (self)

  Get all states.
- def get_transition_list (self)

  Get list of transitions from the transition function.
- def get_transitions (self)

  Get transitions.
- def inverse_ffa (self)

  Get the inverse FFA.
- def merge_equivalent (self, classes)

  Merge equivalent states according to the equivalent classes.
- def merge_states (self, states)

  Merge a set of states (remove those states and replace with one in the set)
- def path_length (self, st1, st2)

  Get length of a shortest path between st1 and st2.
- def reachable_states (self, st_set)

  Get all reachable states from st_set.
- def rename_states (self)
- def successors (self, state, sym=None)

  Get all successors from state over sym.
- def successors_set (self, states, sym=None)

  Get all successors from the set states over sym.
- def to_graphiwiz (self, legend=None)

  Convert the WFA to graphwiz format (for graphical visualization).
- def to_wfa (self)

  Converts FFA to WFA (weighted finite automaton)
- def trim (self)

6.11.1 Detailed Description

General frequency automata (FFA)

6.11.2 Constructor & Destructor Documentation

6.11.2.1 __init__() def ffa.FFA.__init__ (self, states, trans, ini, fin)

Constructor.

Parameters

states	States of the DFFA
trans	Transitions of the DFFA
ini	Initial states
fin	Final states

Reimplemented in fpt.FPT, dffa.DFFA, fpt.FPT, and dffa.DFFA.

6.11.3 Member Function Documentation

6.11.3.1 get_finals() def ffa.FFA.get_finals (self)

Get final states.

Returns

Final states of the FFA

6.11.3.2 get_states() def ffa.FFA.get_states (self)

Get all states.

Returns

All states of the FFA

6.11.3.3 get_transition_list() def ffa.FFA.get_transition_list (self)

Get list of transitions from the transition function.

Returns

List of transitions

6.11.3.4 get_transitions() def ffa.FFA.get_transitions (self)

Get transitions.

Returns

Transitions of the FFA

6.11.3.5 inverse_ffa() def ffa.FFA.inverse_ffa (self)

Get the inverse FFA.

Returns

FFA with the inverse transition function

6.11.3.6 merge_equivalent() def ffa.FFA.merge_equivalent (self, classes)

Merge equivalent states according to the equivalent classes.

Parameters

classes	Partitioning of the states

6.11.3.7 merge_states() def ffa.FFA.merge_states (self, states)

Merge a set of states (remove those states and replace with one in the set)

Parameters

states	States to be merged

6.11.3.8 path_length() `def ffa.FFA.path_length (`

```
self,
st1,
st2 )
```

Get length of a shortest path between st1 and st2.

Parameters

st1	Source state
st2	Destination state

Returns

Length of a shortest path

6.11.3.9 reachable_states() `def ffa.FFA.reachable_states (`

```
self,
st_set )
```

Get all reachable states from st_set.

Parameters

st_set	Set of states

Returns

Set of reachable states

6.11.3.10 rename_states() `def ffa.FFA.rename_states (`

```
self )
```

```
Rename states to consecutive numbers (from 0)
```

6.11.3.11 successors() def ffa.FFA.successors (self, state, sym = None)

Get all successors from state over sym.

Parameters

state	State
sym	Symbol

Returns

Set of all successors

6.11.3.12 successors_set() def ffa.FFA.successors_set (self, states, sym = None)

Get all successors from the set states over sym.

Parameters

states	State
sym	Symbol

Returns

Set of all successors

6.11.3.13 to_graphiwiz() def ffa.FFA.to_graphiwiz (self, legend = None)

Convert the WFA to graphwiz format (for graphical visualization).

Parameters

legend	Legend to be print in the figure

Returns

Graphwiz format of the automaton

6.11.3.14 to_wfa() `def ffa.FFA.to_wfa ( self )`

Converts FFA to WFA (weighted finite automaton)

Returns

FFA represented as WFA

6.11.3.15 trim() `def ffa.FFA.trim ( self )`

```
Remove unreachable states from the automaton.
```

The documentation for this class was generated from the following file:

- ffa.py

6.12 ffa.FFATrans Class Reference

Class representing a transtion of the FFA.

