Detano: Library for Automata-based Detection of Network Anomalies

Generated by Doxygen 1.9.3

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

- dffa.DFFA alergia (dffa.DFFA freq_aut, float alpha, int t0)

  PA learning using the Alergia algorithm.
- Optional[ffa.StateType] choose_blue_state (dffa.DFFA freq_aut, Set[ffa.StateType] blue_set, int t0)

  Chose a blue state from a set of blue states.
- Optional[ffa.StateType] choose_red_state (dffa.DFFA freq_aut, Set[ffa.StateType] red_set, ffa.StateType blue, float 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()

```
dffa.DFFA alergia.alergia (
        dffa.DFFA freq_aut,
        float alpha,
        int 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()

```
Optional[ffa.StateType] alergia.choose_blue_state (
        dffa.DFFA freq_aut,
        Set[ffa.StateType] blue_set,
        int 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()

```
Optional[ffa.StateType] alergia.choose_red_state (
        dffa.DFFA freq_aut,
        Set[ffa.StateType] red_set,
        ffa.StateType blue,
        float 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.

Variables

- ComPairType = FrozenSet[Tuple[str,str]]

5.2.1 Detailed Description

Anomaly detection base class.

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

Author

Vojtěch Havlena

Copyright

5.2.2 Variable Documentation

5.2.2.1 ComPairType `anom_detect_base.ComPairType = FrozenSet[Tuple[str,str]]`

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.

Variables

- DistType = dict[Tuple[T, T], float]
- SortedDictType = List[Tuple[Tuple[T,T], float]]
- T = TypeVar("T")

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

5.4.2 Variable Documentation

5.4.2.1 DistType `distance.DistType = dict[Tuple[T, T], float]`

5.4.2.2 SortedDictType `distance.SortedDictType = List[Tuple[Tuple[T,T], float]]`

5.4.2.3 T `distance.T = TypeVar("T")`

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.

Variables

- StateType = str
- StateWeightType = dict[StateType, int]
- SymbolType = TypeVar("SymbolType")
- TransFuncDetType = dict[StateType, dict[str, "FFATrans"]]
- TransFuncMixType = Union[dict[StateType, dict[str, Set["FFATrans"]]], dict[StateType, dict[str, "FFATrans"]]]
- TransFuncType = dict[StateType, dict[str, Set["FFATrans"]]]

5.6.1 Detailed Description

Class for general frequency automata.

Class providing operations for general (nondeterministic) frequency automata.

Author

Vojtěch Havlena

Copyright

5.6.2 Variable Documentation

5.6.2.1 StateType `ffa.StateType = str`

5.6.2.2 StateWeightType `ffa.StateWeightType = dict[StateType, int]`

5.6.2.3 SymbolType `ffa.SymbolType = TypeVar("SymbolType")`

5.6.2.4 TransFuncDetType `ffa.TransFuncDetType = dict[StateType, dict[str, "FFATrans"]]`

5.6.2.5 TransFuncMixType `ffa.TransFuncMixType = Union[ dict[StateType, dict[str, Set["FFATrans"]]], dict[StateType, dict[str, "FFATrans"]]]`

5.6.2.6 TransFuncType `ffa.TransFuncType = dict[StateType, dict[str, Set["FFATrans"]]]`

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.

Variables

- ConvBaseType = List[ItemType]
- ItemType = TypeVar("ItemType")

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

5.11.2 Variable Documentation

5.11.2.1 ConvBaseType `parser.conversation_parser_base.ConvBaseType = List[ItemType]`

5.11.2.2 ItemType `parser.conversation_parser_base.ItemType = TypeVar("ItemType")`

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.

Variables

- ComPairType = FrozenSet[Tuple[str,str]]
- ConvStrType = List[ConvSymbolType]
- ConvSymbolType = Tuple[str, str]
- RowType = Dict[str, str]

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

5.12.2 Variable Documentation

5.12.2.1 ComPairType `parser.IEC104_conv_parser.ComPairType = FrozenSet[Tuple[str,str]]`

5.12.2.2 ConvStrType `parser.IEC104_conv_parser.ConvStrType = List[ConvSymbolType]`

5.12.2.3 ConvSymbolType `parser.IEC104_conv_parser.ConvSymbolType = Tuple[str, str]`

5.12.2.4 RowType `parser.IEC104_conv_parser.RowType = Dict[str, str]`

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

- List[ConvSymbolType] get_messages (fd)

  Get all messages from a csv file.

Variables

- ComPairType = FrozenSet[Tuple[str,str]]
- ConvStrType = List[ConvSymbolType]
- ConvSymbolType = Dict[str, str]

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() `List[ConvSymbolType] parser.IEC104_parser.get_messages ( fd )`

Get all messages from a csv file.

Parameters

fd	File descriptor

Returns

Messages from the csv file fd

5.13.3 Variable Documentation

5.13.3.1 ComPairType `parser.IEC104_parser.ComPairType = FrozenSet[Tuple[str,str]]`

5.13.3.2 ConvStrType `parser.IEC104_parser.ConvStrType = List[ConvSymbolType]`

5.13.3.3 ConvSymbolType `parser.IEC104_parser.ConvSymbolType = Dict[str, str]`

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

- str convert_to_pritable (str dec, bool 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()

```
str wfa.aux_functions.convert_to_pritable (
        str dec,
        bool 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.

Variables

- StateFloatMapOptType = Optional[dict[StateType, float]]
- StateFloatMapType = dict[StateType, float]
- StateType = TypeVar("StateType")
- SymbolType = TypeVar("SymbolType")
- TransFunctionType = dict[StateType, dict[SymbolType, Set[StateType]]]

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

5.16.2 Variable Documentation

5.16.2.1 StateFloatMapOptType `wfa.core_wfa.StateFloatMapOptType = Optional[dict[StateType, float]]`

5.16.2.2 StateFloatMapType `wfa.core_wfa.StateFloatMapType = dict[StateType, float]`

5.16.2.3 StateType `wfa.core_wfa.StateType = TypeVar("StateType")`

5.16.2.4 SymbolType `wfa.core_wfa.SymbolType = TypeVar("SymbolType")`

5.16.2.5 TransFunctionType `wfa.core_wfa.TransFunctionType = dict[StateType, dict[SymbolType, Set[StateType]]]`

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)
- PrintSymbolType = Union[core_wfa.SymbolType, List[core_wfa.SymbolType]]
- 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

Copyright (C) 2017 Vojtech Havlena, xhavle03@stud.fit.vutbr.cz
This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 2 of the License, or (at your option) any later version.
This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.
You should have received a copy of the GNU General Public License. If not, see http://www.gnu.org/licenses/.

