Fuzzy DL OWL 2

Release 1.0.9

Giuseppe Filippone

Jul 09, 2025

CONTENTS:

Welcome to fuzzy_dl_owl2's documentation!

Part I

Contents:

CHAPTER

ONE

FUZZY DL OWL 2

A python porting of the Fuzzy Description Language and the Fuzzy OWL 2 framework.

A lightweight Python porting of the Fuzzy Description Language (FuzzyDL) and the Fuzzy OWL 2 framework, designed for representing fuzzy logic within description logic and for mapping an knowledge base represented in FuzzyDL to a Fuzzy OWL 2 construct in RDF/XML format.

Features:

- Object-oriented representation of Fuzzy Description Logic elements
- Object-oriented representation of Fuzzy OWL 2 elements
- Mapping from FuzzyDL to Fuzzy OWL 2
- Mapping from Fuzzy OWL 2 to FuzzyDL
- Reasoning in FuzzyDL

1.1 Directory dl-examples

The directory `dl-examples` contains a few examples of Knowledge Bases written using the Fuzzy Description Logic language.

1.2 Project Structure

```
fuzzy_dl_owl2
├── __init__.py
├── fuzzydl
    ├── __init__.py
    ├── assertion
        ├── __init__.py
        ├── assertion.py
        └── atomic_assertion.py
    ├── classification_node.py
    ├── concept
        ├── __init__.py
        ├── all_some_concept.py
        ├── approximation_concept.py
        ├── atomic_concept.py
        ├── choquet_integral.py
        ├── concept.py
        ├── concrete
            ├── __init__.py
            ├── __pycache__
            ├── crisp_concrete_concept.py
            ├── fuzzy_concrete_concept.py
```

(continues on next page)

(continued from previous page)

```
│   │   │   ├── fuzzy_number
│   │   │   │   ├── __init__.py
│   │   │   │   └── triangular_fuzzy_number.py
│   │   │   ├── left_concrete_concept.py
│   │   │   ├── linear_concrete_concept.py
│   │   │   ├── modified_concrete_concept.py
│   │   │   ├── right_concrete_concept.py
│   │   │   ├── trapezoidal_concrete_concept.py
│   │   │   └── triangular_concrete_concept.py
│   │   ├── ext_threshold_concept.py
│   │   ├── has_value_concept.py
│   │   ├── implies_concept.py
│   │   ├── interface
│   │   │   ├── __init__.py
│   │   │   ├── __pycache__
│   │   │   ├── has_concept_interface.py
│   │   │   ├── has_concepts_interface.py
│   │   │   ├── has_role_concept_interface.py
│   │   │   ├── has_role_interface.py
│   │   │   ├── has_value_interface.py
│   │   │   └── has_weighted_concepts_interface.py
│   │   ├── modified
│   │   │   ├── __init__.py
│   │   │   ├── linearly_modified_concept.py
│   │   │   ├── modified_concept.py
│   │   │   └── triangularly_modified_concept.py
│   │   ├── negated_nominal.py
│   │   ├── operator_concept.py
│   │   ├── owa_concept.py
│   │   ├── qowa_concept.py
│   │   ├── quasi_sugeno_integral.py
│   │   ├── self_concept.py
│   │   ├── sigma_concept.py
│   │   ├── sigma_count.py
│   │   ├── string_concept.py
│   │   ├── sugeno_integral.py
│   │   ├── threshold_concept.py
│   │   ├── truth_concept.py
│   │   ├── value_concept.py
│   │   ├── weighted_concept.py
│   │   ├── weighted_max_concept.py
│   │   ├── weighted_min_concept.py
│   │   ├── weighted_sum_concept.py
│   │   └── weighted_sum_zero_concept.py
│   ├── concept_equivalence.py
│   ├── concrete_feature.py
│   ├── degree
│   │   ├── __init__.py
│   │   ├── degree_expression.py
│   │   ├── degree_numeric.py
│   │   ├── degree_variable.py
│   │   └── degree.py
│   ├── domain_axiom.py
│   ├── exception
│   │   ├── __init__.py
│   │   ├── fuzzy_ontology_exception.py
```

(continues on next page)

(continued from previous page)

```
    │   └── inconsistent_ontology_exception.py
    ├── feature_function.py
    ├── fuzzydl_to_owl2.py
    ├── general_concept_inclusion.py
    ├── individual
    │   ├── __init__.py
    │   ├── created_individual.py
    │   ├── individual.py
    │   └── representative_individual.py
    ├── knowledge_base.py
    ├── label.py
    ├── milp
    │   ├── __init__.py
    │   ├── expression.py
    │   ├── inequation.py
    │   ├── milp_helper.py
    │   ├── show_variables_helper.py
    │   ├── solution.py
    │   ├── term.py
    │   └── variable.py
    ├── modifier
    │   ├── __init__.py
    │   ├── linear_modifier.py
    │   ├── modifier.py
    │   └── triangular_modifier.py
    ├── parser
    │   ├── __init__.py
    │   └── dl_parser.py
    ├── primitive_concept_definition.py
    ├── query
    │   ├── __init__.py
    │   ├── all_instances_query.py
    │   ├── bnp_query.py
    │   ├── classification_query.py
    │   ├── defuzzify
    │   │   ├── __init__.py
    │   │   ├── defuzzify_query.py
    │   │   ├── lom_defuzzify_query.py
    │   │   ├── mom_defuzzify_query.py
    │   │   └── som_defuzzify_query.py
    │   ├── instance_query.py
    │   ├── kb_satisfiable_query.py
    │   ├── max
    │   │   ├── __init__.py
    │   │   ├── max_instance_query.py
    │   │   ├── max_query.py
    │   │   ├── max_related_query.py
    │   │   ├── max_satisfiable_query.py
    │   │   └── max_subsumes_query.py
    │   ├── min
    │   │   ├── __init__.py
    │   │   ├── min_instance_query.py
    │   │   ├── min_query.py
    │   │   ├── min_related_query.py
    │   │   ├── min_satisfiable_query.py
    │   │   └── min_subsumes_query.py
```

(continues on next page)

(continued from previous page)

```
│   │   ├── query.py
│   │   ├── related_query.py
│   │   ├── satisfiable_query.py
│   │   └── subsumption_query.py
│   ├── range_axiom.py
│   ├── relation.py
│   ├── restriction
│   │   ├── __init__.py
│   │   ├── has_value_restriction.py
│   │   └── restriction.py
│   ├── role_parent_with_degree.py
│   └── util
│       ├── __init__.py
│       ├── config_reader.py
│       ├── constants.py
│       ├── util.py
│       └── utils.py
└── fuzzyowl2
    ├── __init__.py
    ├── fuzzyowl2_to_fuzzydl.py
    ├── fuzzyowl2.py
    ├── owl_types
    │   ├── __init__.py
    │   ├── choquet_concept.py
    │   ├── concept_definition.py
    │   ├── fuzzy_datatype.py
    │   ├── fuzzy_modifier.py
    │   ├── fuzzy_nominal_concept.py
    │   ├── fuzzy_property.py
    │   ├── left_shoulder_function.py
    │   ├── linear_function.py
    │   ├── linear_modifier.py
    │   ├── modified_concept.py
    │   ├── modified_function.py
    │   ├── modified_property.py
    │   ├── owa_concept.py
    │   ├── property_definition.py
    │   ├── qowa_concept.py
    │   ├── quasi_sugeno_concept.py
    │   ├── right_shoulder_function.py
    │   ├── sugeno_concept.py
    │   ├── trapezoidal_function.py
    │   ├── triangular_function.py
    │   ├── triangular_modifer.py
    │   ├── weighted_concept.py
    │   ├── weighted_max_concept.py
    │   ├── weighted_min_concept.py
    │   ├── weighted_sum_concept.py
    │   └── weighted_sum_zero_concept.py
    ├── parser
    │   ├── __init__.py
    │   ├── owl2_parser.py
    │   └── owl2_xml_parser.py
    └── util
        ├── __init__.py
        ├── constants.py
```

(continues on next page)

(continued from previous page)

```
        └── fuzzy_xml.py
```

1.3 Test

The directory `test` contains the `unittest` files. In particular, the file `test_suite.py` contains all the test suite. The directory `examples/TestSuite` contains all the knowledge bases used for the tests.

1.4 License

CHAPTER

TWO

FUNDING

We acknoledge financial support from the **Sustainability Decision Framework** (**SDF**) Research Project – CUP **B79J23000540005** – Grant Assignment Decree No. **5486** adopted on **2023-08-04**.

CHAPTER

THREE

INSTALLATION AND CONFIGURATION

Check the repository.

```
pip install fuzzy-dl-owl2
```

Examples of supported Fuzzy Description Logic Constructs

Python Class	Description
AtomicConcept	Define an atomic concept
ChoquetIntegral	Define a choquet integral concept
ApproximationConcept	Define a tight/upper/* lower/upper approximation concept

3.1 Configuration of the MILP solver

Since version 1.0.1 uses `Gurobi Optimizer` (see gurobipy for the Fuzzy DL reasoning, please create a GUROBI license to use this library.

For the configuration, create a `CONFIG.ini` file in the same directory used for the execution of the library. Example of your execution directory:

```
your_directory
├── CONFIG.ini
├── your_file.py
```

The file `CONFIG.ini` is structured as follows:

```
[DEFAULT]
debugPrint = False
epsilon = 0.001
maxIndividuals = -1
owlAnnotationLabel = fuzzyLabel
milpProvider = mip
```

Configuration Variable	Description
debugPrint	Enable/disable debugging
epsilon	Define the precision of the solution. For instance, epsilon = 0.001 means that the solution will be calculated with an accuracy to the third decimal place
maxIndividuals	Define the maximal number of individuals to handle. The value -1 indicates that there is no maximum
owlAnnotationLabel	Define the Annotation label used to build the Fuzzy OWL 2 RDF/XML ontology
milpProvider	Define the MILP provider used by the reasoner. The supported providers are listed below.

Supported MILP Providers:

Provider	milpProvider
Gurobi	gurobi
CPLEX	pulp_cplex
CBC	pulp
GLPK	pulp_glpk
HiGHS	pulp_highs
MIP	mip

3.2 MILP Provider Usage and Configuration

3.2.1 GUROBI

- Install gurobipy:

```
pip install gurobipy==12.0.0
```

- Download the GUROBI license from their website.
- Add Gurobi to the PATH

3.2.2 MIP

- Install Python MIP:

```
pip install mip==1.16rc0
```

3.2.3 GLPK

- Install GLPK v5.0 and GMP v6.3.0
- Install Python pulp:

```
pip install pulp==3.2.1
```

- Add GLPK to the PATH

3.2.4 CBC

- Install CBC
- Install Python pulp:

```
pip install pulp==3.2.1
```

- Add CBC to the PATH

3.2.5 CPLEX

- Install CPLEX v22.11
- Install Python pulp:

```
pip install pulp==3.2.1
```

- Add CPLEX to the PATH

3.2.6 HiGHS

- Install HiGHS v1.10.0
- Install python pulp:

```
pip install pulp==3.2.1
```

- Add HiGHS to the PATH

CHAPTER

FOUR

FUZZY DESCRIPTION LOGIC GRAMMATICS

4.1 String and Numbers

```
name    := ["][a-zA-Z_][a-zA-Z0-9_]*["]
numbers := [+-]? [0-9]+(\.[0-9]+)
```

4.2 Define the semantics of the knowledge base

```
logic          := 'lukasiewicz' | 'zadeh' | 'classical'
define_logic   := '(' 'define-fuzzy-logic' ["] logic ["] ')'
```

4.3 Define truth constants

```
constant := '(' 'define-truth-constant' name numbers ')'
```

- Example: **(define-truth-constant V 5.3)** defines the truth constant named V with value 5.3.

4.4 Define modifiers

Modifiers change the membership function of a fuzzy concept.

```
modifier    := (
    '(' 'define-modifier' name 'linear-modifier' '(' numbers ')' ')'
# linear hedge with c > 0
    | '(' 'define-modifier' name 'triangular-modifier' '(' numbers ',' numbers ','
numbers ')' ')'      # triangular function
)
```

4.5 Define concrete fuzzy concepts

```
concept_type    := (
        'crisp' '(' numbers ',' numbers ',' numbers ',' numbers ')'
# crisp interval
        | 'left-shoulder' '(' numbers ',' numbers ',' numbers ',' numbers ')'
# left-shoulder function
        | 'right-shoulder' '(' numbers ',' numbers ',' numbers ',' numbers ')'
# right-shoulder function
        | 'triangular' '(' numbers ',' numbers ',' numbers ',' numbers ')'
# triangular function
        | 'trapezoidal' '(' numbers ',' numbers ',' numbers ',' numbers ',' numbers ')
```

(continues on next page)

(continued from previous page)

```
↪'     # trapezoidal function
       | 'linear' '(' numbers ',' numbers ',' numbers ',' numbers ')'
# linear function
       | 'modified' '(' name ',' name ')'
# modified datatype
    )
fuzzy_concept   := '(' 'define-fuzzy-concept' name concept_type ')'
```

- Note: the fuzzy concept **modified** applies only to modifiers and datatype restrictions. Example: **(define-fuzzy-concept CONCEPT modified(MOD, F))**, where **CONCEPT** is the name of the created concrete fuzzy concept, **MOD** is the name of an already defined modifier, and **F** is the name of an already defined datatype restriction.

4.6 Define fuzzy numbers

```
fuzzy_number_range      := '(' 'define-fuzzy-number-range' numbers numbers ')'    # if
fuzzy numbers are used, then define the range [k1, k2]
fuzzy_number_expression := (
    name
    | numbers                                    # if fuzzy number is a real number 'n',
then it is considered as (n, n, n)
    | '(' numbers ',' numbers ',' numbers ')'
    | '(' 'f+' fuzzy_number_expression+ ')'                              # addition of
fuzzy numbers
    | '(' 'f-' fuzzy_number_expression fuzzy_number_expression ')'     # subtraction
of fuzzy numbers
    | '(' 'f*' fuzzy_number_expression+ ')'                              # product of
fuzzy numbers
    | '(' 'f/' fuzzy_number_expression fuzzy_number_expression ')'     # division of
fuzzy numbers
)
fuzzy_number            := '(' 'define-fuzzy-number' name fuzzy_number_expression ')'
```

4.6.1 Fuzzy Number Operations

The fuzzy number system supports several types of expressions, each with specific mathematical definitions:

4.6.2 Basic Fuzzy Number Representation

(a, b, c) represents a fuzzy number with the mathematical definition (a, b, c).

4.6.3 Real Number Conversion

n represents a real number that is automatically converted to the fuzzy number (n, n, n).

4.6.4 Addition of Fuzzy Numbers

$(f+ \ f_1 \ f_2 \ \ldots \ f_n)$ performs addition of multiple fuzzy numbers, defined mathematically as $(\sum_{i=0}^{n} a_i, \sum_{i=0}^{n} b_i, \sum_{i=0}^{n} c_i)$.

4.6.5 Subtraction of Fuzzy Numbers

$(f- \ f_1 \ f_2)$ performs subtraction between two fuzzy numbers, defined mathematically as $(a_1 - c_2, b_1 - b_2, c_1 - a_2)$.

4.6.6 Product of Fuzzy Numbers

$(f* \ f_1 \ f_2 \ \ldots \ f_n)$ performs multiplication of multiple fuzzy numbers, defined mathematically as $(\prod_{i=0}^{n} a_i, \prod_{i=0}^{n} b_i, \prod_{i=0}^{n} c_i)$.

4.6.7 Division of Fuzzy Numbers

$(f/ \ f_1 \ f_2)$ performs division between two fuzzy numbers, defined mathematically as $(\frac{a_1}{c_2}, \frac{b_1}{b_2}, \frac{c_1}{a_2})$.

4.7 Define Features, i.e., functional datatypes

```
feature         := '(' 'functional' name ')'        # first, define the feature
feature_range   := (
    '(' 'range' name '*integer*' numbers numbers ')'
    | '(' 'range' name '*real*' numbers numbers ')'
    | '(' 'range' name '*string*' ')'
    | '(' 'range' name '*boolean*' ')'
)
```

4.7.1 Feature Range Specifications

Features can be defined with different data types and ranges, each serving specific purposes in the fuzzy description logic system:

4.7.2 Feature Definition

(functional F) defines the feature F as a functional datatype that can be used throughout the knowledge base.

4.7.3 Integer Range Features

(range F `*integer*` k_1 k_2) specifies that the range of feature F consists of integer numbers within the interval $[k_1, k_2]$.

4.7.4 Real Number Range Features

(range F `*real*` k_1 k_2) specifies that the range of feature F consists of rational numbers within the interval $[k_1, k_2]$.

4.7.5 String Features

(range F `*string*`) specifies that the range of feature F consists of string values, allowing textual data to be handled within the fuzzy logic framework.

4.7.6 Boolean Features

(range F `*boolean*`) specifies that the range of feature F consists of boolean values (true/false), enabling logical propositions within the system.

4.8 Datatype/feature restrictions

```
# (>= ...) = at least datatype restriction
# (<= ...) = at most datatype restriction
# (= ...)  = exact datatype restriction

restriction_function    := (
    numbers
```

(continues on next page)

(continued from previous page)

```
    | name
    | numbers [*]? restriction_function
    | restriction_function [-] restriction_function
    | (restriction_function [+])+ restriction_function
)
restriction             := '(' ('>=' | '<=', '=') name (name | restriction_function |
fuzzy_number) ')'
```

4.8.1 Datatype Restriction Operations

The system supports various types of datatype restrictions that allow precise control over feature values and relationships:

4.8.2 Greater Than or Equal Restrictions with Variables

(>= F **variable**) defines a restriction where feature F must be greater than or equal to a variable, mathematically expressed as $\sup_{b \in \Delta_D}[F^{\mathcal{I}}(x, b) \otimes (b \geq \text{variable})]$.

4.8.3 Less Than or Equal Restrictions with Variables

(<= F **variable**) defines a restriction where feature F must be less than or equal to a variable, mathematically expressed as $\sup_{b \in \Delta_D}[F^{\mathcal{I}}(x, b) \otimes (b \leq \text{variable})]$.

4.8.4 Equality Restrictions with Variables

(= F **variable**) defines an exact equality restriction for feature F, mathematically expressed as $\sup_{b \in \Delta_D}[F^{\mathcal{I}}(x, b) \otimes (b = \text{variable})]$.

4.8.5 Greater Than or Equal Restrictions with Fuzzy Numbers

(>= F **fuzzy_number**) extends the comparison to fuzzy numbers, mathematically defined as $\sup_{b', b \in \Delta_D}[F^{\mathcal{I}}(x, b) \otimes (b \geq b') \otimes \text{fuzzy\_number}(b')^{\mathcal{I}}]$.

4.8.6 Less Than or Equal Restrictions with Fuzzy Numbers

(<= F **fuzzy_number**) provides fuzzy number comparison for upper bounds, mathematically defined as $\sup_{b', b \in \Delta_D}[F^{\mathcal{I}}(x, b) \otimes (b \leq b') \otimes \text{fuzzy\_number}(b')^{\mathcal{I}}]$.

4.8.7 Equality Restrictions with Fuzzy Numbers

(= F **fuzzy_number**) establishes equality constraints with fuzzy numbers, mathematically expressed as $\sup_{b', b \in \Delta_D}[F^{\mathcal{I}}(x, b) \otimes (b = b') \otimes \text{fuzzy\_number}(b')^{\mathcal{I}}]$.

4.8.8 Function-Based Restrictions

(>= F function$(F_1, \ldots, F_n)$), (<= F function$(F_1, \ldots, F_n)$), and (= F function$(F_1, \ldots, F_n)$) allow restrictions based on complex functions of multiple features, with mathematical definitions following the pattern $\sup_{b \in \Delta_D}[F^{\mathcal{I}}(x, b) \otimes (b \text{ op } \text{function}(F_1, \ldots, F_n)^{\mathcal{I}})]$ where **op** represents the comparison operator.

4.8.9 Variable and Value Constraints

In datatype restrictions, the variable **variable** must be declared as a **(free variable)** before its use in a datatype restriction, utilizing the **constraints** defined below. The value for b must be within the range $[k_1, k_2]$ subset or equivalent to $[-k_\infty, k_\infty]$ of the feature F, and the values for **variable**, $\mathbf{function}(F_1, \ldots, F_n)$ and the range of **fuzzy_number** must be in $[-k_\infty, k_\infty]$, where k_∞ is the maximal representable integer.

In datatype restrictions, the variable **variable** may be replaced with a value, such as an integer, a real number, a string, or a boolean constant (true, false), depending on the range of the feature F.

4.8.10 MILP Solver Constraints

The value of k_∞ varies depending on the MILP solver used, reflecting computational limitations and precision requirements:

MILP Solver	k_∞
Gurobi	$1000 \cdot ((1 \ll 31) - 1)$
PULP CBC	$(1 \ll 31) - 1$
MIP	$(1 \ll 31) - 1$
PULP GLPK	$(1 \ll 28) - 1$
PULP HiGHS	$(1 \ll 28) - 1$
PULP CPLEX	$(1 \ll 28) - 1$

- Note: The value of k_∞ is different for some MILP solvers for computational issues. In particular, higher values lead to the accumulation of errors, which can distort the results. The values currently given give the same results for the test files provided.

4.9 Constraints

```
operator                := '>=' | '<=' | '='
term                    := numbers | name | numbers [*] term | name [*] term
expression              := (term [+])+ term
inequation_constraint   := expression operator numbers
constraints             := '(' 'constraints' (
    inequality_constraint
    | 'binary' name             # binary variable in {0, 1}
    | 'free' name               # continuous variable in (-inf, +inf)
) ')'
```

4.10 Show statements

```
statements = (
    '(' 'show-concrete-fillers' name+ ')'               # show value of the fillers
    | '(' 'show-concrete-fillers-for' name{2, } ')'     # show value of the fillers
for an individual
    | '(' 'show-concrete-instance-for' name{3, } ')'    # show degrees of being the
filler of individual instance of a concept
    | '(' 'show-abstract-fillers' name+ ')'             # show fillers and membership
to any concept
    | '(' 'show-abstract-fillers-for' name{2, } ')'     # show fillers for an
individuals and membership to any concept
    | '(' 'show-concepts' name+ ')'                     # show membership of
individuals to any concept
    | '(' 'show-instances' name+ ')'                    # show value of the instances
of the listed concepts
    | '(' 'show-variables' name+ ')'                    # show value of the listed
variables
    | '(' 'show-language' ')'                           # show language of the KB,
from ALC to SHIF(D)
)
```

4.10.1 Show Statement Operations

The system provides various show statements for debugging and analyzing the knowledge base contents:

4.10.2 Concrete Filler Display

(show-concrete-fillers $F_1 \ldots F_n$**)** displays the values of the fillers of the features F_i, providing concrete instantiations of feature relationships.

4.10.3 Individual-Specific Concrete Fillers

(show-concrete-fillers-for ind $F_1 \ldots F_n$**)** shows the values of the fillers of features F_i specifically for the individual **ind**, allowing targeted analysis of individual properties.

4.10.4 Concept Instance Degrees

(show-concrete-instance-for ind $F\ C_1 \ldots C_n$**)** displays the degrees of being the F filler of the individual **ind** instance of concepts C_i, revealing membership strengths in fuzzy concepts.

4.10.5 Abstract Filler Analysis

(show-abstract-fillers $R_1 \ldots R_n$**)** shows fillers of roles R_i and their membership to any concept, providing comprehensive relationship analysis.

4.10.6 Individual Abstract Fillers

(show-abstract-fillers-for ind $R_1 \ldots R_n$**)** displays fillers of roles R_i for the individual **ind** and their membership to any concept, focusing on specific individual relationships.

4.10.7 Concept Membership Display

(show-concepts $a_1 \ldots a_n$**)** shows membership of the individuals a_i to any concept, revealing the concept classification of individuals.

4.10.8 Concept Instance Values

(show-instances $C_1 \ldots C_n$**)** displays the values of the instances of the concepts C_i, showing which individuals belong to specific concepts.

4.10.9 Variable Value Display

(show-variables $x_1 \ldots x_n$**)** shows the values of the variables x_i, useful for debugging constraint satisfaction problems.

4.10.10 Knowledge Base Language Display

(show-language) shows the language of the knowledge base, indicating the description logic expressivity from $\mathcal{ALC}$ to $\mathcal{SHIF}(\mathbf{D})$.

4.11 Crisp declarations

```
crisp_concepts  := '(' 'crisp-concept' name+ ')' # the listed concepts are crisp
crisp_roles     := '(' 'crisp-role' name+ ')' # the listed roles are crisp
```

4.12 Fuzzy relations

```
fuzzy_similarity    := '(' 'define-fuzzy-similarity'   name ')' # fuzzy similarity
relation
fuzzy_equivalence   := '(' 'define-fuzzy-equivalence'  name ')' # fuzzy equivalence
relation
```

4.13 Concept expressions

```
concept := (
    '*top*'                                                     # top concept
    | '*bottom*'                                                # bottom concept
    | name                                                      # atomic concept or
concrete fuzzy concept
    | restriction                                               # datatype restriction
    | '(' 'and' concept concept ')'                             # concept conjunction
    | '(' 'g-and' concept concept ')'                           # Goedel conjunction
    | '(' 'l-and' concept concept ')'                           # Lukasiewicz
conjunction
    | '(' 'or' concept concept ')'                              # concept disjunction
    | '(' 'g-or' concept concept ')'                            # Goedel disjunction
    | '(' 'l-or' concept concept ')'                            # Lukasiewicz
disjunction
    | '(' 'not' concept ')'                                     # concept negation
    | '(' 'implies' concept concept ')'                         # concept implication
    | '(' 'g-implies' concept concept ')'                       # Goedel implication
    | '(' 'l-implies' concept concept ')'                       # Lukasiewicz
implication
    | '(' 'kd-implies' concept concept ')'                      # Kleene-Dienes
implication
    | '(' 'all' name concept ')'                                # universal role
restriction
    | '(' 'some' name concept ')'                               # existential role
restriction
    | '(' 'some' name name ')'                                  # individual value
restriction
    | '(' 'ua' name concept ')'                                 # upper approximation
    | '(' 'lua' name concept ')'                                # loose upper
approximation
    | '(' 'tua' name concept ')'                                # tight upper
approximation
    | '(' 'la' name concept ')'                                 # lower approximation
    | '(' 'lla' name concept ')'                                # loose lower
approximation
    | '(' 'tla' name concept ')'                                # tight lower
approximation
    | '(' 'self' concept ')'                                    # local reflexivity
concept
    | '(' name concept ')'                                      # modifier applied to
concept
    | '(' fuzzy_number ')'                                      # fuzzy number
    | '(' '[' ('>=' | '<=') name ']' concept ')'                # threshold concept
    | '(' numbers concept ')'                                   # weighted concept
    | '(' 'w-sum' ('(' numbers concept ')')+ ')'                # weighted sum concept
    | '(' 'w-max' ('(' numbers concept ')')+ ')'                # weighted max concept
    | '(' 'w-min' ('(' numbers concept ')')+ ')'                # weighted min concept
```

(continues on next page)

(continued from previous page)

```
    | '(' 'w-sum-zero' ('(' numbers concept ')')+ ')'          # weighted sum zero
concept
    | '(' 'owa' numbers+ concept+ ')'                          # OWA aggregation
operator
    | '(' 'q-owa' name concept+ ')'                            # quantifier-guided
OWA
    | '(' 'choquet' numbers+ concept+ ')'                      # Choquet integral
    | '(' 'sugeno' numbers+ concept+ ')'                       # Sugeno integral
    | '(' 'q-sugeno' numbers+ concept+ ')'                     # Quasi-Sugeno
integral
    | '(' 'sigma-count' name concept '{' name+ '}' name ')'    # Sigma-count concept
)
```

4.13.1 Basic Concept Expressions

The fuzzy description logic provides fundamental concept building blocks:

4.13.2 Top Concept

top represents the universal concept that always evaluates to true, mathematically defined as $\top = 1$.

