FunCs

Florin Leon

Version 1.0

23 February 2019

Table of Contents

Namespace Index

Packages

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

Hierarchical Index

Class Hierarchy

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

Class Index

Class List

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

Namespace Documentation

ExtensionMethods Namespace Reference

Classes

- class **IEnumerable<double>**
- class **IEnumerable<int>**
- class **IEnumerable<OptionF<T>>**
- class **IEnumerable<string>**
- class **string**

FunCs Namespace Reference

Classes

- struct **DoubleF**
  Represents an immutable double-precision floating-point number.
- class **ExpertMatchF**
  The class that implements different pattern matching cases on single lists or lists of facts. General pattern matching is performed using the Rete algorithm implemented in the Clips expert system tool.
- class **ExpertMatchF<T>**
  The class that implements pattern matching on options.
- struct **IntF**
  Represents an immutable 32-bit signed integer.
- class **OptionF<T>**
  The option type is used when an actual value may not exist. An option has an underlying type and can hold a value of that type, i.e. Some(value), or it may contain no value, i.e. None.

Class Documentation

FunCs.DoubleF Struct Reference

Represents an immutable double-precision floating-point number.

Public Member Functions

- **DoubleF** (double value)
  Initializes a new instance of the ***DoubleF*** *structure.*

- override string **ToString** ()
  Converts the numeric value of this instance to its equivalent string representation.

Static Public Member Functions

- static implicit **operator double** (**DoubleF** d)
  DoubleF *can be implicitly converted to double so that it can be used for all the operations defined for double.*

Properties

- double **Value** `[get]`
  The read-only double value.

Detailed Description

Represents an immutable double-precision floating-point number.

Constructor & Destructor Documentation

FunCs.DoubleF.DoubleF (double *value*)

Initializes a new instance of the **DoubleF** structure.

Parameters:

value	The real value used for initialization

Member Function Documentation

static implicit FunCs.DoubleF.operator double (DoubleF *d*)`[static]`

DoubleF can be implicitly converted to double so that it can be used for all the operations defined for double.

override string FunCs.DoubleF.ToString ()

Converts the numeric value of this instance to its equivalent string representation.

Property Documentation

double FunCs.DoubleF.Value `[get]`

The read-only double value.

FunCs.ExpertMatchF Class Reference

The class that implements different pattern matching cases on single lists or lists of facts. General pattern matching is performed using the Rete algorithm implemented in the Clips expert system tool.

Public Member Functions

- **ExpertMatchF** (string list)
  Initializes a new instance of the ***ExpertMatchF*** *class.*

- **ExpertMatchF** (List< string > factList)
  Initializes a new instance of the ***ExpertMatchF*** *class.*

- bool **MatchListEmpty** ()
  Returns true if the list is empty.

- bool **MatchListHeadTail** (out string head, out List< string > tail)
  Identifies the first item of the list and the rest of the list. It returns false if the list is empty.

- bool **MatchListGeneral** (string pattern, out Dictionary< string, string > results)
  Matches an arbitrary pattern on the list of facts. It returns false if the pattern cannot be matched on the facts.

- bool **MatchMultiple** (List< string > patterns, out List< Dictionary< string, string >> results)
  Matches arbitrary patterns on the list of facts. It returns false if the patterns cannot be matched on the facts.

- bool **MatchMultiple** (List< string > patterns, string constraints, out List< Dictionary< string, string >> results)
  Matches arbitrary patterns on the list of facts. It returns false if the pattern cannot be matched on the facts.

Detailed Description

The class that implements different pattern matching cases on single lists or lists of facts. General pattern matching is performed using the Rete algorithm implemented in the Clips expert system tool.

Constructor & Destructor Documentation

FunCs.ExpertMatchF.ExpertMatchF (string *list*)

Initializes a new instance of the **ExpertMatchF** class.

Parameters:

list	A list of string items that will be used for pattern matching. To increase clarity and to stress that it is a list, it is recommended to enclose the items within square

	brackets

FunCs.ExpertMatchF.ExpertMatchF (List< string > *factList*)

Initializes a new instance of the **ExpertMatchF** class.

Parameters:

factList	A list of facts that will be used for pattern matching, where each fact is a list of string items. The facts are general and should not contain square brackets.

Member Function Documentation

bool FunCs.ExpertMatchF.MatchListEmpty ()

Returns true if the list is empty.

bool FunCs.ExpertMatchF.MatchListHeadTail (out string *head*, out List< string > *tail*)

Identifies the first item of the list and the rest of the list. It returns false if the list is empty.