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 PrintSymbolType `wfa.core_wfa_export.PrintSymbolType = Union[core_wfa.SymbolType, List[core_wfa.SymbolType]]`

5.17.2.3 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

- StateFloatMapOptType = Optional[dict[StateType, float]]
- StateFloatMapType = dict[StateType, float]
- StateType = int
- SymbolType = TypeVar("SymbolType")
- float THRESHOLD = 0.0

  Threshold for sparse matrices.
- TransFunctionType = dict[StateType, dict[SymbolType, Set[StateType]]]

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 StateFloatMapOptType

```
wfa.matrix_wfa.StateFloatMapOptType = Optional[dict[StateType, float]]
```

5.18.2.2 StateFloatMapType

```
wfa.matrix_wfa.StateFloatMapType = dict[StateType, float]
```

5.18.2.3 StateType

```
wfa.matrix_wfa.StateType = int
```

5.18.2.4 SymbolType

```
wfa.matrix_wfa.SymbolType = TypeVar("SymbolType")
```

5.18.2.5 THRESHOLD

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

Threshold for sparse matrices.

5.18.2.6 TransFunctionType `wfa.matrix_wfa.TransFunctionType = dict[StateType, dict[SymbolType, Set[StateType]]]`

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

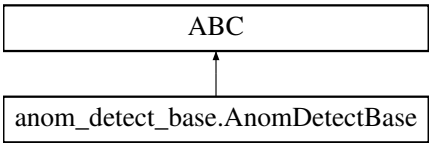
Copyright

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, core_wfa.CoreWFA aut, List window, ComPairType compair)

  Abstract apply detection on a given window.
- def detect (self, List window, ComPairType compair, float accelerate=0.0)

  Abstract anomaly detection.
- def dpa_selection (self, List window, ComPairType 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,
    core_wfa.CoreWFA aut,
    List window,
    ComPairType 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,
    List window,
    ComPairType compair,
    float  accelerate = 0.0 )
```

Abstract anomaly detection.

Parameters

window	List of messages corresponding to a single window to be checked
compair	Pair of communicating devices
accelerate	Use acceleration with the given value (if a detection value is below accelerate, the detection analysis terminates without computing all detection values).

Returns

abstact detection values

6.1.2.3 dpa_selection()

```
def anom_detect_base.AnomDetectBase.dpa_selection (
        self,
    List window,
    ComPairType compair )
```

Abstract DPA selection.

Parameters

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

Returns

Selected DPA

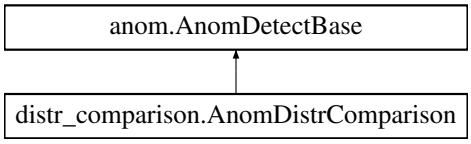
The documentation for this class was generated from the following file:

- 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, dict[anom.ComPairType, List[core_wfa.CoreWFA]] aut_map, Callable learning_procedure)
  Constructor.
- float apply_detection (self, core_wfa.CoreWFA aut, List window, anom.ComPairType compair)
  Apply distribution-comparison-based anomaly detection.
- List[float] detect (self, List window, anom.ComPairType compair, float accelerate=0.0)
  Detect if anomaly occurrs in the given window.
- List[core_wfa.CoreWFA] dpa_selection (self, List window, anom.ComPairType compair)
  Select appropriate DPA according to a communication window and a communication pair.
- None remove_euclid_similar (self, float max_error)
  Remove Euclid similar automata from the golden map (with the error bounded by max_error).
- None remove_identical (self)
  Remove identical automata from the golden map.

Static Public Member Functions

- float euclid_distance (core_wfa.CoreWFA aut1, core_wfa.CoreWFA 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.
- test_fa

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, dict[anom.ComPairType, List[core_wfa.CoreWFA]] aut_map, Callable 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() `float distr_comparison.AnomDistrComparison.apply_detection ( self, core_wfa.CoreWFA aut, List window, anom.ComPairType 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()

```
List[float] distr_comparison.AnomDistrComparison.detect (
    self,
    List window,
    anom.ComPairType compair,
    float  accelerate = 0.0 )
```

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
accelerate	Use acceleration with the given value (if a detection value is below accelerate, the detection analysis terminates without computing all detection values).

Returns

List of floats representing distance between golden automata and a window

6.2.3.3 dpa_selection()

```
List[core_wfa.CoreWFA] distr_comparison.AnomDistrComparison.dpa_selection (
    self,
    List window,
    anom.ComPairType 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() float distr_comparison.AnomDistrComparison.euclid_distance (core_wfa.CoreWFA *aut1*, core_wfa.CoreWFA *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() None distr_comparison.AnomDistrComparison.remove_euclid_similar (*self*, float *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() None 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.

6.2.4.3 test_fa `distr_comparison.AnomDistrComparison.test_fa`

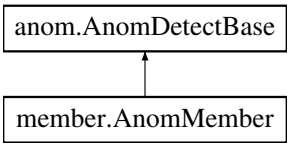
The documentation for this class was generated from the following file:

- 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, dict[anom.ComPairType, List[core_wfa.CoreWFA]] aut_map, Callable learning_procedure)
  Constructor.
- def apply_detection (self, core_wfa.CoreWFA aut, List window, anom.ComPairType compair)
  Apply member-based anomaly detection.
- List[float] detect (self, List window, anom.ComPairType compair, float accelerate=0.0)
  Detect if anomaly occurrs in the given window.
- List[core_wfa.CoreWFA] dpa_selection (self, List window, anom.ComPairType 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,
dict[anom.ComPairType, List[core_wfa.CoreWFA]] aut_map,
Callable 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,
core_wfa.CoreWFA aut,
List window,
anom.ComPairType 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() `List[float] member.AnomMember.detect (`

```
 self,
List window,
anom.ComPairType compair,
float  accelerate = 0.0 )
```

Detect if anomaly occurrs in the given window.