4.13.3 Bottom Concept

bottom represents the empty concept that always evaluates to false, mathematically defined as $\bot = 0$.

4.13.4 Atomic Concepts

A represents an atomic concept A, evaluated as $A^{\mathcal{I}}(x)$ in the interpretation.

4.13.5 Concrete Fuzzy Concepts

CFC represents a concrete fuzzy concept (such as crisp, left-shoulder, and so on), evaluated as $\mathrm{CFC}^{\mathcal{I}}(x)$.

4.13.6 Datatype Restrictions

DR represents a datatype restriction, evaluated as $\mathrm{DR}^{\mathcal{I}}(x)$.

4.13.7 Logical Connectives

The system supports various logical operations with different semantics:

4.13.8 Standard Conjunction

(and C_1 C_2**)** performs concept conjunction of C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \otimes C_2^{\mathcal{I}}(x)$.

4.13.9 Goedel Conjunction

(g-and C_1 C_2**)** performs Goedel conjunction of C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \otimes_G C_2^{\mathcal{I}}(x)$.

4.13.10 Lukasiewicz Conjunction

(l-and C_1 C_2**)** performs Lukasiewicz conjunction of C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \otimes_L C_2^{\mathcal{I}}(x)$.

4.13.11 Standard Disjunction

(or C_1 C_2**)** performs concept disjunction of C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \oplus C_2^{\mathcal{I}}(x)$.

4.13.12 Goedel Disjunction

(g-or C_1 C_2**)** performs Goedel disjunction of C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \oplus_G C_2^{\mathcal{I}}(x)$.

4.13.13 Lukasiewicz Disjunction

(l-or C_1 C_2**)** performs Lukasiewicz disjunction of C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \oplus_L C_2^{\mathcal{I}}(x)$.

4.13.14 Negation

(not C**)** performs concept C negation, defined as $\ominus_L C^{\mathcal{I}}(x)$.

4.13.15 Implication Operations

Various forms of implication are supported for different logical semantics:

4.13.16 Standard Implication

(implies C_1 C_2**)** establishes concept implication between C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \Rightarrow C_2^{\mathcal{I}}(x)$.

4.13.17 Goedel Implication

(g-implies C_1 C_2**)** establishes Goedel implication between C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \Rightarrow_G C_2^{\mathcal{I}}(x)$.

4.13.18 Lukasiewicz Implication

(l-implies C_1 C_2**)** establishes Lukasiewicz implication between C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \Rightarrow_L C_2^{\mathcal{I}}(x)$.

4.13.19 Kleene-Dienes Implication

(kd-implies C_1 C_2**)** establishes Kleene-Dienes implication between C_1 and C_2, defined as $C_1^{\mathcal{I}}(x) \Rightarrow_{\mathrm{KD}} C_2^{\mathcal{I}}(x)$.

4.13.20 Role Restrictions

Role-based concept formation allows complex relationship modeling:

4.13.21 Universal Role Restriction

(all R C**)** creates a universal role R restriction for concept C, defined as $\inf_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)$.

4.13.22 Existential Role Restriction

(some R C**)** creates an existential role R restriction for concept C, defined as $\sup_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)$.

4.13.23 Individual Value Restriction

(some R a**)** creates an individual value restriction for role R and individual a, defined as $R^{\mathcal{I}}(x,a)$.

4.13.24 Approximation Concepts

Rough set theory concepts for handling uncertainty:

4.13.25 Upper Approximation

(ua s C**)** defines upper approximation for a fuzzy relation s and concept C, calculated as $\sup_{y \in \Delta^{\mathcal{I}}} s^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)$.

4.13.26 Loose Upper Approximation

(lua s C**)** defines loose upper approximation for a fuzzy relation s and concept C, calculated as $\sup_{z \in X} s^{\mathcal{I}}(x,z) \otimes \sup_{y \in \Delta^{\mathcal{I}}} s^{\mathcal{I}}(y,z) \otimes C^{\mathcal{I}}(x)$.

4.13.27 Tight Upper Approximation

(tua s C**)** defines tight upper approximation for a fuzzy relation s and concept C, calculated as $\inf_{z \in X} s^{\mathcal{I}}(x,z) \Rightarrow \sup_{y \in \Delta^{\mathcal{I}}} s^{\mathcal{I}}(y,z) \otimes C^{\mathcal{I}}(x)$.

4.13.28 Lower Approximation

(la s C**)** defines lower approximation for a fuzzy relation s and concept C, calculated as $\inf_{y \in \Delta^{\mathcal{I}}} s^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)$.

4.13.29 Loose Lower Approximation

(lla s C**)** defines loose lower approximation for a fuzzy relation s and concept C, calculated as $\sup_{z \in X} s^{\mathcal{I}}(x,z) \otimes \inf_{y \in \Delta^{\mathcal{I}}} s^{\mathcal{I}}(y,z) \otimes C^{\mathcal{I}}(x)$.

4.13.30 Tight Lower Approximation

(tla s C**)** defines tight lower approximation for a fuzzy relation s and concept C, calculated as $\inf_{z \in X} s^{\mathcal{I}}(x,z) \Rightarrow \inf_{y \in \Delta^{\mathcal{I}}} s^{\mathcal{I}}(y,z) \otimes C^{\mathcal{I}}(x)$.

4.13.31 Special Concept Constructs

Advanced concept formation techniques:

4.13.32 Self Reflexivity

(self C) defines a local reflexivity concept, evaluated as $C^{\mathcal{I}}(x)(x,x)$.

4.13.33 Modified Concepts

(MOD C) applies modifier MOD to concept C, evaluated as $f_m(C^{\mathcal{I}}(x))$, where f_m is the modifier function associated to MOD.

4.13.34 Fuzzy Numbers in Concepts

(FN) represents fuzzy number FN as a concept, evaluated as $\mathrm{FM}^{\mathcal{I}}(x)$.

4.13.35 Threshold Concepts

([>= var] C) creates a threshold concept that returns $C^{\mathcal{I}}(x)$ if $C^{\mathcal{I}}(x) \geq \mathrm{var}$, otherwise returns 0.

([<= var] C) creates a threshold concept that returns $C^{\mathcal{I}}(x)$ if $C^{\mathcal{I}}(x) \leq \mathrm{var}$, otherwise returns 0.

4.13.36 Weighted Concepts

Concepts with numerical weights for aggregation:

4.13.37 Basic Weighted Concept

(n C) creates a weighted concept C with weight n, evaluated as $nC^{\mathcal{I}}(x)$.

4.13.38 Weighted Sum

(w-sum $(n_1\ C_1) \ldots (n_k\ C_k)$**)** performs weighted sum of concepts, calculated as $\sum_{i=1}^{k} n_i C_i^{\mathcal{I}}(x)$.

4.13.39 Weighted Maximum

(w-max $(v_1\ C_1) \ldots (v_k\ C_k)$**)** performs weighted max of concepts, calculated as $\max_{i=1}^{k} \min v_i, x_i$.

4.13.40 Weighted Minimum

(w-min $(v_1\ C_1) \ldots (v_k\ C_k)$**)** performs weighted min of concepts, calculated as $\min_{i=1}^{k} \max 1 - v_i, x_i$.

4.13.41 Weighted Sum Zero

(w-sum-zero $(n_1\ C_1) \ldots (n_k\ C_k)$**)** performs weighted sum with zero-constraint, returning 0 if any $C_i^{\mathcal{I}}(x) = 0$ for some $i \in 1, \ldots, k$, otherwise $\sum_{i=1}^{k} n_i C_i^{\mathcal{I}}(x)$.

4.13.42 Aggregation Operators

Advanced aggregation techniques for concept combination:

4.13.43 OWA Aggregation

(owa $(w_1, \ldots, w_n)$ $(C_1, \ldots, C_n)$**)** performs OWA aggregation, calculated as $\sum_{i=1}^{n} w_i y_i$.

4.13.44 Quantifier-Guided OWA

(q-owa Q $(C_1, \ldots, C_n)$**)** performs quantifier-guided OWA with quantifier Q (where Q is a right-shoulder or linear function), calculated as $\sum_{i=1}^{n} w_i y_i$, where $w_i = Q(\frac{i}{n}) - Q(\frac{i-1}{n})$.

4.13.45 Choquet Integral

(choquet $(w_1, \ldots, w_n)$ $(C_1, \ldots, C_n)$**)** computes the Choquet integral, calculated as $y_1 w_1 + \sum_{i=2}^{n} (y_i - y_{i-1}) w_i$.

4.13.46 Sugeno Integral

(sugeno $(v_1, \ldots, v_n)$ $(C_1, \ldots, C_n)$**)** computes the Sugeno integral, calculated as $\max_{i=1}^{n} \min y_i, mu_i$.

4.13.47 Quasi-Sugeno Integral

(q-sugeno $(v_1, \ldots, v_n)$ $(C_1, \ldots, C_n)$**)** computes the Quasi-Sugeno integral, calculated as $\max_{i=1}^{n} y_i \otimes_L mu_i$.

4.13.48 Sigma-Count Concept

(sigma-count $R\ C\ a_1\ \ldots\ a_k\ F_C$**)** creates a Sigma-Count concept with role R and associated to the concept C, the individuals a_i and the fuzzy concrete concept F_C.

4.13.49 Important Constraints and Notes

Several constraints apply to the proper use of these concept expressions:

- $n_1, \ldots, n_k \in [0, 1]$, with $\sum_{i=1}^{k}\ n_i \leq 1$
- $w_1, \ldots, w_n \in [0, 1]$, with $\sum_{i=1}^{n}\ w_i = 1$
- $v_1, \ldots, v_n \in [0, 1]$, with $\max_{i=1}^{n}\ v_i = 1$

- y_i is the i-largest of the $C_i^{\mathcal{I}}(x)$
- ow_i is the weight v_i of the i-largest of the $C_i^{\mathcal{I}}(x)$
- mu_i is defined as follows: $mu_1 = ow_1$, and $mu_i = ow_i \oplus mu_{i-1}$ for $i \in 2, \ldots, n$
- Fuzzy numbers can only appear in existential, universal and datatype restrictions
- In threshold concepts **var** may be replaced with $w \in [0, 1]$
- Fuzzy relations s should be previously defined as fuzzy similarity relation or a fuzzy equivalence relation as **(define-fuzzy-similarity s)** or **(define-fuzzy-equivalence s)**, respectively
- Fuzzy concrete concept F_C in **sigma-count** concept has to be previously defined as **left-shoulder**, **right-shoulder** or **triangular** concept with **define-fuzzy-concept**

4.14 Axioms

```
degree := (
    numbers             # a rational number
    | expression        # a linear expression
    | name              # variable or an already defined truth constant
)
axioms := (
    '(' 'instance' name concept degree? ')'             # concept assertion
    | '(' 'related' name name name degree? ')'          # role assertion
    | '(' 'implies' concept concept numbers? ')'        # General Concept Inclusion
(GCI) with degree 'numbers'
    | '(' 'g-implies' concept concept numbers? ')'      # Goedel GCI with degree
↪'numbers'
    | '(' 'kd-implies' concept concept numbers? ')'     # Kleene-Dienes GCI with
degree 'numbers'
    | '(' 'l-implies' concept concept numbers? ')'      # Lukasiewicz GCI with degree
↪'numbers'
    | '(' 'z-implies' concept concept numbers? ')'      # Zadeh's set GCI with degree
↪'numbers'
    | '(' 'define-concept' name concept ')'             # concept definition
    | '(' 'define-primitive-concept' name concept ')'   # concept subsumption
    | '(' 'equivalent-concepts' concept concept ')'     # equivalent concept
definition
    | '(' 'disjoint' concept+  ')'                      # concept disjointness
    | '(' 'disjoint-union' concept+ ')'                 # disjoint union of concepts
    | '(' 'range' name concept ')'                      # range restriction of a
concept
    | '(' 'domain' name concept ')'                     # domain restriction of a
concept
    | '(' 'functional' name ')'                         # functional role
    | '(' 'inverse-functional' name ')'                 # inverse functional role
    | '(' 'reflexive' name ')'                          # reflexive role
    | '(' 'symmetric' name ')'                          # symmetric role
    | '(' 'transitive' name ')'                         # transitive role
    | '(' 'implies-role' name name numbers? ')'         # Role Implication Axiom (RIA)
    | '(' 'inverse' name name ')'                       # inverse role
)
```

4.14.1 Assertion Axioms

Basic statements about individuals and their relationships:

4.14.2 Concept Assertions

(instance a C d**)** asserts that individual a is an instance of concept C with degree d, formally defined as $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq d$.

4.14.3 Role Assertions

(related a b R d**)** asserts that individuals a and b are related by role R with degree d, formally defined as $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \geq d$.

4.14.4 General Concept Inclusion Axioms

Various forms of concept subsumption with different logical semantics:

4.14.5 Standard General Concept Inclusion

(implies C_1 C_2 **d)** establishes that concept C_1 implies concept C_2 with degree d, formally defined as $\inf_{x \in \Delta^{\mathcal{I}}} C_1^{\mathcal{I}}(x) \Rightarrow C_2^{\mathcal{I}}(x) \geq d$.

4.14.6 Goedel General Concept Inclusion

(g-implies C_1 C_2 **d)** establishes Goedel implication between concepts with degree d, formally defined as $\inf_{x \in \Delta^{\mathcal{I}}} C_1^{\mathcal{I}}(x) \Rightarrow_G C_2^{\mathcal{I}}(x) \geq d$.

4.14.7 Kleene-Dienes General Concept Inclusion

(kd-implies C_1 C_2 **d)** establishes Kleene-Dienes implication between concepts with degree d, formally defined as $\inf_{x \in \Delta^{\mathcal{I}}} C_1^{\mathcal{I}}(x) \Rightarrow_{\mathrm{KD}} C_2^{\mathcal{I}}(x) \geq d$.

4.14.8 Lukasiewicz General Concept Inclusion

(l-implies C_1 C_2 **d)** establishes Lukasiewicz implication between concepts with degree d, formally defined as $\inf_{x \in \Delta^{\mathcal{I}}} C_1^{\mathcal{I}}(x) \Rightarrow_L C_2^{\mathcal{I}}(x) \geq d$.

4.14.9 Zadeh General Concept Inclusion

(z-implies C_1 C_2 **d)** establishes Zadeh set implication between concepts with degree d, formally defined as $\inf_{x \in \Delta^{\mathcal{I}}} C_1^{\mathcal{I}}(x) \Rightarrow_Z C_2^{\mathcal{I}}(x) \geq d$.

4.14.10 Concept Definition Axioms

Formal concept definitions and relationships:

4.14.11 Complete Concept Definition

(define-concept A C**)** provides a complete definition of atomic concept A as equivalent to complex concept C, formally defined as $\forall_{x \in \Delta^{\mathcal{I}}} A^{\mathcal{I}}(x) = C^{\mathcal{I}}(x)$.

4.14.12 Primitive Concept Definition

(define-primitive-concept A C**)** establishes that atomic concept A is subsumed by complex concept C, formally defined as $\inf_{x \in \Delta^{\mathcal{I}}} A^{\mathcal{I}}(x) \leq C^{\mathcal{I}}(x)$.

4.14.13 Concept Equivalence

(equivalent-concepts C_1 C_2**)** establishes that two concepts are equivalent, formally defined as $\forall_{x \in \Delta^{\mathcal{I}}} C_1^{\mathcal{I}}(x) = C_2^{\mathcal{I}}(x)$.

4.14.14 Concept Disjointness and Union

Axioms for controlling concept overlap:

4.14.15 Concept Disjointness

(disjoint $C_1 \ldots C_k$**)** declares that the listed concepts are mutually disjoint, equivalent to (implies (g-and C_i C_j) `*bottom*`) for all pairs, formally $\forall_{i,j \in 1,\ldots,k, i<j} (C_i^{\mathcal{I}}(x) \otimes_G C_j^{\mathcal{I}}(x)) \Rightarrow \bot$.

4.14.16 Disjoint Union

(disjoint-union $C_1 \ldots C_k$**)** establishes that the first concept is the disjoint union of the others, formally $C_1 = \bigoplus_{i=2}^{k} C_i$ and $\forall_{i,j \in 1,\ldots,k, i<j} (C_i^{\mathcal{I}}(x) \otimes_G C_j^{\mathcal{I}}(x)) \Rightarrow \bot$.

4.14.17 Role Restriction Axioms

Domain and range constraints for roles:

4.14.18 Role Range Restriction

(range R C**)** restricts the range of role R to concept C, equivalent to (implies `*top*` (all R C)), formally $\top \Rightarrow \inf_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x, y) \Rightarrow C^{\mathcal{I}}(y)$.

4.14.19 Role Domain Restriction

(domain R C**)** restricts the domain of role R to concept C, equivalent to (implies (some R `*top*`) C), formally $\sup_{y \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(x, y) \otimes \top \Rightarrow C^{\mathcal{I}}(x)$.

4.14.20 Role Property Axioms

Fundamental properties that roles can possess:

4.14.21 Functional Roles

(functional R**)** declares role R as functional, meaning each individual can be related to at most one other individual, formally $R^{\mathcal{I}}(a, b) = R^{\mathcal{I}}(a, c) \rightarrow b = c$.

4.14.22 Inverse Functional Roles

(inverse-functional R**)** declares role R as inverse functional, meaning each individual can be the target of at most one relationship, formally $R^{\mathcal{I}}(b, a) = R^{\mathcal{I}}(c, a) \rightarrow b = c$.

4.14.23 Reflexive Roles

(reflexive R**)** declares role R as reflexive, meaning every individual is related to itself, formally $\forall_{a \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(a, a) = 1$.

4.14.24 Symmetric Roles

(symmetric R**)** declares role R as symmetric, meaning if a is related to b, then b is related to a, formally $\forall_{a,b \in \Delta^{\mathcal{I}}} R^{\mathcal{I}}(a, b) = R^{\mathcal{I}}(b, a)$.

4.14.25 Transitive Roles

(transitive R**)** declares role R as transitive, meaning the relationship chains together, formally $\forall_{a,b\in\Delta^{\mathcal{I}}} R^{\mathcal{I}}(a,b) \geq \sup_{c\in\Delta^{\mathcal{I}}} R^{\mathcal{I}}(a,c) \otimes R^{\mathcal{I}}(c,b)$.

4.14.26 Role Relationship Axioms

Axioms governing relationships between roles:

4.14.27 Role Implication

(implies-role R_1 R_2 d**)** establishes that role R_1 implies role R_2 with degree d, formally $\inf_{x,y\in\Delta^{\mathcal{I}}} R_1^{\mathcal{I}}(x,y) \Rightarrow_L R_2^{\mathcal{I}}(x,y) \geq d$.

4.14.28 Inverse Roles

(inverse R_1 R_2**)** declares that role R_1 is the inverse of role R_2, formally $R_1^{\mathcal{I}} \equiv (R_2^{\mathcal{I}})^{-1}$.

4.14.29 Important Constraints

Several important constraints must be observed when using axioms:

- Transitive roles cannot be functional
- In Zadeh logic, $\Rightarrow$ is Zadeh's set inclusion

4.15 Queries

```
queries := (
      '(' 'sat?' ')'                # is Knowledge base consistent?
    | '(' 'max-instance?' name concpet ')'
    | '(' 'min-instance?' name concept ')'
    | '(' 'all-instances?' concept ')'
    | '(' 'max-related?' name name name ')'
    | '(' 'min-related?' name name name')'
    | '(' 'max-subs?' concept concept ')'
    | '(' 'min-subs?' concept concept ')'
    | '(' 'max-g-subs?' concept concept ')'
    | '(' 'min-g-subs?' concept concept ')'
    | '(' 'max-l-subs?' concept concept ')'
    | '(' 'min-l-subs?' concept concept ')'
    | '(' 'max-kd-subs?' concept concept ')'
    | '(' 'min-kd-subs?' concept concept ')'
    | '(' 'max-sat?' concept name? ')'
    | '(' 'min-sat?' concept name? ')'
    | '(' 'max-var?' name ')'
    | '(' 'min-var?' name ')'
    | '(' 'defuzzify-lom?' concept name name ')'    # Defuzzify using the largest of
the maxima
    | '(' 'defuzzify-mom?' concept name name ')'    # Defuzzify using the middle of
the maxima
    | '(' 'defuzzify-som?' concept name name ')'    # Defuzzify using the smallest of
the maxima
    | '(' 'bnp?' name ')'                           # Computes the Best Non-Fuzzy
Performance (BNP) of a fuzzy number
)
```

4.15.1 Knowledge Base Consistency Queries

Fundamental queries for knowledge base validation:

4.15.2 Satisfiability Check

(sat?) checks if the knowledge base $\mathcal{K}$ is consistent, determining whether there exists a valid interpretation that satisfies all axioms.

4.15.3 Instance Membership Queries

Queries for determining individual concept membership:

4.15.4 Maximum Instance Membership

(max-instance? a C**)** computes the maximum degree to which individual a belongs to concept C, formally $\sup n \mid \mathcal{K} \models$ (instance a C n).

4.15.5 Minimum Instance Membership

(min-instance? a C**)** computes the minimum degree to which individual a belongs to concept C, formally $\inf n \mid \mathcal{K} \models$ (instance a C n).

4.15.6 All Instance Memberships

(all-instances? C**)** computes (min-instance? a C) for every individual a in the knowledge base, providing a comprehensive view of concept membership.

4.15.7 Role Relationship Queries

Queries for analyzing relationships between individuals:

4.15.8 Maximum Role Relationship

(max-related? a b R**)** computes the maximum degree to which individuals a and b are related by role R, formally $\sup n \mid \mathcal{K} \models$ (related a b R n).

4.15.9 Minimum Role Relationship

(min-related? a b R**)** computes the minimum degree to which individuals a and b are related by role R, formally $\inf n \mid \mathcal{K} \models$ (related a b R n).

4.15.10 Concept Subsumption Queries

Queries for analyzing hierarchical relationships between concepts:

4.15.11 Maximum Standard Subsumption

(max-subs? C D**)** computes the maximum degree to which concept D subsumes concept C, formally $\sup n \mid \mathcal{K} \models$ (implies D C n).

4.15.12 Minimum Standard Subsumption

(min-subs? C D**)** computes the minimum degree to which concept D subsumes concept C, formally $\inf n \mid \mathcal{K} \models$ (implies D C n).

4.15.13 Maximum Goedel Subsumption

(max-g-subs? C D**)** computes the maximum degree of Goedel subsumption between concepts, formally $\sup\ n \mid \mathcal{K} \models$ (g-implies D C n).

4.15.14 Minimum Goedel Subsumption

(min-g-subs? C D**)** computes the minimum degree of Goedel subsumption between concepts, formally $\inf\ n \mid \mathcal{K} \models$ (g-implies D C n).

4.15.15 Maximum Lukasiewicz Subsumption

(max-l-subs? C D**)** computes the maximum degree of Lukasiewicz subsumption between concepts, formally $\sup\ n \mid \mathcal{K} \models$ (l-implies D C n).

4.15.16 Minimum Lukasiewicz Subsumption

(min-l-subs? C D**)** computes the minimum degree of Lukasiewicz subsumption between concepts, formally $\inf\ n \mid \mathcal{K} \models$ (l-implies D C n).

4.15.17 Maximum Kleene-Dienes Subsumption

(max-kd-subs? C D**)** computes the maximum degree of Kleene-Dienes subsumption between concepts, formally $\sup\ n \mid \mathcal{K} \models$ (kd-implies D C n).

4.15.18 Minimum Kleene-Dienes Subsumption

(min-kd-subs? C D**)** computes the minimum degree of Kleene-Dienes subsumption between concepts, formally $\inf\ n \mid \mathcal{K} \models$ (kd-implies D C n).

4.15.19 Concept Satisfiability Queries

Queries for analyzing concept satisfiability across all models:

4.15.20 Maximum Concept Satisfiability

(max-sat? C a**)** computes the maximum satisfiability of concept C across all interpretations, formally $\sup_{\mathcal{I}} \sup_{a \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(a)$.

4.15.21 Minimum Concept Satisfiability

(min-sat? C a**)** computes the minimum satisfiability of concept C across all interpretations, formally $\inf_{\mathcal{I}} \inf_{a \in \Delta^{\mathcal{I}}} C^{\mathcal{I}}(a)$.

4.15.22 Variable Optimization Queries

Queries for optimizing variable values within constraints:

4.15.23 Maximum Variable Value

(max-var? var) computes the maximum value that variable **var** can take while maintaining knowledge base consistency, formally $\sup$ var $\mid \mathcal{K}$ is consistent.

4.15.24 Minimum Variable Value

(min-var? var) computes the minimum value that variable **var** can take while maintaining knowledge base consistency, formally $\inf$ var $\mid \mathcal{K}$ is consistent.

4.15.25 Defuzzification Queries

Queries for converting fuzzy values to crisp values using different strategies:

4.15.26 Largest of Maxima Defuzzification

(defuzzify-lom? C a F**)** defuzzifies the value of feature F using the largest of the maxima strategy, where concept C represents Mamdani/Rules IF-THEN fuzzy rules for determining the feature value.

4.15.27 Middle of Maxima Defuzzification

(defuzzify-mom? C a F**)** defuzzifies the value of feature F using the middle of the maxima strategy, providing a balanced approach to crisp value selection.

4.15.28 Smallest of Maxima Defuzzification

(defuzzify-som? C a F**)** defuzzifies the value of feature F using the smallest of the maxima strategy, offering a conservative approach to defuzzification.

4.15.29 Fuzzy Number Performance Analysis

4.15.30 Best Non-Fuzzy Performance

(bnp? f**)** computes the Best Non-Fuzzy Performance (BNP) of a fuzzy number f, providing a measure of how well the fuzzy number can be represented as a crisp value.

4.15.31 Important Notes on Defuzzification

In defuzzify queries, the concept C represents several Mamdani/Rules IF-THEN fuzzy rules expressing how to obtain the value of the concrete feature F. These rules form the basis for the fuzzy inference process that determines the appropriate crisp output value.

CHAPTER

FIVE

FUZZY CONCEPTS

The fuzzy concept currently implemented are as follows.

5.1 Base Concepts

5.1.1 Concept

Defines a base class of any concept. This is the fundamental class from which all other concept implementations derive.

5.1.2 AtomicConcept

Defines an atomic concept. This represents the smallest, indivisible unit in the fuzzy logic system.

5.1.3 TruthConcept

Defines the top $\top$ and bottom $\bot$ concepts. These represent absolute truth and absolute falsehood respectively in the logical system.

5.2 Logical Operators

5.2.1 OperatorConcept

It is the class to handle a logic (Zadeh, Lukasiewicz, and product) connectives (AND, OR, NOT) between fuzzy concepts. This class implements the fundamental logical operations in fuzzy logic systems.

5.2.2 AllSomeConcept

Defines a universal ($\forall$) and existential ($\exists$) restrictions on fuzzy concepts. These allow for expressing quantifications in the fuzzy domain.

5.2.3 ImpliesConcept

Defines a zadeh and goedel implies concept. This handles implication operations in fuzzy logic.

5.3 Fuzzy Integrals

5.3.1 ChoquetIntegral

Defines a Choquet integral of fuzzy concept. This integral is used for aggregating information when measures are non-additive.

5.3.2 SugenoIntegral

Defines a Sugeno integral of fuzzy concept. This is a particular type of non-linear integral with respect to fuzzy measures.

5.3.3 QsugenoIntegral

Defines a Quasi-Sugeno integral of fuzzy concept. This extends the Sugeno integral with additional flexibility.

5.4 OWA Concepts

5.4.1 OwaConcept

Defines a OWA concept. OWA (Ordered Weighted Averaging) operators provide a family of aggregation operators.

