this version: August 17, 2011 22:02

Representing expressions in the semantic memory

Peter Schodl
Arnold Neumaier

Fakultät für Mathematik, Universität Wien
Nordbergstr. 15, A-1090 Wien, Austria
WWW: http://www.mat.univie.ac.at/~neum/FMathL.html

Contents

Acknowledgements. Earlier versions of this paper were discussed in meetings of the FMathL project. In particular, Kevin Kofler and Hermann Schichl contributed significantly through their remarks.

Support by the Austrian Science Fund (FWF) under contract number P20631 is greatfully acknowledged.

1 Introduction

This paper assumes [1].

We aim to represent in a transparent fashion all possible mathematical expression in the semantic memory.

An **operation** is anything that can be applied to mathematical expressions $E_1, E_2, \ldots$ such that the result E is an expression again. We call $E_1, E_2, \ldots$ the **subexpressions** of E. In particular, all standard functions, binary operations, and relations are operations, and so are quantification, merging expressions to form a set, a vector, etc. The operations are those categories that match the category `Expression` in the sense of [1].

We store the information in a fashion inspired by automatic differentiation, meaning we proceed from the most elementary subexpressions (its variables and constants) to the more complicated subexpressions by applying operations until the expression is fully covered.

2 The representation

Let the record `#handle` contain the expression E. Then we say that `#handle` is the **handle** of E. From the handle of some expression, the espression E itself and the free variables of E have to be accessable easily from `#handle`. The nodes representing the free variables of E are stored in `#handle.free` in the following fashion: For every node `#var` representing a free variable of E, we have `#handle.free.#var=#var`. If some expression does not have any free variables, then `#handle.VAR` is nonempty but does not have children.

The expression itself is constructed from its subexpressions in a recursive way, with constants and variables being expressions without subexpressions. The operation that is applied to the subexpressions of E is represented in the object `#handle.type`, the same object that is used for the typing of `#handle` (see [1]). How the subexpressions of E are represented in relation to `#handle` depends on the kind of operation, see below.

When an operation is applied to subexpressions, the free variables of the combined expression form the union of the free variables of the subexpressions, minus the variables that are bound by the operation. Every variable `#var` that is bound by application of the operation represented in `#handle.type` is stored as `#handle.binds.#var=#var`.

3 Types of expressions

We now illustrate the different types of expressions: since we build up all expressions from variables, constants and the application of operations to subexpressions, we have to describe the representation of these.

The handle of the expression is always denoted by `#handle` or `#h`.

3.1 Constants

There are currently three types of constants: strings, integers and floats, and all of them are represented in a similar fashion. The actual constant is always stored as the value of the handle of the constant. The record `#h` representing a constant has `#h.type=String` if `#h` is a string, `#h.type=Integer` if `#h` is an integer, and `#h.type=Float` if `#h` is a float.

For example, the string "Hello world" is represented as

$$\texttt{\#h} \xrightarrow{\texttt{type}} \texttt{String}$$

where `#h` is some anonymous node with `VALUE(#h) = Hello world`.

The type declaration of the constant types are:

```
String, Integer, Float:
nothingElse>
```

The MATLAB creation code to create a string "Hello world" in record `string1` is

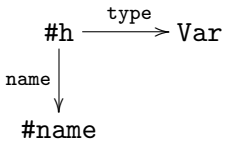
```
string1 = createstring('Hello world');
```

The corresponding commands for integers and floats are `createint` and `createdouble`.

3.2 Variables

The record `#h` representing a variable has `#h.type=Var`.

A name can, but need not be assigned to the variable. A variable with name `x1` is represented as



where the object `#name` is a string containing `x1`.

The type declaration of variables are:

```
Var:
optinal> name=String
nothingElse>
```

The MATLAB creation code to create a variable with name `x1` in record `variable1` is

```
variable1 = createvar('x1');
```

3.3 Operations with fixed arguments

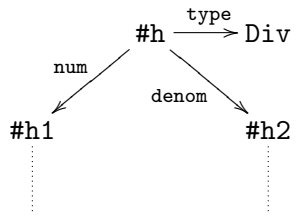
These are the operations that only allow a certain number of subexpressions to be applied to, and these subexpressions have a known role in the resulting expression E.