6.12.1 Detailed Description

Class representing a transtion of the FFA.

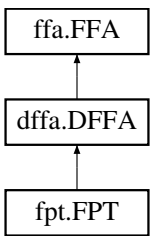
The documentation for this class was generated from the following file:

- ffa.py

6.13 fpt.FPT Class Reference

Frequency prefix tree (FPT)

Inheritance diagram for fpt.FPT:



Public Member Functions

- def __init__ (self)

  Default constructor.
- def __init__ (self, states, trans, ini, fin)

  Constructor.
- def __str__ (self)

  Convert to a string representation.
- def add_string (self, string, label=0)

  Add string to the frequency prefix tree.
- def add_string_list (self, lst, label=0)

  Add a list of strings to frequency prefix tree.
- def count_label_edges (self, label)

  Count edges with labels corresponding to label.
- def get_leaves (self)

  Get leaves (states without outgoing transitions)
- def show (self)

  Convert the FPT to a string representation.
- def suffix_minimize (self)

  Merge equivalent backward deterministic states.

Public Attributes

- flanguages

Additional Inherited Members

6.13.1 Detailed Description

Frequency prefix tree (FPT)

6.13.2 Constructor & Destructor Documentation

6.13.2.1 __init__() [1/2] def fpt.FPT.__init__ (self, states, trans, ini, fin)

Constructor.

Parameters

states	States of the DFFA
trans	Transitions of the DFFA
ini	Initial states
fin	Final states

Reimplemented from dffa.DFFA.

6.13.2.2 __init__() [2/2] def fpt.FPT.__init__ (self)

Default constructor.

Reimplemented from dffa.DFFA.

6.13.3 Member Function Documentation

6.13.3.1 __str__() def fpt.FPT.__str__ (self)

Convert to a string representation.

6.13.3.2 add_string() def fpt.FPT.add_string (self, string, label = 0)

Add string to the frequency prefix tree.

Parameters

string	String to be added to the FPT
label	Label of the new added string

6.13.3.3 add_string_list() def fpt.FPT.add_string_list (self, lst, label = 0)

Add a list of strings to frequency prefix tree.

Parameters

lst	List of strings to be added to the FPT
label	Label of the new added string

6.13.3.4 count_label_edges() def fpt.FPT.count_label_edges (self, label)

Count edges with labels corresponding to label.

Parameters

label	Label of an edge

Returns

Number of edge labelled by label

6.13.3.5 get_leaves() def fpt.FPT.get_leaves (self)

Get leaves (states without outgoing transitions)

Returns

Set of leaves

6.13.3.6 show() def fpt.FPT.show (self)

Convert the FPT to a string representation.

Returns

String representation of the FPT

6.13.3.7 suffix_minimize() def fpt.FPT.suffix_minimize (self)

Merge equivalent backward deterministic states.

6.13.4 Member Data Documentation

6.13.4.1 flanguages `fpt.FPT.flanguages`

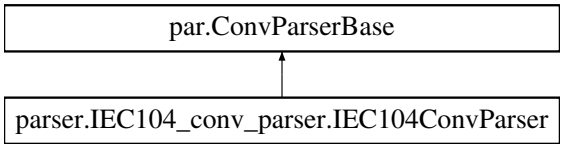
The documentation for this class was generated from the following file:

- fpt.py

6.14 parser.IEC104_conv_parser.IEC104ConvParser Class Reference

Class for parsing IEC104 conversations from already divided messages.

Inheritance diagram for parser.IEC104_conv_parser.IEC104ConvParser:



Public Member Functions

- def __init__ (self, inp, pr=None)
  Constructor taking a list of messages (each message is a dictionary)
- def get_all_conversations (self, proj=None)
  Get all conversations (possibly filter by communication pairs)
- def get_conversation (self)
  Get a following conversation from already divided messages.
- def get_line (self)
  Get a next line.
- def parse_conversations (self)
  Parse and store all conversations.
- def parse_data (self, data)
  Parse data.
- def split_communication_pairs (self)
  Split input according to the communication pairs.
- def split_to_windows (self, dur)
  Split input according to time windows.