5.4.2 QowaConcept

Defines a quantified-guided OWA concept. This extends the OWA concept with quantifier-guided behavior.

5.5 Approximation and Threshold Concepts

5.5.1 ApproximationConcept

Defines uppers and lowers approximation concept. This handles rough set approximations in the fuzzy context.

5.5.2 ThresholdConcept

Defines a positives and negatives threshold concept. This implements threshold-based classification in fuzzy systems.

5.5.3 ExtendedThresholdConcept

Defines a extended positives and negatives threshold concept. This provides enhanced threshold functionality beyond the basic threshold concept.

5.6 Value-Based Concepts

5.6.1 HasValueConcept

Defines a concept associated with a value. This links concepts to specific values in the domain.

5.6.2 ValueConcept

Defines a datatype restriction (at most, at least, and exact) concept. This handles numerical constraints and restrictions.

5.6.3 SelfConcept

Defines a self reflexivity concept. This implements self-referential properties in fuzzy logic.

5.6.4 NegatedNominal

Defines a negated nominal concept. This handles the negation of nominal (named) concepts.

5.7 Weighted Concepts

5.7.1 WeightedConcept

Defines a weighted concept. This provides basic weighting functionality for concepts.

5.7.2 WeightedMinConcept

Defines a weighted min concept. This implements weighted minimum operations.

5.7.3 WeightedMaxConcept

Defines a weighted max concept. This implements weighted maximum operations.

5.7.4 WeightedSumConcept

Defines a weighted sum concept. This implements weighted summation operations.

5.7.5 WeightedSumZeroConcept

Defines a weighted sum-zero concept. This implements weighted sum operations with zero-centering.

5.8 Counting and Modification Concepts

5.8.1 SigmaConcept

Defines a sigma-count concept. This handles counting operations in fuzzy contexts.

5.8.2 ModifiedConcept

Defines the base class for modified (linear and triangular) concepts. This is the parent class for concepts that can be modified through various functions.

5.8.3 LinearlyModifiedConcept

Define a linearly modified concept. This applies linear modifications to base concepts.

5.8.4 TriangularlyModifiedConcept

Define a triangularly modified concept. This applies triangular function modifications to base concepts.

5.9 Concrete Fuzzy Concepts

5.9.1 FuzzyConcreteConcept

Defines the base class for concrete (crisp, left-shoulder, right-shoulder, and so on) concepts. This is the parent class for all concrete implementations of fuzzy concepts.

5.9.2 CrispConcreteConcept

Define a crisp concept. This represents classical, non-fuzzy (crisp) concepts within the fuzzy framework.

5.9.3 ModifiedConcreteConcept

It is a modified datatype concept. This applies modifications to concrete datatype concepts.

5.9.4 LinearConcreteConcept

It is a concept defined by a linear function. This implements concepts using linear mathematical functions.

5.9.5 LeftConcreteConcept

It is a concept defined by a left-shoulder function. This implements concepts using left-shoulder membership functions.

5.9.6 RightConcreteConcept

It is a concept defined by a right-shoulder function. This implements concepts using right-shoulder membership functions.

5.9.7 TriangularConcreteConcept

It is a concept defined by a triangular function. This implements concepts using triangular membership functions.

5.9.8 TrapezoidalConcreteConcept

It is a concept defined by a trapezoidal function. This implements concepts using trapezoidal membership functions.

5.9.9 TriangularFuzzyNumber

It is a sub-class of the TriangularConcreteConcept and represents a fuzzy triangular number. This provides a specialized implementation for triangular fuzzy numbers.

CHAPTER

SIX

USAGE

6.1 Reasoning

6.1.1 Knowledge base in example.fdl

```
(define-fuzzy-logic lukasiewicz)
(define-modifier very linear-modifier(0.8))
(define-fuzzy-concept eq243 crisp(0, 400, 243, 243))
(define-fuzzy-concept geq300 crisp(0, 400, 300, 400))
(define-fuzzy-concept High right-shoulder(0, 400, 180, 250))
(define-concept SportCar (and Car (some speed (very High))))
(instance ferrari (and Car (some speed geq300)) 1)
(instance audi (and Car (some speed eq243)) 1)

(min-instance? audi SportCar)
```

6.1.2 Python code

```
from fuzzy_dl_owl2 import DLParser

DLParser.main("./example.fdl")  # "Is audi instance of SportCar ? >= 0.92"
```

6.2 Fuzzy OWL 2

6.2.1 From *.fdl to *.owl

```
from fuzzy_dl_owl2 import FuzzydlToOwl2

fdl = FuzzydlToOwl2("./example.fdl", "example.owl")
fdl.run()  # save example.owl in the subdirectory "./results"
```

6.2.2 From *.owl to *.fdl

```
from fuzzy_dl_owl2 import FuzzyOwl2ToFuzzyDL

fdl = FuzzyOwl2ToFuzzyDL("./results/example.owl", "example.fdl")
fdl.translate_owl2ontology()  # save example.fdl in the subdirectory "./results"
```

CHAPTER

SEVEN

API REFERENCE

This page contains auto-generated API reference documentation[1].

7.1 fuzzy_dl_owl2

7.1.1 Submodules

7.1.1.1 fuzzy_dl_owl2.fuzzydl

7.1.1.1.1 Submodules

fuzzy_dl_owl2.fuzzydl.assertion

Submodules

fuzzy_dl_owl2.fuzzydl.assertion.assertion

Classes

Assertion

Module Contents

class Assertion(
    ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
    c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
    d: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
)

__eq__(*value: Self*) → bool

__ne__(*value: Self*) → bool

__repr__() → str

__str__() → str

clone() → Self

equals(*ass: Self*) → bool

get_concept() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_individual() → *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

[1] Created with sphinx-autoapi

get_lower_limit() → *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

get_name_without_degree() → str

get_type() → *fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*

set_individual(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual) → None

set_lower_limit(*deg:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree) → None

concept: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

degree: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

individual: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

fuzzy_dl_owl2.fuzzydl.assertion.atomic_assertion

Classes

AtomicAssertion

Module Contents

class AtomicAssertion(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *degree:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree)

__str__() → str

get_concept_name() → str

get_degree() → *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

c: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

degree: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

fuzzy_dl_owl2.fuzzydl.classification_node

Classes

ClassificationNode

Module Contents

class ClassificationNode(*name: str*)

__hash__() → int

__repr__() → str

__str__() → str

add_input_edge(*node: Self*, *n: float*) → None

add_label(*c: str*) → None

add_ouput_edge(*node: Self*, *n: float*) → None

get_full_name() → str

get_immediate_predecessors() → set[Self]

get_immediate_successors() → set[Self]

get_output_edges() → dict[Self, float]

has_name(*name: str*) → bool

is_nothing() → bool

is_thing() → bool

remove_input_edge(*node: Self*, *n: float*) → None

remove_ouput_edge(*node: Self*, *n: float*) → None

EQUIVALENT_NAMES: set[str]

INPUT_EDGES: dict[Self, float]

OUTPUT_EDGES: dict[Self, float]

fuzzy_dl_owl2.fuzzydl.concept

Submodules

fuzzy_dl_owl2.fuzzydl.concept.all_some_concept

Attributes

All	
Some	

Classes

AllSomeConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class AllSomeConcept(
role: str,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c_type: fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_role_concept_interface.HasRoleConceptInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static all(*role: str*, *concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str

get_atoms() → list[Self]

get_roles() → set[str]

is_complemented_atomic() → bool

static new(
c_type: fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType,
role: str,
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → Self

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static some(*role: str*, *concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

All

Some

fuzzy_dl_owl2.fuzzydl.concept.approximation_concept

Attributes

LooseLowerApprox
LooseUpperApprox
LowerApprox
TightLowerApprox
TightUpperApprox
UpperApprox

Classes

ApproximationConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class ApproximationConcept(
c_type: fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType,
role: str,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_role_concept_interface.HasRoleConceptInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

get_roles() → set[str]

static loose_lower_approx(*role: str*, *c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

static loose_upper_approx(*role: str*, *c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

static lower_approx(*role: str*, *c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static tight_lower_approx(*role: str*, *c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

static tight_upper_approx(*role: str*, *c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

to_all_some_concept() → *fuzzy_dl_owl2.fuzzydl.concept.all_some_concept.AllSomeConcept*

static upper_approx(*role: str*, *c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

INVERSE_APPROXIMATION: dict[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*, *fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

name

LooseLowerApprox

LooseUpperApprox

LowerApprox

TightLowerApprox

TightUpperApprox

UpperApprox

fuzzy_dl_owl2.fuzzydl.concept.atomic_concept

Classes

AtomicConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class AtomicConcept(*name: str*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__eq__(*value: Self*) → bool

__hash__() → int

__invert__() → Self

__ne__(*value: Self*) → bool

__neg__() → Self

__or__(*value: Self*) → Self

__repr__() → str

__rshift__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[Self]

compute_name() → str

get_atomic_concepts() → set[Self]

get_atoms() → list[Self]

get_clauses(*is_type: Callable*) → set[Self]

get_roles() → set[str]

is_atomic() → bool

is_complemented_atomic() → bool

is_concrete() → bool

static new_atomic_concept() → Self

reduce_idempotency(*is_type: Callable*) → Self

replace(*a: Self*, *c: Self*) → Self | None

fuzzy_dl_owl2.fuzzydl.concept.choquet_integral

Classes

ChoquetIntegral	Choquet integral concept.

Module Contents

class ChoquetIntegral(*weights: list[float]*, *concepts:*
*list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_weighted_concepts_interface.HasWeightedConceptsInterface*

Choquet integral concept.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

fuzzy_dl_owl2.fuzzydl.concept.concept

Classes

Concept	Helper class that provides a standard way to create an ABC using
Thing	Helper class that provides a standard way to create an ABC using

Module Contents

class Concept(*c_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType = *ConceptType.ATOMIC*, *name: str = ''*)

Bases: *Thing*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__eq__(*value: Self*) → bool

__iand__(*value: Self*) → Self

__ior__(*value: Self*) → Self

__irshift__(*value: Self*) → Self

__ne__(*value: Self*) → bool

__or__(*value: Self*) → Self

__rshift__(*value: Self*) → Self

__str__() → str

is_atomic() → bool

is_complemented_atomic() → bool

DEFAULT_NAME = 'Concept@'

SPECIAL_STRING = '@'

property name: str

num_new_concepts = 1

property type: *fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*

class Thing

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

abstractmethod __eq__(*value: Self*) → bool

__ge__(*value: Self*) → Self

__gt__(*value: Self*) → Self

__invert__() → Self

__le__(*value: Self*) → Self

__lt__(*value: Self*) → Self

__ne__(*value: Self*) → bool

abstractmethod __neg__() → Self

__repr__() → str

classic_cnf() → Self

classic_dnf() → Self

abstractmethod clone() → Self

abstractmethod compute_atomic_concepts() → set[Self]

abstractmethod compute_name() → str | None

static contains_negated_subconcept(*v: list[Self]*, *cj: Self*) → int

static contains_subconcept(*v: list[Self]*, *cj: Self*) → bool

de_morgan() → Self

distribute(*c_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType) → Self

get_atomic_concepts() → set[Self]

get_atomic_concepts_names() → set[str]

get_atoms() → list[Self]

get_clauses(*is_type: Callable*) → list[Self]

abstractmethod get_roles() → set[str]

goedel_cnf() → Self

goedel_dnf() → Self

has_nominals() → bool

is_concrete() → bool

is_simplified() → bool

This function check if current formula is simplified, i.e., if:

- The only negated elements are literal of kind (~ A), where A is an AtomicProposition
- **The OR operator is between:**
  - Two literals => A | B
  - One literal and a AND => A | (B & C) - (A & B) | C
  - Two (or more) OR => (A & B) | (C & D) | (E & F)
- **The AND operator is between:**
  - Two literals => A & B
  - One literal and a OR => A & (B | C) - (A | B) & C
  - Two (or more) AND => (A | B) & (C | D) & (E | F)
- The only operators are AND, OR and NOT

lukasiewicz_cnf() → Self

lukasiewicz_dnf() → Self

normal_form(*is_type: Callable*) → Self

reduce_double_negation() → Self

reduce_idempotency(*is_type: Callable*) → Self

reduce_quantifiers() → Self

reduce_truth_values() → Self

static remove_element(*v: list[Self]*, *i: int*) → None

abstractmethod replace(*a: Self*, *c: Self*) → Self | None

fuzzy_dl_owl2.fuzzydl.concept.concrete

Submodules

fuzzy_dl_owl2.fuzzydl.concept.concrete.crisp_concrete_concept

Classes

CrispConcreteConcept	Concrete concept defined with a crisp interval.

Module Contents

class CrispConcreteConcept(*name: str*, *k1: float*, *k2: float*, *a: float*, *b: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

Concrete concept defined with a crisp interval.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*x: float*) → float

Get membership degree for a value

property a: float

property b: float

k1: float

k2: float

fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept

Classes

FuzzyConcreteConcept	Fuzzy concrete concept defined with an explicit membership function.

Module Contents

class FuzzyConcreteConcept(*name: str*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, abc.ABC

Fuzzy concrete concept defined with an explicit membership function.

compute_atomic_concepts() → set[Self]

compute_name() → str

abstractmethod get_membership_degree(*value: float*) → float

Get membership degree for a value

get_roles() → set[str]

is_concrete() → bool

replace(
concept1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
concept2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

property k1: float

property k2: float

name: str

fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number

Submodules

fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number

Classes

TriangularFuzzyNumber	Fuzzy number defined with a triangular function.

Module Contents

class TriangularFuzzyNumber(*name: str*, *a: float*, *b: float*, *c: float*)

class TriangularFuzzyNumber(*a: float*, *b: float*, *c: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concrete.triangular_concrete_concept.TriangularConcreteConcept*

Fuzzy number defined with a triangular function.

__add__(*other: Self*) → Self

__and__(*value: Self*) → Self

__eq__(*other: Self*) → bool

__hash__() → int

__mul__(*other: Self*) → Self

__ne__(*other: Self*) → bool

__neg__() → *TriangularFuzzyNumber*

__or__(*value: Self*) → Self

__sub__(*other: Self*) → Self

__truediv__(*other: Self*) → Self

static add(*t1: Self*, *t2: Self*) → Self

Adds two triangular fuzzy numbers.

clone() → Self

compute_name() → str

static divided_by(*t1: Self*, *t2: Self*) → Self

Divides two triangular fuzzy numbers.

get_best_non_fuzzy_performance() → float

Gets the Best Non fuzzy Performance (BNP) of the fuzzy number.

static has_defined_range() → bool

Checks if the range of the fuzzy numbers has been defined.

is_concrete() → bool

is_number() → bool

static minus(*t1: Self*, *t2: Self*) → Self

Subtracts two triangular fuzzy numbers.

static set_range(*min_range: float*, *max_range: float*) → None

static times(*t1: Self*, *t2: Self*) → Self

Multiplies two triangular fuzzy numbers.

K1: float

K2: float

fuzzy_dl_owl2.fuzzydl.concept.concrete.left_concrete_concept

Classes

LeftConcreteConcept	Fuzzy concrete concept defined with a left shoulder function

Module Contents

class LeftConcreteConcept(*name: str*, *k1: float*, *k2: float*, *a: float*, *b: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

Fuzzy concrete concept defined with a left shoulder function

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*value: float*) → float

Get membership degree for a value

property a: float

property b: float

k1: float

k2: float

fuzzy_dl_owl2.fuzzydl.concept.concrete.linear_concrete_concept

Classes

LinearConcreteConcept	Fuzzy concrete concept defined with a left shoulder function

Module Contents

class LinearConcreteConcept(*name: str*, *k1: float*, *k2: float*, *a: float*, *b: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

Fuzzy concrete concept defined with a left shoulder function

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*value: float*) → float

Get membership degree for a value

property a: float

property b: float

k1: float

k2: float

fuzzy_dl_owl2.fuzzydl.concept.concrete.modified_concrete_concept

Classes

ModifiedConcreteConcept	Modified concrete concept.

Module Contents

class ModifiedConcreteConcept(
name: str,
modifier: fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier,
f: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

Modified concrete concept.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*x: float*) → float

Get membership degree for a value

k1: float = 0.0

k2: float = 1.0

property modified: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

property modifier: *fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier*

fuzzy_dl_owl2.fuzzydl.concept.concrete.right_concrete_concept

Classes

RightConcreteConcept	Fuzzy concrete concept defined with a right shoulder function.

Module Contents

class RightConcreteConcept(*name: str*, *k1: float*, *k2: float*, *a: float*, *b: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

Fuzzy concrete concept defined with a right shoulder function.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*x: float*) → float

Get membership degree for a value

property a: float

property b: float

k1: float

k2: float

fuzzy_dl_owl2.fuzzydl.concept.concrete.trapezoidal_concrete_concept

Classes

TrapezoidalConcreteConcept	Fuzzy concrete concept defined with a trapezoidal function.

Module Contents

class TrapezoidalConcreteConcept(*name: str*, *k1: float*, *k2: float*, *a: float*, *b: float*, *c: float*, *d: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

Fuzzy concrete concept defined with a trapezoidal function.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*x: float*) → float

Get membership degree for a value

property a: float

property b: float

property c: float

property d: float

k1: float

k2: float

name: str

fuzzy_dl_owl2.fuzzydl.concept.concrete.triangular_concrete_concept

Classes

TriangularConcreteConcept	Fuzzy concrete concept defined with a triangular function.

Module Contents

class TriangularConcreteConcept(*name: str*, *k1: float*, *k2: float*, *a: float*, *b: float*, *c: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

Fuzzy concrete concept defined with a triangular function.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*x: float*) → float

Get membership degree for a value

property a: float

property b: float

property c: float

k1: float

k2: float

fuzzy_dl_owl2.fuzzydl.concept.ext_threshold_concept

Attributes

ExtendedNegThreshold	
ExtendedPosThreshold	

Classes

ExtThresholdConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class ExtThresholdConcept(
c_type: fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
weight_variable: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_concept_interface.HasConceptInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone()

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

static extended_neg_threshold(*v:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable, *c: Self*) → Self

static extended_pos_threshold(*v:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable, *c: Self*) → Self

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name: str = '([>= Uninferable] Uninferable)'

property weight_variable: *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

ExtendedNegThreshold

ExtendedPosThreshold

fuzzy_dl_owl2.fuzzydl.concept.has_value_concept

Classes

HasValueConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class HasValueConcept(*role: str*, *value: Any*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_value_interface.HasValueInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

get_roles() → set[str]

static has_value(*role: str*, *i: Any*) → Self

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name: str

fuzzy_dl_owl2.fuzzydl.concept.implies_concept

Attributes

GoedelImplies	
KleeneDienesImplies	
LukasiewiczImplies	
ZadehImplies	

Classes

ImpliesConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class ImpliesConcept(
c_type: fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType,
*concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_concepts_interface.HasConceptsInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__eq__(*value: Self*) → bool

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

get_roles() → set[str]

static goedel_implies(
c1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static kleene_dienes_implies(
c1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static lukasiewicz_implies(
c1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static zadeh_implies(
c1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name: str

GoedelImplies

KleeneDienesImplies

LukasiewiczImplies

ZadehImplies

fuzzy_dl_owl2.fuzzydl.concept.interface

Submodules

fuzzy_dl_owl2.fuzzydl.concept.interface.has_concept_interface

Classes

HasConceptInterface	Helper class that provides a standard way to create an ABC using

Module Contents

class HasConceptInterface(*concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

property curr_concept: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

fuzzy_dl_owl2.fuzzydl.concept.interface.has_concepts_interface

Classes

HasConceptsInterface	Helper class that provides a standard way to create an ABC using

Module Contents

class HasConceptsInterface(*concepts: Iterable[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

property concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

fuzzy_dl_owl2.fuzzydl.concept.interface.has_role_concept_interface

Classes

HasRoleConceptInterface	Helper class that provides a standard way to create an ABC using

Module Contents

class HasRoleConceptInterface(*role: str*, *concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.interface.has_role_interface.HasRoleInterface*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_concept_interface.HasConceptInterface*, abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

fuzzy_dl_owl2.fuzzydl.concept.interface.has_role_interface

Classes

HasRoleInterface	Helper class that provides a standard way to create an ABC using

Module Contents

class HasRoleInterface(*role: str*)

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

property role: str

fuzzy_dl_owl2.fuzzydl.concept.interface.has_value_interface

Classes

HasValueInterface	Helper class that provides a standard way to create an ABC using

Module Contents

class HasValueInterface(*role: str*, *value: Any*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.interface.has_role_interface.HasRoleInterface*, abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

property value: Any

fuzzy_dl_owl2.fuzzydl.concept.interface.has_weighted_concepts_interface

Classes

HasWeightedConceptsInterface	Helper class that provides a standard way to create an ABC using

Module Contents

class HasWeightedConceptsInterface(
weights: Iterable[float] | None,
*concepts: Iterable[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.interface.has_concepts_interface.HasConceptsInterface*, abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

property weights: list[float] | None

fuzzy_dl_owl2.fuzzydl.concept.modified

Submodules

fuzzy_dl_owl2.fuzzydl.concept.modified.linearly_modified_concept

Classes

LinearlyModifiedConcept	Fuzzy concept modified with a linear modifier

Module Contents

class LinearlyModifiedConcept(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *mod:* fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.modified.modified_concept.ModifiedConcept*

Fuzzy concept modified with a linear modifier

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

replace(*a:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → Self

fuzzy_dl_owl2.fuzzydl.concept.modified.modified_concept

Classes

ModifiedConcept	Modified fuzzy concept.

Module Contents

class ModifiedConcept(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *mod:* fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_concept_interface.HasConceptInterface*, abc.ABC

Modified fuzzy concept.

__and__() → Self

__neg__() → Self

__or__() → Self

__repr__() → str

__str__() → str

compute_atomic_concepts() → set[Self]

compute_name() → str | None

get_roles() → set[str]

is_concrete() → bool

replace(
concept1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
concept2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

property modifier: *fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier*

fuzzy_dl_owl2.fuzzydl.concept.modified.triangularly_modified_concept

Classes

TriangularlyModifiedConcept	Fuzzy concept modified with a triangular modifier.

Module Contents

class TriangularlyModifiedConcept(
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
mod: fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.modified.modified_concept.ModifiedConcept*

Fuzzy concept modified with a triangular modifier.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

fuzzy_dl_owl2.fuzzydl.concept.negated_nominal

Classes

NegatedNominal	Negated nominal concept. Only used in range restrictions for the moment.

Module Contents

class NegatedNominal(*ind_name: str*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Negated nominal concept. Only used in range restrictions for the moment.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → Self

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str | None

property ind_name: str

name: str = '(not { Uninferable })'

fuzzy_dl_owl2.fuzzydl.concept.operator_concept

Attributes

And
GoedelAnd
GoedelOr
LukasiewiczAnd
LukasiewiczOr
Not
Or

Classes

OperatorConcept	Defines a logic operator concept defined as AND, OR or NOT of concepts.

Module Contents

class OperatorConcept(
c_type: fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType,
*concepts: Iterable[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_concepts_interface.HasConceptsInterface*

Defines a logic operator concept defined as AND, OR or NOT of concepts.

__and__(*value: Self*) → Self

__eq__(*value: Self*) → bool

__hash__() → int

__ne__(*value: Self*) → bool

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

static and_(**concepts:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

clone() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

de_morgan() → Self

distribute(*c_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType) → Self

get_atom() → Self | None

get_atoms() → list[Self]

get_clauses(*is_type: Callable*) → list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

get_roles() → set[str]

static goedel_and(**concepts:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static goedel_or(**concepts:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static is_and(*c_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType) → bool

is_atomic() → bool

is_complemented_atomic() → bool

is_concrete() → bool

static is_not_at_least_value(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_at_most_value(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_choquet(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_concrete(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_exact_value(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_ext_neg_threshold(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_ext_pos_threshold(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_fuzzy_number(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_goedel_implies(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_has_value(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_modified(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_neg_threshold(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_owa(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_pos_threshold(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_qowa(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_quasi_sugeno(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_self(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_sigma_concept(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_sugeno(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_type(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *c_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType) → bool

static is_not_weighted(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_weighted_max(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_weighted_min(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_weighted_sum(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_weighted_sum_zero(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_not_zadeh_implies(*op:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

static is_or(*c_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType) → bool

is_simplified() → bool

This function check if current formula is simplified, i.e., if:

- The only negated elements are literal of kind (~ A), where A is an AtomicProposition
- **The OR operator is between:**
  - Two literals => A | B
  - One literal and a AND => A | (B & C) - (A & B) | C
  - Two (or more) OR => (A & B) | (C & D) | (E & F)

- **The AND operator is between:**
  - Two literals => A & B
  - One literal and a OR => A & (B | C) - (A | B) & C
  - Two (or more) AND => (A | B) & (C | D) & (E | F)
- The only operators are AND, OR and NOT

static lukasiewicz_and(**concepts:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static lukasiewicz_or(**concepts:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

normal_form(*is_type: Callable*) → Self

static not_(*concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static or_(**concepts:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

reduce_double_negation() → Self

reduce_idempotency(*is_type: Callable*) → Self

reduce_quantifiers() → Self

reduce_truth_values() → Self

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

ABSORPTION_OPERATORS: list[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

ALL_OPERATORS: list[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

AND_OPERATORS: list[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

BINARY_OPERATORS: list[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

COMPLEMENT_LAW_OPERATORS: list[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

DISTRIBUTIVE_OPERATORS: list[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

OPERATORS: dict[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*, *fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

OR_OPERATORS: list[*fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*]

property concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

name = '(and)'

type: *fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType*

And

GoedelAnd

GoedelOr

LukasiewiczAnd

LukasiewiczOr

Not

Or

fuzzy_dl_owl2.fuzzydl.concept.owa_concept

Classes

OwaConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class OwaConcept(*weights: list[float]*, *concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_weighted_concepts_interface.HasWeightedConceptsInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept* | None

fuzzy_dl_owl2.fuzzydl.concept.qowa_concept

Classes

QowaConcept	Quantified-guided OWA concept.

Module Contents

class QowaConcept(
quantifier: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept,
*concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.owa_concept.OwaConcept*

Quantified-guided OWA concept.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

compute_weights(*n: int*) → None

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept* | None

name = '(q-owa Uninferable)'

property quantifier: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

type

fuzzy_dl_owl2.fuzzydl.concept.quasi_sugeno_integral

Classes

QsugenoIntegral	Quasi Sugeno integral concept.

Module Contents

class QsugenoIntegral(*weights: list[float]*, *concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.sugeno_integral.SugenoIntegral*

Quasi Sugeno integral concept.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

type

fuzzy_dl_owl2.fuzzydl.concept.self_concept

Classes

SelfConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class SelfConcept(*role: str*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_role_interface.HasRoleInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone()

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

static self(*role: str*) → Self

name = '(self Uninferable)'

fuzzy_dl_owl2.fuzzydl.concept.sigma_concept

Classes

SigmaConcept	Sigma-count concept.

Module Contents

class SigmaConcept(
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
role: str,
*individuals: list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]*,
concrete_concept: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Sigma-count concept.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

get_concept() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_fuzzy_concept() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

get_individuals() → list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]

get_role() → str

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

concept: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

concrete_concept: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*

individuals: list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]

name: str = '(sigma-count Uninferable Uninferable {} Uninferable)'

role: str

fuzzy_dl_owl2.fuzzydl.concept.sigma_count

Classes

SigmaCount	Sigma-count pending tasks.

Module Contents

class SigmaCount(
var: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
*inds: list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]*,
role: str,
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
)

Sigma-count pending tasks.