Parameters

window	List of messages to be inspected
compair	Pair of communicating devices
accelerate	Use acceleration with the given value (if a detection value is below accelerate, the detection analysis terminates without computing all detection values).

Returns

List of detection result for each model

6.3.3.3 dpa_selection() `List[core_wfa.CoreWFA] member.AnomMember.dpa_selection ( self, List window, anom.ComPairType 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.

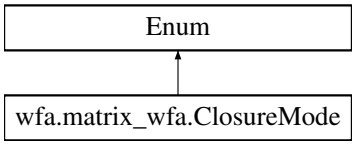
The documentation for this class was generated from the following file:

- 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.

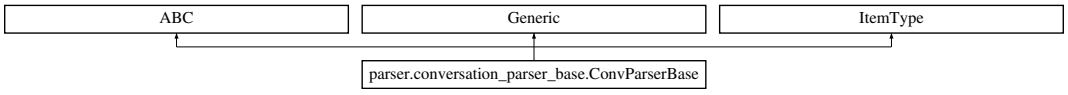
The documentation for this class was generated from the following file:

- 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

- List[ConvBaseType] get_all_conversations (self, Optional[Callable] proj=None)

  Get all conversations (possibly projected by abstraction)
- Optional[ConvBaseType] get_conversation (self)

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

  Parse and store all conversations.
- List["ConvParserBase"] split_communication_pairs (self)

  Split input according to the communication pairs.
- List["ConvParserBase"] split_to_windows (self, float 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()

```
List[ConvBaseType] parser.conversation_parser_base.ConvParser↩
Base.get_all_conversations (
              self,
             Optional[Callable] 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()

```
Optional[ConvBaseType] parser.conversation_parser_base.ConvParser↩
Base.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() List["ConvParserBase"] 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() List["ConvParserBase"] parser.conversation_parser_base.ConvParser↩ Base.split_to_windows (

self,
float *dur*)

Split input according to time windows.

Parameters

dur	Time duration

Returns

List of ConvParserBase (or derived)

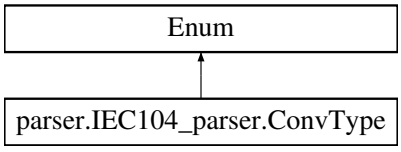
The documentation for this class was generated from the following file:

- 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.

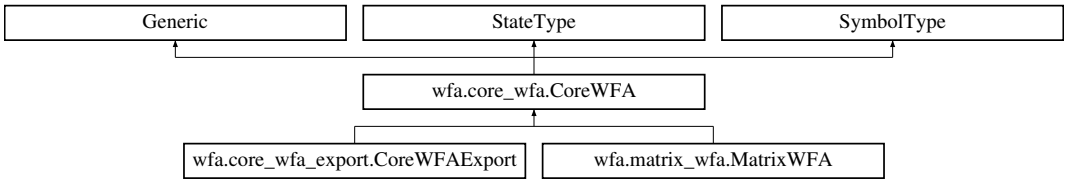
The documentation for this class was generated from the following file:

- 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

- bool __eq__ (self, object other)

  Equality of two WFAs.
- def __hash__ (self)

  Hash method.
- def __init__ (self, List[Transition] transitions=None, StateFloatMapOptType finals=None, StateFloatMapType start=dict(), Optional[List[SymbolType]] alphabet=None)

  Constructor.
- def breadth_first_search (self, StateType state, Set[StateType] visited, dict[StateType, List[Transition]] tr_dict)

  BFS in the automaton graph.
- None complete_wfa (self, StateType trap)

  Complete the automaton.
- "CoreWFA" difference_dwfa (self, "CoreWFA" diff)

  Compute the difference weighted automaton.
- Set[StateType] get_accessible_states (self, Optional[dict[StateType, List[Transition]]] tr_dict=None)

  Get accessible states of the WFA.
- List[SymbolType] get_alphabet (self)

  Get alphabet used by the WFA.
- "CoreWFA" get_automata_restriction (self, Set[StateType] states)

  Get WFA restriction to only states in states.
- Set[StateType] get_coaccessible_states (self, Optional[dict[StateType, List[Transition]]] tr_dict=None)

  Get coaccessible states of the WFA.
- dict[StateType, List[Transition]] get_dictionary_transitions (self)

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

  Get all final states of the WFA.
- List[SymbolType] get_most_probable_string (self)

  Compute the most probable word of the DPA.

- Set[StateType] get_predecessors (self, StateType state)

  Operation that finds predessors of the state state.
- dict[StateType, List[Transition]] get_predecessors_transitions (self)

  Get predecessors of all states of the WFA.
- Optional[dict[object, object]] get_rename_dict (self)

  Get the dictionary containing original state labels and renamed state labels.
- "CoreWFA" get_rev_transitions_aut (self)

  Get automaton with reversed directions of transitios.
- dict[StateType, List[Transition]] 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).
- StateFloatMapType get_starts (self)

  Get the start state (only one start state is allowed).
- TransFunctionType 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)
- List[StateType] get_states (self)

  Get all states of the WFA (the list of states is computed from the transitions).
- List[Transition] get_transitions (self)

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

  Get trimed WFA.
- bool is_deterministic (self)

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

  Apply the function fnc on the symbols of all transitions.
- "CoreWFA" product (self, "CoreWFA" 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.
- None set_all_finals (self)

  Set all states to be final (all having the accepting weight 1.0)
- None set_alphabet (self, List[SymbolType] alph)

  Set the alphabet.
- None set_finals (self, StateFloatMapType finals)

  Set final states of the WFA.
- None set_ones (self)

  Set the weight of all transitions to 1.0.
- def set_starts (self, StateFloatMapType start)

  Set the initial state.
- Optional[float] string_prob_deterministic (self, List[SymbolType] 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,
List[Transition] transitions = None,
StateFloatMapOptType finals = None,
StateFloatMapType  start = dict(),
Optional[List[SymbolType]] 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__() bool wfa.core_wfa.CoreWFA.__eq__ (

```
self,
object 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,
    StateType state,
    Set[StateType] visited,
    dict[StateType, List[Transition]] 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 complete_wfa()