Public Attributes

- compair
- conversations
- index
- input

6.14.1 Detailed Description

Class for parsing IEC104 conversations from already divided messages.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 __init__() def parser.IEC104_conv_parser.IEC104ConvParser.__init__ (self, inp, pr = None)

Constructor taking a list of messages (each message is a dictionary)

Parameters

inp	Input list of messages
pr	A communication pair

6.14.3 Member Function Documentation

6.14.3.1 get_all_conversations() def parser.IEC104_conv_parser.IEC104ConvParser.get_all_conversations (self, proj = None)

Get all conversations (possibly filter by communication pairs)

Parameters

proj	Projection on the messages

Returns

All parsed conversations

6.14.3.2 get_conversation() def parser.IEC104_conv_parser.IEC104ConvParser.get_conversation (self)

Get a following conversation from already divided messages.

Returns

Parsed conversation

6.14.3.3 get_line() def parser.IEC104_conv_parser.IEC104ConvParser.get_line (self)

Get a next line.

Returns

Next line of the buffer

6.14.3.4 parse_conversations() def parser.IEC104_conv_parser.IEC104ConvParser.parse_conversations (self)

Parse and store all conversations.

6.14.3.5 parse_data() def parser.IEC104_conv_parser.IEC104ConvParser.parse_data (self, data)

Parse data.

Parameters

data	Input to be parsed

Returns

List of parsed values

6.14.3.6 split_communication_pairs() def parser.IEC104_conv_parser.IEC104ConvParser.split_communication_pairs (self)

Split input according to the communication pairs.

Returns

List of intances of IEC104ConvParser each for one communication pair

6.14.3.7 split_to_windows() def parser.IEC104_conv_parser.IEC104ConvParser.split_to_windows (self, dur)

Split input according to time windows.

Returns

List of intances of IEC104ConvParser each for one window

6.14.4 Member Data Documentation

6.14.4.1 compair parser.IEC104_conv_parser.IEC104ConvParser.compair

6.14.4.2 conversations parser.IEC104_conv_parser.IEC104ConvParser.conversations

6.14.4.3 index parser.IEC104_conv_parser.IEC104ConvParser.index

6.14.4.4 input parser.IEC104_conv_parser.IEC104ConvParser.input

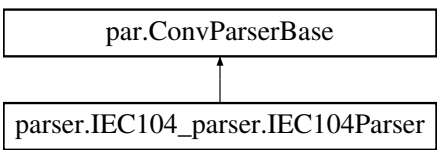
The documentation for this class was generated from the following file:

- IEC104_conv_parser.py

6.15 parser.IEC104_parser.IEC104Parser Class Reference

Class for parsing IEC104 conversations.

Inheritance diagram for parser.IEC104_parser.IEC104Parser:



Public Member Functions

- def __init__ (self, inp, pr=None)

  Constructor taking a list of messages (each message is a dictionary)
- def get_all_conversations (self, proj=None)

  Get all conversations (possibly filter by communication pairs)
- def get_conversation (self)

  Get a following conversation from a list of messages.
- def get_symbol (self, buff_read)

  Get a next message from the buffer.
- def is_conversation_complete (self, conv)

  Check if a given conversation is complete (according to the last packet).
- def parse_conversations (self)

  Parse and store all conversations.
- def return_symbol (self, val, buff_read)

  Return the message to the buffer.
- def split_communication_pairs (self)

  Split input according to the communication pairs.
- def split_to_windows (self, dur)

  Split input according to time windows.

Static Public Member Functions

- def get_initial_type (row)

  Get initial type of a conversation.
- def in_middle_range (row, tp)

  Is the message in the middle of a conversation.
- def is_final (row, tp)

  Is the message final.
- def is_inform_message (row)

  Is the message informal?
- def is_msg_match (compair, val)

  Does the message match communication pair restriction?
- def is_spontaneous (row)

  Is the message spontaneous?