__hash__() → int

__repr__() → str

__str__() → str

clone() → Self

get_concept() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_individual() → *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

get_individuals() → list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]

get_role() → str

get_variable() → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

concept: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

individual: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

individuals: list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]

role: str

variable: *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

fuzzy_dl_owl2.fuzzydl.concept.string_concept

Classes

StringConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class StringConcept(*name: str*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Helper class that provides a standard way to create an ABC using inheritance.

__hash__() → int

__neg__() → Self

clone() → Self

compute_atomic_concepts() → set[Self]

compute_name() → str | None

get_roles() → set[str]

replace(*a: Self*, *c: Self*) → Self | None

fuzzy_dl_owl2.fuzzydl.concept.sugeno_integral

Classes

SugenoIntegral	Sugeno integral concept.

Module Contents

class SugenoIntegral

class SugenoIntegral(*weights: list[float] | None*, *concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_weighted_concepts_interface.HasWeightedConceptsInterface*

Sugeno integral concept.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

fuzzy_dl_owl2.fuzzydl.concept.threshold_concept

Attributes

NegThreshold	
PosThreshold	

Classes

ThresholdConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class ThresholdConcept(
c_type: fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
weight: float,
)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_concept_interface.HasConceptInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

get_roles() → set[str]

static neg_threshold(*w: float*, *c: Self*) → Self

static pos_threshold(*w: float*, *c: Self*) → Self

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name = '([>= Uninferable] Uninferable)'

property weight: float

NegThreshold

PosThreshold

fuzzy_dl_owl2.fuzzydl.concept.truth_concept

Attributes

BOTTOM	
TOP	

Classes

TruthConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class TruthConcept(*c_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__eq__(*value: Self*) → bool

__hash__() → int

__ne__(*value: Self*) → bool

__neg__() → Self

__or__(*value: Self*) → Self

__rshift__(*value:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

get_atomic_concepts() → set[Self]

get_atoms() → list[Self]

static get_bottom()

get_roles() → set[str]

static get_top()

is_atomic() → bool

is_complemented_atomic() → bool

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name = '*top*'

BOTTOM: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

TOP: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

fuzzy_dl_owl2.fuzzydl.concept.value_concept

Classes

ValueConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class ValueConcept(*c_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConceptType, *role: str*, *value: Any*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_value_interface.HasValueInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

static at_least_value(*role: str*, *o: Any*) → Self

static at_most_value(*role: str*, *o: Any*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

static exact_value(*role: str*, *o: Any*) → Self

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name

fuzzy_dl_owl2.fuzzydl.concept.weighted_concept

Classes

WeightedConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedConcept(*weight: float*, *c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_concept_interface.HasConceptInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str | None

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name = '(Uninferable Uninferable)'

property weight: float

fuzzy_dl_owl2.fuzzydl.concept.weighted_max_concept

Classes

WeightedMaxConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedMaxConcept(*weights: list[float]*, *concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_weighted_concepts_interface.HasWeightedConceptsInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name = '(w-max)'

fuzzy_dl_owl2.fuzzydl.concept.weighted_min_concept

Classes

WeightedMinConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedMinConcept(*weights: list[float]*, *concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_weighted_concepts_interface.HasWeightedConceptsInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name = '(w-min)'

fuzzy_dl_owl2.fuzzydl.concept.weighted_sum_concept

Classes

WeightedSumConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedSumConcept(*weights: list[float]*, *concepts:*
*list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_weighted_concepts_interface.HasWeightedConceptsInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name = '(w-sum)'

fuzzy_dl_owl2.fuzzydl.concept.weighted_sum_zero_concept

Classes

WeightedSumZeroConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedSumZeroConcept(*weights: list[float]*, *concepts:* *list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*)

Bases: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*, *fuzzy_dl_owl2.fuzzydl.concept.interface.has_weighted_concepts_interface.HasWeightedConceptsInterface*

Helper class that provides a standard way to create an ABC using inheritance.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_atomic_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

compute_name() → str

get_roles() → set[str]

replace(
a: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

name = '(w-sum-zero)'

fuzzy_dl_owl2.fuzzydl.concept_equivalence

Classes

ConceptEquivalence	Concept equivalence axiom

Module Contents

class ConceptEquivalence(*c1:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *c2:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

Concept equivalence axiom

clone() → Self

get_c1() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_c2() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

c1: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

c2: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

fuzzy_dl_owl2.fuzzydl.concrete_feature

Classes

ConcreteFeature	

Module Contents

class ConcreteFeature(*name: str*)
class ConcreteFeature(*name: str*, *is_boolean: bool*)
class ConcreteFeature(*name: str*, *k1: int*, *k2: int*)
class ConcreteFeature(*name: str*, *k1: float*, *k2: float*)

__repr__() → str

__str__() → str

clone() → Self

get_k1() → float | int | None

get_k2() → float | int | None

get_name() → str

get_type() → *fuzzy_dl_owl2.fuzzydl.util.constants.ConcreteFeatureType*

set_range(*k1: float | int | None*, *k2: float | int | None*) → None

set_type(*new_type:* fuzzy_dl_owl2.fuzzydl.util.constants.ConcreteFeatureType) → None

fuzzy_dl_owl2.fuzzydl.degree

Submodules

fuzzy_dl_owl2.fuzzydl.degree.degree

Classes

Degree	Helper class that provides a standard way to create an ABC using

Module Contents

class Degree

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

abstractmethod __eq__(*degree: Self*) → bool

__ne__(*value: Self*) → bool

__repr__() → str

abstractmethod __str__() → str

abstractmethod add_to_expression(
expression: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

abstractmethod clone() → Self

abstractmethod create_inequality_with_degree_rhs(
expression: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
inequation_type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → *fuzzy_dl_owl2.fuzzydl.milp.inequation.Inequation*

static get_degree(*value*) → Self

Abstractmethod

abstractmethod is_number_not_one() → bool

abstractmethod is_number_zero() → bool

abstractmethod is_numeric() → bool

abstractmethod multiply_constant(*double: float*) →
fuzzy_dl_owl2.fuzzydl.milp.expression.Expression

abstractmethod subtract_from_expression(
expression: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

fuzzy_dl_owl2.fuzzydl.degree.degree_expression

Classes

DegreeExpression	Helper class that provides a standard way to create an ABC using

Module Contents

class DegreeExpression(*expr:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression)

Bases: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

Helper class that provides a standard way to create an ABC using inheritance.

__eq__(*d:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree) → bool

__str__() → str

add_to_expression(*expr:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) →
fuzzy_dl_owl2.fuzzydl.milp.expression.Expression

clone() → Self

create_inequality_with_degree_rhs(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
inequality_type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → *fuzzy_dl_owl2.fuzzydl.milp.inequation.Inequation*

static get_degree(*value:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) → Self

get_expression() → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

is_number_not_one() → bool

is_number_zero() → bool

is_numeric() → bool

multiply_constant(*constant: float*) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

subtract_from_expression(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

expr: *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

fuzzy_dl_owl2.fuzzydl.degree.degree_numeric

Classes

DegreeNumeric	Helper class that provides a standard way to create an ABC using

Module Contents

class DegreeNumeric(*numeric: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

Helper class that provides a standard way to create an ABC using inheritance.

__eq__(*d:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree) → bool

__str__() → str

add_to_expression(*expr:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) →
fuzzy_dl_owl2.fuzzydl.milp.expression.Expression

clone() → Self

create_inequality_with_degree_rhs(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
inequation_type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → *fuzzy_dl_owl2.fuzzydl.milp.inequation.Inequation*

static get_degree(*value: float*) → Self

get_numerical_value() → float

static get_one() → Self

is_number_not_one() → bool

is_number_zero() → bool

is_numeric() → bool

multiply_constant(*constant: float*) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

subtract_from_expression(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

value: float

fuzzy_dl_owl2.fuzzydl.degree.degree_variable

Classes

DegreeVariable	Helper class that provides a standard way to create an ABC using

Module Contents

class DegreeVariable(*variable:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable)

Bases: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

Helper class that provides a standard way to create an ABC using inheritance.

__eq__(*degree:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree) → bool

__str__() → str

add_to_expression(*expr:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

clone() → Self

create_inequality_with_degree_rhs(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
inequality_type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → *fuzzy_dl_owl2.fuzzydl.milp.inequation.Inequation*

static get_degree(*value:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → Self

get_variable() → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

is_number_not_one() → bool

is_number_zero() → bool

is_numeric() → bool

multiply_constant(*constant: float*) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

subtract_from_expression(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

variable: *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

fuzzy_dl_owl2.fuzzydl.domain_axiom

Classes

DomainAxiom	Role domain axiom

Module Contents

class DomainAxiom(*role: str*, *concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

Role domain axiom

concept: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

role: **str**

fuzzy_dl_owl2.fuzzydl.exception

Submodules

fuzzy_dl_owl2.fuzzydl.exception.fuzzy_ontology_exception

Exceptions

FuzzyOntologyException	Common base class for all non-exit exceptions.

Module Contents

exception FuzzyOntologyException(*message: str*)

Bases: Exception

Common base class for all non-exit exceptions.

fuzzy_dl_owl2.fuzzydl.exception.inconsistent_ontology_exception

Exceptions

InconsistentOntologyException	Common base class for all non-exit exceptions.

Module Contents

exception InconsistentOntologyException(*message: str*)

Bases: Exception

Common base class for all non-exit exceptions.

fuzzy_dl_owl2.fuzzydl.feature_function

Classes

FeatureFunction	Function involving several features.

Module Contents

class FeatureFunction(*feature: Self*)
class FeatureFunction(*feature: str*)
class FeatureFunction(*n: float*)
class FeatureFunction(*feature: list[Self]*)
class FeatureFunction(*feature1: Self*, *feature2: Self*)
class FeatureFunction(*n: float*, *feature: Self*)

Function involving several features.

__repr__() → str

__str__() → str

get_features() → set[str]

Gets an array of features that take part in the function.

get_number() → float

get_type() → *fuzzy_dl_owl2.fuzzydl.util.constants.FeatureFunctionType*

to_expression(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression* | None

Gets an array of features that take part in the function.

fuzzy_dl_owl2.fuzzydl.fuzzydl_to_owl2

Classes

FuzzydlToOwl2	Convert FuzzyDL to OWL2

Functions

main()

Module Contents

class FuzzydlToOwl2(*input_file: str*, *output_file: str*, *base_iri: str* = *'http://www.semanticweb.org/ontologies/fuzzydl_ontology#'*)

Convert FuzzyDL to OWL2

add_entity_annotation(*annotation: str*, *entity: pyowl2.abstracts.entity.OWLEntity*) → None

Add annotation to an entity

add_ontology_annotation(*annotation: str*) → None

Add annotation to the ontology

annotate_gci(*gci:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion) → None

Annotate a General Concept Inclusion (GCI)

annotate_pcd(
c1: pyowl2.abstracts.class_expression.OWLClassExpression,
pcd: fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition,
) → None

Annotate a Primitive Concept Definition (PCD)

annotation_property_iri(*o: object*) → pyowl2.base.iri.IRI

Convert datatype to IRI string

class_iri(*o: object*) → pyowl2.base.iri.IRI

Convert class to IRI string

data_property_iri(*o: object*) → pyowl2.base.iri.IRI

Convert data property to IRI string

datatype_iri(*o: object*) → pyowl2.base.iri.IRI

Convert datatype to IRI string

exist_data_property(*role: str*) → bool

Check if a data property exists

exist_object_property(*role: str*) → bool

Check if an object property exists

get_annotations_for_axiom(
value: float | fuzzy_dl_owl2.fuzzydl.degree.degree_numeric.DegreeNumeric,
) → set[pyowl2.base.annotation.OWLAnnotation]

Get annotations for an axiom with degree

get_base(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) →
pyowl2.abstracts.class_expression.OWLClassExpression

Get the base class for a concept

get_class(*name: str*) → pyowl2.abstracts.class_expression.OWLClassExpression

get_class(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) →
pyowl2.abstracts.class_expression.OWLClassExpression

Get or create an OWL class

get_data_property(
role: str,
) → pyowl2.expressions.data_property.OWLDataProperty | pyowl2.expressions.object_property.OWLObjectProperty

Get or create a data property

get_individual(*name: str*) → pyowl2.individual.named_individual.OWLNamedIndividual

Get or create a named individual

get_new_atomic_class(*name: str*) → pyowl2.abstracts.class_expression.OWLClassExpression

Get or create a new atomic class

get_object_property(
role: str,
) → pyowl2.expressions.data_property.OWLDataProperty | pyowl2.expressions.object_property.OWLObjectProperty

Get or create an object property

individual_iri(*o: object*) → pyowl2.base.iri.IRI

Convert individual object to IRI string

iri(*o: object*, *iri_type: type = OWLClass*) → pyowl2.base.iri.IRI

Convert object to IRI string

object_property_iri(*o: object*) → pyowl2.base.iri.IRI

Convert object property to IRI string

run() → None

Execute the conversion process

to_owl_annotation(*annotation: str*) → pyowl2.base.annotation.OWLAnnotation

Convert a string to an OWL annotation

concepts: dict[str, pyowl2.abstracts.class_expression.OWLClassExpression]

datatypes: dict[str, pyowl2.base.datatype.OWLDatatype]

fuzzyLabel: pyowl2.base.annotation_property.OWLAnnotationProperty

input_FDL: str

modifiers: dict[str, pyowl2.base.datatype.OWLDatatype]

num_classes: int = 0

ontology: pyowl2.ontology.OWLOntology

ontology_iri: pyowl2.base.iri.IRI

ontology_path: str = 'Uninferable#'

output_FOWL: str

main()

fuzzy_dl_owl2.fuzzydl.general_concept_inclusion

Classes

GeneralConceptInclusion	General concept inclusion axiom.

Module Contents

class GeneralConceptInclusion(
subsumer: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
subsumed: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
type_: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
)

General concept inclusion axiom.

__eq__(*other: Self*) → bool

__ge__(*other: Self*) → bool

__gt__(*other: Self*) → bool

__hash__() → int

__le__(*other: Self*) → bool

__lt__(*other: Self*) → bool

__ne__(*other: Self*) → bool

__repr__() → str

__str__() → str

clone() → Self

get_degree() → *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

get_subsumed() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_subsumer() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_type() → *fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType*

set_degree(*deg:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree) → None

set_subsumed(*new_concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

set_subsumer(*new_concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

degree: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

subsumed: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

subsumer: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

type: *fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType*

fuzzy_dl_owl2.fuzzydl.individual

Submodules

fuzzy_dl_owl2.fuzzydl.individual.created_individual

Classes

CreatedIndividual

Module Contents

class CreatedIndividual(
name: str,
parent: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual | *None* = *None*,
role_name: str | *None* = *None*,
)

class CreatedIndividual(*name: str*)

Bases: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

__eq__(*value: Self*) → bool

__ge__(*value: Self*) → bool

__gt__(*value: Self*) → bool

__hash__() → int

__le__(*value: Self*) → bool

__lt__(*value: Self*) → bool

__ne__(*value: Self*) → bool

__str__() → str

clone() → Self

clone_special_attributes(*ind: Self*) → None

get_depth() → int

get_integer_id() → int

get_parent() → *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual* | None

get_parent_name() → str

get_representative_if_exists(
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
f_name: str,
f: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber,
) → Self | None

Return b individual p with b representative of b set of individuals if it exists. Given b fuzzy number F, b representative individual is the set of individuals that are greater or equal (or less or equal) than F. The representative individual is related to p via b concrete feature f.

Parameters

- **type** (InequalityType) – Type of the representative individual (GREATER_EQUAL, LESS_EQUAL)
- **f_name** (*str*) – Name of the feature for which the individual is b filler
- **f** (TriangularFuzzyNumber) – Fuzzy number

Returns
A new individual with b representative individual

Return type
Optional[*Self*]

get_role_name() → str

individual_set_intersection_of(
set1: sortedcontainers.SortedSet[Self],
set2: sortedcontainers.SortedSet[Self],
) → sortedcontainers.SortedSet[Self]

Gets the intersection of two concept labels.

is_blockable() → bool

is_concrete() → bool

mark_indirectly_blocked() → None

Marks the subtree of a node as indirectly blocked

set_concrete_individual() → None

Sets that the individual is concrete.

fuzzy_dl_owl2.fuzzydl.individual.individual

Classes

Individual

Module Contents

class Individual(*name: str*)

__eq__(*value: Self*) → bool

__ne__(*value: Self*) → bool

__repr__() → str

__str__() → str

add_concept(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

add_concrete_restriction(*f_name: str*, *ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

Adds a negated datatype restriction to the individual.

add_to_nominal_list(*ind_name: str*) → None

clone() → Self

clone_attributes(*ind: Self*) → None

get_concepts() → set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

get_nominal_list() → set[str]

abstractmethod get_representative_if_exists(
type: fuzzy_dl_owl2.fuzzydl.util.constants.RepresentativeIndividualType,
f_name: str,
f: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber,
)

is_blockable() → bool

prune() → None

set_label(*ind_name: str*) → None

set_name(*name: str*) → None

DEFAULT_NAME: str = 'i'

concrete_role_restrictions: dict[str, list[*fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion*]]

fillers_to_show: dict[str, set[str]]

list_of_concepts: set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

name: str

nominal_list: set[str]

not_self_roles: set[str] = []

role_relations: dict[str, list[*fuzzy_dl_owl2.fuzzydl.relation.Relation*]]

role_restrictions: dict[str, list[*fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction*]]

fuzzy_dl_owl2.fuzzydl.individual.representative_individual

Classes

RepresentativeIndividual	New concrete individual being a representative of a set of individuals.

Module Contents

class RepresentativeIndividual(
c_type: fuzzy_dl_owl2.fuzzydl.util.constants.RepresentativeIndividualType,
f_name: str,
f: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber,
ind: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
)

New concrete individual being a representative of a set of individuals. Given an individual p and a fuzzy number F, a representative individual is the set of individuals that are greater or equal (or less or equal) than F. Then, p is related to the representative individual in some way.

get_feature_name() → str

get_fuzzy_number() → *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber*

get_individual() → *fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*

get_type() → *fuzzy_dl_owl2.fuzzydl.util.constants.RepresentativeIndividualType*

f: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber*

f_name: str

ind: *fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*

type: *fuzzy_dl_owl2.fuzzydl.util.constants.RepresentativeIndividualType*

fuzzy_dl_owl2.fuzzydl.knowledge_base

Classes

ClassicalSolver	Solver for classical logic semantics.
CreatedIndividualHandler	
DatatypeReasoner	
IndividualHandler	
KnowledgeBase	
LukasiewiczSolver	
ZadehSolver	Solver for Zadeh fuzzy logic semantics.

Module Contents

class ClassicalSolver

Solver for classical logic semantics.

static solve_all(
rel: fuzzy_dl_owl2.fuzzydl.relation.Relation,
restrict: fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction,
kb: KnowledgeBase,
) → None

Solves a universal restriction fuzzy assertion with respect to a reference fuzzy KB.

static solve_and(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a conjunction fuzzy assertion with respect to a reference fuzzy KB.

static solve_or(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a disjunction fuzzy assertion with respect to a reference fuzzy KB.

static solve_some(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a existential restriction fuzzy assertion with respect to a reference fuzzy KB.

class CreatedIndividualHandler

static get_representative(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
f_name: str,
f: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber,
kb: KnowledgeBase,
) → *fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*

Gets b individual p with b representative of b set of individuals. Given b fuzzy number F, b representative individual is the set of individuals that are greater or equal (or less or equal) than F. The representative individual is related to p via b concrete feature f.

Parameters

- **current_individual** (CreatedIndividual) – The individual we want the representative
- **type** (InequalityType) – Type of the representative individual (GREATER_EQUAL, LESS_EQUAL)
- **f_name** (*str*) – Name of the feature for which the individual is b filler
- **f** (TriangularFuzzyNumber) – Fuzzy number
- **kb** (KnowledgeBase) – KnowledgeBase

Returns
A new individual with b representative individual

Return type
CreatedIndividual

static is_blocked(*current_individual:* fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual, *kb:* KnowledgeBase) → bool

Gets if the individual is blocked with respect to a fuzzy KB.

static is_directly_anywhere_pairwise_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Test if the individual is anywhere pair-wise directly blocked with respect to a fuzzy KB

static is_directly_anywhere_simple_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Gets if the individual is directly anywhere simple blocked with respect to a fuzzy KB.
Case SUBSET or SET blocking. It is assumed that the individual and all ancestors are not blocked.

static is_directly_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Gets if the individual is directly blocked with respect to a fuzzy KB. A node v is directly blocked iff none of its ancestors are blocked and there exists an ancestor w such that L(v) = L(w), where L(*) is the set of Concept's labels for a node. In this case we say that w directly blocks v.

static is_directly_pairwise_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Test if the individual is pair-wise directly blocked with respect to a fuzzy KB.

static is_directly_simple_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Gets if the individual is directly blocked with respect to a fuzzy KB.
Case SUBSET or SET blocking It is assumed that the individual and all ancestors are not blocked

static is_indirectly_anywhere_pairwise_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Gets if the individual is indirectly anywhere pairwise blocked with respect to a fuzzy KB.

static is_indirectly_anywhere_simple_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Gets if the individual is indirectly anywhere blocked with respect to a fuzzy KB. Case SUBSET or SET blocking.

static is_indirectly_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Gets if the individual is indirectly blocked with respect to a fuzzy KB. A node v is indirectly blocked iff one of its ancestors are blocked.

static is_indirectly_pairwise_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Gets if the individual is indirectly blocked with respect to a fuzzy KB.

static is_indirectly_simple_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Gets if the individual is indirectly blocked with respect to a fuzzy KB. Case SUBSET or SET blocking. A node v is indirectly blocked iff one of its ancestors are blocked.

static mark_indirectly_simple_unchecked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → None

Marks the subtree of a node as indirectly unblocked

static match_concept_labels(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → bool

Checks if two individuals match concept labels

static match_set_concept_labels(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
) → bool

Check that two concept labels are equal

static match_subset_concept_labels(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
) → bool

Check that every concept in the labels of this is also in b

static matching_individual(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → set[*fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*]

Checks if there is a matching individual to this one

static unblock(*current_individual:* fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual, *kb:* KnowledgeBase) → None

static unblock_directly_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → None

Unblocks an directly blocked individual.

static unblock_indirectly_blocked(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → None

Unblocks an indirectly blocked individual.

static unblock_pairwise(
current_individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
) → None

Unblock the individual

static update_role_successors(*name: str*, *role_name: str*, *kb:* KnowledgeBase) → None

class DatatypeReasoner

static apply_at_least_value_rule(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

static apply_at_most_value_rule(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

static apply_exact_value_rule(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

static apply_not_at_least_value_rule(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
ass: fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion,
kb: KnowledgeBase,
) → None

static apply_not_at_most_value_rule(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,

ass: fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion,
kb: KnowledgeBase,
) → None

static apply_not_exact_value_rule(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
ass: fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion,
kb: KnowledgeBase,
) → None

static apply_not_rule(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
ass: fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion,
kb: KnowledgeBase,
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → None

static apply_rule(
ass: fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion,
kb: KnowledgeBase,
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → None

static geq_equation(
y: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static get_bounds(*t:* fuzzy_dl_owl2.fuzzydl.concrete_feature.ConcreteFeature) → list[float] | None

static get_created_individual_and_variables(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role: str,
t: fuzzy_dl_owl2.fuzzydl.concrete_feature.ConcreteFeature,
k: list[float],
kb: KnowledgeBase,
) → list[Any]

static get_feature(*f_name: str*, *kb:* KnowledgeBase) →
fuzzy_dl_owl2.fuzzydl.concrete_feature.ConcreteFeature

static get_xb(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
t: fuzzy_dl_owl2.fuzzydl.concrete_feature.ConcreteFeature,
kb: KnowledgeBase,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

static rule_feature_function(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
t: fuzzy_dl_owl2.fuzzydl.concrete_feature.ConcreteFeature,
fun: fuzzy_dl_owl2.fuzzydl.feature_function.FeatureFunction,
kb: KnowledgeBase,
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
k: list[float],
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → None

static rule_not_simple_restriction(
n: Any,
kb: KnowledgeBase,
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
k: list[float],
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → None

static rule_not_triangular_fuzzy_number(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
f_name: str,
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
n: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber,
k: list[float],
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → None

static rule_simple_restriction(
n: Any,
kb: KnowledgeBase,
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
k: list[float],
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → None

static rule_triangular_fuzzy_number(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
kb: KnowledgeBase,
f_name: str,
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
n: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber,
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → None

static write_feature_equation(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
fun: fuzzy_dl_owl2.fuzzydl.feature_function.FeatureFunction,
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
k: list[float],
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
kb: KnowledgeBase,
)

static write_fuzzy_number_equation(
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_b_prime: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,

type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
kb: KnowledgeBase,
)

static write_not_feature_equation(
deg: fuzzy_dl_owl2.fuzzydl.degree.degree_expression.DegreeExpression,
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
k: list[float],
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
kb: KnowledgeBase,
) → None

static write_not_fuzzy_number_equation(
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_b_prime: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_b_prime_is_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_is_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
k: list[float],
type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
kb: KnowledgeBase,
) → None

class IndividualHandler

static add_not_self_restriction(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual, *role: str*, *kb:* KnowledgeBase) → None

static add_relation(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role_name: str,
b: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
kb: KnowledgeBase,
) → *fuzzy_dl_owl2.fuzzydl.relation.Relation* | None

Adds b relation to the individual.

static add_restriction(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role_name: str,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
kb: KnowledgeBase,
) → None

static add_restriction(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role_name: str,
ind_name: str,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
kb: KnowledgeBase,
) → None

static common_part_add_restriction(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role_name: str,
restrict: fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction,
kb: KnowledgeBase,
) → None

static solve_not_self_rule(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual, *role_name: str*, *kb:* KnowledgeBase) → None

Apply not self rule.

static solve_relation_restriction(
rel: fuzzy_dl_owl2.fuzzydl.relation.Relation,
restrict: fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction,
kb: KnowledgeBase,
) → None

Apply b universal restriction to b relation of the individual.

static unblock_simple(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual, *kb:* KnowledgeBase) → None

Unblock the individual.
Case subset/set blocking

class KnowledgeBase

add_assertion(*new_ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

add_assertion(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
n: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → None

add_assertion(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
restrict: fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction,
) → None

Adds a fuzzy assertion.

add_assertions(*list_of_assertions: list[*fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion*]*) → None

Adds a list of fuzzy assertions.

add_atomic_concepts_disjoint(*disjoint_concepts: list[str]*) → None

Adds some disjoint concept axioms.

Parameters
disjoint_concepts (*list[str]*) – A vector of concept names.

add_axiom_to_A_equiv_C(*a: str*, *conc:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

add_axiom_to_A_is_a_C(
a: str,
pcd: fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition,
*pcd_dict: dict[str, set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]]*,
) → None

add_axiom_to_C_is_a_A(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
logic_type: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
) → None

Adds a GCI (conc2, conc1, degree, type) to add_axiom_to_C_is_a_A.

Parameters

- **conc1** (Concept) – Subsumer concept.

- **conc2** (Concept) – Subsumed concept.
- **degree** (Degree) – Lower bound for the degree.
- **logic_type** (LogicOperatorType) – Type of the GCI (semantics according to the implication).

add_axiom_to_C_is_a_D(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
logic_type: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
) → None

Adds a GCI (conc2, conc1, degree, type) to axioms_C_is_a_D.

Parameters

- **conc1** (Concept) – Subsumer concept.
- **conc2** (Concept) – Subsumed concept.
- **degree** (Degree) – Lower bound for the degree.
- **logic_type** (LogicOperatorType) – Type of the GCI (semantics according to the implication).

add_axiom_to_C_is_a_X(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
logic_type: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
atomic: bool,
) → None

Adds a GCI (conc2, conc1, degree, type) to axioms_C_is_a_A or axioms_C_is_a_D.