For example, the operation "squareroot" has one argument, the *radicand*, a fraction has two arguments, the *numerator* and the *denominator*, etc.

This reflects in the way these expressions are represented. For an expression E represented as record `#h` the subexpressions will be represented in `#h.#field` where the name of `#field` will usually unambiguously clarify the role of the subexpression in `#h.#field` for the expression in `#h`.

For example, consider an expression E with E_1 being it numerator and E_2 the denominator, hence $E = \frac{E_1}{E_2}$.

If E_1 is represented in `#h1` and E_2 is represented in `#h2` then the representation of the expression E in record `#h` (omitting the free variables) is:



The type declaration of a division is:

```
Div:
allOf> num=Expression, denom=Expression
```

The MATLAB creation code to create the division of the expression E_1 in record `expr1` and expression E_2 in record `expr2` to be stored in record `division1` is

```
division1 = create('Div',expr1,expr2);
```

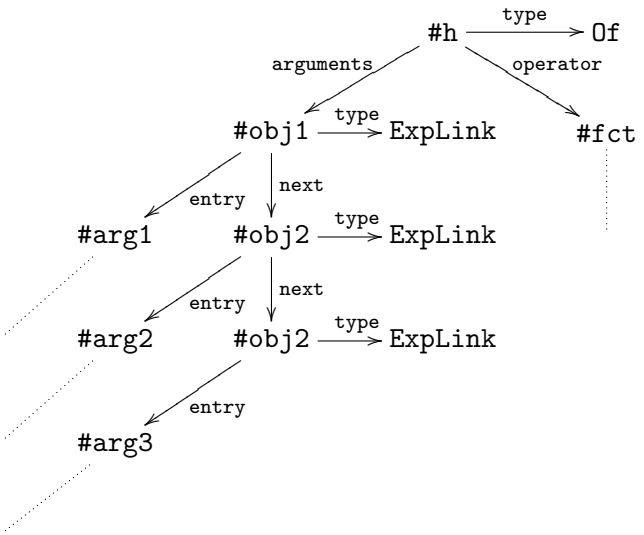
where the order of the arguments is significant: the first argument always contains the type, the

3.4 General n-ary Operations

Another kind of operations are those that admit an arbitrary number of subexpressions to be applied to, but all of these are treated equally. But there may still be a known number of subexpressions aside of these that have a fixed role.

For example, a case dinstinction between n cases, and as an extra argument the case "otherwise", or the application of a function f to n arguments. In these cases, the n arguments are always represented as a linked list.

For example, consider the expression $f(x_1, x_2, x_3)$ where f is represented in `#fct` and x_i is represented in `#arg`i. Then the representation of the expression $f(x_1, x_2, x_3)$ in record `#h` (again omitting the free variables) is:



The type declaration of this application is:

```
Of:
allOf> operator=Expression, arguments=ExpLink

ExpLink:
allOf> entry=Expression
optional> next=ExpLink
```

The MATLAB creation code to create $f(x_1, x_2, x_3)$ with f in record `func1` and x_1, x_2, x_3 in records `arg1, arg2, arg3` respectively in record `functionapplic1` is

```
functionapplic1 = create('Of',func1,{arg1,arg2,arg3});
```

3.5 Example

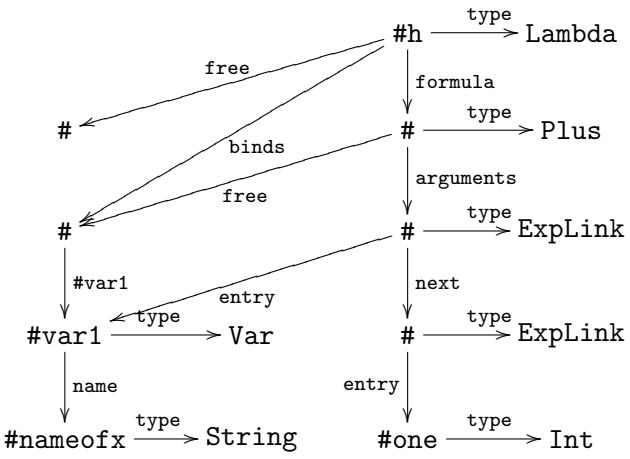
We give a complete record, with the bound variables and the free variables.

