Algebra

Release 0.1

SymCollab

Nov 27, 2020

CONTENTS:

CHAPTER

ONE

TERM LIBRARY

The term library is responsible for the creation of fundamental algebraic structures such as variables, constants, and functions and their combinations to make terms.

This library also contains helper functions that can be useful in algorithms that operate on terms.

class algebra.term.**Constant**(*symbol: str*, *sort: Optional[*algebra.term.Sort*] = None*)

A symbolic representation of a constant.

Parameters

- **symbol** (*str*) – The name of the constant.
- **sort** (*AnySort*) – The sort in which the constant belongs to.

Notes

For historical reasons, a constant is represented as a zero-arity FuncTerm.

Examples

```
>>> from algebra import Constant
>>> a = Constant("a")
>>> a
a
```

class algebra.term.**Equation**(*l: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *r: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*)

A structure to hold a unification problem.

Parameters

- **l** (*Term*) – The left hand side of the equation.
- **r** (*Term*) – The right hand side of the equation.

Examples

```
>>> from algebra import *
>>> x = Variable("x")
>>> a = Constant("a")
>>> Equation(x, a)
x = a
```

class algebra.term.**FuncTerm**(*function:* algebra.term.Function, *args*)

A symbolic representation of the instantiation of a function.

Parameters

- **function** (Function) – The function in which the FuncTerm was instantiated from.
- **args** (*{Variable, Constant, FuncTerm}*) – The arguments of the function.

Examples

```
>>> from algebra import *
>>> f = Function("f", 1)
>>> a = Constant("a")
>>> f(a)
f(a)
```

class algebra.term.**Function**(*symbol: str*, *arity: int*, *domain_sort: Union[*algebra.term.Sort, *None, List[Optional[*algebra.term.Sort*]]] = None*, *range_sort: Optional[*algebra.term.Sort*] = None*)

A symbolic representation of a function.

This class provides a callable symbolic representation of a function that can be used to create instantiations of terms called FuncTerms.

Parameters

- **symbol** (*str*) – The symbol to print out in the textual representation.
- **arity** (*int*) – The number of arguments that this function takes.
- **domain_sort** (*Union[AnySort, List[AnySort]], default=Any*) – The sort in which to restrict the input to. Defaults to AnySort. If given a list of sorts, then each argument is matched to a sort.
- **range_sort** (*AnySort*) – The sort to restrict the output to.

Examples

```
>>> from algebra import *
>>> x = Variable("x")
>>> f = Function("f", arity = 1)
>>> f(x)
f(x)
```

class algebra.term.**Sort**(*name: str*, *parent_sort: Optional[*Sort*] = None*)

A set that holds symbolic terms.

A sort is by default a subset of the universal sort though it can subsort of another sort if specified.

Parameters

- **name** (*str*) – The name of the sort.
- **parent_sort** (Sort) – The sort in which this sort is a subset of.

Notes

A real life example of this is the sort of fractions. The integer sort is a subsort of the fraction sort.

Examples

```
>>> from algebra import Sort
>>> fractions = Sort("Q")
>>> integers = Sort("Z", parent_sort=fractions)
>>> integers < fractions
True
```

subset_of(*sort*) → bool
Returns true is this sort is a subset of the sort given

class algebra.term.**Variable**(*symbol: str*, *sort: Optional[*algebra.term.Sort*] = None*)
A symbolic representation of a variable.

Parameters

- **symbol** (*str*) – The symbol to print out in the textual representation.
- **sort** (*AnySort*) – The sort in which the variable belongs to.

Examples

```
>>> from algebra import Variable
>>> Variable("x")
x
```

algebra.term.**count_occurence**(*subterm: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *term: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*)
Count the number of occurences of the subterm in the term.

Examples

```
>>> from algebra import *
>>> f = Function("f", 2)
>>> h = Function("h", 1)
>>> x = Variable("x")
>>> count_occurence(h(x), f(h(x), f(x, h(x))))
2
```

algebra.term.**depth**(*t: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *depth_level: int = 0*)
Returns the depth of a term.