Public Attributes

- buffer
- compair
- conversations
- incomplete
- index
- input

6.15.1 Detailed Description

Class for parsing IEC104 conversations.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 __init__() def parser.IEC104_parser.IEC104Parser.__init__ (self, inp, pr = None)

Constructor taking a list of messages (each message is a dictionary)

Parameters

inp	Input list of messages
pr	A communication pair

6.15.3 Member Function Documentation

6.15.3.1 get_all_conversations() def parser.IEC104_parser.IEC104Parser.get_all_conversations (self, proj = None)

Get all conversations (possibly filter by communication pairs)

Parameters

proj	Projection on the messages

Returns

All parsed conversations

6.15.3.2 get_conversation() def parser.IEC104_parser.IEC104Parser.get_conversation (self)

Get a following conversation from a list of messages.

It implements just a couple of cases (definitely not all of them)

Returns

Parsed conversation

6.15.3.3 get_initial_type() def parser.IEC104_parser.IEC104Parser.get_initial_type (row) [static]

Get initial type of a conversation.

Parameters

row	Message

Returns

Type of the conversation initialized by the message row

6.15.3.4 get_symbol() def parser.IEC104_parser.IEC104Parser.get_symbol (self, buff_read)

Get a next message from the buffer.

Parameters

buff_read	Buffer

Returns

Next message in the buffer

6.15.3.5 in_middle_range() def parser.IEC104_parser.IEC104Parser.in_middle_range (row, tp) [static]

Is the message in the middle of a conversation.

Parameters

row	Message
tp	Type of the conversation

Returns

True – the message is in the middle of a conversation of that type

6.15.3.6 is_conversation_complete() def parser.IEC104_parser.IEC104Parser.is_conversation_↩ complete (self, conv)

Check if a given conversation is complete (according to the last packet).

Parameters

conv	Parsed conversation

Returns

: True – the message is complete

6.15.3.7 is_final() def parser.IEC104_parser.IEC104Parser.is_final (row, tp) [static]

Is the message final.

Parameters

row	Message
tp	Type of the conversation

Returns

True – the message is final

6.15.3.8 is_inform_message() def parser.IEC104_parser.IEC104Parser.is_inform_message (row) [static]

Is the message informal?

Parameters

row	Message

Returns

True – informal message

6.15.3.9 is_msg_match() def parser.IEC104_parser.IEC104Parser.is_msg_match (compair, val) [static]

Does the message match communication pair restriction?

Parameters

compair	A communication pair (IP, port)
val	A message

Returns

Is the message sent by the compair?

6.15.3.10 is_spontaneous() def parser.IEC104_parser.IEC104Parser.is_spontaneous (row) [static]

Is the message spontaneous?

Parameters

row	Message

Returns

True – spontaneous message

6.15.3.11 parse_conversations() def parser.IEC104_parser.IEC104Parser.parse_conversations (self)

Parse and store all conversations.

6.15.3.12 return_symbol() def parser.IEC104_parser.IEC104Parser.return_symbol (self, val, buff_read)

Return the message to the buffer.

Parameters

val	Value to be inserted
buff_read	Is it read from the buffer

6.15.3.13 split_communication_pairs() def parser.IEC104_parser.IEC104Parser.split_communication↩

```
_pairs (
            self )
```

Split input according to the communication pairs.

Returns

List of intances of IEC104Parser each for one communication pair

6.15.3.14 split_to_windows() def parser.IEC104_parser.IEC104Parser.split_to_windows (

```
            self,
            dur )
```

Split input according to time windows.

Returns

List of intances of IEC104Parser each for one window

6.15.4 Member Data Documentation

6.15.4.1 buffer parser.IEC104_parser.IEC104Parser.buffer

6.15.4.2 compair parser.IEC104_parser.IEC104Parser.compair

6.15.4.3 conversations parser.IEC104_parser.IEC104Parser.conversations

6.15.4.4 incomplete parser.IEC104_parser.IEC104Parser.incomplete

6.15.4.5 index parser.IEC104_parser.IEC104Parser.index

6.15.4.6 input `parser.IEC104_parser.IEC104Parser.input`

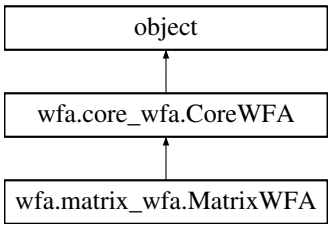
The documentation for this class was generated from the following file:

- IEC104_parser.py

6.16 wfa.matrix_wfa.MatrixWFA Class Reference

Class for matrix operations with WFAs involving matrix operations.