Parameters

- **conc1** (Concept) – Subsumer concept.
- **conc2** (Concept) – Subsumed concept.
- **degree** (Degree) – Lower bound for the degree.
- **logic_type** (LogicOperatorType) – Type of the GCI (semantics according to the implication).
- **atomic** (*bool*) – true for C isA A; false for C isA D

add_axiom_to_do_A_is_a_X(*a: str*, *pcd:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → None

add_axiom_to_inc(*a: str*, *pcd:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → None

add_axioms_to_tg() → None

add_concept(*concept_name: str*, *conc:* fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept) → None

Adds a fuzzy concept to the array of concepts in the fuzzy KB.

add_concepts_disjoint(*disjoint_concepts: list[str]*) → None

add_concepts_disjoint(*c1: str*, *c2: str*) → None

add_concepts_disjoint(
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
d: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → None

Adds some disjoint concept axioms.

add_created_individual(*ind_name: str*, *ind:* fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual) → None

Adds a created individual to the KB.

add_datatype_restriction(
restriction_type: fuzzy_dl_owl2.fuzzydl.util.constants.RestrictionType,
o: Any,
f_name: str,
) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Adds a datatype restriction of the form (restriction_type, f_name, o).

Parameters

- **restriction_type** (RestrictionType) – Type of the datatype restriction.
- **o** (*Any*) – Value of the datatype restriction.
- **f_name** (*str*) – Concrete feature.

Returns
A datatype restriction.

Return type
Concept

add_disjoint_union_concept(*disjoint_union_concepts: list[str]*) → None

Adds a disjoint union concept axiom.

Parameters
disjoint_union_concepts (*list[str]*) – A vector of concepts names.

add_equivalence_relation(*role: str*) → None

Adds a fuzzy equivalence relation.

add_equivalent_concepts(*equiv_concepts: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]*) → None

Adds some equivalent concept axioms.

Parameters
equiv_concepts (*list[*Concept*]*) – An array list of vector of equivalent fuzzy concepts.

add_equivalent_roles(*equiv_roles: list[str]*) → None

Adds some equivalent funcRole axioms.

Parameters
equiv_roles (*list[str]*) – An array list of equivalent fuzzy funcRole names.

add_fuzzy_number(
f_name: str,
f: fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber,
) → None

Adds a fuzzy number to the fuzzy KB.

add_gci(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,

degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
logic_type: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
) → None

Adds a General Concept Inclusion (conc2, conc1, degree, type).

Parameters

- **conc1** (Concept) – Subsumer concept.
- **conc2** (Concept) – Subsumed concept.
- **degree** (Degree) – Lower bound for the degree.
- **logic_type** (LogicOperatorType) – Type of the GCI (semantics according to the implication).

add_individual(*ind_name: str*, *ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual) → None

Adds a individual to the KB.

add_individual_to_concept(*concept_id: int*, *ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual) → None

Add the individual a to the individual list of the concept.

add_inverse_roles(*role: str*, *inv_role: str*) → None

Adds an inverse funcRole axiom.

add_labels_with_nodes(*node: str*, *ind_name: str*) → None

add_modifier(*mod_name: str*, *mod:* fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier) → None

Adds a fuzzy modifier to the fuzzy KB.

add_mutually_disjoint(*c1: str*, *c2: str*) → None

add_negated_datatype_restriction(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

add_negated_equations(
i: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → None

For some and all concepts, add x_{v:C} = 1 - x_{v:not C}.

add_parent_recursively(*role_c: str*, *all_parents: dict[str, float]*, *current_role: str*, *n1: float*) → None

Used in the computation of the transitive closure of the Role Inclusion Axioms.

add_relation(
ind_A: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role: str,
ind_B: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → *fuzzy_dl_owl2.fuzzydl.relation.Relation*

Adds a fuzzy relation of the form (ind_A, ind_B, role, degree)

Parameters

- **ind_A** (Individual) – A subbject individual.
- **role** (*str*) – An abstract role.
- **ind_B** (Individual) – An object individual.
- **degree** (Degree) – Lower bound for the degree.

Returns
Added relation.

Return type
Relation

add_relation_with_role_parent(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role_c: str,
role_p: str,
n: float,
) → None

add_relation_with_role_parent_in_lukasiewicz(*r:* fuzzy_dl_owl2.fuzzydl.relation.Relation, *role_p: str*, *n: float*) → None

add_similarity_relation(*role: str*) → None

Adds a fuzzy similarity relation.

add_simple_inverse_roles(*role: str*, *inv_role: str*) → None

States that two roles are inverse without recursion.

add_subsumption(
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
logic_type: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
) → None

Adds a General Concept Inclusion (conc2, conc1, degree, type) even if the left side is atomic.

Parameters

- **conc1** (Concept) – Subsumed concept.
- **conc2** (Concept) – Subsumer concept.
- **degree** (Degree) – Lower bound for the degree.
- **logic_type** (LogicOperatorType) – Type of the GCI (semantics according to the implication).

add_tdef_links(*g: networkx.DiGraph*, *A_t_C: dict[str, int]*, *use_tdr: bool*) → bool

We return true if we know that htere are cycles because of t_synonyms. False does not mean that there are no cycles!

add_tdr_links(*g: networkx.DiGraph*, *A_t_C: dict[str, int]*, *used_roles: set[str]*, *v: int*) → bool

We return true if we know that there are cycles because of t_synonyms. False does not mean that there are no cycles!

add_tinc_links(*g: networkx.DiGraph*, *A_t_C: dict[str, int]*, *use_tdr: bool*) → bool

We return true if we know that there are cycles because of t_synonyms. False does not mean that there are no cycles!

add_tmp_feature(*feature: str*) → None

Add a feature from the DL parser.

check_fuzzy_number_concept_exists(*conc_name: str*) → bool

Checks if there exists a fuzzy number with the indicated name.

check_individual_exists(*ind_name: str*) → bool

Checks if there exists an individual with the given name.

check_role(*role_name: str*, *conc:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

Checks the disjointness between abstract and concrete roles.

Parameters

- **role_name** (*str*) – A role name.

- **conc** (Concept) – A concept appearing in a restrictions involving the role.

check_trans_role_applied(
rel: fuzzy_dl_owl2.fuzzydl.relation.Relation,
restrict: fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction,
) → bool

Checks if transitivity has been applied to a universal restriction.

Parameters

- **rel** (Relation) – A relation.
- **restrict** (Restriction) – A restriction.

Returns
true if the transitivity rule has been applied; false otherwise.

Return type
bool

classify() → None

clone() → Self

Gets a copy of a knowledge base.

clone_without_abox() → Self

Gets a copy of a knowledge base except the ABox.

compute_blocking_type() → None

Computes the type of the blocking in {NO_BLOCKING, SUBSET_BLOCKING, SET_BLOCKING, (ANYWHERE) DOUBLE_BLOCKING}. If the type is in {SUBSET_BLOCKING, SET_BLOCKING, (ANYWHERE) DOUBLE_BLOCKING}, it also computes whether it is dynamic or not.

compute_language() → None

Computes the language of the fuzzy KB, from ALC to SHIF(D).

compute_variables_old_calculus(*fcc:* fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept) → None

concept_absorption(*pcd:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition, *atomic: bool*) → bool

concept_absorption(*tau:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion, *atomic: bool*) → bool

concept_exists(*name: str*) → bool

Checks if there exists a concept with the given name.

convert_strings_into_integers() → None

Transforms string datatype restrictions into integer datatype restrictions.

create_roles_with_all_parents() → None

Computes transitive closure of the Role Inclusion Axioms.

create_roles_with_trans_children() → None

Used in the computation of the transitive closure of the Role Inclusion Axioms.

define_atomic_concept(
concept_name: str,
conc: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
implication: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
n: float,
) → None

Adds an atomic fuzzy concept definition.

define_boolean_concrete_feature(*fun_role: str*) → None

Define a concrete feature with range boolean.

Parameters
fun_role (*str*) – Name of the concrete feature.

define_concept(*concept_name: str*, *conc:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

Adds a fuzzy concept definition.

define_concreate_feature(*role: str*) → None

define_equivalent_concepts(
c1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → None

Adds a concept equivalence axiom.

define_integer_concrete_feature(*fun_role: str*, *d1: int*, *d2: int*) → None

Define a concrete feature with range integers in [d1, d2].

Parameters

- **fun_role** (*str*) – Name of the concrete feature.
- **d1** (*int*) – Lower bound of the range.
- **d2** (*int*) – Upper bound of the range.

define_real_concrete_feature(*fun_role: str*, *d1: float*, *d2: float*) → None

Define a concrete feature with range real numbers in [d1, d2].

Parameters

- **fun_role** (*str*) – Name of the concrete feature.
- **d1** (*int*) – Lower bound of the range.
- **d2** (*int*) – Upper bound of the range.

define_string_concrete_feature(*fun_role: str*) → None

Define a concrete feature with range string.

Parameters
fun_role (*str*) – Name of the concrete feature.

define_synonym(*concept_name_1: str*, *concept_name_2: str*) → None

Adds a fuzzy synonym definition.

define_synonyms(*concept_name_1: str*, *concept_name_2: str*) → None

Adds a fuzzy synonym definition.

definition_absorption(*gci:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion) → bool

Parameters
gci (GeneralConceptInclusion) – A GCI.

Returns
true if there are changes; false otherwise.

Return type
bool

definition_absorption_to_do(*pcd:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → bool

Parameters
pcd (PrimitiveConceptDefinition) – A primitive concept definition.

Returns
true if there are changes; false otherwise.

Return type
bool

degree_if_not_one(*deg:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree) → str

degree_if_not_one(*d: float*) → str

Return a string representation of the degree if it is different to 1.0.

disjoint_with_defined_concept(*a: str*) → bool

Computes if there is some disjoint(a, b) in tDis with b being a head of an axiom in Tdef

exists_primite_concept_definition(
*pcds: set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]*,
pcd: fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition,
) → bool

exit_condition() → None

Add every GCI to tG using the form *top* isA (C -> D).

exit_condition_A_is_a_X(*pcd:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → None

exit_condition_C_is_a_X(*gci:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion) → None

form_inv_role_inc_axioms() → None

Computes relations for the inverse roles and Role Inclusion Axioms (R => P, n) implies (inv(R) => inv(P), n)

form_inv_role_relations() → None

Computes relations for the inverse roles

form_inv_trans_roles() → None

Computes relations for the inverse roles and transitive roles.

gci_transform_define_atomic_concept(
concept_name: str,
conc: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
implication: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
n: float,
)

gci_transformation(*tau:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion, *atomic: bool*) → bool

gci_transformation(*pcd:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → bool

gci_transformation_add_axiom_to_C_is_a_X(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
logic_type: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
) → None

gci_transformations_A_is_a_C() → None

gci_transformations_C_is_a_A() → None

gci_transformations_C_is_a_D() → None

get_A_t_C() → dict[str, int]

get_classification_node() → *fuzzy_dl_owl2.fuzzydl.classification_node.ClassificationNode* | None

get_concept(*name: str*) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Gets a concept with indicated name.

get_concept_from_number(*n: int*) → str | None

Gets the concept name encoded by a number.

get_correct_version_of_individual(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

get_correct_version_of_individual(*rel:* fuzzy_dl_owl2.fuzzydl.relation.Relation) → None

Use right version of the individual (needed when we clone the KB or merge individuals)

get_inclusion_degree(*subsumed: str*, *subsumer: str*) → float

Computes the inclusion degree between two roles.

Parameters

- **subsumed** (*str*) – Subsumed funcRole.
- **subsumer** (*str*) – Subsumer funcRole.

Returns

Inclusion degree of subsumed in subsumer.

Return type

float

get_individual(
ind_name: str,
) → *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual* | *fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*

Gets an individual with the indicated name (creating it if necessary).

get_individuals() → dict[str, *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]

Gets all individuals of the KB.

get_inverses_of_inverse_role(*role: str*) → set[str] | None

Gets the set of inverse roles of some inverse of a given role.

get_language() → str

Gets the language of the fuzzy KB, from ALC to SHIF(D).

get_logic() → *fuzzy_dl_owl2.fuzzydl.util.constants.FuzzyLogic*

Gets the fuzzy logic of the fuzzy knowledge base.

get_named_individuals() → list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]

get_new_atomic_concept() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_new_concrete_individual(
parent: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
f_name: str,
) → *fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*

get_new_individual() → *fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*

get_new_individual(
parent: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
f_name: str,
) → *fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*

get_new_individual_common_code(
parent: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
f_name: str,
) → *fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*

get_number_from_concept(*concept_name: str*) → int

Gets a number to encode a concept name.

get_number_of_domain_restrictions() → int

get_number_of_range_restrictions() → int

get_subsumption_flags(*b:* fuzzy_dl_owl2.fuzzydl.classification_node.ClassificationNode) → float

Retrieves the value subFlags(a, b)

get_tmp_feature(*feature: str*) → str

Gets a feature from the DL parser.

get_truth_constants(*s: str*) → float | None

Gets a truth constant from the DL parser.

goedel_implies(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → None

Adds a Goedel General Concept Inclusion.

Parameters

- **conc1** (Concept) – Subsumed concept.
- **conc2** (Concept) – Subsumer concept.
- **degree** (Degree) – Lower bound for the degree.

has_functional_abstract_roles() → bool

has_nominals_in_abox() → bool

Checks if the ABox contains the b-some constructor.

Returns
true if the ABox contains the b-some constructor; false otherwise.

Return type
bool

has_nominals_in_tbox() → bool

Checks if the TBox contains the b-some constructor.

Returns
true if the TBox contains the b-some constructor; false otherwise.

Return type
bool

has_only_crisp_sub_concepts(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

Checks if a concept c is only composed of crisp concepts or not.

implies(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → None

Adds a General Concept Inclusion (conc1, conc2, degree).

Parameters

- **conc1** (Concept) – Subsumed concept.
- **conc2** (Concept) – Subsumer concept.
- **degree** (Degree) – Lower bound for the degree.

is_assertion_processed(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

Checks if an assertion has already been processed.

is_atomic_crisp_concept(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

Checks if a concept is atomic and crisp.

is_classified() → bool

Checks if the knowledge base has already been classified.

is_concrete_type(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → bool

Computes if the type is one of the concretes (concrete, fuzzy number, or their complements)

is_crisp_concept(*concept_name: str*) → bool

Checks if a concept is crisp.

Parameters
concept_name (*str*) – Name of the concept.

Returns
true if the semantics is classical logic or if the concept is crisp, false otherwise.

Return type
bool

is_crisp_role(*role_name: str*) → bool

Checks if a role is crisp.

Parameters
role_name (*str*) – Name of the role.

Returns
true if the semantics is classical logic or if the role is crisp, false otherwise.

Return type
bool

is_lazy_unfoldable() → bool

Checks if the fuzzy KB is already lazy unfoldable.

is_loaded() → bool

Checks if the fuzzy KB is loaded.

is_redundant_A_is_a_C(
concept_name: str,
conc: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
implication: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
n: float,
) → bool

is_redundant_gci(
C: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
D: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
implication: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
n: float,
) → bool

Checks if C => D redundant.

Parameters

- **C** (Concept) – Subsumed concept.
- **D** (Concept) – Subsumer concept.
- **implication** (LogicOperatorType) – A fuzzy implication.
- **n** (*float*) – Degree of truth.

Raises
InconsistentOntologyException – If C is *top* concept and D is *bottom* concept.

is_tbox_acyclic() → bool

Check if t_inclusions cup t_definitions is acyclic

kleene_dienes_implies(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → None

Adds a Kleene-Dienes General Concept Inclusion.

Parameters

- **conc1** (Concept) – Subsumed concept.
- **conc2** (Concept) – Subsumer concept.
- **degree** (Degree) – Lower bound for the degree.

lukasiewicz_implies(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → None

Adds a Lukasiewicz General Concept Inclusion.

Parameters

- **conc1** (Concept) – Subsumed concept.
- **conc2** (Concept) – Subsumer concept.
- **degree** (Degree) – Lower bound for the degree.

mark_process_assertion(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

Marks assertion as processed.

merge(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
b: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
) → None

Merges two individuals.

Parameters

- **a** (Individual) – An individual. As an effect, it will contain a merged individual.
- **b** (Individual) – Another individual.

merge_fillers(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual, *func_role: str*) → None

If individual ind has two or more fillers via the functional role funcRole, they are merged into just one filler concept.

Parameters

- **ind** (Individual) – Subject individual.
- **func_role** (*str*) – A functional role.

nominal_absorption(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → bool

optimize(*e:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

It optimizes an expression.

Parameters
e (Expression) – Expression to be optimized.

Returns
An optimal solution of the expression.

Return type
Solution

partition_loop_A_is_a_B() → None

partition_loop_A_is_a_C() → None

partition_loop_C_is_a_A() → None

partition_loop_C_is_a_D() → None

partition_loop_to_do_A_is_a_B() → None

partition_loop_to_do_A_is_a_C() → None

preprocess_tbox() → None

Computes if the fuzzy KB has an acyclic TBox. If not, add primitive and concept definitions as GCIs.

print_tbox() → None

read_object_from_file(*file_path: str*) → *KnowledgeBase*

remove_A_is_a_B(*key: str*, *pcd:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → None

remove_A_is_a_C(*key: str*, *pcd:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → None

remove_A_is_a_X(
key: str,
pcd: fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition,
pcd_dict: *dict[str,* *set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]]*,
) → None

remove_A_is_a_X(
key: str,
pcd: fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition,
atomic: bool,
) → None

remove_C_is_a_A(*key: str*, *gci:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion) → None

remove_C_is_a_D(*key: str*, *gci:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion) → None

remove_C_is_a_X(*key: str*, *gci:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion, *atomic: bool*) → None

represent_tbox_with_gcis() → None

restrict_range(*x_b:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable, *k1: float*, *k2: float*) → None

restrict_range(
x_b: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_f: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
k1: float,
k2: float,
) → None

Restricts the range of a variable to [k1, k2].

role_absorption(*tau:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → bool

role_absorption(*tau:* fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion, *atomic: bool*) → bool

role_domain(*role: str*, *conc:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

Adds a domain funcRole axiom.

role_implies(*subsumed: str*, *subsumer: str*) → None

role_implies(*subsumed: str*, *subsumer: str*, *n: float*) → None

Adds a Role Inclusion Axiom (subsumed, subsumer, degree).

role_is_functional(*role: str*) → None

Adds a functional funcRole axiom.

role_is_inverse_functional(*role: str*) → None

Adds an inverse functional funcRole axiom.

role_is_reflexive(*role: str*) → None

Adds a reflexive funcRole axiom.

role_is_symmetric(*role: str*) → None

Adds a symmetric funcRole axiom.

role_is_transitive(*role: str*) → None

Adds a transitive funcRole axiom.

role_range(*role: str*, *conc:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

Adds a funcRole range axiom.

role_subsumes(*subsumer: str*, *subsumed: str*, *n: float*) → None

Adds a Role Inclusion Axiom (subsumer, subsumed, degree).

Parameters

- **subsumer** (*str*) – Subsumer funcRole.
- **subsumed** (*str*) – Subsumed funcRole.
- **n** (*float*) – Lower bound for the degree.

role_subsumes_bool(*subsumer: str*, *subsumed: str*, *n: float*) → bool

role_subsumes_bool(*subsumer: str*, *subsumed: str*, *n: float*, *p_list: dict[str, dict[str, float]]*) → bool

Adds a Role Inclusion Axiom (subsumer, subsumed, degree).

rule_all(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_and(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_ass_nom(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
v: str,
) → None

Applies the rule AssNom to a node v and an assertion <a : C>.

Parameters

- **a** (Individual) – Individual of an assertion.
- **c** (Concept) – Concept of an assertion.
- **v** (*str*) – Node that is an a-node.

rule_atomic(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_bottom(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_choquet(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented(
i: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → None

rule_complemented_at_least_datatype_restriction(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
ass: fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion,
) → None

rule_complemented_at_most_datatype_restriction(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
ass: fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion,
) → None

rule_complemented_atomic(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_choquet(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_complex_assertion(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_concrete(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_exact_datatype_restriction(
b: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
ass: fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion,
) → None

rule_complemented_extended_negative_threshold(*ass:* fuzzy_dl_-owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_extended_positive_threshold(*ass:* fuzzy_dl_-owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_fuzzy_number(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_goedel_implication(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_has_value(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_lazy_unfolding(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_modified(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_negative_threshold(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_owa(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_positive_threshold(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_quantified_owa(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_quasi_sugeno(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_self(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_sigma_concept(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_sugeno(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_weighted_concept(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_weighted_max(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_weighted_min(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_weighted_sum(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_weighted_sum_zero(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_complemented_zadeh_implication(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_concrete(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_domain_lazy_unfolding(*domain_role: str*, *rel:* fuzzy_dl_owl2.fuzzydl.relation.Relation) → None

rule_extended_negative_threshold(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_extended_positive_threshold(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_fuzzy_number(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_goedel_and(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_goedel_implication(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_goedel_or(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_has_value(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_lazy_unfolding(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_loose_lower_approximation(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_loose_upper_approximation(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_lower_approximation(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_lukasiewicz_and(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_lukasiewicz_or(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_modified(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_n2() → None

rule_n3() → None

rule_negative_threshold(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_or(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_owa(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_positive_threshold(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_quantified_owa(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_quasi_sugeno(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_range_lazy_unfolding(*range_role: str*, *rel:* fuzzy_dl_owl2.fuzzydl.relation.Relation) → None

rule_self(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_sigma_concept(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_some(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_sugeno(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_threshold_common(
x_a_in_c: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x_a_in_tc: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
y: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
) → None

rule_tight_lower_approximation(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_tight_upper_approximation(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_top(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_upper_approximation(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_weighted_concept(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_weighted_max(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_weighted_min(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_weighted_sum(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_weighted_sum_zero(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

rule_zadeh_implication(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

save_absorbed_tbox_to_file(*output: Callable*) → None

save_tbox_common_part_to_file(*output: Callable*) → None

save_tbox_to_file(*output: Callable*) → None

save_to_file(*file_name: str*) → None
Saves a fuzzy KB into a text file.

set_crisp_concept(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None
Defines a concept to be crisp.

set_crisp_role(*role_name: str*) → None
Defines a role to be crisp.

set_dynamic_blocking() → None
Sets dynamic blocking unless the current blocking is pairwise blocking.

set_logic(*logic:* fuzzy_dl_owl2.fuzzydl.util.constants.FuzzyLogic) → None
Sets the fuzzy logic of the fuzzy knowledge base.

set_truth_constants(*s: str*, *w: float*) → None
Sets a truth constant from the DL parser.

set_unsatisfiable_KB() → None

show_statistics() → None

solve_abox() → None

Solves all the fuzzy assertions.

solve_assertions() → None

Solves all the fuzzy assertions.

solve_cardinality_list() → None

Solve the list of sigma-count pending tasks

solve_choquet_integral_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
c: fuzzy_dl_owl2.fuzzydl.concept.choquet_integral.ChoquetIntegral,
) → None

Solves an assertion of the form (individual, concept) with respect to a fuzzy KB.

solve_choquet_integral_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
c: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
) → None

Solves an assertion of the form (individual, not concept) with respect to a fuzzy KB.

solve_concept_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → None

solve_concept_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
lower_limit: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → None

solve_concrete_value_assertions() → None

Solves the datatypes restrictions.

solve_crisp_concrete_concept_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concrete.crisp_concrete_concept.CrispConcreteConcept,
) → None

This function define the equations for the individual belonging to the crisp set.

Parameters

- **ind** (Individual) – current individual
- **Variables** –
  - x => variable associated with the individual
  - x' => generic variable associated with an individual belonging to this crisp concept
- **lines** (*Draw the four*) –
  - (b, 1) – (k_2, 0) -> y_2 <= (x - k_2) / (k_2 - b)
  - (a, 1) – (k_2, 0) -> y_1 <= (x - k_2) / (k_2 - a)
  - (b, 1) – (k_1, 0) -> y_3 >= (x - k_1) / (k_1 - b)
  - (a, 1) – (k_1, 0) -> y_2 >= (x - k_1) / (k_1 - a)
- **constraints** (*Along with the following*) –
  - y_1 + y_2 + y_3 = 1
  - x' + y_1 + y_3 <= 1

- x' - y_2 >= 0

solve_domain_and_range_axioms() → None

Solves all the domain and range restrictions.

solve_functional_roles() → None

Solves the functional role axioms.

solve_fuzzy_concrete_concept_complement_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
lower_limit: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
curr_concept: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
) → None

Solves an assertion of the form (individual, complement of the concept, degree) with respect to a fuzzy KB.

solve_gci(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
gci: fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion,
) → None

solve_gci(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual) → None

Solves a GCI for a given individual.

solve_goedel_gci(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
gci: fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion,
) → None

solve_inverse_roles() → None

Solves the inverse funcRole axioms.

solve_kb() → None

Prepares the fuzzy knowledge base to reason with it.

solve_kleene_dienes_gci(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
gci: fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion,
) → None

solve_left_concrete_concept_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concrete.left_concrete_concept.LeftConcreteConcept,
) → None

solve_linear_concrete_concept_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concrete.linear_concrete_concept.LinearConcreteConcept,
) → None

solve_linear_modifier_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
con: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
modifier: fuzzy_dl_owl2.fuzzydl.modifier.linear_modifier.LinearModifier,
) → None

solve_lukasiewicz_gci(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
gci: fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion,
) → None

solve_modifier_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
modifier: fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier,
) → None

Solves an assertion of the form (individual, concept, lower degree) with respect to a fuzzy KB.

solve_modifier_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → None

Solves an assertion of the form (individual, negated concept, lower degree) with respect to a fuzzy KB.

solve_one_exist_assertion() → None

Solves one existential assertion.

solve_owa_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
c: fuzzy_dl_owl2.fuzzydl.concept.owa_concept.OwaConcept | fuzzy_dl_owl2.fuzzydl.concept.qowa_concept.QowaConcept,
) → None

Solves an assertion of the form (individual, concept) with respect to a fuzzy KB.

solve_owa_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
curr_concept: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
) → None

Solves an assertion of the form (individual, not concept) with respect to a fuzzy KB.

solve_reflexive_role(*role: str*) → None

Solves a reflexive funcRole axiom.

solve_reflexive_roles(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual) → None

solve_reflexive_roles() → None

Solves a reflexive funcRole axiom.

solve_right_concrete_concept_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concrete.right_concrete_concept.RightConcreteConcept,
) → None

solve_role_inclusion_axioms() → None

solve_role_inclusion_axioms(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
r: fuzzy_dl_owl2.fuzzydl.relation.Relation,
) → None

Solves the fuzzy funcRole inclusion axioms.

solve_sugeno_integral_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.sugeno_integral.SugenoIntegral | fuzzy_dl_owl2.fuzzydl.concept.quasi_sugeno_integral.QsugenoIntegral,
) → None

Solves an assertion of the form (individual, concept) with respect to a fuzzy KB.

solve_sugeno_integral_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
curr_concept: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
) → None

Solves an assertion of the form (individual, not concept) with respect to a fuzzy KB.

solve_trapezoidal_concrete_concept_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concrete.trapezoidal_concrete_concept.TrapezoidalConcreteConcept,
) → None

solve_triangular_concrete_concept_assertion(
individual: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concrete.triangular_concrete_concept.TriangularConcreteConcept,
) → None

solve_triangular_modifier_assertion(
individual: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
modifier: fuzzy_dl_owl2.fuzzydl.modifier.triangular_modifier.TriangularModifier,
) → None

solve_w_max_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.weighted_max_concept.WeightedMaxConcept,
) → None

Solves an assertion of the form (individual, concept) with respect to a fuzzy KB.

solve_w_max_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
curr_concept: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
) → None

Solves an assertion of the form (individual, not concept) with respect to a fuzzy KB.

solve_w_min_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.weighted_min_concept.WeightedMinConcept,
) → None

Solves an assertion of the form (individual, concept) with respect to a fuzzy KB.

solve_w_min_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
curr_concept: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
) → None

Solves an assertion of the form (individual, not concept) with respect to a fuzzy KB.

solve_w_sum_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.weighted_sum_concept.WeightedSumConcept,
) → None

Solves an assertion of the form (individual, concept) with respect to a fuzzy KB.

solve_w_sum_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
curr_concept: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
) → None

Solves an assertion of the form (individual, not concept) with respect to a fuzzy KB.

solve_w_sum_zero_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept: fuzzy_dl_owl2.fuzzydl.concept.weighted_sum_zero_concept.WeightedSumZeroConcept,
) → None

Solves an assertion of the form (individual, concept) with respect to a fuzzy KB.

solve_w_sum_zero_complemented_assertion(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
curr_concept: fuzzy_dl_owl2.fuzzydl.concept.operator_concept.OperatorConcept,
) → None

Solves an assertion of the form (individual, not concept) with respect to a fuzzy KB.

solve_zadeh_gci(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
gci: fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion,
) → None

synonym_absorption_A_is_a_B(*pcd1:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → bool

Absorbs synonyms in axioms_A_is_a_B.