Parameters

- **t** (*Term*) – The term to check the depth of.

- **depth_level** (*int*) – Used internally to keep track of the recursion

Examples

```
>>> from algebra import *
>>> f = Function("f", 2)
>>> x = Variable("x")
>>> a = Constant("a")
>>> depth(f(f(x,a), f(x,a)))
2
```

algebra.term.**get_constants**(*t:* *Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *unique:* *Literal[False]*) → List[*algebra.term.Constant*]
algebra.term.**get_constants**(*t:* *Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *unique:* *Literal[True]*) → Set[*algebra.term.Constant*]

Get the constants inside a term

Parameters

- **t** (*Term*) – The term to look for constants.
- **unique** (*bool*) – If true, then we only show each constant once.

Examples

```
>>> from algebra import *
>>> f = Function("f", 2)
>>> x = Variable("x")
>>> a = Constant("a")
>>> get_constants(f(x, f(x, a)))
[a]
```

algebra.term.**get_vars**(*t:* *Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *unique: Literal[False]*) → List[*algebra.term.Variable*]
algebra.term.**get_vars**(*t:* *Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *unique: Literal[True]*) → Set[*algebra.term.Variable*]

Get the variables inside a term

Parameters

- **t** (*Term*) – The term to look for variables.
- **unique** (*bool*) – If true, then we only show each variable once.

Examples

```
>>> from algebra import *
>>> f = Function("f", 2)
>>> x = Variable("x")
>>> a = Constant("a")
>>> get_vars(f(x, f(x, a)))
[x, x]
```

algebra.term.**get_vars_or_constants**(*t: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *unique: Literal[False]*) → List[Union[*algebra.term.Variable*, *algebra.term.Constant*]]

algebra.term.**get_vars_or_constants**(*t: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*, *unique: Literal[True]*) → Set[Union[*algebra.term.Variable*, *algebra.term.Constant*]]

Get the variables and constants inside a term

Parameters

- **t** (*Term*) – The term to look for variables and constants.
- **unique** (*bool*) – If true, then we only show each variable and constant once.

Examples

```
>>> from algebra import *
>>> f = Function("f", 2)
>>> x = Variable("x")
>>> a = Constant("a")
>>> get_vars_or_constants(f(x, f(x, a)))
[x, x, a]
```

CHAPTER

TWO

TERM SUBSTITUTION

This module is responsible for describing substitutions which are mappings between variables and terms, as well as the application of them.

exception algebra.substitute.**SortMismatch**
Raise when there is a sort mismatch.

class algebra.substitute.**SubstituteTerm**
Represents a substitution from variables to terms.

Examples

```
>>> from algebra import *
>>> x = Variable("x")
>>> a = Constant("a")
>>> sigma = SubstituteTerm()
>>> sigma.add(x, a)
>>> x * sigma
a
```

add(*variable:* algebra.term.Variable, *term:* *Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*)
Adds a mapping from a variable to a term

Parameters

- **variable** (Variable) – The variable to replace
- **term** (*Term*) – The term to replace the variable with.

Examples

```
>>> from algebra import *
>>> x = Variable("x")
>>> a = Constant("a")
>>> sigma = SubstituteTerm()
>>> sigma.add(x, a)
>>> print(sigma)
{ x -> a }
```

domain() → List[*algebra.term.Variable*]
Grabs the domain (the left side) of the substitutions.

Warning: Do not pair this call with range as ordering is not guarenteed.

range() → List[Union[*algebra.term.Variable*, *algebra.term.Constant*, *algebra.term.FuncTerm*]]
Grabs the range (the right side) of the substitutions.