Parameters:

head	The head of the list, i.e. the first item
tail	The rest of the list, starting with the second item

bool FunCs.ExpertMatchF.MatchListGeneral (string *pattern*, out Dictionary< string, string > *results*)

Matches an arbitrary pattern on the list of facts. It returns false if the pattern cannot be matched on the facts.

Parameters:

pattern	A pattern containing items to be matched and variables, identified by ? for a single word or $? for multiple words, e.g. ?a or $?b. At least one variable must be named in the pattern
results	A dictionary that contains the values of the variables, e.g. results["?a"] contains the value of that variable after the pattern matching

bool FunCs.ExpertMatchF.MatchMultiple (List< string > *patterns*, out List< Dictionary< string, string >> *results*)

Matches arbitrary patterns on the list of facts. It returns false if the patterns cannot be matched on the facts.

Parameters:

patterns	A list of patterns containing items to be matched and variables, identified by ? for a single word or $? for multiple words, e.g. ?a or $?b. At least one variable must be named in the patterns
results	A dictionary that contains the values of the variables, e.g. results["?a"] contains the value of that variable after the pattern matching

bool FunCs.ExpertMatchF.MatchMultiple (List< string > *patterns*, string *constraints*, out List< Dictionary< string, string >> *results*)

Matches arbitrary patterns on the list of facts. It returns false if the pattern cannot be matched on the facts.

Parameters:

patterns	A list of patterns containing items to be matched and variables, identified by ? for a single word or $? for multiple words, e.g. ?a or $?b. At least one variable must be named in the patterns
constraints	A logical expression which contains the conditions that the matched variable values must satisfy. The Clips language syntax and operators are used to describe the constraints
results	A dictionary that contains the values of the variables, e.g. results["?a"] contains the value of that variable after the pattern matching

FunCs.ExpertMatchF<T> Class Reference

The class that implements pattern matching on options.

Public Member Functions

- **ExpertMatchF** (OptionF< T > option)
  *Initializes a new instance of the **ExpertMatchF** class.*

- bool **MatchOptionSome** (out T some)
  Returns true if the option contains a value and false otherwise.

- bool **MatchOptionNone** ()
  Returns true if the option does not contain a value and false otherwise.

Detailed Description

The class that implements pattern matching on options.

Member Function Documentation

FunCs.ExpertMatchF<T>.ExpertMatchF (OptionF< T > *option*)

Initializes a new instance of the **ExpertMatchF** class.

Parameters:

option	An option object that will be used for pattern matching

bool FunCs.ExpertMatchF<T>.MatchOptionSome (out T *some*)

Returns true if the option contains a value and false otherwise.

Parameters:

some	The value of the option

bool FunCs.ExpertMatchF<T>.MatchOptionNone ()

Returns true if the option does not contain a value and false otherwise.

ExtensionMethods.IEnumerable<double> Class Reference

Public Member Functions

- **string ToStringF** (int noDecimals=3)
  Converts the collection of doubles to a string representation.

- bool **MatchF** (out double head, out IEnumerable< double > tail)
  Identifies the first item of the list and the rest of the list. It returns false if the list is empty.

Member Function Documentation

string ExtensionMethods.IEnumerable<double>.ToStringF (int *noDecimals* = 3)

Converts the collection of doubles to a string representation.

Parameters:

noDecimals	The number of decimals places to be used when formatting the collection elements

bool ExtensionMethods.IEnumerable<double>.MatchF (out double *head*, out IEnumerable< double > *tail*)

Identifies the first item of the list and the rest of the list. It returns false if the list is empty.

Parameters:

head	The head of the list, i.e. the first item
tail	The rest of the list, starting with the second item

ExtensionMethods.IEnumerable<int> Class Reference

Public Member Functions

- **string ToStringF** ()
  Converts the collection of integers to a string representation.

- bool **MatchF** (out int head, out IEnumerable< int > tail)
  Identifies the first item of the list and the rest of the list. It returns false if the list is empty.

Member Function Documentation

string ExtensionMethods.IEnumerable<int>.ToStringF ()

Converts the collection of integers to a string representation.

bool ExtensionMethods.IEnumerable<int>.MatchF (out int *head*, out IEnumerable< int > *tail*)

Identifies the first item of the list and the rest of the list. It returns false if the list is empty.

Parameters:

head	The head of the list, i.e. the first item
tail	The rest of the list, starting with the second item

ExtensionMethods.IEnumerable<OptionF<T>> Class Reference

Public Member Functions

- IEnumerable< T > **WhereSome< T >** ()
  Filters a sequence of OptionF objects and returns the list of values of the objects which are Some.