Returns
true if there are changes; false otherwise.

Return type
bool

synonym_absorption_to_do_A_is_a_B(*pcd1:* fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition) → bool

Absorbs synonyms in axioms_to_do_A_is_a_B. note that A => B is in t_inclusions.

Returns
true if there are changes; false otherwise.

Return type
bool

unblock_children(*ancestor: str*) → None

Unblocks the children of the individual with the given name.

Parameters
ancestor (*str*) – Name of the ancestor individual.

unblock_individual(*node_name: str*) → None

Unblocks the individual and descendants of the individual with the given name. :param node_name: Name of the ancestor individual. :type node_name: str

write_object_to_file(*file_path: str*) → None

zadeh_implies(
conc1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
conc2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → None

Adds a Zadeh General Concept Inclusion.

Parameters

- **conc1** (Concept) – Subsumed concept.
- **conc2** (Concept) – Subsumer concept.

ABOX_EXPANDED: bool = False

CLASSIFIED: bool = False

KB_LOADED: bool = False

KB_UNSAT: bool = False

abstract_roles: set[str]

acyclic_tbox: bool = False

applied_trans_role_rules: list[str] = []

assertions: list[*fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion*] = []

atomic_concepts: dict[str, *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

axioms_A_equiv_C: dict[str, set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]]

axioms_A_is_a_B: dict[str, set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]]

axioms_A_is_a_C: dict[str, set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]]

axioms_C_equiv_D:
list[*fuzzy_dl_owl2.fuzzydl.concept_equivalence.ConceptEquivalence*] = []

axioms_C_is_a_A: dict[str,
set[*fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion*]]

axioms_C_is_a_D: dict[str,
set[*fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion*]]

axioms_to_do_A_is_a_B: dict[str, set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]]

axioms_to_do_A_is_a_C: dict[str, set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]]

axioms_to_do_C_is_a_A: dict[str,
set[*fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion*]]

axioms_to_do_C_is_a_D: dict[str,
set[*fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion*]]

axioms_to_do_tmp_A_is_a_C: dict[str, set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]]

axioms_to_do_tmp_C_is_a_A: dict[str,
set[*fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion*]]

axioms_to_do_tmp_C_is_a_D: dict[str,
set[*fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion*]]

blocked_assertions: dict[str,
list[*fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion*]]

blocked_exist_assertions: dict[str,
list[*fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion*]]

blocking_dynamic: bool = False

blocking_type: *fuzzy_dl_owl2.fuzzydl.util.constants.BlockingDynamicType*

concept_individual_list: dict[int, sortedcontainers.SortedSet[*fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual*]]

concrete_concepts: dict[str, *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*]

concrete_features: dict[str, *fuzzy_dl_owl2.fuzzydl.concrete_feature.ConcreteFeature*]

concrete_fuzzy_concepts: bool = False

concrete_roles: set[str]

directly_blocked_children: dict[str, list[str]]

disjoint_variables: dict[str, set[str]]

domain_restrictions: dict[str, set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]]

exist_assertions: list[*fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion*] = []

functional_roles: set[str]

fuzzy_numbers: dict[str, *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber*]

individuals: dict[str, *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]

inverse_functional_roles: set[str]

inverse_roles: dict[str, set[str]]

labels_with_nodes: dict[str, set[str]]

language: str = ''

lazy_unfondable: bool = False

max_depth: int = 1

milp: *fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper*

modifiers: dict[str, *fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier*]

nodes_classification: list[*fuzzy_dl_owl2.fuzzydl.classification_node.ClassificationNode*] = []

num_assertions: int = 0

num_defined_concepts: int = 0

num_defined_individuals: int = 0

num_relations: int = 0

number_of_concepts: dict[str, int]

number_of_roles: dict[str, int]

old_01_variables: int = 0

old_binary_variables: int = 0

order: dict[str, int]

positive_concrete_value_assertions: list[*fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion*] = []

processed_assertions: set[int]

r_successors: dict[str, list[str]]

range_restrictions: dict[str, set[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]]

reflexive_roles: set[str]

roles_with_all_parents: dict[str, dict[str, float]]

roles_with_parents: dict[str, dict[str, float]]

roles_with_trans_children: dict[str, list[str]]

rule_acyclic_tbox: bool = False

rules_applied: dict[*fuzzy_dl_owl2.fuzzydl.util.constants.KnowledgeBaseRules*, int]

show_language: bool = False

similarity_relations: set[str]

subsumption_flags: dict[str, dict[str, float]]

symmetric_roles: set[str]

t_G: list[*fuzzy_dl_owl2.fuzzydl.general_concept_inclusion.GeneralConceptInclusion*] = []

t_definitions: dict[str, *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*]

t_disjoints: dict[str, set[str]]

t_inclusions: dict[str, set[*fuzzy_dl_owl2.fuzzydl.primitive_concept_definition.PrimitiveConceptDefinition*]]

t_synonyms: dict[str, set[str]]

temp_relations_list: dict[str, list[*fuzzy_dl_owl2.fuzzydl.relation.Relation*]]

temp_string_concept_list: list[*fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*] = []

temp_string_list: list[str] = []

tmp_features: list[str] = []

transitive_roles: set[str]

truth_constants: dict[str, float]

x_prime_individuals: dict[str, list[str]]

y_prime_individuals: dict[str, list[str]]

class LukasiewiczSolver

static and_(*n1: float*, *n2: float*) → float

Gets the value n1 and n2, according to Lukasiewicz t-norm

static and_equation(
*x: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]*,
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: float,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_geq_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_geq_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: float,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_leq_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute z <= x1 AND x2

static or_equation(
*x: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]*,
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute z = x1 OR x2 OR … OR xN

static solve_all(
rel: fuzzy_dl_owl2.fuzzydl.relation.Relation,
restrict: fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction,
kb: KnowledgeBase,
) → None

Solves a universal restriction fuzzy assertion with respect to a reference fuzzy KB.

static solve_and(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a conjunction fuzzy assertion with respect to a reference fuzzy KB.

static solve_or(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a disjunction fuzzy assertion with respect to a reference fuzzy KB.

static solve_some(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a existential restriction fuzzy assertion with respect to a reference fuzzy KB.

class ZadehSolver

Solver for Zadeh fuzzy logic semantics.

static and_(*n1: float*, *n2: float*) → float

Gets the value n1 and n2, according to Goedel t-norm

static and_equation(
*x: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]*,
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_equation(
*x: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]*,
t: fuzzy_dl_owl2.fuzzydl.milp.term.Term,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: float,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_equation(
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_geq_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_geq_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: float,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static and_leq_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,

x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute z <= x1 AND x2

static and_negated_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: float,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute z = (1 - x1) AND x2

static goedel_implies_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute z = x1 G-implies x2

static goedel_not_equation(
y: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute y = NOT z

static kleene_dienes_implies_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute z <= x1 KD-implies x2

static or_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static or_equation(
*x: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]*,
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static or_negated_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: float,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute z = (1 - x1) OR x2

static solve_all(
rel: fuzzy_dl_owl2.fuzzydl.relation.Relation,
restrict: fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction,
kb: KnowledgeBase,
) → None

Solves a universal restriction fuzzy assertion with respect to a reference fuzzy KB.

static solve_and(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a conjunction fuzzy assertion with respect to a reference fuzzy KB.

static solve_or(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a disjunction fuzzy assertion with respect to a reference fuzzy KB.

static solve_some(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *kb:* KnowledgeBase) → None

Solves a existential restriction fuzzy assertion with respect to a reference fuzzy KB.

static zadeh_implies_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static zadeh_implies_equation(
z: float,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

static zadeh_implies_leq_equation(
z: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
x2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
milp: fuzzy_dl_owl2.fuzzydl.milp.milp_helper.MILPHelper,
) → None

Compute z <= x1 Z-implies x2, where x1 is binary

fuzzy_dl_owl2.fuzzydl.label

Classes

Label	Label (weighted concept used in created individuals)

Module Contents

class Label(*concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *weight:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree)

Label (weighted concept used in created individuals)

__eq__(*cw: Self*) → bool

__ne__(*cw: Self*) → bool

__str__() → str

static weights_equal(*w1:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree, *w2:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree) → bool

Checks if two degrees are equal

concept: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

weight: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

fuzzy_dl_owl2.fuzzydl.milp

Submodules

fuzzy_dl_owl2.fuzzydl.milp.expression

Classes

Expression	Linear expression of the form c + c1 * x1 + c2 * x2 + ... + cN * xN

Module Contents

class Expression(*constant: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*)

class Expression(*constant: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*, **terms:* fuzzy_dl_owl2.fuzzydl.milp.term.Term)

class Expression(**terms:* fuzzy_dl_owl2.fuzzydl.milp.term.Term)

class Expression(*expr: Self*)

class Expression(*v: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*] | set[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]*)

Linear expression of the form c + c1 * x1 + c2 * x2 + … + cN * xN

__add__(*value: int | float | Self |* fuzzy_dl_owl2.fuzzydl.milp.term.Term) → Self

__eq__(*value: Self*) → bool

__hash__() → int

__mul__(*scalar: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*) → Self

__ne__(*value: Self*) → bool

__neg__() → Self

__radd__(*scalar: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*) → Self

__repr__() → str

__rmul__(*scalar: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*) → Self

__rsub__(*scalar: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*) → Self

__str__() → str

Gets a printable name of the expression.

__sub__(*expr: int | float | Self |* fuzzy_dl_owl2.fuzzydl.milp.term.Term) → Self

__truediv__(*scalar: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*) → Self

static add_constant(*expr: Self*, *constant: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*) → Self

Adds a constant to an expression.

static add_expressions(*expr1: Self*, *expr2: Self*) → Self

Adds two expressions.

add_term(*term:* fuzzy_dl_owl2.fuzzydl.milp.term.Term) → None

Adds a term to an expression.

static add_term_(*exp: Self*, *term:* fuzzy_dl_owl2.fuzzydl.milp.term.Term) → Self

Adds a term to an expression.

clone() → Self

get_constant() → fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER

get_constant_term(*var:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER

Given a variable, gets its coefficient in the expression.

get_terms() → list[*fuzzy_dl_owl2.fuzzydl.milp.term.Term*]

increment_constant() → None

Increments the constant in one.

static multiply_constant(*expr: Self*, *constant: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*) → Self

Multiplies a constant and an expression.

static negate_expression(*expr: Self*) → Self

Changes the sign of all the elements of an expression.

set_constant(*constant: fuzzy_dl_owl2.fuzzydl.util.constants.NUMBER*) → None

static subtract_expressions(*expr1: Self*, *expr2: Self*) → Self

Substracts two expressions.

fuzzy_dl_owl2.fuzzydl.milp.inequation

Attributes

EqualTo	
GreaterThan	
LessThan	

Classes

Inequation	Inequality of the form c + c1 * x1 + c2 * x2 + ... (>= | <= | =) 0.

Module Contents

class Inequation(*exp:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression, *i_type:* fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType)

Inequality of the form c + c1 * x1 + c2 * x2 + ... (>= | <= | =) 0.

__eq__(*value: Self*) → bool

__hash__() → int

__ne__(*value: Self*) → bool

__repr__() → str

__str__() → str

Gets a printable name of the object.

clone() → Self

static equal_to(*exp:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) → Self

get_constant() → float

get_string_type() → str

Gets a string representation of the type.

get_terms() → list[*fuzzy_dl_owl2.fuzzydl.milp.term.Term*]

get_type() → *fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType*

static greater_then(*exp:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) → Self

is_zero() → bool

static less_than(*exp:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) → Self

expr: ***fuzzy_dl_owl2.fuzzydl.milp.expression.Expression***

type: ***fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType***

EqualTo

GreaterThan

LessThan

fuzzy_dl_owl2.fuzzydl.milp.milp_helper

Classes

MILPHelper	MILP problem manager, storing the problem and calling an external solver.

Module Contents

class MILPHelper

MILP problem manager, storing the problem and calling an external solver.

add_cardinality_list(*sc:* fuzzy_dl_owl2.fuzzydl.concept.sigma_count.SigmaCount) → None

SigmaCount(r,C,O,d)^I(w) = d^I(xSigma)

Parameters

sc (SigmaCount) – xSigma: Free variable taking the value sigma_{i2 in O} r(i1, i2) otimes C(i2) i1: Name of an individual, subject of the relation. O: Set of individuals candidates to be the object of the relation. r: Role. C: Concept.

add_contradiction() → None

Add a contradiction to make the fuzzy KB unsatisfiable

add_crisp_concept(*concept_name: str*) → None

Defines a concept to be crisp.

add_crisp_role(*role_name: str*) → None

Defines a role to be crisp.

add_equality(*var1:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable, *var2:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → None

Add an equality of the form: var1 = var2.

add_new_constraint(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
constraint_type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
) → None

add_new_constraint(*x:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable, *n: float*) → None

add_new_constraint(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion, *n: float*) → None

add_new_constraint(*x:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable, *d:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree) → None

add_new_constraint(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → None

add_new_constraint(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
constraint_type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
) → None

add_new_constraint(
expr: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
constraint_type: fuzzy_dl_owl2.fuzzydl.util.constants.InequalityType,
n: float,
) → None

add_string_feature(*role: str*) → None

Adds a string feature.

add_string_value(*value: str*, *int_value: int*) → None

Relates the value of a string feature with an integer value.

Parameters

- **value** (*str*) – Value of a string feature.
- **int_value** (*int*) – Corresponding integer value.

change_variable_names(*old_name: str*, *new_name: str*, *old_is_created_individual: bool*) → None

Replaces the name of the variables including an individual name with the name of another individual name.

Parameters

- **old_name** (*str*) – Old individual name.
- **new_name** (*str*) – New individual name.
- **old_is_created_individual** (*bool*) – Indicates whether the old individual is a created individual or not.

check_if_replacement_is_needed(
v1: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
s1: str,
v2: fuzzy_dl_owl2.fuzzydl.milp.variable.Variable,
s2: str,
) → bool

clone() → Self

exists_nominal_variable(*i: str*) → bool

Checks if there exists a variable taking the value of an individual i belonging to the nominal concept {i}.

exists_variable(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
b: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role: str,
) → bool

Checks if a variable taking the value of a role assertion exists.

Parameters

- **a** (Individual) – Object individual.
- **b** (Individual) – Subject individual.
- **role** (*str*) – A role name.

get_name_for_integer(*i: int*) → str | None

Gets the name of the i-th variable.

get_negated_nominal_variable(*i1: str*, *i2: str*) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

Gets a variable taking the value of an individual i1 not belonging to the nominal concept {i2}.

Parameters

- **i1** (*str*) – An individual that is subject of the assertion.
- **i2** (*str*) – An individual representing the nominal concept.

Returns
A variable taking the value of the assertion i1: not {i2}.

Return type
Variable

get_new_variable(*v_type:* fuzzy_dl_owl2.fuzzydl.util.constants.VariableType) →
fuzzy_dl_owl2.fuzzydl.milp.variable.Variable

Gets a new variable with the indicated type.

get_nominal_variable(*i1: str*) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_nominal_variable(*i1: str*, *i2: str*) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_number_for_assertion(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → int

Gets an integer codification of an assertion.

get_ordered_permutation(
*x: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]*,
) → list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]

get_ordered_permutation(
*x: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]*,
*z: list[list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]]*,
) → list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]

get_variable(*var_name: str*) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(
var_name: str,
v_type: fuzzy_dl_owl2.fuzzydl.util.constants.VariableType,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) →
fuzzy_dl_owl2.fuzzydl.milp.variable.Variable

get_variable(*rel:* fuzzy_dl_owl2.fuzzydl.relation.Relation) →
fuzzy_dl_owl2.fuzzydl.milp.variable.Variable

get_variable(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
restrict: fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
concept_name: str,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
b: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role: str,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
b: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role: str,
v_type: fuzzy_dl_owl2.fuzzydl.util.constants.VariableType,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(
a: str,
b: str,
role: str,
v_type: fuzzy_dl_owl2.fuzzydl.util.constants.VariableType,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(
ind: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
) → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

get_variable(
ind: fuzzy_dl_owl2.fuzzydl.individual.created_individual.CreatedIndividual,
v_type: fuzzy_dl_owl2.fuzzydl.util.constants.VariableType,
) → None

has_nominal_variable(*terms: list[*fuzzy_dl_owl2.fuzzydl.milp.term.Term*]*) → bool

Checks if a collection of terms has a nominal variable.

has_variable(*name: str*) → bool

has_variable(*ass:* fuzzy_dl_owl2.fuzzydl.assertion.assertion.Assertion) → bool

is_crisp_concept(*concept_name: str*) → bool

Checks if a concept is crisp or not.

is_crisp_role(*role_name: str*) → bool

Checks if a role is crisp or not.

is_nominal_variable(*i: str*) → bool

Checks if a variable 'i' is a nominal variable.

optimize(*objective:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) →
fuzzy_dl_owl2.fuzzydl.milp.solution.Solution | None

It optimizes an expression using a solvers from MILPProvider.

Parameters
objective (Expression) – Expression to be optimized.

Raises
ValueError – If MILPProvider is not known.

Returns
An optimal solution of the expression

Return type
Optional[*Solution*]

print_instance_of_labels(*f_name: str*, *ind_name: str*, *value: float*) → None

print_instance_of_labels(*name: str*, *value: float*) → None

Shows the membership degrees to some linguistic labels.

set_binary_variables() → None

Transforms every [0,1]-variable into a {0,1} variable.

set_nominal_variables(*value: bool*) → None

solve_gurobi(
objective: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution* | None

Solves a MILP problem using Gurobi.

solve_mip(*objective:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression) →
fuzzy_dl_owl2.fuzzydl.milp.solution.Solution | None

solve_pulp(
objective: fuzzy_dl_owl2.fuzzydl.milp.expression.Expression,
) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution* | None

PARTITION: bool = False

PRINT_LABELS: bool = True

PRINT_VARIABLES: bool = True

cardinalities: list[*fuzzy_dl_owl2.fuzzydl.concept.sigma_count.SigmaCount*] = []

constraints: list[*fuzzy_dl_owl2.fuzzydl.milp.inequation.Inequation*] = []

crisp_concepts: set[str]

crisp_roles: set[str]

nominal_variables: bool = False

number_of_variables: dict[str, int]

show_vars: *fuzzy_dl_owl2.fuzzydl.milp.show_variables_helper.ShowVariablesHelper*

string_features: set[str]

string_values: dict[int, str]

variables: list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*] = []

fuzzy_dl_owl2.fuzzydl.milp.show_variables_helper

Classes

ShowVariablesHelper

Module Contents

class ShowVariablesHelper

add_abstract_filler_to_show(*role_name: str*) → None

add_abstract_filler_to_show(*role_name: str*, *ind_name: str*) → None

Shows the membership degree to some atomic concepts of the fillers of an abstract role.

add_concept_to_show(*conc_name: str*) → None

Show membership degree of every instance of an atomic concept.

add_concrete_filler_to_show(*f_name: str*) → None

add_concrete_filler_to_show(*f_name: str*, *ind_name: str*) → None

add_concrete_filler_to_show(
f_name: str,
ind_name: str,
*ar: list[*fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*]*,
) → None

Shows the value of the fillers of a concrete feature.

add_individual_to_show(*ind_name: str*) → None

Shows the value of an individual to every atomic concept.

add_variable(*var:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable, *name_to_show: str*) → None

Add a variable to shown, showing it with a given name.

Parameters

- **var** (Variable) – A variable.
- **name_to_show** (*str*) – Name of the variable when shown.

clone() → Self

get_labels(*var_name: str*) →
list[*fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*]

Gets the fuzzy concrete concepts marked to be shown for a variable.

get_name(*var:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → str

Gets the name of a variable.

get_variables() → list[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*]

Gets the variables to be shown.

show_abstract_role_fillers(*role_name: str*, *ind_name: str*) → bool

Returns whether a given individuals is marked for showing every filler of an abstract role.

Parameters

- **role_name** (*str*) – Name of the abstract role.
- **ind_name** (*str*) – Name of the individual.

show_concepts(*concept_name: str*) → bool

Returns whether an atomic concept is marked to show the membership degree of every individual.

Parameters
concept_name (*str*) – Name of atomic concept.

Returns
true if the concept is marked to be shown; false otherwise.

Return type
bool

show_concrete_fillers(*f_name: str*, *ind_name: str*) → bool

Returns whether a given individuals is marked for showing every filler of a concrete feature.

Parameters

- **f_name** (*str*) – Name of the concrete feature.
- **ind_name** (*str*) – Name of the individual.

show_individuals(*ind_name: str*) → bool

Checks whether an individual is marked to be shown or not.

show_variable(*var:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → bool

Checks whether the variable exists or not.

abstract_fillers: dict[str, set[str]]

concepts: set[str]

concrete_fillers: dict[str, set[str]]

global_abstract_fillers: set[str]

global_concrete_fillers: set[str]

individuals: set[str]

labels_for_fillers: dict[str, list[*fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_concrete_concept.FuzzyConcreteConcept*]]

variables: dict[*fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*, str]

fuzzy_dl_owl2.fuzzydl.milp.solution

Classes

Solution

Module Contents

class Solution(*consistent: bool*)
class Solution(*sol: float*)

__hash__() → int

__repr__() → str

__str__() → str

add_showed_variable(*var_name: str*, *value: float*) → None

Sets the value of a showed variable.

get_showed_variables() → dict[str, float]

Gets the values of some variables after solving a query over a consistent KB.

get_solution() → bool | float

Gets the solution to some query over a consistent KB.

is_consistent_kb() → bool

Indicates whether the original KB is consistent or not.

CONSISTENT_KB: bool = True

INCONSISTENT_KB: bool = False

fuzzy_dl_owl2.fuzzydl.milp.term

Classes

Term

Module Contents

class Term(*coeff: float*, *var:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable)
class Term(*var:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable)

__add__(*term: Self*) → Self

__eq__(*term: Self*) → bool

__hash__() → int

__mul__(*scalar: float*) → Self

__ne__(*term: Self*) → bool

__neg__() → Self

__repr__() → str

__rmul__(*scalar: float*) → Self

__str__() → str

__sub__(*term: Self*) → Self

__truediv__(*scalar: float*) → Self

clone() → Self

get_coeff() → float

get_var() → *fuzzy_dl_owl2.fuzzydl.milp.variable.Variable*

fuzzy_dl_owl2.fuzzydl.milp.variable

Attributes

BinaryVar
FreeVar

continues on next page

Table 80 – continued from previous page

IntegerVar
UpVar

Classes

Variable

Module Contents

class Variable(*name: str*, *v_type:* fuzzy_dl_owl2.fuzzydl.util.constants.VariableType)

__eq__(*value: Self*) → bool

__hash__() → int

__ne__(*value: object*) → bool

__repr__() → str

__str__() → str

clone() → Self

static get_binary_variable(*name: str*) → Self

static get_continuous_variable(*name: str*) → Self

get_datatype_filler_type() → bool

static get_integer_variable(*name: str*) → Self

get_lower_bound() → float

static get_new_variable(*v_type:* fuzzy_dl_owl2.fuzzydl.util.constants.VariableType) → Self

static get_semi_continuous_variable(*name: str*) → Self

get_type() → *fuzzy_dl_owl2.fuzzydl.util.constants.VariableType*

get_upper_bound() → float

set_binary_variable() → None

set_datatype_filler_variable() → None

set_name(*name: str*) → None

set_type(*v_type:* fuzzy_dl_owl2.fuzzydl.util.constants.VariableType) → None

VARIABLE_NAME: str = 'y'

VARIABLE_NUMBER: int = 0

datatype_filler: bool = False

lower_bound: float = 0.0

name: str

type: *fuzzy_dl_owl2.fuzzydl.util.constants.VariableType* = None

upper_bound: float = 0.0

BinaryVar

FreeVar

IntegerVar

UpVar

fuzzy_dl_owl2.fuzzydl.modifier

Submodules

fuzzy_dl_owl2.fuzzydl.modifier.linear_modifier

Classes

LinearModifier	Linear modifier with parameter c

Module Contents

class LinearModifier(*name: str*, *c: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier*

Linear modifier with parameter c

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*value: float*) → float

Gets the image in [0,1] of a real number to the modifier.

Parameters
value (*float*) – A real number in the range of values of the modifier function.

Returns
Image in [0,1] of x to the explicit modifier function.

Return type
float

modify(
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
) → *fuzzy_dl_owl2.fuzzydl.concept.modified.linearly_modified_concept.LinearlyModifiedConcept*

Modifies a fuzzy concept.

Parameters
concept (Concept) – A fuzzy concept

Returns
Fuzzy concept resulting from the application of the modifier to c.

Return type
Concept

property a: float

property b: float

property c: float

fuzzy_dl_owl2.fuzzydl.modifier.modifier

Classes

Modifier	Fuzzy modifier.

Module Contents

class Modifier(*name: str*)

Bases: abc.ABC

Fuzzy modifier.

__repr__() → str

__str__() → str

abstractmethod clone() → Self

abstractmethod compute_name() → str

abstractmethod get_membership_degree(*value: float*) → float

Gets the image in [0,1] of a real number to the modifier.

Parameters
value (*float*) – A real number in the range of values of the modifier function.

Returns
Image in [0,1] of x to the explicit modifier function.

Return type
float

abstractmethod modify(*concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Modifies a fuzzy concept.

Parameters
concept (Concept) – A fuzzy concept

Returns
Fuzzy concept resulting from the application of the modifier to c.

Return type
Concept

set_name(*name: str*) → None

name: str

fuzzy_dl_owl2.fuzzydl.modifier.triangular_modifier

Classes

TriangularModifier	Fuzzy modifier.

Module Contents

class TriangularModifier(*name: str*, *a: float*, *b: float*, *c: float*)

Bases: *fuzzy_dl_owl2.fuzzydl.modifier.modifier.Modifier*

Fuzzy modifier.

__and__(*value: Self*) → Self

__hash__() → int

__neg__() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

__or__(*value: Self*) → Self

clone() → Self

compute_name() → str

get_membership_degree(*x: float*) → float

Gets the image in [0,1] of a real number to the modifier.

Parameters
value (*float*) – A real number in the range of values of the modifier function.

Returns
Image in [0,1] of x to the explicit modifier function.