```
None wfa.core_wfa.CoreWFA.complete_wfa (
        self,
    StateType trap )
```

Complete the automaton.

New transitions have the weight 0.0

Parameters

trap	New trap (sink) state (assuming not to be in the set of states)

6.7.3.5 difference_dwfa()

```
"CoreWFA" wfa.core_wfa.CoreWFA.difference_dwfa (
        self,
    "CoreWFA" diff )
```

Compute the difference weighted automaton.

First, convert the second automaton to a complete WFA with all transitions 1.0. Then, switch the accepting states to obtain a complementary automaton. Finally, return the product with the original automaton.

Parameters

diff	Second automaton

Returns

Difference automaton

6.7.3.6 get_accessible_states()

```
Set[StateType] wfa.core_wfa.CoreWFA.get_accessible_states (
        self,
        Optional[dict[StateType, List[Transition]]] tr_dict = None )
```

Get accessible states of the WFA.

Parameters

tr_dict	Transition dictionary.

Returns

The list of accessible states.

6.7.3.7 get_alphabet()

```
List[SymbolType] 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.

6.7.3.8 get_automata_restriction()

```
"CoreWFA" wfa.core_wfa.CoreWFA.get_automata_restriction (
        self,
        Set[StateType] 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)

6.7.3.9 get_coaccessible_states()

```
Set[StateType] wfa.core_wfa.CoreWFA.get_coaccessible_states (
        self,
        Optional[dict[StateType, List[Transition]]] tr_dict = None )
```

Get coaccessible states of the WFA.

Parameters

tr_dict	Transition dictionary.

Returns

The list of coaccessible states.

6.7.3.10 get_dictionary_transitions()

```
dict[StateType, List[Transition]] 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)

6.7.3.11 get_finals()

```
StateFloatMapType 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)

6.7.3.12 get_most_probable_string()

```
List[SymbolType] wfa.core_wfa.CoreWFA.get_most_probable_↩
string (
        self )
```

Compute the most probable word of the DPA.

Returns

A word with a highest probability

6.7.3.13 get_predecessors()

```
Set[StateType] wfa.core_wfa.CoreWFA.get_predecessors (
        self,
        StateType state )
```

Operation that finds predessors of the state state.

Parameters

state	The state whose predessors are found.

Returns

List of predecessors

6.7.3.14 get_predecessors_transitions() dict[StateType, List[Transition]] wfa.core_wfa.CoreWFA.get_predecessors_transitions (self)

Get predecessors of all states of the WFA.

Returns

Dict: State -> List(State)

6.7.3.15 get_rename_dict() Optional[dict[object, object]] 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).

6.7.3.16 get_rev_transitions_aut() "CoreWFA" wfa.core_wfa.CoreWFA.get_rev_transitions_aut (self)

Get automaton with reversed directions of transitios.

Returns

WFA with reversed transitions.

6.7.3.17 get_single_dictionary_transitions()

```
dict[StateType, List[Transition]] 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)

6.7.3.18 get_starts()

```
StateFloatMapType wfa.core_wfa.CoreWFA.get_starts (
        self )
```

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

Returns

Start state.

6.7.3.19 get_state_symbol_dict()

```
TransFunctionType 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)

6.7.3.20 get_states()

```
List[StateType] 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.

6.7.3.21 get_transitions() List[Transition] wfa.core_wfa.CoreWFA.get_transitions (self)

Get all transitions of the WFA.

Returns

List of all transitions

6.7.3.22 get_trim_automaton() "CoreWFA" wfa.core_wfa.CoreWFA.get_trim_automaton (self)

Get trimed WFA.

Returns

Trimmed WFA.

6.7.3.23 is_deterministic() bool wfa.core_wfa.CoreWFA.is_deterministic (self)

Is the WFA deterministic.

Returns

True deterministic, otherwise False

6.7.3.24 map_symbols() def wfa.core_wfa.CoreWFA.map_symbols (self, Callable fnc)

Apply the function fnc on the symbols of all transitions.

Parameters

fnc	Function applied on symbols

6.7.3.25 product() "CoreWFA" wfa.core_wfa.CoreWFA.product (self, "CoreWFA" aut)

Perform the product of two WFAs.

Parameters

aut	Second automaton for the product.

Returns

WFA representing the product of WFAs

6.7.3.26 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

6.7.3.27 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.

6.7.3.28 set_all_finals() `None wfa.core_wfa.CoreWFA.set_all_finals (`

```
self )
```

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

6.7.3.29 set_alphabet() `None wfa.core_wfa.CoreWFA.set_alphabet (`

```
self,
List[SymbolType] alph )
```

Set the alphabet.

Parameters

alph	New alphabet

6.7.3.30 set_finals()

```
None wfa.core_wfa.CoreWFA.set_finals (
        self,
        StateFloatMapType finals )
```

Set final states of the WFA.

Parameters

finals	Dictionary of final states and their weight of accepting.

6.7.3.31 set_ones()

```
None wfa.core_wfa.CoreWFA.set_ones (
        self )
```

Set the weight of all transitions to 1.0.

6.7.3.32 set_starts()

```
def wfa.core_wfa.CoreWFA.set_starts (
        self,
        StateFloatMapType start )
```

Set the initial state.

Parameters

start	New initial state

6.7.3.33 string_prob_deterministic()

```
Optional[float] wfa.core_wfa.CoreWFA.string_prob_deterministic
(
        self,
        List[SymbolType] word )
```

Compute the probability of the word word.