Consider the expression:

$$\lambda x.x + 1$$

The subexpressions are the variable x and the constant 1, the result of the operator `Plus` to these, resulting in $x + 1$ (having free variable x) and lastly the application of the operator `Lambda` binding variable x, hence resulting in $\lambda x.x + 1$ which has no free variable.

Anonymous nodes that are not referred to are simply denoted by `#`. Assume that `VALUE(#one) = 1` and `VALUE(#nameofx) = x`.



4 The typesheet for expressions

The following is the typesheet that defines all the types that can be considered as operators to form expressions:

```
Expressions::

! Expression types
! ----------------
!
! Peter Schodl
!
! Feb 21, 2011
!
! This type system defines the types needed to process expression.

! To type tighter:
!!! Make Term and Expression
!!! type narrower: from AND to  /  over  (a new type FromTo)


Alternative:
  allOf> linkedList = AlternativeLink

AlternativeLink:
  allOf> entry = Expression
  optional> next = AlternativeLink

Bracket:
 allOf> entry = Expression

Cases:
```

```
  allOf> linkedList = CasesLink
  optional> otherwise = Expression

CasesLink:
  allOf> formula = Expression
         condition = Expression
  optional> next = CasesLink

Chain:
  allOf> firstrel = Expression
         linkedList = ExpLink
ExpLink:
  allOf> entry = Expression
  optional> next = ExpLink

Diag:
  allOf> linkedList = ExpLink

Div:
  allOf> nom = Expression
         den = Expression

Dummy:
  allOf> entry = Expression

Equal:
  allOf> lhs = Expression
         rhs = Expression
  optional> above = Text

Eval:
  allOf> formula = Expression
         binds = VarList
  optional> index = Expression
            from = Expression
            to = Expression

Forall:
  allOf> formula = Expression
         scopedvar = Expression
         binds = VarList

InvisMult:
  allOf> linkedList = ExpLink

Interval:
  allOf> lower = Expression
         upper = Expression

Integral:
  allOf> formula = Expression
         variable = IndexedVar
         binds = VarList
  optional> index = Expression
```

```
            from = Expression
            to = Expression

List:
   allOf> linkedList = ExpLink
   optional> leftBr = Brackets
             separator = Separators
             rightBr = Brackets

Lambda:
  allOf> formula = Expression
         variable = IndexedVar
         binds = VarList

Max:
  allOf> formula = Expression
  optional> binds = VarList
            index = Expression

Min:
  allOf> formula = Expression
  optional> binds = VarList
            index = Expression

Mid:
  allOf> lhs = Expression
         rhs = Expression

Matrix:
  allOf> linkedList = RowLink

Norm:
  allOf> formula = Expression
  optional> index = Expression

Of:
  allOf> operator = Expression
         arguments = Expression

Or:
  allOf> linkedList = ExpLink

OtherInterval:
  someOf> lowerclosed = Expression
          loweropen = Expression
          upperopen = Expression
          upperclosed = Expression

Partial:
  allOf> linkedList = ExpLink

Prime:
  allOf> entry = Expression
```

```
Power:
  allOf> base = Expression
         exponent = Expression

Prob:
  allOf> event = Expression
  optional> condition = Expression

Row:
  allOf> linkedList = ExpLink

RowLink:
  allOf> entry = Row
  optional> next = RowLink

Restriction:
  allOf> formula = Expression
         restriction = Expression
  optional> binds = VarList
            if = Expression
            forsome = Expression

Relation:
  allOf> lhs = RelationLhs
         rhs = Expression
         relation = RelationSymbols
  optional> above = Text

Script:
  allOf> formula = Expression
  someOf> sub = Expression
          sup = Expression
          lsup = Expression
          lsub = Expression

Sqrt:
  allOf> radicand = Expression

Set2Exp:
  allOf> lhs = Expression
         rhs = Expression
  optional> binds = VarList

Sum:
  allOf> formula = Expression
         binds = VarList
  someOf> index = Expression
          from = Expression
          to = Expression

Set:
  allOf> scopedvar = Expression
         condition = Expression
  optional> binds = VarList
```

```
SetUnion:
  allOf> linkedList = ExpLink

SetProduct:
  allOf> linkedList = ExpLink

SetBucket:
  allOf> linkedList = ExpLink

SignedSum:
  allOf> linkedList = SignedSumLink

SignedSumLink:
  allOf> sign = Signs
         entry = Expression
  optional> next = SignedSumLink

Text:
  allOf> entry = Object

Var:
  nothingElse>
  optional> name = String

VarList:
  itself> IndexedVar

Vector:
  allOf> linkedList = ExpLink

Separators:
atomic> SepKomma,SepColon,SepSemicolon,SepBlank,None

Brackets:
atomic> BrLeftRound,BrRightRound,BrLeftSquare,BrRightSquare,None

RelationSymbols:
atomic> LessEq,Less,In,Greater,GreaterEq,EqualByDef,EqualSign

Signs:
atomic> InvisPlusSign,MinusSign,PlusSign

Expression:
  union> Alternative,Bracket,Cases,Chain,Diag,Div,Dummy,Equal
  union> Eval,Forall,InvisMult,Interval,Integral,List,Lambda,Max
  union> Min,Mid,Matrix,Norm,Of,Or,OtherInterval,Partial,Prime
  union> Power,Prob,Restriction,Relation,Script,Sqrt,Set2Exp,Sum
  union> Set,SetBucket,SetProduct,SignedSum,SetUnion,Var,Vector,Dummy
  union> String,Integer,Double,Separators,Signs,RelationSymbols

RelationLhs:
  union> Expression, VarList
```

```
IndexedVar:
  union> Var, Script
```