Return type
float

modify(*concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

Modifies a fuzzy concept.

Parameters
concept (Concept) – A fuzzy concept

Returns
Fuzzy concept resulting from the application of the modifier to c.

Return type
Concept

property a: float

property b: float

property c: float

fuzzy_dl_owl2.fuzzydl.parser

Submodules

fuzzy_dl_owl2.fuzzydl.parser.dl_parser

Attributes

FILENAME
LOG_DIR
TODAY

Classes

DLParser

Module Contents

class DLParser

Bases: object

static get_grammatics() → pyparsing.ParserElement

This function generate the grammatics to parse the predicate wih formula "formula".

Parameters
formula (= *The predicate formula used for the parsing.*)

Return type
The parsed result given by pyparsing.

static get_kb(**args*) → tuple[*fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase*, list[*fuzzy_dl_owl2.fuzzydl.query.query.Query*]]

static load_config(**args*) → None

static main(**args*) → None

static parse_string(*instring: str*) → pyparsing.ParseResults

static parse_string_opt(*filename: str*) → pyparsing.ParseResults

kb: *fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase* = None

queries_list: list[*fuzzy_dl_owl2.fuzzydl.query.query.Query*] = []

FILENAME: str

LOG_DIR: str

TODAY: datetime.datetime

fuzzy_dl_owl2.fuzzydl.primitive_concept_definition

Classes

PrimitiveConceptDefinition	General concept inclusion axiom.

Module Contents

class PrimitiveConceptDefinition(
defined: str,
definition: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
implication: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
degree: float,
)

General concept inclusion axiom.

__eq__(*other: Self*) → bool

__ge__(*other: Self*) → bool

__gt__(*other: Self*) → bool

__hash__() → int

__le__(*other: Self*) → bool

__lt__(*other: Self*) → bool

__ne__(*other: Self*) → bool

__repr__() → str

__str__() → str

clone() → Self

get_defined_concept() → str

get_definition() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_degree() → float

get_type() → *fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType*

set_definition(*definition:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept) → None

set_degree(*deg: float*) → None

defined: str

definition: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

degree: float

implication: *fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType*

fuzzy_dl_owl2.fuzzydl.query

Submodules

fuzzy_dl_owl2.fuzzydl.query.all_instances_query

Classes

AllInstancesQuery	Min instance query for every individual of a knowledge base.

Module Contents

class AllInstancesQuery(*concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Min instance query for every individual of a knowledge base.

__str__() → str

Solves the query over a fuzzy KB

get_degrees() → list[float]

get_individuals() → list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*]

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

solve_new(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Specific algorithm to solve the instance retrieval.

conc

degrees: list[float] = []

individuals: list[*fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*] = []

name = 'Instances of Uninferable?'

fuzzy_dl_owl2.fuzzydl.query.bnp_query

Classes

BnpQuery	Helper class that provides a standard way to create an ABC using

Module Contents

class BnpQuery(*c:* fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber)

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

c: *fuzzy_dl_owl2.fuzzydl.concept.concrete.fuzzy_number.triangular_fuzzy_number.TriangularFuzzyNumber*

fuzzy_dl_owl2.fuzzydl.query.classification_query

Classes

ClassificationQuery	Helper class that provides a standard way to create an ABC using

Module Contents

class ClassificationQuery

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.defuzzify

Submodules

fuzzy_dl_owl2.fuzzydl.query.defuzzify.defuzzify_query

Classes

DefuzzifyQuery	Helper class that provides a standard way to create an ABC using

Module Contents

class DefuzzifyQuery(
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
feature_name: str,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Helper class that provides a standard way to create an ABC using inheritance.

abstractmethod get_obj_expression(*variable:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution* | None

Solve the query using given knowledge base

a: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

conc: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

f_name: **str**

obj_expr: *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression* = **None**

fuzzy_dl_owl2.fuzzydl.query.defuzzify.lom_defuzzify_query

Classes

LomDefuzzifyQuery	Largest of maxima defuzzification query

Module Contents

class LomDefuzzifyQuery(
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
feature_name: str,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.defuzzify.defuzzify_query.DefuzzifyQuery*

Largest of maxima defuzzification query

__str__() → str

Solves the query over a fuzzy KB

get_obj_expression(*variable:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

fuzzy_dl_owl2.fuzzydl.query.defuzzify.mom_defuzzify_query

Classes

MomDefuzzifyQuery	Middle of maxima defuzzification query.

Module Contents

class MomDefuzzifyQuery(
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
feature_name: str,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.defuzzify.defuzzify_query.DefuzzifyQuery*

Middle of maxima defuzzification query.

__str__() → str

Solves the query over a fuzzy KB

get_obj_expression(*variable:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.defuzzify.som_defuzzify_query

Classes

SomDefuzzifyQuery	Smallest of maxima defuzzification query.

Module Contents

class SomDefuzzifyQuery(
c: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
ind: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
f_name: str,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.defuzzify.defuzzify_query.DefuzzifyQuery*

Smallest of maxima defuzzification query.

__str__() → str

Solves the query over a fuzzy KB

get_obj_expression(*q:* fuzzy_dl_owl2.fuzzydl.milp.variable.Variable) → *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

fuzzy_dl_owl2.fuzzydl.query.instance_query

Classes

InstanceQuery	Instance checking query

Module Contents

class InstanceQuery(
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
individual: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*, abc.ABC

Instance checking query

conc: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

ind: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

obj_expr: *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression* = None

fuzzy_dl_owl2.fuzzydl.query.kb_satisfiable_query

Classes

KbSatisfiableQuery	Knowledge base satisfiability degree

Module Contents

class KbSatisfiableQuery

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Knowledge base satisfiability degree

__str__() → str

Solves the query over a fuzzy KB

is_consistent_kb(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → bool

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.max

Submodules

fuzzy_dl_owl2.fuzzydl.query.max.max_instance_query

Classes

MaxInstanceQuery	Lowest upper bound of a concept assertion

Module Contents

class MaxInstanceQuery(
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
individual: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.instance_query.InstanceQuery*

Lowest upper bound of a concept assertion

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.max.max_query

Classes

MaxQuery	Maximize expression query

Module Contents

class MaxQuery(*expr:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression)

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Maximize expression query

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

obj_expr: *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression*

fuzzy_dl_owl2.fuzzydl.query.max.max_related_query

Classes

MaxRelatedQuery	Lowest upper bound of a role assertion (ind1, ind2, role)

Module Contents

class MaxRelatedQuery(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
b: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role_name: str,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.related_query.RelatedQuery*

Lowest upper bound of a role assertion (ind1, ind2, role)

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

ind1: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

ind2: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

role: **str**

fuzzy_dl_owl2.fuzzydl.query.max.max_satisfiable_query

Classes

MaxSatisfiableQuery	Maximal satisfiability degree of a fuzzy concept.

Module Contents

class MaxSatisfiableQuery(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

class MaxSatisfiableQuery(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *a:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual)

Bases: *fuzzy_dl_owl2.fuzzydl.query.satisfiable_query.SatisfiableQuery*

Maximal satisfiability degree of a fuzzy concept.

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.max.max_subsumes_query

Classes

MaxSubsumesQuery	Maximize subsumption query.

Module Contents

class MaxSubsumesQuery(
c1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
type_: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.subsumption_query.SubsumptionQuery*

Maximize subsumption query.

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.min

Submodules

fuzzy_dl_owl2.fuzzydl.query.min.min_instance_query

Classes

MinInstanceQuery	Greatest lower bound of a concept assertion.

Module Contents

class MinInstanceQuery(
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
individual: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.instance_query.InstanceQuery*

Greatest lower bound of a concept assertion.

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.min.min_query

Classes

MinQuery	Minimize expression query.

Module Contents

class MinQuery(*expr:* fuzzy_dl_owl2.fuzzydl.milp.expression.Expression)

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Minimize expression query.

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Solve the query using given knowledge base

obj_expr

fuzzy_dl_owl2.fuzzydl.query.min.min_related_query

Classes

MinRelatedQuery	Greatest lower bound of a role assertion (ind1, ind2, role).

Module Contents

class MinRelatedQuery(
a: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
b: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
role_name: str,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.related_query.RelatedQuery*

Greatest lower bound of a role assertion (ind1, ind2, role).

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

ind1: ***fuzzy_dl_owl2.fuzzydl.individual.individual.Individual***

ind2: ***fuzzy_dl_owl2.fuzzydl.individual.individual.Individual***

role: **str**

fuzzy_dl_owl2.fuzzydl.query.min.min_satisfiable_query

Classes

MinSatisfiableQuery	Minimal satisfiability degree of a fuzzy concept.

Module Contents

class MinSatisfiableQuery(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

class MinSatisfiableQuery(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *a:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual)

Bases: *fuzzy_dl_owl2.fuzzydl.query.satisfiable_query.SatisfiableQuery*

Minimal satisfiability degree of a fuzzy concept.

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.min.min_subsumes_query

Classes

MinSubsumesQuery	Minimize subsumption query.

Module Contents

class MinSubsumesQuery(
c1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
type_: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.subsumption_query.SubsumptionQuery*

Minimize subsumption query.

__str__() → str

Solves the query over a fuzzy KB

preprocess(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

solve(*kb:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

fuzzy_dl_owl2.fuzzydl.query.query

Classes

Query	Helper class that provides a standard way to create an ABC using

Module Contents

class Query

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

abstractmethod __str__() → str

Solves the query over a fuzzy KB

get_total_time() → float

abstractmethod preprocess(*knowledge_base:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → None

Performs some preprocessing steps of the query over a fuzzy KB.

set_initial_time() → None

set_total_time() → None

abstractmethod solve(*knowledge_base:* fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase) → *fuzzy_dl_owl2.fuzzydl.milp.solution.Solution*

Solve the query using given knowledge base

initial_time: int = 0

total_time: int = 0

fuzzy_dl_owl2.fuzzydl.query.related_query

Classes

RelatedQuery	Entailment of a role assertion query

Module Contents

class RelatedQuery

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*, abc.ABC

Entailment of a role assertion query

ind1: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual* **= None**

ind2: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual* **= None**

obj_expr: *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression* **= None**

role: **str = None**

fuzzy_dl_owl2.fuzzydl.query.satisfiable_query

Classes

SatisfiableQuery	Fuzzy concept satisfiability query.

Module Contents

class SatisfiableQuery(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept, *a:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual)

class SatisfiableQuery(*c:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Fuzzy concept satisfiability query.

fuzzy_dl_owl2.fuzzydl.query.subsumption_query

Classes

SubsumptionQuery	Helper class that provides a standard way to create an ABC using

Module Contents

class SubsumptionQuery(
c1: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
c2: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
s_type: fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType,
)

Bases: *fuzzy_dl_owl2.fuzzydl.query.query.Query*

Helper class that provides a standard way to create an ABC using inheritance.

c1: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

c2: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

obj_expr: *fuzzy_dl_owl2.fuzzydl.milp.expression.Expression* **= None**

type: *fuzzy_dl_owl2.fuzzydl.util.constants.LogicOperatorType*

fuzzy_dl_owl2.fuzzydl.range_axiom

Classes

RangeAxiom	Role range axiom

Module Contents

class RangeAxiom(*role: str*, *concept:* fuzzy_dl_owl2.fuzzydl.concept.concept.Concept)

Role range axiom

concept: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

role: **str**

fuzzy_dl_owl2.fuzzydl.relation

Classes

Relation	Represents a role assertion of the form (object individual, role, lower bound for the degree) with respect to a subject individual.

Module Contents

class Relation(
role_name: str,
ind1: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
ind2: fuzzy_dl_owl2.fuzzydl.individual.individual.Individual,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
)

Represents a role assertion of the form (object individual, role, lower bound for the degree) with respect to a subject individual.

__repr__() → str

__str__() → str

clone() → Self

get_degree() → *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

get_name_without_degree() → str

Gets a printable name of the role assertion without the lower bound

get_object_individual() → *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

get_role_name() → str

get_subject_individual() → *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

set_object_individual(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual) → None

set_subject_individual(*ind:* fuzzy_dl_owl2.fuzzydl.individual.individual.Individual) → None

degree: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

ind_a: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

ind_b: *fuzzy_dl_owl2.fuzzydl.individual.individual.Individual*

role_name: str

fuzzy_dl_owl2.fuzzydl.restriction

Submodules

fuzzy_dl_owl2.fuzzydl.restriction.has_value_restriction

Classes

HasValueRestriction	Universal restriction formed by a role, a individual and a lower bound degree.

Module Contents

class HasValueRestriction(*role_name: str*, *individual: str*, *degree:* fuzzy_dl_owl2.fuzzydl.degree.degree.Degree)

Bases: *fuzzy_dl_owl2.fuzzydl.restriction.restriction.Restriction*

Universal restriction formed by a role, a individual and a lower bound degree.

get_individual() → str

get_name_without_degree() → str

Gets the name of the restriction without the degree.

ind_name: str

fuzzy_dl_owl2.fuzzydl.restriction.restriction

Classes

Restriction	Universal restriction formed by a role, a concept and a lower bound degree.

Module Contents

class Restriction(
role_name: str,
concept: fuzzy_dl_owl2.fuzzydl.concept.concept.Concept,
degree: fuzzy_dl_owl2.fuzzydl.degree.degree.Degree,
)

Universal restriction formed by a role, a concept and a lower bound degree.

__repr__() → str

__str__() → str

clone() → Self

get_concept() → *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

get_degree() → *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

get_name_without_degree() → str

Gets the name of the restriction without the degree.

get_role_name() → str

concept: *fuzzy_dl_owl2.fuzzydl.concept.concept.Concept*

degree: *fuzzy_dl_owl2.fuzzydl.degree.degree.Degree*

role_name: str

fuzzy_dl_owl2.fuzzydl.role_parent_with_degree

Classes

RoleParentWithDegree	Pair of elements (role, degree in [0,1]).

Module Contents

class RoleParentWithDegree(*parent: str*, *degree: float*)

Pair of elements (role, degree in [0,1]). Given a role, represents a role parent and the inclusion degree.

get_degree() → float

get_parent() → str

degree: float

parent: str

fuzzy_dl_owl2.fuzzydl.util

Submodules

fuzzy_dl_owl2.fuzzydl.util.config_reader

Classes

ConfigReader

Module Contents

class ConfigReader

static load_parameters(*config_file: str*, *args: list[str]*) → None

ANYWHERE_DOUBLE_BLOCKING: bool = True

ANYWHERE_SIMPLE_BLOCKING: bool = True

DEBUG_PRINT: bool = True

EPSILON: float = 0.001

MAX_INDIVIDUALS: int = -1

MILP_PROVIDER: *fuzzy_dl_owl2.fuzzydl.util.constants.MILPProvider*

NUMBER_DIGITS: int = 2

OPTIMIZATIONS: int = 1

OWL_ANNOTATION_LABEL: str = 'fuzzyLabel'

RULE_ACYCLIC_TBOXES: bool = True

fuzzy_dl_owl2.fuzzydl.util.constants

Attributes

KNOWLEDGE_BASE_SEMANTICS
MAXVAL
MAXVAL2
NUMBER
RESULTS_PATH
SEPARATOR
STAR_SEPARATOR

Classes

BlockingDynamicType	Create a collection of name/value pairs.
ConceptType	Create a collection of name/value pairs.
ConcreteFeatureType	Create a collection of name/value pairs.
CreatedIndividualBlockingType	Create a collection of name/value pairs.
FeatureFunctionType	Create a collection of name/value pairs.
FuzzyDLKeyword	Create a collection of name/value pairs.
FuzzyLogic	Enum where members are also (and must be) strings
InequalityType	Enum where members are also (and must be) strings
KnowledgeBaseRules	Create a collection of name/value pairs.
LogicOperatorType	Create a collection of name/value pairs.
MILPProvider	Enum where members are also (and must be) strings
RepresentativeIndividualType	Create a collection of name/value pairs.
RestrictionType	Create a collection of name/value pairs.
VariableType	Enum where members are also (and must be) strings

Module Contents

class BlockingDynamicType(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

ANYWHERE_DOUBLE_BLOCKING = 6

ANYWHERE_SET_BLOCKING = 5

ANYWHERE_SUBSET_BLOCKING = 4

DOUBLE_BLOCKING = 3

NO_BLOCKING = 0

SET_BLOCKING = 2

SUBSET_BLOCKING = 1

class ConceptType(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

ALL = 7

AND = 0

ATOMIC = 17

AT_LEAST_VALUE = 22

AT_MOST_VALUE = 21

BOTTOM = 20

CHOQUET_INTEGRAL = 49

COMPLEMENT = 18

CONCRETE = 39

CONCRETE_COMPLEMENT = 40

EXACT_VALUE = 23

EXT_NEG_THRESHOLD = 37

EXT_POS_THRESHOLD = 35

FUZZY_NUMBER = 44

FUZZY_NUMBER_COMPLEMENT = 10

GOEDEL_AND = 1

GOEDEL_IMPLIES = 15

GOEDEL_OR = 4

HAS_VALUE = 62

```
LOOSE_LOWER_APPROX = 14

LOOSE_UPPER_APPROX = 13

LOWER_APPROX = 9

LUKASIEWICZ_AND = 2

LUKASIEWICZ_OR = 5

MODIFIED = 41

MODIFIED_COMPLEMENT = 42

NEG_THRESHOLD = 33

NOT_AT_LEAST_VALUE = 25

NOT_AT_MOST_VALUE = 24

NOT_CHOQUET_INTEGRAL = 52

NOT_EXACT_VALUE = 26

NOT_EXT_NEG_THRESHOLD = 38

NOT_EXT_POS_THRESHOLD = 36

NOT_GOEDEL_IMPLIES = 16

NOT_HAS_VALUE = 63

NOT_NEG_THRESHOLD = 34

NOT_OWA = 47

NOT_POS_THRESHOLD = 32

NOT_QUANTIFIED_OWA = 48

NOT_QUASI_SUGENO_INTEGRAL = 54

NOT_SELF = 61

NOT_SIGMA_CONCEPT = 67

NOT_SUGENO_INTEGRAL = 53

NOT_WEIGHTED = 28

NOT_W_MAX = 56

NOT_W_MIN = 58

NOT_W_SUM = 30

NOT_W_SUM_ZERO = 60

NOT_ZADEH_IMPLIES = 65

OR = 3

OWA = 45

POS_THRESHOLD = 31
```

QUANTIFIED_OWA = 46

QUASI_SUGENO_INTEGRAL = 51

SELF = 43

SIGMA_CONCEPT = 66

SOME = 6

SUGENO_INTEGRAL = 50

TIGHT_LOWER_APPROX = 12

TIGHT_UPPER_APPROX = 11

TOP = 19

UPPER_APPROX = 8

WEIGHTED = 27

W_MAX = 55

W_MIN = 57

W_SUM = 29

W_SUM_ZERO = 59

ZADEH_IMPLIES = 64

class ConcreteFeatureType(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

BOOLEAN = 3

INTEGER = 1

REAL = 2

STRING = 0

class CreatedIndividualBlockingType(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

BLOCKED = 0

NOT_BLOCKED = 1

UNCHECKED = 2

class FeatureFunctionType(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

ATOMIC = 0

NUMBER = 1

PRODUCT = 5

SUBTRACTION = 3

SUM = 2

class FuzzyDLKeyword(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__eq__(*value: object*) → bool

__repr__() → str

__str__() → str

get_name() → str

get_value() → pyparsing.CaselessKeyword | pyparsing.Literal

ALL

ALL_INSTANCES_QUERY

AND

BINARY

BNP_QUERY

BOOLEAN

BOTTOM

CHOQUET

CLASSICAL

CONSTRAINTS

CRISP

CRISP_CONCEPT

CRISP_ROLE

DEFINE_CONCEPT

DEFINE_FUZZY_CONCEPT

DEFINE_FUZZY_EQUIVALENCE

DEFINE_FUZZY_LOGIC

DEFINE_FUZZY_NUMBER

DEFINE_FUZZY_NUMBER_RANGE

DEFINE_FUZZY_SIMILARITY

DEFINE_MODIFIER

DEFINE_PRIMITIVE_CONCEPT

DEFINE_TRUTH_CONSTANT

DEFUZZIFY_LOM_QUERY

DEFUZZIFY_MOM_QUERY

DEFUZZIFY_SOM_QUERY

DISJOINT

DISJOINT_UNION

DOMAIN

EQUALS

EQUIVALENT_CONCEPTS

FEATURE_DIV

FEATURE_MUL

FEATURE_SUB

FEATURE_SUM

FREE

FUNCTIONAL

GOEDEL_AND

GOEDEL_IMPLIES

GOEDEL_OR

GREATER_THAN_OR_EQUAL_TO

HAS_VALUE

IMPLIES

IMPLIES_ROLE

INSTANCE

INTEGER

INVERSE

INVERSE_FUNCTIONAL

KLEENE_DIENES_IMPLIES

LEFT_SHOULDER

LESS_THAN_OR_EQUAL_TO

LINEAR

LINEAR_MODIFIER

LOOSE_LOWER_APPROXIMATION

LOOSE_UPPER_APPROXIMATION

LOWER_APPROXIMATION

LUKASIEWICZ

LUKASIEWICZ_AND

LUKASIEWICZ_IMPLIES

LUKASIEWICZ_OR

MAX_G_SUBS_QUERY

MAX_INSTANCE_QUERY

MAX_KD_SUBS_QUERY

MAX_L_SUBS_QUERY

MAX_RELATED_QUERY

MAX_SAT_QUERY

MAX_SUBS_QUERY

MAX_VAR_QUERY

MIN_G_SUBS_QUERY

MIN_INSTANCE_QUERY

MIN_KD_SUBS_QUERY

MIN_L_SUBS_QUERY

MIN_RELATED_QUERY

MIN_SAT_QUERY

MIN_SUBS_QUERY

MIN_VAR_QUERY

MODIFIED

MUL

NOT

OR

OWA

QUASI_SUGENO

Q_OWA

RANGE

REAL

REFLEXIVE

RELATED

RIGHT_SHOULDER

SAT_QUERY

SELF

SHOW_ABSTRACT_FILLERS

SHOW_ABSTRACT_FILLERS_FOR

SHOW_CONCEPTS

SHOW_CONCRETE_FILLERS

SHOW_CONCRETE_FILLERS_FOR

SHOW_CONCRETE_INSTANCE_FOR

SHOW_INSTANCES

SHOW_LANGUAGE

SHOW_VARIABLES

SIGMA_COUNT

SOME

STRING

SUB

SUGENO

SUM

SYMMETRIC

TIGHT_LOWER_APPROXIMATION

TIGHT_UPPER_APPROXIMATION

TOP

TRANSITIVE

TRAPEZOIDAL

TRIANGULAR

TRIANGULAR_MODIFIER

UPPER_APPROXIMATION

W_MAX

W_MIN

W_SUM

W_SUM_ZERO

ZADEH

ZADEH_IMPLIES

class FuzzyLogic

Bases: enum.StrEnum

Enum where members are also (and must be) strings

__repr__() → str

Return repr(self).

__str__() → str

Return str(self).

CLASSICAL = 'classical'

LUKASIEWICZ = 'lukasiewicz'

ZADEH = 'zadeh'

class InequalityType

Bases: enum.StrEnum

Enum where members are also (and must be) strings

__repr__() → str

Return repr(self).

__str__() → str

Return str(self).

EQUAL = '='

GREATER_THAN = '>'

LESS_THAN = '<'

class KnowledgeBaseRules(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

RULE_ATOMIC = 0

RULE_BOTTOM = 11

RULE_CHOQUET_INTEGRAL = 30

RULE_COMPLEMENT = 1

RULE_CONCRETE = 14

RULE_DATATYPE = 18

RULE_FUZZY_NUMBER = 20

RULE_GOEDEL_ALL = 8

RULE_GOEDEL_AND = 2

RULE_GOEDEL_IMPLIES = 12

RULE_GOEDEL_OR = 4

RULE_GOEDEL_SOME = 6

RULE_HAS_VALUE = 44

RULE_LUKASIEWICZ_ALL = 9

RULE_LUKASIEWICZ_AND = 3

RULE_LUKASIEWICZ_OR = 5

RULE_LUKASIEWICZ_SOME = 7

RULE_MODIFIED = 16

RULE_NOT_CHOQUET_INTEGRAL = 31

RULE_NOT_CONCRETE = 15

RULE_NOT_DATATYPE = 19

RULE_NOT_FUZZY_NUMBER = 21

RULE_NOT_GOEDEL_IMPLIES = 13

RULE_NOT_HAS_VALUE = 45

RULE_NOT_MODIFIED = 17

RULE_NOT_OWA = 27

RULE_NOT_QUASI_SUGENO_INTEGRAL = 35

RULE_NOT_SELF = 37

RULE_NOT_SIGMA_COUNT = 49

RULE_NOT_SUGENO_INTEGRAL = 33

RULE_NOT_THRESHOLD = 25

RULE_NOT_WEIGHTED = 23

RULE_NOT_W_MAX = 41

RULE_NOT_W_MIN = 39

RULE_NOT_W_SUM = 29

RULE_NOT_W_SUM_ZERO = 43

RULE_NOT_ZADEH_IMPLIES = 47

RULE_OWA = 26

RULE_QUASI_SUGENO_INTEGRAL = 34

RULE_SELF = 36

RULE_SIGMA_COUNT = 48

RULE_SUGENO_INTEGRAL = 32

RULE_THRESHOLD = 24

RULE_TOP = 10

RULE_WEIGHTED = 22

RULE_W_MAX = 40

RULE_W_MIN = 38

RULE_W_SUM = 28

RULE_W_SUM_ZERO = 42

RULE_ZADEH_IMPLIES = 46

class LogicOperatorType(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

GOEDEL = 1

KLEENE_DIENES = 2

LUKASIEWICZ = 0

ZADEH = 3

class MILPProvider

Bases: enum.StrEnum

Enum where members are also (and must be) strings

static from_str(*value: str*) → Self

GUROBI

MIP

PULP

PULP_CPLEX

PULP_GLPK

PULP_HIGHS

class RepresentativeIndividualType(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

GREATER_EQUAL = 0

LESS_EQUAL = 1

class RestrictionType(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__repr__() → str

__str__() → str

AT_LEAST_VALUE = 1

AT_MOST_VALUE = 0

EXACT_VALUE = 2

class VariableType

Bases: enum.StrEnum

Enum where members are also (and must be) strings

__repr__() → str

Return repr(self).

__str__() → str

Return str(self).