Parameters

word	Word

Returns

Probability of word

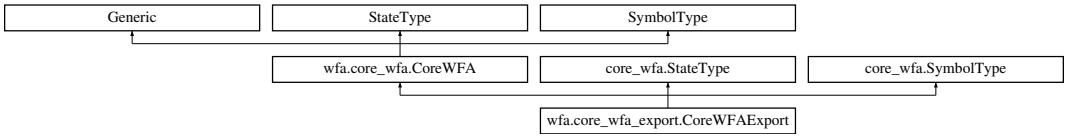
The documentation for this class was generated from the following file:

- 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, List[core_wfa.Transition] transitions=None, core_wfa.StateFloatMapOptType finals=None, core_wfa.StateFloatMapType start=dict(), Optional[List[core_wfa.SymbolType]] alphabet=None)

  Constructor.
- dict[Tuple[core_wfa.StateType, core_wfa.StateType], Tuple[List[core_wfa.SymbolType], float]] get_aggregated_transitions (self)

  Get aggregated transitions (merging transitions which differs only on symbol into a transition labeled with the list of symbols).
- str to_dot (self, bool aggregate=True, Optional[dict[core_wfa.StateType, str]] state_label=None, str legend=None)

  Convert the WFA to dot format (for graphical visualization).
- str to_fa_format (self, bool initial=False, bool 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,
    List[core_wfa.Transition] transitions = None,
    core_wfa.StateFloatMapOptType finals = None,
    core_wfa.StateFloatMapType  start = dict(),
    Optional[List[core_wfa.SymbolType]] 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()

```
dict[Tuple[core_wfa.StateType, core_wfa.StateType], Tuple[List[core_wfa.Sym
float]] 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()

```
str wfa.core_wfa_export.CoreWFAExport.to_dot (
            self,
          bool aggregate = True,
          Optional[dict[core_wfa.StateType, str]] state_label = None,
          str 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()

```
str wfa.core_wfa_export.CoreWFAExport.to_fa_format (
            self,
          bool initial = False,
          bool 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)

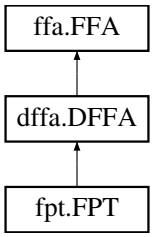
The documentation for this class was generated from the following file:

- 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, Set[ffa.StateType] states, ffa.TransFuncDetType trans, ffa.StateWeightType ini, ffa.StateWeightType fin, Optional[ffa.StateType] root=None)

  Constructor.
- bool alergia_compatible (self, ffa.StateType qa, ffa.StateType qb, float alpha)

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

  Get the root (initial) state.
- core_wfa_export.CoreWFAExport normalize (self)

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

  Compute frequency of a state (number of strings accepted at the state or leaving the state).
- None stochastic_fold (self, ffa.StateType red, ffa.StateType blue)

  Fold frequencies from subtree given by blue root into the automaton rooted at the red state.
- None stochastic_merge (self, ffa.StateType red, ffa.StateType blue)

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

Static Public Member Functions

- bool alergia_test (float f1, float n1, float f2, float n2, float 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__() def dffa.DFFA.__init__ (

```
self,
Set[ffa.StateType] states,
ffa.TransFuncDetType trans,
ffa.StateWeightType ini,
ffa.StateWeightType fin,
Optional[ffa.StateType]  root = None )
```

Constructor.

Parameters

states	States of the DFFA
trans	Transitions of the DFFA
ini	Initial states
fin	Final states
root	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() bool dffa.DFFA.alergia_compatible (

```
self,
ffa.StateType qa,
ffa.StateType qb,
float 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()

```
bool dffa.DFFA.alergia_test (
        float f1,
        float n1,
        float f2,
        float n2,
        float 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()

```
ffa.StateType dffa.DFFA.get_root (
        self )
```

Get the root (initial) state.

Returns

Root (initial) state

6.9.3.4 normalize()

```
core_wfa_export.CoreWFAExport 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()

```
float dffa.DFFA.state_freq (
        self,
        ffa.StateType 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()

```
None dffa.DFFA.stochastic_fold (
        self,
    ffa.StateType red,
    ffa.StateType 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()

```
None dffa.DFFA.stochastic_merge (
        self,
    ffa.StateType red,
    ffa.StateType blue )
```

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

Parameters

red	Red state
blue	Blue state

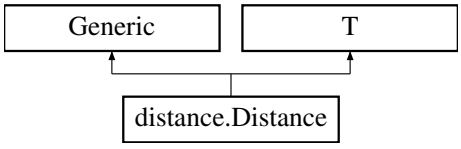
The documentation for this class was generated from the following file:

- dffa.py

6.10 distance.Distance Class Reference

Class removing items from a set causing the minimum error.

Inheritance diagram for distance.Distance:



Public Member Functions

- def __init__ (self, DistType dists, List[T] pts)

  Constructor.
- Set[T] compute_subset_error (self, float 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, DistType dists, List[T] 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() `Set[T] distance.Distance.compute_subset_error ( self, float 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`

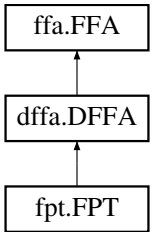
The documentation for this class was generated from the following file:

- distance.py

6.11 ffa.FFA Class Reference

General frequency automata (FFA)

Inheritance diagram for ffa.FFA:



Public Member Functions

- def __init__ (self, Set[StateType] states, TransFuncMixType trans, StateWeightType ini, StateWeightType fin)
  Constructor.
- StateWeightType get_finals (self)
  Get final states.
- Set[StateType] get_states (self)
  Get all states.
- List[FFATrans] get_transition_list (self)
  Get list of transitions from the transition function.
- TransFuncMixType get_transitions (self)
  Get transitions.
- "FFA" inverse_ffa (self)
  Get the inverse FFA.
- None merge_equivalent (self, Set[Set[StateType]] classes)
  Merge equivalent states according to the equivalent classes.

- None merge_states (self, Set[StateType] states)

  Merge a set of states (remove those states and replace with one in the set)
- Optional[int] path_length (self, StateType st1, StateType st2)

  Get length of a shortest path between st1 and st2.
- Set[StateType] reachable_states (self, Set[StateType] st_set)

  Get all reachable states from st_set.
- None rename_states (self)
- Set[StateType] successors (self, StateType state, Optional[str] sym=None)

  Get all successors from state over sym.
- Set[StateType] successors_set (self, Set[StateType] states, Optional[str] sym=None)

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

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

  Converts FFA to WFA (weighted finite automaton)
- None 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,
Set[StateType] states,
TransFuncMixType trans,
StateWeightType ini,
StateWeightType fin )
```

Constructor.

Parameters

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

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

6.11.3 Member Function Documentation

6.11.3.1 get_finals() `StateWeightType ffa.FFA.get_finals ( self )`

Get final states.

Returns

Final states of the FFA

6.11.3.2 get_states() `Set[StateType] ffa.FFA.get_states ( self )`

Get all states.

Returns

All states of the FFA

6.11.3.3 get_transition_list() `List[FFATrans] ffa.FFA.get_transition_list ( self )`

Get list of transitions from the transition function.

Returns

List of transitions

6.11.3.4 get_transitions() `TransFuncMixType ffa.FFA.get_transitions ( self )`

Get transitions.

Returns

Transitions of the FFA

6.11.3.5 inverse_ffa() `"FFA" ffa.FFA.inverse_ffa ( self )`

Get the inverse FFA.

Returns

FFA with the inverse transition function

6.11.3.6 merge_equivalent() `None ffa.FFA.merge_equivalent ( self, Set[Set[StateType]] classes )`

Merge equivalent states according to the equivalent classes.

Parameters

classes	Partitioning of the states

6.11.3.7 merge_states()