5 Not yet implemented

The following expressions involve some types that are not implemented yet:

1. Ellipsis for operations, e.g.,
$$\int \ldots \int f(x_1, \ldots, x_n)\ \mathrm{d}x_1 \ldots \mathrm{d}x_n$$
2. Diagrams
3. Tables
4. $\int \frac{x\ \mathrm{d}x}{1+x^2}$ instead of $\int \frac{x}{1+x^2}\mathrm{d}x$
5. Other integral styles (Lebesgue etc.).
6. $m(a) \begin{cases} > 0 & \text{if } A \in \mathcal{A} \\ = 0 & \text{otherwise} \end{cases}$

6 Examples of expressions

We give examples for the representation of expressions in the SM as records.
The following table gives a small statistic of the examples:

Example	# visible symbols	# LaTeX-characters	# sems
1	6	19	21
2	10	38	31
3	14	78	37
4	22	76	60
5	15	65	53
6	7	36	10
7	2	8	15
8	11	88	54
9	24	99	33
10	18	113	39
11	7	29	16
12	11	54	32
13	10	50	28
14	6	95	34
15	33	99	86
16	14	51	42
17	17	64	53
18	20	76	73
19	16	45	28
20	22	73	90
21	9	32	45
22	6	16	19
23	11	51	57
24	28	159	75
25	17	38	23
26	19	105	50
27	18	84	83
28	13	62	39
29	15	73	60
30	17	75	46
31	16	56	41