Warning: Do not pair this call with domain as ordering is not guarenteed.

remove(*variable:* algebra.term.Variable)
Removes a mapping from a variable

replace(*variable:* algebra.term.Variable, *term: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*)
Replaces a mapping from a variable with another term

CHAPTER

THREE

SYMPY INTEGRATION

This module is an interface layer between our term library and sympy.

algebra.sympy.**sympyToTerm**(*symterm: sympy.core.symbol.Symbol*) → Union[*algebra.term.Variable*, *algebra.term.Constant*]
algebra.sympy.**sympyToTerm**(*symterm: sympy.core.function.FunctionClass*) → *algebra.term.Function*
algebra.sympy.**sympyToTerm**(*symterm: sympy.core.function.Function*) → *algebra.term.FuncTerm*

Converts a sympy term to a term. See notes for caveats.

Parameters **symterm** ({*sympy.Symbol*, *sympy.FunctionClass*, *sympy.Function*}) – A term from the sympy library. See notes for specifics.

Notes

This function isn't meant to take an arbitrary sympy term but one that is formatted the following way:

- Variables end in *_variable*
- Constants end in *_constant*
- Functions with arity x end with *_x*

Examples

```
>>> from algebra import sympyToTerm
>>> import sympy
>>> f = sympy.Function("f_2")
>>> sympyToTerm(f)
f
```

algebra.sympy.**termToSympy**(*term:* algebra.term.Constant) → sympy.core.symbol.Symbol
algebra.sympy.**termToSympy**(*term:* algebra.term.Variable) → sympy.core.symbol.Symbol
algebra.sympy.**termToSympy**(*term:* algebra.term.Function) → sympy.core.function.FunctionClass
algebra.sympy.**termToSympy**(*term:* algebra.term.FuncTerm) → sympy.core.function.Function

Converts a term to a sympy term

Parameters **term** (*Term*) – The term from this library to turn into a sympy term.

Examples

```
>>> from algebra import *
>>> f = Function("f", 1)
>>> a = Constant("a")
>>> termToSympy(f(a))
f_1(a_constant)
```

CHAPTER

FOUR

TERM STRING PARSER

This module is responsible for creating terms out of strings, given a set of known terms.

class algebra.parser.**Parser**

A string parser library.

Given a signature, the parser can then take a string that represents a term and create the term using the algebra library.

Examples

```
>>> from algebra import *
>>> f = Function("f", 2)
>>> x = Variable("x")
>>> p = Parser()
>>> p.add(f)
>>> p.add(x)
>>> p.parse("f(x,x)")
f(x, x)
```

add(*term: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.Function*]*)

Adds a term to the parser.

parse(*x: str*) → Union[*algebra.term.Variable*, *algebra.term.Constant*, *algebra.term.FuncTerm*, *algebra.term.Function*]

Attempt to parse a string given the parser's existing signature.

remove(*term: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.Function*]*)

Remove a term from the parser.

CHAPTER

FIVE

TERM DAG LIBRARY

This module contains a direct acyclic graph representation of a term. This is mainly included for applications that require better performance characteristics than the recursive definition.

class algebra.dag.**TermDAG**(*term: Union[*algebra.term.Variable, algebra.term.Constant, algebra.term.FuncTerm*]*)

A directed acyclic graph representation of a term.

The dag implements struture sharing so that every subterm in the DAG is unique.

Notes

You can think of this as a tree. At the top of the tree is the outermost function/variable/constant. For each argument that a function has, it will point an array from that function to the argument.

bs_edge_traversal()
Breadth-first traversal of the edges

bs_node_traversal()
Breadth-frist traversal of the nodes

df_edge_traversal()
Depth-first traversal of the edges

df_node_traversal()
Depth-first traversal of the nodes

leaves()
Leaves of a TermDAG

parents(*term*)
Parents of a term in the TermDAG

show()
Plot the directed acyclic graph of the TermDAG

algebra.dag.**termDAGSubstitute**(*dag*, *variable*, *replacement_term*)

Performs a variable substitution in a TermDAG.

Parameters

- **dag** (TermDAG) – The DAG representation of a term.
- **variable** (Variable) – The variable to replace within the DAG.
- **replacement_term** (*Term*) – The term to replace the variable in the DAG with.

PYTHON MODULE INDEX

a

INDEX