Member Function Documentation

IEnumerable<T> ExtensionMethods.IEnumerable<OptionF<T>>.WhereSome< T > ()

Filters a sequence of OptionF objects and returns the list of values of the objects which are Some.

ExtensionMethods.IEnumerable<string> Class Reference

Public Member Functions

- **string ToStringF** ()
  Converts the collection of strings to a string representation.

- bool **MatchF** (out **string** head, out IEnumerable< **string** > tail)
  Identifies the first item of the list and the rest of the list. It returns false if the list is empty.

- bool **MatchF** (**string** pattern)
  Matches an empty pattern on the list.

- bool **MatchF** (**string** pattern, out **string** var1)
  Matches an arbitrary pattern on the list. It returns false if the pattern cannot be matched.

- bool **MatchF** (**string** pattern, out **string** var1, out **string** var2)
  Matches an arbitrary pattern on the list. It returns false if the pattern cannot be matched.

- bool **MatchF** (**string** pattern, out **string** var1, out **string** var2, out **string** var3)
  Matches an arbitrary pattern on the list. It returns false if the pattern cannot be matched.

- bool **MatchF** (**string** pattern, out **string** var1, out **string** var2, out **string** var3, out **string** var4)
  Matches an arbitrary pattern on the list. It returns false if the pattern cannot be matched.

Member Function Documentation

string ExtensionMethods.IEnumerable<string>.ToStringF ()

Converts the collection of strings to a string representation.

bool ExtensionMethods.IEnumerable<string>.MatchF (out string *head*, out IEnumerable< string > *tail*)

Identifies the first item of the list and the rest of the list. It returns false if the list is empty.

Parameters:

head	The head of the list, i.e. the first item
tail	The rest of the list, starting with the second item

bool ExtensionMethods.IEnumerable<string>.MatchF (string *pattern*)

Matches an empty pattern on the list.

Parameters:

pattern	A pattern containing items to be matched

bool ExtensionMethods.IEnumerable<string>.MatchF (string *pattern*, out string *var1*)

Matches an arbitrary pattern on the list. It returns false if the pattern cannot be matched.

Parameters:

pattern	A pattern with one variable
var1	The value of the first variable after pattern matching

bool ExtensionMethods.IEnumerable<string>.MatchF (string *pattern*, out string *var1*, out string *var2*)

Matches an arbitrary pattern on the list. It returns false if the pattern cannot be matched.

Parameters:

pattern	A pattern with two variables
var1	The value of the first variable after pattern matching
var2	The value of the second variable after pattern matching

bool ExtensionMethods.IEnumerable<string>.MatchF (string *pattern*, out string *var1*, out string *var2*, out string *var3*)

Matches an arbitrary pattern on the list. It returns false if the pattern cannot be matched.

Parameters:

pattern	A pattern with three variables
var1	The value of the first variable after pattern matching
var2	The value of the second variable after pattern matching
var3	The value of the third variable after pattern matching

bool ExtensionMethods.IEnumerable<string>.MatchF (string *pattern*, out string *var1*, out string *var2*, out string *var3*, out string *var4*)

Matches an arbitrary pattern on the list. It returns false if the pattern cannot be matched.

Parameters:

pattern	A pattern with four variables
var1	The value of the first variable after pattern matching
var2	The value of the second variable after pattern matching
var3	The value of the third variable after pattern matching
var4	The value of the fourth variable after pattern matching

FunCs.IntF Struct Reference

Represents an immutable 32-bit signed integer.

Public Member Functions

- **IntF** (int value)
  *Initializes a new instance of the **IntF** structure.*

- override string **ToString** ()
  Converts the numeric value of this instance to its equivalent string representation.

Static Public Member Functions

- static implicit **operator int** (**IntF** i)
  ***IntF** can be implicitly converted to int so that it can be used for all the operations defined for int.*

Properties

- int **Value** `[get]`
  The read-only int value.

Detailed Description

Represents an immutable 32-bit signed integer.

Constructor & Destructor Documentation

FunCs.IntF.IntF (int *value*)

Initializes a new instance of the **IntF** structure.

Parameters:

value	The integer value used for initialization

Member Function Documentation

static implicit FunCs.IntF.operator int (IntF *i*) `[static]`

IntF can be implicitly converted to int so that it can be used for all the operations defined for int.

override string FunCs.IntF.ToString ()

Converts the numeric value of this instance to its equivalent string representation.

Property Documentation

int FunCs.IntF.Value `[get]`

The read-only int value.

FunCs.OptionF<T> Class Reference

The option type is used when an actual value may not exist. An option has an underlying type and can hold a value of that type, i.e. Some(value), or it may contain no value, i.e. None.
Inherits IEnumerable<T>.