Inheritance diagram for wfa.matrix_wfa.MatrixWFA:



Public Member Functions

- def __init__ (self, transitions=None, finals=None, start=None, alphabet=None)

  Constructor.
- def are_states_compatible (self)

  Check whether the states of the WFA are compatible with matrix operations (states are labeled with consequtive numbers from 0 to n-1).
- def compute_language_probability (self, closure_mode, sparse=False, iterations=0, debug=False)

  Compute the total probability of the WFA's language.
- def compute_transition_closure (self, closure_mode, sparse=False, iterations=0, debug=False)

  Compute transition closure by a specified method (assume that the conditions for given method are met).
- def get_final_ones (self, sparse=False)

  Get a vector with items 1.0 corresponding to final states (other states are set to 0).
- def get_final_vector (self, sparse=False)

  Get a vector with final weights corresponding to the WFA.
- def get_initial_vector (self, sparse=False)

  Get a vector of initial weights.
- def get_transition_matrix (self, sparse=False)

  Get a transition matrix corresponding to the WFA.

6.16.1 Detailed Description

Class for matrix operations with WFAs involving matrix operations.

6.16.2 Constructor & Destructor Documentation

6.16.2.1 __init__() def wfa.matrix_wfa.MatrixWFA.__init__ (

```
self,
transitions = None,
finals = None,
start = None,
alphabet = None )
```

Constructor.

Parameters

transitions	Transitions
finals	Final states with weights
start	Initial state
alphabet	Alphabet

Reimplemented from wfa.core_wfa.CoreWFA.

6.16.3 Member Function Documentation

6.16.3.1 are_states_compatible() `def wfa.matrix_wfa.MatrixWFA.are_states_compatible ( self )`

Check whether the states of the WFA are compatible with matrix operations (states are labeled with consequtive numbers from 0 to n-1).

Returns

Compatibility of states

6.16.3.2 compute_language_probability() `def wfa.matrix_wfa.MatrixWFA.compute_language_probability ( self, closure_mode, sparse = False, iterations = 0, debug = False )`

Compute the total probability of the WFA's language.

Parameters

closure_mode	Method for computing the transition closure (ClosureMode).
sparse	Use sparse matrices
iterations	Maximum number of iteration (in the case of iterative methods).
debug	Show debug info.

Returns

Weight of the language (float)

6.16.3.3 compute_transition_closure() def wfa.matrix_wfa.MatrixWFA.compute_transition_closure (self, closure_mode, sparse = False, iterations = 0, debug = False)

Compute transition closure by a specified method (assume that the conditions for given method are met).

Parameters

closure_mode	Method for computing the transition closure (ClosureMode).
sparse	Use sparse matrices
iterations	Maximum number of iteration (in the case of iterative methods).
debug	Show debug info.

Returns

Transition closure (Numpy.matrix)

6.16.3.4 get_final_ones() def wfa.matrix_wfa.MatrixWFA.get_final_ones (self, sparse = False)

Get a vector with items 1.0 corresponding to final states (other states are set to 0).

Parameters

sparse	Use sparse matrices

Returns

Numpy.matrix (final states are set to one).

6.16.3.5 get_final_vector() def wfa.matrix_wfa.MatrixWFA.get_final_vector (self, sparse = False)

Get a vector with final weights corresponding to the WFA.

Parameters

sparse	Use sparse matrices

Returns

Final vector (Numpy.matrix)

6.16.3.6 get_initial_vector() `def wfa.matrix_wfa.MatrixWFA.get_initial_vector ( self, sparse = False )`

Get a vector of initial weights.

Parameters

sparse	Use sparse matrices

Returns

Vector of initial weights (Numpy.matrix).

6.16.3.7 get_transition_matrix() `def wfa.matrix_wfa.MatrixWFA.get_transition_matrix ( self, sparse = False )`

Get a transition matrix corresponding to the WFA.