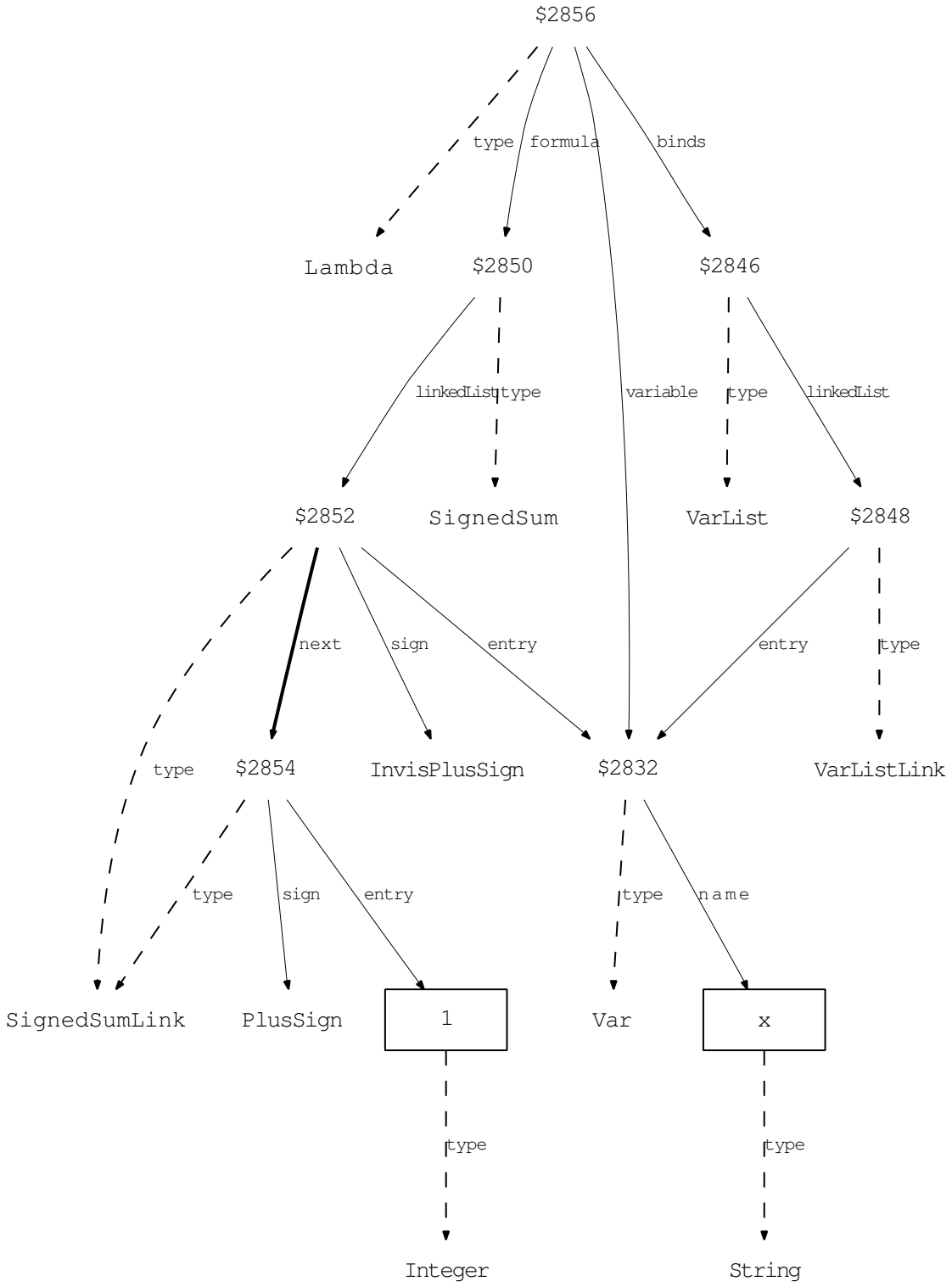
In the following examples, the entry of `#o.free` containing the free variables of the expression, is missing!