```
None ffa.FFA.merge_states (
        self,
        Set[StateType] 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()

```
Optional[int] ffa.FFA.path_length (
        self,
        StateType st1,
        StateType 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()

```
Set[StateType] ffa.FFA.reachable_states (
        self,
        Set[StateType] 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() None ffa.FFA.rename_states (self)

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

6.11.3.11 successors() Set[StateType] ffa.FFA.successors (self, StateType state, Optional[str] 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() Set[StateType] ffa.FFA.successors_set (self, Set[StateType] states, Optional[str] 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()

```
str ffa.FFA.to_graphiwiz (
        self,
        str 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()

```
core_wfa_export.CoreWFAExport ffa.FFA.to_wfa (
        self )
```

Converts FFA to WFA (weighted finite automaton)

Returns

FFA represented as WFA

6.11.3.15 trim()

```
None ffa.FFA.trim (
        self )
```

Remove unreachable states from the automaton.

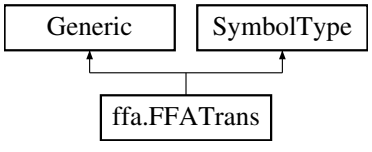
The documentation for this class was generated from the following file:

- ffa.py

6.12 ffa.FFATrans Class Reference

Class representing a transtion of the FFA.

Inheritance diagram for ffa.FFATrans:



6.12.1 Detailed Description

Class representing a transtion of the FFA.

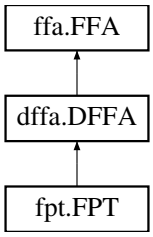
The documentation for this class was generated from the following file:

- 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, Set[ffa.StateType] states, ffa.TransFuncDetType trans, ffa.StateWeightType ini, ffa.StateWeightType fin)
  Constructor.
- str __str__ (self)
  Convert to a string representation.
- None add_string (self, str string, int label=0)
  Add string to the frequency prefix tree.
- None add_string_list (self, List[str] lst, int label=0)
  Add a list of strings to frequency prefix tree.
- int count_label_edges (self, int label)
  Count edges with labels corresponding to label.
- Set[ffa.StateType] get_leaves (self)
  Get leaves (states without outgoing transitions)
- str show (self)
  Convert the FPT to a string representation.
- None suffix_minimize (self)
  Merge equivalent backward deterministic states.

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,
Set[ffa.StateType] states,
ffa.TransFuncDetType trans,
ffa.StateWeightType ini,
ffa.StateWeightType 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__() str fpt.FPT.__str__ (

```
self )
```

Convert to a string representation.

6.13.3.2 add_string() None fpt.FPT.add_string (

```
self,
str string,
int 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()

```
None fpt.FPT.add_string_list (
        self,
    List[str] lst,
    int 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()

```
int fpt.FPT.count_label_edges (
        self,
    int 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()

```
Set[ffa.StateType] fpt.FPT.get_leaves (
        self )
```

Get leaves (states without outgoing transitions)

Returns

Set of leaves

6.13.3.6 show() `str fpt.FPT.show ( self )`

Convert the FPT to a string representation.

Returns

String representation of the FPT

6.13.3.7 suffix_minimize() `None fpt.FPT.suffix_minimize ( self )`

Merge equivalent backward deterministic states.

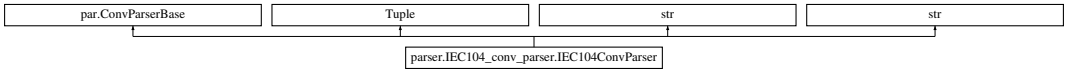
The documentation for this class was generated from the following file:

- 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, List[RowType] inp, Optional[ComPairType] pr=None)
  Constructor taking a list of messages (each message is a dictionary)
- List[ConvStrType] get_all_conversations (self, Optional[Callable] proj=None)
  Get all conversations (possibly filter by communication pairs)
- Optional[ConvStrType] get_conversation (self)
  Get a following conversation from already divided messages.
- RowType get_line (self)
  Get a next line.
- None parse_conversations (self)
  Parse and store all conversations.
- ConvStrType parse_data (self, str data)
  Parse data.
- List["IEC104ConvParser"] split_communication_pairs (self)
  Split input according to the communication pairs.
- List["IEC104ConvParser"] split_to_windows (self, float 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, List[RowType] inp, Optional[ComPairType] 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() List[ConvStrType] parser.IEC104_conv_parser.IEC104ConvParser.↩ get_all_conversations (self, Optional[Callable] 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() Optional[ConvStrType] 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() RowType parser.IEC104_conv_parser.IEC104ConvParser.get_line (self)

Get a next line.

Returns

Next line of the buffer

6.14.3.4 parse_conversations() None parser.IEC104_conv_parser.IEC104ConvParser.parse_conversations (self)

Parse and store all conversations.

6.14.3.5 parse_data() ConvStrType parser.IEC104_conv_parser.IEC104ConvParser.parse_data (self, str data)

Parse data.