BINARY

CONTINUOUS

INTEGER

SEMI_CONTINUOUS

KNOWLEDGE_BASE_SEMANTICS: *FuzzyLogic*

MAXVAL: float = 2147483647000

MAXVAL2: float = 4294967294000

NUMBER

RESULTS_PATH: str

SEPARATOR: str = '------------------------'

STAR_SEPARATOR: str = '************************'

fuzzy_dl_owl2.fuzzydl.util.util

Attributes

FILENAME
LOG_DIR
TODAY
logger

Classes

Util

Module Contents

class Util

static debug(*message: str*) → None

static error(*message: str*) → None

static has_integer_value(*d: float*) → bool

static info(*message: str*) → None

static log2(*n: float*) → int

static order(*v: list[Any]*) → None

static round(*x: float*) → float

static warning(*message: str*) → None

FILENAME: str

LOG_DIR: str

TODAY: datetime.datetime

logger

fuzzy_dl_owl2.fuzzydl.util.utils

Attributes

FULL_CLASS_DEBUG_PRINT

Functions

class_debugging()	Decorator to wrap all methods of a class using debugging_wrapper.
debugging_wrapper(cls, func)	Debugging wrapper that prints before and after the method call.
recursion_unlimited(func)	

Module Contents

class_debugging()

Decorator to wrap all methods of a class using debugging_wrapper.

debugging_wrapper(*cls*, *func*)

Debugging wrapper that prints before and after the method call.

recursion_unlimited(*func: Callable*)

FULL_CLASS_DEBUG_PRINT: bool = False

7.1.1.2 fuzzy_dl_owl2.fuzzyowl2

7.1.1.2.1 Submodules

fuzzy_dl_owl2.fuzzyowl2.fuzzyowl2

Classes

FuzzyOwl2

Module Contents

class FuzzyOwl2(*input_file: str*, *output_file: str*, *base_iri: str* = *'http://www.semanticweb.org/ontologies/fuzzydl_ontology#'*)

Bases: `object`

get_atomic_concept_name(*c: pyowl2.base.owl_class.OWLClass*) → str

get_atomic_data_property_name(*p: pyowl2.expressions.data_property.OWLDataProperty*) → str

get_atomic_object_property_name(*p: pyowl2.expressions.object_property.OWLObjectProperty*) → str

get_bottom_concept_name() → str

get_bottom_data_property_name() → str

get_bottom_object_property_name() → str

get_class_name(*c: pyowl2.abstracts.class_expression.OWLClassExpression*) → str

get_data_all_values_from_name(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange,
) → str

get_data_exact_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange = *None*,
) → str

get_data_has_value_name(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
literal: pyowl2.literal.literal.OWLLiteral,
) → str

get_data_max_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange = *None*,
) → str

get_data_min_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange = *None*,
) → str

get_data_property_name(*p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression*) → str

get_data_some_values_from_name(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange,
) → str

get_individual_name(*i: pyowl2.abstracts.individual.OWLIndividual*) → str | None

get_object_all_values_from_name(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → str

get_object_complement_of_name(*c: pyowl2.abstracts.class_expression.OWLClassExpression*) → str

get_object_exact_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression = None,
) → str

get_object_has_self_name(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → str

get_object_has_value_name(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
i: pyowl2.abstracts.individual.OWLIndividual,
) → str

get_object_intersection_of_name(*operands: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → str

get_object_max_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression = None,
) → str

get_object_min_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression = None,
) → str

get_object_one_of_name(*ind_set: set[pyowl2.abstracts.individual.OWLIndividual]*) → str

get_object_property_name(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → str

get_object_some_values_from_name(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → str

get_object_union_of_name(*operands: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → str

get_short_name(*e: pyowl2.abstracts.entity.OWLEntity*) → str

get_top_concept_name() → str

get_top_data_property_name() → str

get_top_object_property_name() → str

process_concept_annotations() → None

process_datatype_annotations() → None

process_ontology_annotations() → None

process_ontology_axioms() → None

process_property_annotations() → None

translate_owl2ontology() → None

write_asymmetric_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_choquet_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.choquet_concept.ChoquetConcept) → None

write_concept_assertion_axiom(
i: pyowl2.abstracts.individual.OWLIndividual,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
d: float,
) → None

write_concept_declaration(*c: pyowl2.abstracts.class_expression.OWLClassExpression*) → None

write_data_property_assertion_axiom(
i: pyowl2.abstracts.individual.OWLIndividual,
lit: pyowl2.literal.literal.OWLLiteral,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
d: float,
) → None

write_data_property_declaration(*dp: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression*) → None

write_data_property_domain_axiom(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → None

write_data_property_range_axiom(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange,
) → None

write_different_individuals_axiom(*ind_set: set[pyowl2.abstracts.individual.OWLIndividual]*) → None

write_disjoint_classes_axiom(*class_set: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → None

write_disjoint_data_properties_axiom(
class_set: set[pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression],
) → None

write_disjoint_object_properties_axiom(
class_set: set[pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression],
) → None

write_disjoint_union_axiom(*class_set: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → None

write_equivalent_classes_axiom(*class_set: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → None

write_equivalent_data_properties_axiom(
class_set: set[pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression],
) → None

write_equivalent_object_properties_axiom(
class_set: set[pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression],
) → None

write_functional_data_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_functional_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_fuzzy_logic(*logic: str*) → None

write_fuzzy_nominal_concept_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_nominal_concept.FuzzyNominalConcept,
) → None

write_inverse_functional_object_property_axiom(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
) → None

write_inverse_object_property_axiom(
p1: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
p2: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
) → None

write_irreflexive_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_left_shoulder_function_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.left_shoulder_function.LeftShoulderFunction,
) → None

write_linear_function_definition(*name: str*, *dat:* fuzzy_dl_owl2.fuzzyowl2.owl_types.linear_function.LinearFunction) → None

write_linear_modifier_definition(*name: str*, *mod:* fuzzy_dl_owl2.fuzzyowl2.owl_types.linear_modifier.LinearModifier) → None

write_modified_concept_definition(*name: str*, *dat:* fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_concept.ModifiedConcept) → None

write_modified_function_definition(*name: str*, *dat:* fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_function.ModifiedFunction) → None

write_modified_property_definition(*name: str*, *dat:* fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_property.ModifiedProperty) → None

write_negative_data_property_assertion_axiom(
i: pyowl2.abstracts.individual.OWLIndividual,
lit: pyowl2.literal.literal.OWLLiteral,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
d: float,
) → None

write_negative_object_property_assertion_axiom(
i1: pyowl2.abstracts.individual.OWLIndividual,
i2: pyowl2.abstracts.individual.OWLIndividual,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
d: float,
) → None

write_object_property_assertion_axiom(
i1: pyowl2.abstracts.individual.OWLIndividual,
i2: pyowl2.abstracts.individual.OWLIndividual,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
d: float,
) → None

write_object_property_declaration(*op: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_object_property_domain_axiom(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → None

write_object_property_range_axiom(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → None

write_owa_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.owa_concept.OwaConcept) → None

write_qowa_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.qowa_concept.QowaConcept) → None

write_quasi_sugeno_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.quasi_sugeno_concept.QsugenoConcept,
) → None

write_reflexive_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_right_shoulder_function_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.right_shoulder_function.RightShoulderFunction,
) → None

write_same_individual_axiom(*ind_set: set[pyowl2.abstracts.individual.OWLIndividual]*) → None

write_sub_data_property_of_axiom(
subproperty: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
superproperty: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
d: float,
) → None

write_sub_object_property_of_axiom(
subproperty: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
superproperty: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
d: float,
) → None

write_sub_property_chain_of_axiom(
chain: list[pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression],
superproperty: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
d: float,
) → None

write_subclass_of_axiom(
subclass: pyowl2.abstracts.class_expression.OWLClassExpression,
superclass: pyowl2.abstracts.class_expression.OWLClassExpression,
d: float,
) → None

write_sugeno_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.sugeno_concept.SugenoConcept) → None

write_symmetric_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_transitive_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_trapezoidal_function_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.trapezoidal_function.TrapezoidalFunction,
) → None

write_triangular_function_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.triangular_function.TriangularFunction,
) → None

write_triangular_modifier_definition(
name: str,
mod: fuzzy_dl_owl2.fuzzyowl2.owl_types.triangular_modifer.TriangularModifier,
) → None

write_weighted_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_concept.WeightedConcept) → None

write_weighted_max_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_max_concept.WeightedMaxConcept,
) → None

write_weighted_min_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_min_concept.WeightedMinConcept,
) → None

write_weighted_sum_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_sum_concept.WeightedSumConcept,
) → None

write_weighted_sum_zero_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_sum_zero_concept.WeightedSumZeroConcept,
) → None

NEG_INFINITY: float = -10000.0

POS_INFINITY: float = 10000.0

defined_concepts: dict[str, *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]

defined_properties: dict[str, *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]

fuzzy_datatypes: dict[str, *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]

fuzzy_label: pyowl2.base.annotation_property.OWLAnnotationProperty

fuzzy_modifiers: dict[str, *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]

ontologies: set[pyowl2.ontology.OWLOntology]

ontology: pyowl2.ontology.OWLOntology

ontology_iri

ontology_path

output_dl: str

processed_axioms: set[str]

fuzzy_dl_owl2.fuzzyowl2.fuzzyowl2_to_fuzzydl

Classes

FuzzyOwl2ToFuzzyDL

Module Contents

class FuzzyOwl2ToFuzzyDL(*input_file: str*, *output_file: str*, *base_iri: str* = *'http://www.semanticweb.org/ontologies/fuzzydl_ontology#'*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.fuzzyowl2.FuzzyOwl2*

get_atomic_concept_name(*c: pyowl2.base.owl_class.OWLClass*) → str

get_atomic_data_property_name(*p: pyowl2.expressions.data_property.OWLDataProperty*) → str

get_atomic_object_property_name(*p: pyowl2.expressions.object_property.OWLObjectProperty*) → str

get_bottom_concept_name() → str

get_bottom_data_property_name() → str

get_bottom_object_property_name() → str

get_data_all_values_from_name(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange,
) → str

get_data_exact_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange = None,
) → str

get_data_has_value_name(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
literal: pyowl2.literal.literal.OWLLiteral,
) → str

get_data_max_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange = None,
) → str

get_data_min_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange = None,
) → str

get_data_some_values_from_name(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange,
) → str

get_individual_name(*i: pyowl2.abstracts.individual.OWLIndividual*) → str

get_object_all_values_from_name(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → str

get_object_complement_of_name(*c: pyowl2.abstracts.class_expression.OWLClassExpression*) → str

get_object_exact_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression = None,
) → str

get_object_has_self_name(*p: pyowl2.abstracts.object_property_-expression.OWLObjectPropertyExpression*) → str

get_object_has_value_name(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
i: pyowl2.abstracts.individual.OWLIndividual,
) → str

get_object_intersection_of_name(*operands: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → str

get_object_max_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression = None,
) → str

get_object_min_cardinality_restriction(
cardinality: int,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression = None,
) → str

get_object_one_of_name(*ind_set: set[pyowl2.abstracts.individual.OWLIndividual]*) → str

get_object_some_values_from_name(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → str

get_object_union_of_name(*operands: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → str

get_short_name(*s: pyowl2.abstracts.entity.OWLEntity | str*)

get_top_concept_name() → str

get_top_data_property_name() → str

get_top_object_property_name() → str

static is_reserved_word(*s: str*) → bool

write_asymmetric_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_choquet_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.choquet_concept.ChoquetConcept) → None

write_concept_assertion_axiom(
i: pyowl2.abstracts.individual.OWLIndividual,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
d: float,
) → None

write_concept_declaration(*c: pyowl2.abstracts.class_expression.OWLClassExpression*) → None

write_data_property_assertion_axiom(
i: pyowl2.abstracts.individual.OWLIndividual,
lit: pyowl2.literal.literal.OWLLiteral,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
d: float,
) → None

write_data_property_declaration(*dp: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression*) → None

write_data_property_domain_axiom(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → None

write_data_property_range_axiom(
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
range: pyowl2.abstracts.data_range.OWLDataRange,
) → None

write_different_individuals_axiom(*ind_set: set[pyowl2.abstracts.individual.OWLIndividual]*) → None

write_disjoint_classes_axiom(*class_set: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → None

write_disjoint_data_properties_axiom(
class_set: set[pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression],
) → None

write_disjoint_object_properties_axiom(
class_set: set[pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression],
) → None

write_disjoint_union_axiom(*class_set: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → None

write_equivalent_classes_axiom(*class_set: set[pyowl2.abstracts.class_expression.OWLClassExpression]*) → None

write_equivalent_data_properties_axiom(
class_set: set[pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression],
) → None

write_equivalent_object_properties_axiom(
class_set: set[pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression],
) → None

write_functional_data_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_functional_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_fuzzy_logic(*logic: str*) → None

write_fuzzy_nominal_concept_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_nominal_concept.FuzzyNominalConcept,
) → None

write_inverse_functional_object_property_axiom(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
) → None

write_inverse_object_property_axiom(
p1: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
p2: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
) → None

write_irreflexive_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_left_shoulder_function_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.left_shoulder_function.LeftShoulderFunction,
) → None

write_linear_function_definition(*name: str*, *dat:* fuzzy_dl_owl2.fuzzyowl2.owl_types.linear_function.LinearFunction) → None

write_linear_modifier_definition(*name: str*, *mod:* fuzzy_dl_owl2.fuzzyowl2.owl_types.linear_modifier.LinearModifier) → None

write_modified_concept_definition(*name: str*, *dat:* fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_concept.ModifiedConcept) → None

write_modified_function_definition(*name: str*, *dat:* fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_function.ModifiedFunction) → None

write_modified_property_definition(*name: str*, *dat:* fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_property.ModifiedProperty) → None

write_negative_data_property_assertion_axiom(
i: pyowl2.abstracts.individual.OWLIndividual,
lit: pyowl2.literal.literal.OWLLiteral,
p: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
d: float,
) → None

write_negative_object_property_assertion_axiom(
i1: pyowl2.abstracts.individual.OWLIndividual,
i2: pyowl2.abstracts.individual.OWLIndividual,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
d: float,
) → None

write_object_property_assertion_axiom(
i1: pyowl2.abstracts.individual.OWLIndividual,
i2: pyowl2.abstracts.individual.OWLIndividual,
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
d: float,
) → None

write_object_property_declaration(*op: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_object_property_domain_axiom(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → None

write_object_property_range_axiom(
p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
c: pyowl2.abstracts.class_expression.OWLClassExpression,
) → None

write_owa_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.owa_concept.OwaConcept) → None

write_qowa_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.qowa_concept.QowaConcept) → None

write_quasi_sugeno_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.quasi_sugeno_concept.QsugenoConcept,
) → None

write_reflexive_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_right_shoulder_function_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.right_shoulder_function.RightShoulderFunction,
) → None

write_same_individual_axiom(*ind_set: set[pyowl2.abstracts.individual.OWLIndividual]*) → None

write_sub_data_property_of_axiom(
subproperty: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
superproperty: pyowl2.abstracts.data_property_expression.OWLDataPropertyExpression,
d: float,
) → None

write_sub_object_property_of_axiom(
subproperty: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
superproperty: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
d: float,
) → None

write_sub_property_chain_of_axiom(
chain: list[pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression],
superproperty: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression,
d: float,
) → None

write_subclass_of_axiom(
subclass: pyowl2.abstracts.class_expression.OWLClassExpression,
superclass: pyowl2.abstracts.class_expression.OWLClassExpression,
d: float,
) → None

write_sugeno_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.sugeno_concept.SugenoConcept) → None

write_symmetric_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_transitive_object_property_axiom(*p: pyowl2.abstracts.object_property_expression.OWLObjectPropertyExpression*) → None

write_trapezoidal_function_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.trapezoidal_function.TrapezoidalFunction,
) → None

write_triangular_function_definition(
name: str,
dat: fuzzy_dl_owl2.fuzzyowl2.owl_types.triangular_function.TriangularFunction,
) → None

write_triangular_modifier_definition(
name: str,
mod: fuzzy_dl_owl2.fuzzyowl2.owl_types.triangular_modifer.TriangularModifier,
) → None

write_weighted_concept_definition(*name: str*, *c:* fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_concept.WeightedConcept) → None

write_weighted_max_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_max_concept.WeightedMaxConcept,
) → None

write_weighted_min_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_min_concept.WeightedMinConcept,
) → None

write_weighted_sum_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_sum_concept.WeightedSumConcept,
) → None

write_weighted_sum_zero_concept_definition(
name: str,
c: fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_sum_zero_concept.WeightedSumZeroConcept,
) → None

DOUBLE_MAX_VALUE: float

DOUBLE_MIN_VALUE: float

EPSILON: float = 0.001

INTEGER_MAX_VALUE: int = 100000000

INTEGER_MIN_VALUE: int = -100000000

boolean_datatypes: set[str]

data_properties: set[str]

numerical_datatypes: set[str]

object_properties: set[str]

processed_functional_data_properties: set[str]

processed_functional_object_properties: set[str]

string_datatypes: set[str]

fuzzy_dl_owl2.fuzzyowl2.owl_types

Submodules

fuzzy_dl_owl2.fuzzyowl2.owl_types.choquet_concept

Classes

ChoquetConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class ChoquetConcept(*weights: list[float]*, *concepts: list[str]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_concepts() → list[str]

get_weights() → list[float]

fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition

Classes

ConceptDefinition	Helper class that provides a standard way to create an ABC using

Module Contents

class ConceptDefinition(*type:* fuzzy_dl_owl2.fuzzyowl2.util.constants.ConceptType)

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

__repr__() → str

get_type() → *fuzzy_dl_owl2.fuzzyowl2.util.constants.ConceptType*

fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_datatype

Classes

FuzzyDatatype	Helper class that provides a standard way to create an ABC using

Module Contents

class FuzzyDatatype

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

__repr__() → str

get_max_value() → float

get_min_value() → float

set_max_value(*max: float*) → None

set_min_value(*min: float*) → None

fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_modifier

Classes

FuzzyModifier	Helper class that provides a standard way to create an ABC using

Module Contents

class FuzzyModifier

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_nominal_concept

Classes

FuzzyNominalConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class FuzzyNominalConcept(*n: float*, *i: str*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_degree() → float

get_individual() → str

fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_property

Classes

FuzzyProperty	Helper class that provides a standard way to create an ABC using

Module Contents

class FuzzyProperty

Bases: abc.ABC

Helper class that provides a standard way to create an ABC using inheritance.

fuzzy_dl_owl2.fuzzyowl2.owl_types.left_shoulder_function

Classes

LeftShoulderFunction	Helper class that provides a standard way to create an ABC using

Module Contents

class LeftShoulderFunction(*a: float*, *b: float*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_datatype.FuzzyDatatype*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_a() → float

get_b() → float

fuzzy_dl_owl2.fuzzyowl2.owl_types.linear_function

Classes

LinearFunction	Helper class that provides a standard way to create an ABC using

Module Contents

class LinearFunction(*a: float*, *b: float*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_datatype.FuzzyDatatype*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_a() → float

get_b() → float

fuzzy_dl_owl2.fuzzyowl2.owl_types.linear_modifier

Classes

LinearModifier	Helper class that provides a standard way to create an ABC using

Module Contents

class LinearModifier(*c: float*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_modifier.FuzzyModifier*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_c() → float

fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_concept

Classes

ModifiedConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class ModifiedConcept(*mod: str*, *c: str*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_fuzzy_concept() → str

get_fuzzy_modifier() → str

fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_function

Classes

ModifiedFunction	Helper class that provides a standard way to create an ABC using

Module Contents

class ModifiedFunction(*mod: str*, *d: str*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_datatype.FuzzyDatatype*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_d() → str

get_mod() → str

fuzzy_dl_owl2.fuzzyowl2.owl_types.modified_property

Classes

ModifiedProperty	Helper class that provides a standard way to create an ABC using

Module Contents

class ModifiedProperty(*mod: str*, *prop: str*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_property.FuzzyProperty*

Helper class that provides a standard way to create an ABC using inheritance.

__repr__() → str

__str__() → str

get_fuzzy_modifier() → str

get_property() → str

fuzzy_dl_owl2.fuzzyowl2.owl_types.owa_concept

Classes

OwaConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class OwaConcept(*weights: list[float]*, *concepts: list[str]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_concepts() → list[str]

get_weights() → list[float]

fuzzy_dl_owl2.fuzzyowl2.owl_types.property_definition

Classes

PropertyDefinition

Module Contents

class PropertyDefinition(*mod: str*, *prop: str*)

get_fuzzy_modifier() → str

get_property() → str

fuzzy_dl_owl2.fuzzyowl2.owl_types.qowa_concept

Classes

QowaConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class QowaConcept(*q: str*, *concepts: list[str]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_concepts() → list[str]

get_quantifier() → str

fuzzy_dl_owl2.fuzzyowl2.owl_types.quasi_sugeno_concept

Classes

QsugenoConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class QsugenoConcept(*weights: list[float]*, *concepts: list[str]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_concepts() → list[str]

get_weights() → list[float]

fuzzy_dl_owl2.fuzzyowl2.owl_types.right_shoulder_function

Classes

RightShoulderFunction	Helper class that provides a standard way to create an ABC using

Module Contents

class RightShoulderFunction(*a: float*, *b: float*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_datatype.FuzzyDatatype*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_a() → float

get_b() → float

fuzzy_dl_owl2.fuzzyowl2.owl_types.sugeno_concept

Classes

SugenoConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class SugenoConcept(*weights: list[float]*, *concepts: list[str]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_concepts() → list[str]

get_weights() → list[float]

fuzzy_dl_owl2.fuzzyowl2.owl_types.trapezoidal_function

Classes

TrapezoidalFunction	Helper class that provides a standard way to create an ABC using

Module Contents

class TrapezoidalFunction(*a: float*, *b: float*, *c: float*, *d: float*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_datatype.FuzzyDatatype*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_a() → float

get_b() → float

get_c() → float

get_d() → float

fuzzy_dl_owl2.fuzzyowl2.owl_types.triangular_function

Classes

TriangularFunction	Helper class that provides a standard way to create an ABC using

Module Contents

class TriangularFunction(*a: float*, *b: float*, *c: float*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_datatype.FuzzyDatatype*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_a() → float

get_b() → float

get_c() → float

fuzzy_dl_owl2.fuzzyowl2.owl_types.triangular_modifer

Classes

TriangularModifier	Helper class that provides a standard way to create an ABC using

Module Contents

class TriangularModifier(*a: float*, *b: float*, *c: float*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.fuzzy_modifier.FuzzyModifier*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_a() → float

get_b() → float

get_c() → float

fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_concept

Classes

WeightedConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedConcept(*n: float*, *c: str*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_fuzzy_concept() → str

get_number() → float

fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_max_concept

Classes

WeightedMaxConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedMaxConcept(*wc: list[*fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_weighted_concepts() → list[*fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]

fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_min_concept

Classes

WeightedMinConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedMinConcept(*wc: list[*fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_weighted_concepts() → list[*fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]

fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_sum_concept

Classes

WeightedSumConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedSumConcept(*wc: list[*fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_weighted_concepts() → list[*fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]

fuzzy_dl_owl2.fuzzyowl2.owl_types.weighted_sum_zero_concept

Classes

WeightedSumZeroConcept	Helper class that provides a standard way to create an ABC using

Module Contents

class WeightedSumZeroConcept(*wc: list[*fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]*)

Bases: *fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*

Helper class that provides a standard way to create an ABC using inheritance.

__str__() → str

get_weighted_concepts() → list[*fuzzy_dl_owl2.fuzzyowl2.owl_types.concept_definition.ConceptDefinition*]

fuzzy_dl_owl2.fuzzyowl2.parser

Submodules

fuzzy_dl_owl2.fuzzyowl2.parser.owl2_parser

Classes

FuzzyOwl2Parser

Module Contents

class FuzzyOwl2Parser

Bases: object

static get_grammatics() → pyparsing.ParserElement

This function generate the grammatics to parse the predicate wih formula "formula".

Parameters
formula (= *The predicate formula used for the parsing.*)

Return type
The parsed result given by pyparsing.

static load_config(**args*) → None

static main(*annotation: str*, **args*,) → tuple[*fuzzy_dl_owl2.fuzzydl.knowledge_base.KnowledgeBase*, list[*fuzzy_dl_owl2.fuzzydl.query.query.Query*]]

static parse_string(*instring: str*, *parse_all: bool = False*, ** (Keyword-only parameters separator (PEP 3102))*, *parseAll: bool = False*) → pyparsing.ParseResults

fuzzy_dl_owl2.fuzzyowl2.parser.owl2_xml_parser

Classes

FuzzyOwl2XMLParser

Module Contents

class FuzzyOwl2XMLParser

Bases: object

static get_caseless_attrib(*attrib: dict[str, str]*, *key: str*) → str | None

static load_config(**args*) → None

static main(*annotation: str*, **args*) → Any

static parse_string(*instring: str*) → pyparsing.ParseResults

fuzzy_dl_owl2.fuzzyowl2.util

Submodules

fuzzy_dl_owl2.fuzzyowl2.util.constants

Classes

ConceptType	Enum where members are also (and must be) strings
FuzzyOWL2Keyword	Create a collection of name/value pairs.

Module Contents

class ConceptType

Bases: enum.StrEnum

Enum where members are also (and must be) strings

__repr__() → str

Return repr(self).

__str__() → str

Return str(self).

CHOQUET

FUZZY_NOMINAL

MODIFIED_CONCEPT

OWA

QUANTIFIED_OWA

QUASI_SUGENO

SUGENO

WEIGHTED_CONCEPT

WEIGHTED_MAX

WEIGHTED_MIN

WEIGHTED_SUM

WEIGHTED_SUM_ZERO

class FuzzyOWL2Keyword(**args*, ***kwds*)

Bases: enum.Enum

Create a collection of name/value pairs.

Example enumeration:

```
>>> class Color(Enum):
...     RED = 1
...     BLUE = 2
...     GREEN = 3
```

Access them by:

- attribute access:

```
>>> Color.RED
<Color.RED: 1>
```

- value lookup:

```
>>> Color(1)
<Color.RED: 1>
```

- name lookup:

```
>>> Color['RED']
<Color.RED: 1>
```

Enumerations can be iterated over, and know how many members they have:

```
>>> len(Color)
3
```

```
>>> list(Color)
[<Color.RED: 1>, <Color.BLUE: 2>, <Color.GREEN: 3>]
```

Methods can be added to enumerations, and members can have their own attributes – see the documentation for details.

__eq__(*value: object*) → bool

__repr__() → str

__str__() → str

get_name() → str

get_str_value() → str

get_tag_name() → str

get_value() → pyparsing.CaselessKeyword | pyparsing.Word

A

AXIOM

B

BASE

C

CHOQUET

CLOSE_TAG

CONCEPT

CONCEPT_NAMES

CRISP

D

DATATYPE

DEGREE_DEF

DEGREE_VALUE

EQUAL

FUZZY_LABEL

FUZZY_LOGIC

FUZZY_OWL_2

FUZZY_TYPE

GEQ

GOEDEL

GRE

INDIVIDUAL

LEFT_SHOULDER

LEQ

LES

LINEAR

LOGIC

LUKASIEWICZ

MODIFIED

MODIFIER

NAME

NOMINAL

ONTOLOGY

OPEN_TAG

OWA

PRODUCT

QUANTIFIER

QUASI_SUGENO

Q_OWA

RIGHT_SHOULDER

ROLE

SINGLE_CLOSE_TAG

SLASH

SUGENO

TRAPEZOIDAL

TRIANGULAR

TYPE

WEIGHT

WEIGHTED

WEIGHTED_MAXIMUM

WEIGHTED_MINIMUM

WEIGHTED_SUM

WEIGHTED_SUMZERO

WEIGHTS

ZADEH

fuzzy_dl_owl2.fuzzyowl2.util.fuzzy_xml

Classes

FuzzyXML

Module Contents

class FuzzyXML

Bases: `object`

static build_concept_xml(*concept_type: str*, *attrib: dict[str, str] = dict()*) → xml.etree.ElementTree.Element

static build_datatype_xml(*datatype_type: str*, *attrib: dict[str, str] = dict()*) → xml.etree.ElementTree.Element

static build_degree_xml(*value: int | float*, *attrib: dict[str, str] = dict()*) → xml.etree.ElementTree.Element

static build_logic_xml(*logic: str*, *attrib: dict[str, str] = dict()*) → xml.etree.ElementTree.Element

static build_main_xml(*fuzzy_type: str*) → xml.etree.ElementTree.Element

static build_modifier_xml(*modifier_type: str*, *attrib: dict[str, str] = dict()*) → xml.etree.ElementTree.Element

static build_names_xml(*concepts: list[pyowl2.abstracts.class_expression.OWLClassExpression]*) → xml.etree.ElementTree.Element

static build_weights_xml(*weights: list[float]*) → xml.etree.ElementTree.Element

static to_str(*element: xml.etree.ElementTree.Element*) → str

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z