Example 1.

$$\lambda x.x + 1$$

```
1 \lambda x . x + 1
```

```
$2856.type=Lambda
$2856.formula=$2850
$2856.binds=$2846
$2856.variable=$2832
$2832.type=Var
$2832.name=$2834
$2834.type=String
$2846.type=VarList
$2846.linkedList=$2848
$2848.type=VarListLink
$2848.entry=$2832
$2850.type=SignedSum
$2850.linkedList=$2852
$2852.type=SignedSumLink
$2852.next=$2854
$2852.sign=InvisPlusSign
$2852.entry=$2832
$2854.type=SignedSumLink
$2854.sign=PlusSign
$2854.entry=$2844
$2844.type=Integer
VALUE($2834) = x
VALUE($2844) = 1
```

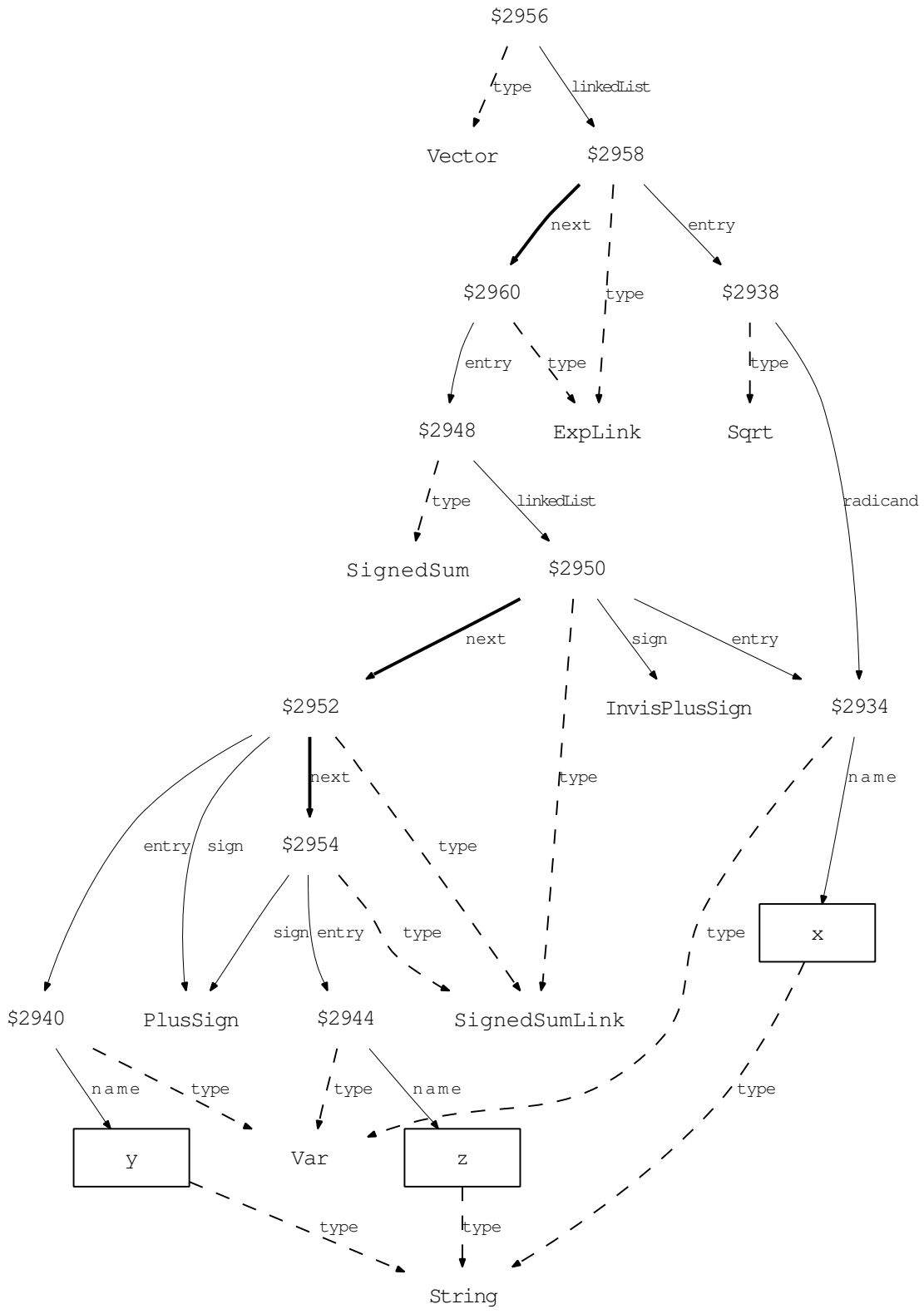

$2856
type
formula
binds
Lambda
$2850
$2846
linkedList
type
variable
type
linkedList
$2852
SignedSum
VarList
$2848
next
sign
entry
entry
type
type
$2854
InvisPlusSign
$2832
VarListLink
type
sign
entry
type
name
SignedSumLink
PlusSign
1
Var
x
type
type
Integer
String

Example 2.

$$\left(\sqrt{x}, x + y + z\right)$$

```
\left(\sqrt{ x } , x + y + z \right)
```

```
$2956.type=Vector
$2956.linkedList=$2958
$2958.type=ExpLink
$2958.next=$2960
$2958.entry=$2938
$2938.type=Sqrt
$2938.radicand=$2934
$2934.type=Var
$2934.name=$2936
$2936.type=String
$2960.type=ExpLink
$2960.entry=$2948
$2948.type=SignedSum
$2948.linkedList=$2950
$2950.type=SignedSumLink
$2950.next=$2952
$2950.sign=InvisPlusSign
$2950.entry=$2934
$2952.type=SignedSumLink
$2952.next=$2954
$2952.sign=PlusSign
$2952.entry=$2940
$2940.type=Var
$2940.name=$2942
$2942.type=String
$2954.type=SignedSumLink
$2954.sign=PlusSign
$2954.entry=$2944
$2944.type=Var
$2944.name=$2946
$2946.type=String
VALUE($2936) = x
VALUE($2942) = y
VALUE($2946) = z
```