Public Member Functions

- IEnumerator< T > **GetEnumerator** ()
  Returns an enumerator that iterates through the collection.

- override string **ToString** ()
  Converts the option to a string representation: Some(value) or None.

- override bool **Equals** (object obj)
  Determines whether the specified option object is equal to the current option object.

- OptionF< R > **Select< R >** (Func< T, R > f)
  Projects the current type of option into a new type of option. An equivalent name in other functional programming languages is Map.

- OptionF< R > **SelectMany< R >** (Func< T, OptionF< R >> f)
  Projects the current type of option into a new type of option and flattens the result. An equivalent name in other functional programming languages is Bind.

- override int **GetHashCode** ()
  Returns the hash code of the current option object.

- bool **MatchSomeF< T >** (out T some)
  Returns true if the option contains a value and false otherwise.

- bool **MatchNoneF< T >** ()
  Returns true if the option does not contain a value and false otherwise.

Static Public Member Functions

- static OptionF< T > **Some** (T value)
  Creates an option that has a given value.

- static OptionF< T > **None** ()
  Creates an option with no value.

Properties

- bool **IsSome** `[get]`
  Returns true if the option has a value and false if it has no value.

- bool **IsNone** `[get]`
  Returns true if the option has no value and false if it has a value.

- T **Value** `[get]`
  Returns the value of the option. It throws an exception if the option has no value.

Detailed Description

The option type is used when an actual value may not exist. An option has an underlying type and can hold a value of that type, i.e. Some(value), or it may contain no value, i.e. None.

Member Function Documentation

static OptionF<T> FunCs.OptionF<T>.Some (T *value*) `[static]`

Creates an option that has a given value.

Parameters:

value	A non-null value

static OptionF<T> FunCs.OptionF<T>.None () `[static]`

Creates an option with no value.

IEnumerator<T> FunCs.OptionF<T>.GetEnumerator ()

Returns an enumerator that iterates through the collection.

override string FunCs.OptionF<T>.ToString ()

Converts the option to a string representation: Some(value) or None.

override bool FunCs.OptionF<T>.Equals (object *obj*)

Determines whether the specified option object is equal to the current option object.

OptionF<R> FunCs.OptionF<T>.Select< R > (Func< T, R > *f*)

Projects the current type of option into a new type of option. An equivalent name in other functional programming languages is Map.

Template Parameters:

R	The type of the option returned by the transform function.

Parameters:

f	A transform function to apply to the current option.

OptionF<R> FunCs.OptionF<T>.SelectMany< R > (Func< T, OptionF< R >> *f*)

Projects the current type of option into a new type of option and flattens the result. An equivalent name in other functional programming languages is Bind.

Template Parameters:

R	The type of the option returned by the transform function.

Parameters:

f	A transform function to apply to the current option.

override int FunCs.OptionF<T>.GetHashCode ()

Returns the hash code of the current option object.

bool FunCs.OptionF<T>.MatchSomeF< T > (out T *some*)

Returns true if the option contains a value and false otherwise.

Parameters:

some	The value of the option

bool FunCs.OptionF<T>.MatchNoneF< T > ()

Returns true if the option does not contain a value and false otherwise.

Property Documentation

bool FunCs.OptionF<T>.IsSome `[get]`

Returns true if the option has a value and false if it has no value.

bool FunCs.OptionF<T>.IsNone `[get]`

Returns true if the option has no value and false if it has a value.

T FunCs.OptionF<T>.Value `[get]`

Returns the value of the option. It throws an exception if the option has no value.

ExtensionMethods.string Class Reference

Public Member Functions

- IEnumerable< int > **ToIntEnumF** ()
- IEnumerable< double > **ToDoubleEnumF** ()
  Converts a string that represents a list of real numbers into the corresponding IEnumerable(double).

- IEnumerable< **string** > **ToStringEnumF** (char separator=' ')
  Converts a string that represents a list of strings into the corresponding IEnumerable(string).

Member Function Documentation

IEnumerable<int> ExtensionMethods.string.ToIntEnumF ()

Converts a string that represents a list of integers into the corresponding IEnumerable(int).

IEnumerable<double> ExtensionMethods.string.ToDoubleEnumF ()

Converts a string that represents a list of real numbers into the corresponding IEnumerable(double).

IEnumerable<string> ExtensionMethods.string.ToStringEnumF (char *separator* = ' ')

Converts a string that represents a list of strings into the corresponding IEnumerable(string).

Parameters:

separator	A separator used to split the list

Index