Parameters

sparse	Use sparse matrices

Returns

Transition matrix (Numpy.matrix)

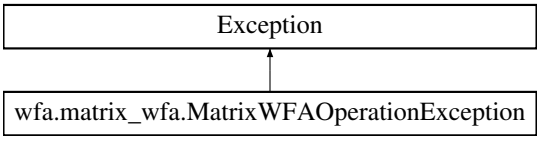
The documentation for this class was generated from the following file:

- matrix_wfa.py

6.17 wfa.matrix_wfa.MatrixWFAOperationException Class Reference

Exception for invalid operations and errors during the closure computing.

Inheritance diagram for wfa.matrix_wfa.MatrixWFAOperationException:



Public Member Functions

- def __init__ (self, msg)

  Constructor.
- def __str__ (self)

  Convert to string.

Public Attributes

- msg

6.17.1 Detailed Description

Exception for invalid operations and errors during the closure computing.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 __init__() `def wfa.matrix_wfa.MatrixWFAOperationException.__init__ ( self, msg )`

Constructor.

Parameters

msg	Error message

6.17.3 Member Function Documentation

6.17.3.1 __str__() `def wfa.matrix_wfa.MatrixWFAOperationException.__str__ ( self )`

Convert to string.

Returns

Error message

6.17.4 Member Data Documentation

6.17.4.1 msg `wfa.matrix_wfa.MatrixWFAOperationException.msg`

The documentation for this class was generated from the following file:

- matrix_wfa.py

6.18 packet_loss.PacketLoss Class Reference

Language-based approach for a detection of packet losses.

Static Public Member Functions

- def compatible_strings (str1, str2)
  Compute edit distance (assuming only the delete operation) between two strings.

6.18.1 Detailed Description

Language-based approach for a detection of packet losses.

6.18.2 Member Function Documentation

6.18.2.1 compatible_strings() `def packet_loss.PacketLoss.compatible_strings ( str1, str2 ) [static]`

Compute edit distance (assuming only the delete operation) between two strings.

Parameters

str1	First string
str2	Second string

Returns

edit distance of str1 and str2 (can be used to compute the number of lost packets)

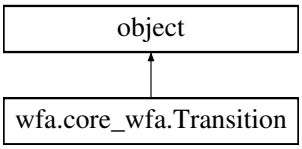
The documentation for this class was generated from the following file:

- packet_loss.py

6.19 wfa.core_wfa.Transition Class Reference

Class for the represention of a WFA transition.

Inheritance diagram for wfa.core_wfa.Transition:



Public Member Functions

- def __eq__ (self, other)
  Equality of two transitions.
- def __hash__ (self)
  Hash method.
- def __init__ (self, src, dest, sym, weight)
  Constructor.
- def __ne__ (self, other)
  Inequality of two transitions.
- def __repr__ (self)
  String representation.
- def __str__ (self)
  String representation.

Public Attributes

- count
- dest
- src
- symbol
- weight

6.19.1 Detailed Description

Class for the represention of a WFA transition.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 __init__() def wfa.core_wfa.Transition.__init__ (self, src, dest, sym, weight)

Constructor.

Parameters

src	Source state
dest	Destination state
sym	Symbol
weight	Weight of the transition

6.19.3 Member Function Documentation

6.19.3.1 __eq__() `def wfa.core_wfa.Transition.__eq__ ( self, other )`

Equality of two transitions.

Parameters

other	Other transition

Returns

True – both transitions are equal

6.19.3.2 __hash__() `def wfa.core_wfa.Transition.__hash__ ( self )`

Hash method.

Returns

Hash

6.19.3.3 __ne__() `def wfa.core_wfa.Transition.__ne__ ( self, other )`

Inequality of two transitions.

Parameters

other	Other transition

Returns

True – both transitions are NOT equal

6.19.3.4 __repr__() def wfa.core_wfa.Transition.__repr__ (self)

String representation.

Returns

String representation of the transition

6.19.3.5 __str__() def wfa.core_wfa.Transition.__str__ (self)

String representation.