Parameters

data	Input to be parsed

Returns

List of parsed values

6.14.3.6 split_communication_pairs() List["IEC104ConvParser"] parser.IEC104_conv_parser.IEC104↩ ConvParser.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() `List["IEC104ConvParser"] parser.IEC104_conv_parser.IEC104Conv↩Parser.split_to_windows (`

```
self,
float 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`

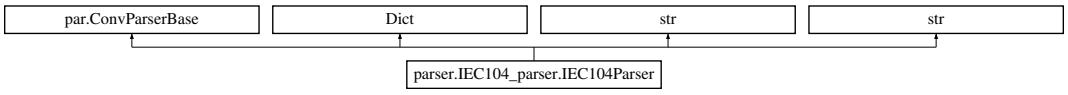
The documentation for this class was generated from the following file:

- 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, List[ConvSymbolType] inp, Optional[ComPairType] pr=None)

  Constructor taking a list of messages (each message is a dictionary)
- List[ConvStrType] get_all_conversations (self, Optional[Callable] proj=None)

  Get all conversations (possibly filter by communication pairs)
- Optional[ConvStrType] get_conversation (self)

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

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

  Check if a given conversation is complete (according to the last packet).
- None parse_conversations (self)

  Parse and store all conversations.
- None return_symbol (self, ConvSymbolType val, bool buff_read)

  Return the message to the buffer.
- List["IEC104Parser"] split_communication_pairs (self)

  Split input according to the communication pairs.
- List["IEC104Parser"] split_to_windows (self, float dur)

  Split input according to time windows.

Static Public Member Functions

- ConvType get_initial_type (ConvSymbolType row)

  Get initial type of a conversation.
- bool in_middle_range (ConvSymbolType row, ConvType tp)

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

  Is the message final.
- bool is_inform_message (ConvSymbolType row)

  Is the message informal?
- bool is_msg_match (ComPairType compair, ConvSymbolType val)

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

  Is the message spontaneous?

Public Attributes

- 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,
List[ConvSymbolType] inp,
Optional[ComPairType] 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() List[ConvStrType] parser.IEC104_parser.IEC104Parser.get_all_↩ conversations (

```
self,
Optional[Callable] 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() Optional[ConvStrType] 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()

```
ConvType parser.IEC104_parser.IEC104Parser.get_initial_type (
        ConvSymbolType 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()

```
ConvSymbolType parser.IEC104_parser.IEC104Parser.get_symbol (
        self,
        bool 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()

```
bool parser.IEC104_parser.IEC104Parser.in_middle_range (
        ConvSymbolType row,
        ConvType 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()

```
bool parser.IEC104_parser.IEC104Parser.is_conversation_complete (
            self,
            ConvStrType 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()

```
bool parser.IEC104_parser.IEC104Parser.is_final (
            ConvSymbolType row,
            ConvType 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()

```
bool parser.IEC104_parser.IEC104Parser.is_inform_message (
            ConvSymbolType row )  [static]
```

Is the message informal?

Parameters

row	Message

Returns

True – informal message

6.15.3.9 is_msg_match()

```
bool parser.IEC104_parser.IEC104Parser.is_msg_match (
        ComPairType compair,
        ConvSymbolType 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()

```
bool parser.IEC104_parser.IEC104Parser.is_spontaneous (
        ConvSymbolType row )  [static]
```

Is the message spontaneous?

Parameters

row	Message

Returns

True – spontaneous message

6.15.3.11 parse_conversations()

```
None parser.IEC104_parser.IEC104Parser.parse_conversations (
        self )
```

Parse and store all conversations.

6.15.3.12 return_symbol()

```
None parser.IEC104_parser.IEC104Parser.return_symbol (
        self,
        ConvSymbolType val,
        bool 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() List["IEC104Parser"] 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() List["IEC104Parser"] parser.IEC104_parser.IEC104Parser.split_to_windows (

```
self,
float 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 compair parser.IEC104_parser.IEC104Parser.compair

6.15.4.2 conversations parser.IEC104_parser.IEC104Parser.conversations

6.15.4.3 incomplete parser.IEC104_parser.IEC104Parser.incomplete

6.15.4.4 index parser.IEC104_parser.IEC104Parser.index

6.15.4.5 input `parser.IEC104_parser.IEC104Parser.input`

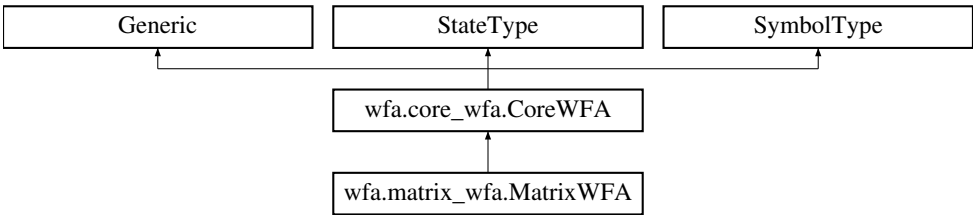
The documentation for this class was generated from the following file:

- 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, List[core_wfa.Transition] transitions=None, StateFloatMapOptType finals=None, StateFloatMapType start=dict(), Optional[List[SymbolType]] alphabet=None)

  Constructor.
- bool 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).
- float compute_language_probability (self, ClosureMode closure_mode, bool sparse=False, int iterations=0, bool debug=False)

  Compute the total probability of the WFA's language.
- numpy.matrix compute_transition_closure (self, ClosureMode closure_mode, bool sparse=False, int iterations=0, bool debug=False)

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

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

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

  Get a vector of initial weights.
- numpy.matrix get_transition_matrix (self, bool 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,
List[core_wfa.Transition] transitions = None,
StateFloatMapOptType finals = None,
StateFloatMapType  start = dict(),
Optional[List[SymbolType]] 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() bool 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() float wfa.matrix_wfa.MatrixWFA.compute_language_↩ probability (

```
 self,
ClosureMode closure_mode,
bool sparse = False,
int iterations = 0,
bool 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()

```
numpy.matrix wfa.matrix_wfa.MatrixWFA.compute_transition_closure (
              self,
             ClosureMode closure_mode,
             bool sparse = False,
             int iterations = 0,
             bool 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()

```
numpy.matrix wfa.matrix_wfa.MatrixWFA.get_final_ones (
              self,
             bool 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() numpy.matrix wfa.matrix_wfa.MatrixWFA.get_final_vector (self, bool 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() numpy.matrix wfa.matrix_wfa.MatrixWFA.get_initial_vector (self, bool 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() numpy.matrix wfa.matrix_wfa.MatrixWFA.get_transition_matrix (self, bool sparse = False)

Get a transition matrix corresponding to the WFA.

Parameters

sparse	Use sparse matrices

Returns

Transition matrix (Numpy.matrix)

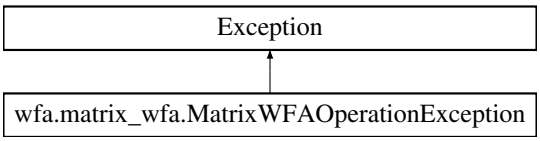
The documentation for this class was generated from the following file:

- 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, str msg)
  Constructor.
- str __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, str msg )`

Constructor.

Parameters

msg	Error message

6.17.3 Member Function Documentation

6.17.3.1 __str__() `str 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

- bool compatible_strings (str str1, str 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() `bool packet_loss.PacketLoss.compatible_strings ( str str1, str 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)

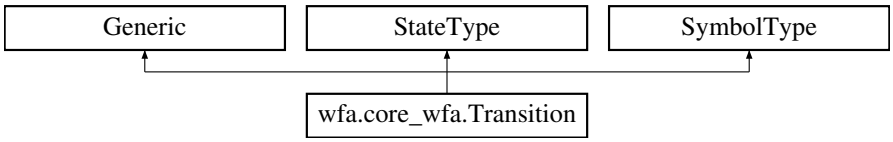
The documentation for this class was generated from the following file:

- 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

- bool __eq__ (self, object other)
  Equality of two transitions.
- def __hash__ (self)
  Hash method.
- def __init__ (self, StateType src, StateType dest, SymbolType sym, float weight)
  Constructor.
- bool __ne__ (self, object other)
  Inequality of two transitions.
- str __repr__ (self)
  String representation.
- str __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, StateType src, StateType dest, SymbolType sym, float 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__() bool wfa.core_wfa.Transition.__eq__ (self, object 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__() `bool wfa.core_wfa.Transition.__ne__ ( self, object other )`

Inequality of two transitions.

Parameters

other	Other transition

Returns

True – both transitions are NOT equal

6.19.3.4 __repr__() `str wfa.core_wfa.Transition.__repr__ ( self )`

String representation.

Returns

String representation of the transition

6.19.3.5 __str__() `str 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`

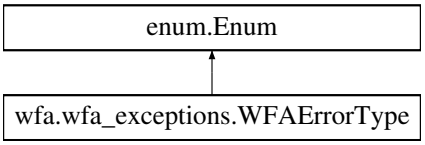
The documentation for this class was generated from the following file:

- 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.

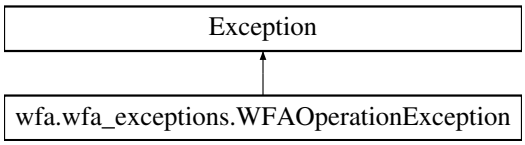
The documentation for this class was generated from the following file:

- 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, str msg, WFAErrorType err_type=WFAErrorType.general_error)
  Constructor.
- str __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,
    str msg,
    WFAErrorType 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__() `str 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.

Variables

- distance.DistType = dict[Tuple[T, T], float]
- distance.SortedDictType = List[Tuple[Tuple[T,T], float]]
- distance.T = TypeVar("T")

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.

Variables

- anom_detect_base.ComPairType = FrozenSet[Tuple[str,str]]

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

- dffa.DFFA alergia.alergia (dffa.DFFA freq_aut, float alpha, int t0)

  PA learning using the Alergia algorithm.
- Optional[ffa.StateType] alergia.choose_blue_state (dffa.DFFA freq_aut, Set[ffa.StateType] blue_set, int t0)

  Chose a blue state from a set of blue states.
- Optional[ffa.StateType] alergia.choose_red_state (dffa.DFFA freq_aut, Set[ffa.StateType] red_set, ffa.StateType blue, float 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.

Variables

- ffa.StateType = str
- ffa.StateWeightType = dict[StateType, int]
- ffa.SymbolType = TypeVar("SymbolType")
- ffa.TransFuncDetType = dict[StateType, dict[str, "FFATrans"]]
- ffa.TransFuncMixType = Union[dict[StateType, dict[str, Set["FFATrans"]]], dict[StateType, dict[str, "FFATrans"]]]
- ffa.TransFuncType = dict[StateType, dict[str, Set["FFATrans"]]]

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.

Variables

- parser.conversation_parser_base.ConvBaseType = List[ItemType]
- parser.conversation_parser_base.ItemType = TypeVar("ItemType")

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.

Variables

- parser.IEC104_conv_parser.ComPairType = FrozenSet[Tuple[str,str]]
- parser.IEC104_conv_parser.ConvStrType = List[ConvSymbolType]
- parser.IEC104_conv_parser.ConvSymbolType = Tuple[str, str]
- parser.IEC104_conv_parser.RowType = Dict[str, str]

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

- List[ConvSymbolType] parser.IEC104_parser.get_messages (fd)

  Get all messages from a csv file.

Variables

- parser.IEC104_parser.ComPairType = FrozenSet[Tuple[str,str]]
- parser.IEC104_parser.ConvStrType = List[ConvSymbolType]
- parser.IEC104_parser.ConvSymbolType = Dict[str, str]

7.15 aux_functions.py File Reference

Namespaces

- namespace wfa
- namespace wfa.aux_functions

  Auxiliary functions for WFAs.

Functions

- str wfa.aux_functions.convert_to_pritable (str dec, bool 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.

Variables

- wfa.core_wfa.StateFloatMapOptType = Optional[dict[StateType, float]]
- wfa.core_wfa.StateFloatMapType = dict[StateType, float]
- wfa.core_wfa.StateType = TypeVar("StateType")
- wfa.core_wfa.SymbolType = TypeVar("SymbolType")
- wfa.core_wfa.TransFunctionType = dict[StateType, dict[SymbolType, Set[StateType]]]

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)
- wfa.core_wfa_export.PrintSymbolType = Union[core_wfa.SymbolType, List[core_wfa.SymbolType]]
- 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

- wfa.matrix_wfa.StateFloatMapOptType = Optional[dict[StateType, float]]
- wfa.matrix_wfa.StateFloatMapType = dict[StateType, float]
- wfa.matrix_wfa.StateType = int
- wfa.matrix_wfa.SymbolType = TypeVar("SymbolType")
- float wfa.matrix_wfa.THRESHOLD = 0.0

  Threshold for sparse matrices.
- wfa.matrix_wfa.TransFunctionType = dict[StateType, dict[SymbolType, Set[StateType]]]

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