Returns

String representation of the transition

6.19.4 Member Data Documentation

6.19.4.1 count wfa.core_wfa.Transition.count

6.19.4.2 dest wfa.core_wfa.Transition.dest

6.19.4.3 src wfa.core_wfa.Transition.src

6.19.4.4 symbol wfa.core_wfa.Transition.symbol

6.19.4.5 weight `wfa.core_wfa.Transition.weight`

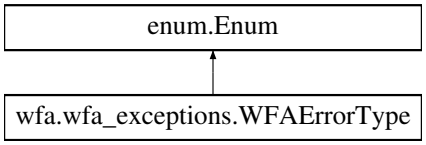
The documentation for this class was generated from the following file:

- core_wfa.py

6.20 wfa.wfa_exceptions.WFAErrorType Class Reference

Error types for WFAs.

Inheritance diagram for wfa.wfa_exceptions.WFAErrorType:



Static Public Attributes

- int general_error = 0
  General error.
- int not_DAG = 1
  Not directed acyclic graph.

6.20.1 Detailed Description

Error types for WFAs.

6.20.2 Member Data Documentation

6.20.2.1 general_error `int wfa.wfa_exceptions.WFAErrorType.general_error = 0 [static]`

General error.

6.20.2.2 not_DAG `int wfa.wfa_exceptions.WFAErrorType.not_DAG = 1 [static]`

Not directed acyclic graph.

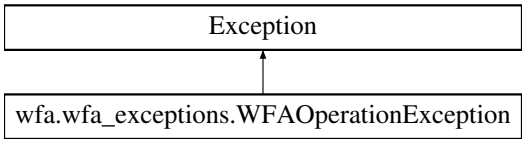
The documentation for this class was generated from the following file:

- wfa_exceptions.py

6.21 wfa.wfa_exceptions.WFAOperationException Class Reference

Exception used when an error during parsing is occured.

Inheritance diagram for wfa.wfa_exceptions.WFAOperationException:



Public Member Functions

- def __init__ (self, msg, err_type=WFAErrorType.general_error)
  Constructor.
- def __str__ (self)
  Convert to string.

Public Attributes

- err_type
- msg

6.21.1 Detailed Description

Exception used when an error during parsing is occured.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 __init__() `def wfa.wfa_exceptions.WFAOperationException.__init__ (`

```
self,
msg,
err_type = WFAErrorType.general_error )
```

Constructor.

Parameters

msg	Error message
err_type	Error Type

6.21.3 Member Function Documentation

6.21.3.1 __str__() `def wfa.wfa_exceptions.WFAOperationException.__str__ ( self )`

Convert to string.

Returns

Error message

6.21.4 Member Data Documentation

6.21.4.1 err_type `wfa.wfa_exceptions.WFAOperationException.err_type`

6.21.4.2 msg `wfa.wfa_exceptions.WFAOperationException.msg`

The documentation for this class was generated from the following file:

- wfa_exceptions.py

7 File Documentation

7.1 distance.py File Reference

Classes

- class distance.Distance

  Class removing items from a set causing the minimum error.

Namespaces

- namespace distance

  Class for removing similar automata in a set.

7.2 anom_detect_base.py File Reference

Classes

- class anom_detect_base.AnomDetectBase

  Base class providing an interface for concrete detections.

Namespaces

- namespace anom_detect_base
  Anomaly detection base class.

7.3 distr_comparison.py File Reference

Classes

- class distr_comparison.AnomDistrComparison
  Anomaly detection based on comparing distributions.

Namespaces

- namespace distr_comparison
  Distribution-based anomaly detection.

Variables

- bool distr_comparison.SPARSE = False
  Use sparse matrices to comput the Euclid distance.

7.4 member.py File Reference

Classes

- class member.AnomMember
  Anomaly detection based on a single message reasoning.

Namespaces

- namespace member
  Member-based anomaly detection.

7.5 packet_loss.py File Reference

Classes

- class packet_loss.PacketLoss
  Language-based approach for a detection of packet losses.

Namespaces

- namespace packet_loss
  Packet-loss detection.

7.6 alergia.py File Reference

Namespaces

- namespace alergia

  Alergia algorithm.

Functions

- def alergia.alergia (freq_aut, alpha, t0)

  PA learning using the Alergia algorithm.
- def alergia.choose_blue_state (freq_aut, blue_set, t0)

  Chose a blue state from a set of blue states.
- def alergia.choose_red_state (freq_aut, red_set, blue, alpha)

  Chose a red state from a set of red states.

7.7 dffa.py File Reference

Classes

- class dffa.DFFA

  Deterministic frequency automaton class.

Namespaces

- namespace dffa

  Class for deterministic frequency automata.

7.8 ffa.py File Reference

Classes

- class ffa.FFA

  General frequency automata (FFA)
- class ffa.FFATrans

  Class representing a transtion of the FFA.

Namespaces

- namespace ffa

  Class for general frequency automata.

7.9 fpt.py File Reference

Classes

- class fpt.FPT

  Frequency prefix tree (FPT)

Namespaces

- namespace fpt

  Class for frequency prefix tree automataa.

7.10 __init__.py File Reference

7.11 __init__.py File Reference

7.12 conversation_parser_base.py File Reference

Classes

- class parser.conversation_parser_base.ConvParserBase

  Base class for parsing conversations.

Namespaces

- namespace parser
- namespace parser.conversation_parser_base

  Dividing list of messages into conversations – base class.

7.13 IEC104_conv_parser.py File Reference

Classes

- class parser.IEC104_conv_parser.IEC104ConvParser

  Class for parsing IEC104 conversations from already divided messages.

Namespaces

- namespace parser
- namespace parser.IEC104_conv_parser

  Parsing files with already divided conversations.

7.14 IEC104_parser.py File Reference

Classes

- class parser.IEC104_parser.ConvType

  Type of a conversation.
- class parser.IEC104_parser.IEC104Parser

  Class for parsing IEC104 conversations.

Namespaces

- namespace parser
- namespace parser.IEC104_parser

  Dividing list of messages into conversations.

Functions

- def parser.IEC104_parser.get_messages (fd)

  Get all messages from a csv file.

7.15 aux_functions.py File Reference

Namespaces

- namespace wfa
- namespace wfa.aux_functions

  Auxiliary functions for WFAs.

Functions

- def wfa.aux_functions.convert_to_pritable (dec, dot=False)

  Convert string containing also non-printable characters to printable hexa number.

7.16 core_wfa.py File Reference

Classes

- class wfa.core_wfa.CoreWFA

  Basic class for representation of WFA.
- class wfa.core_wfa.Transition

  Class for the represention of a WFA transition.

Namespaces

- namespace wfa
- namespace wfa.core_wfa

  Core class for working with WFAs.

7.17 core_wfa_export.py File Reference

Classes

- class wfa.core_wfa_export.CoreWFAExport

  Class for exporting WFAs to a text format.

Namespaces

- namespace wfa
- namespace wfa.core_wfa_export

  Class for exporting WFAs in a textual format.

Variables

- int wfa.core_wfa_export.PRECISE = 3

  Precise of float numbers (for output)
- int wfa.core_wfa_export.SYMBOLS = 25

  Max number of symbols on transition (DOT format)

7.18 matrix_wfa.py File Reference

Classes

- class wfa.matrix_wfa.ClosureMode

  Ignore a particular warning.
- class wfa.matrix_wfa.MatrixWFA

  Class for matrix operations with WFAs involving matrix operations.
- class wfa.matrix_wfa.MatrixWFAOperationException

  Exception for invalid operations and errors during the closure computing.

Namespaces

- namespace wfa
- namespace wfa.matrix_wfa

  Class for working with a computation of language weights.

Variables

- float wfa.matrix_wfa.THRESHOLD = 0.0

  Threshold for sparse matrices.

7.19 wfa_exceptions.py File Reference

Classes

- class wfa.wfa_exceptions.WFAErrorType

  Error types for WFAs.
- class wfa.wfa_exceptions.WFAOperationException

  Exception used when an error during parsing is occured.

Namespaces

- namespace wfa
- namespace wfa.wfa_exceptions

  Exception class for specifying errors when working with WFAs.

Index