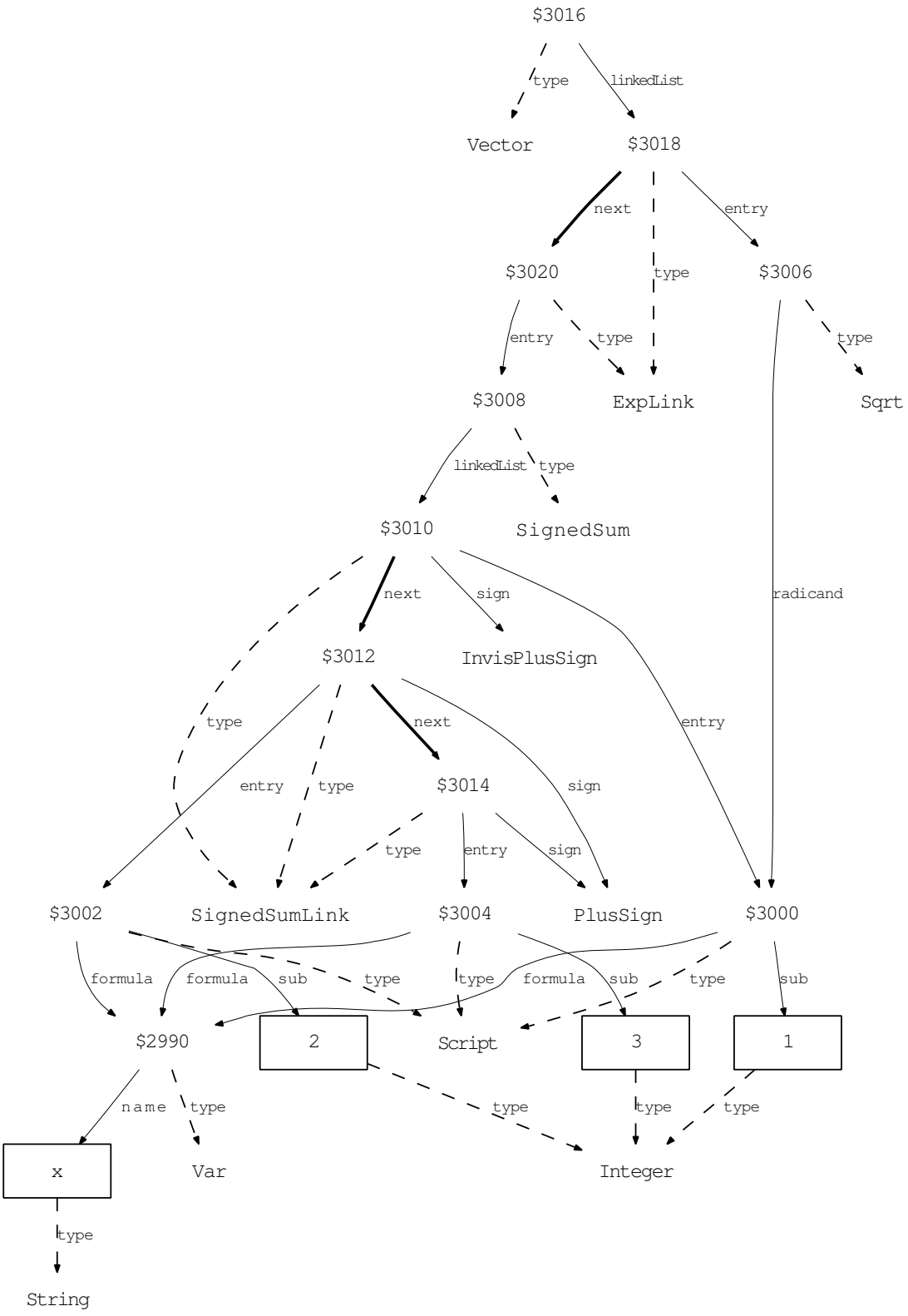

$2956
type
linkedList
Vector
$2958
next
entry
$2960
type
$2938
entry
type
type
$2948
ExpLink
Sqrt
type
linkedList
radicand
SignedSum
$2950
next
sign
entry
$2952
InvisPlusSign
$2934
next
type
name
entry
sign
$2954
type
sign
entry
type
type
x
$2940
PlusSign
$2944
SignedSumLink
name
type
type
name
type
y
Var
z
type
type
String

Example 3. Similar as before, but now the only variable is the vector x.

$$(\sqrt{x_1}, x_1 + x_2 + x_3)$$

```
\left(\sqrt{ {}{x}_{ 1 } } , {}{x}_{ 1 } + {}{x}_{ 2 } + {}{x}_
   { 3 } \right)
```

```
$3016.type=Vector
$3016.linkedList=$3018
$3018.type=ExpLink
$3018.next=$3020
$3018.entry=$3006
$3006.type=Sqrt
$3006.radicand=$3000
$3000.type=Script
$3000.formula=$2990
$3000.sub=$2994
$2990.type=Var
$2990.name=$2992
$2992.type=String
$2994.type=Integer
$3020.type=ExpLink
$3020.entry=$3008
$3008.type=SignedSum
$3008.linkedList=$3010
$3010.type=SignedSumLink
$3010.next=$3012
$3010.sign=InvisPlusSign
$3010.entry=$3000
$3012.type=SignedSumLink
$3012.next=$3014
$3012.sign=PlusSign
$3012.entry=$3002
$3002.type=Script
$3002.formula=$2990
$3002.sub=$2996
$2996.type=Integer
$3014.type=SignedSumLink
$3014.sign=PlusSign
$3014.entry=$3004
$3004.type=Script
$3004.formula=$2990
$3004.sub=$2998
$2998.type=Integer
VALUE($2992) = x
VALUE($2994) = 1
VALUE($2996) = 2
VALUE($2998) = 3
```

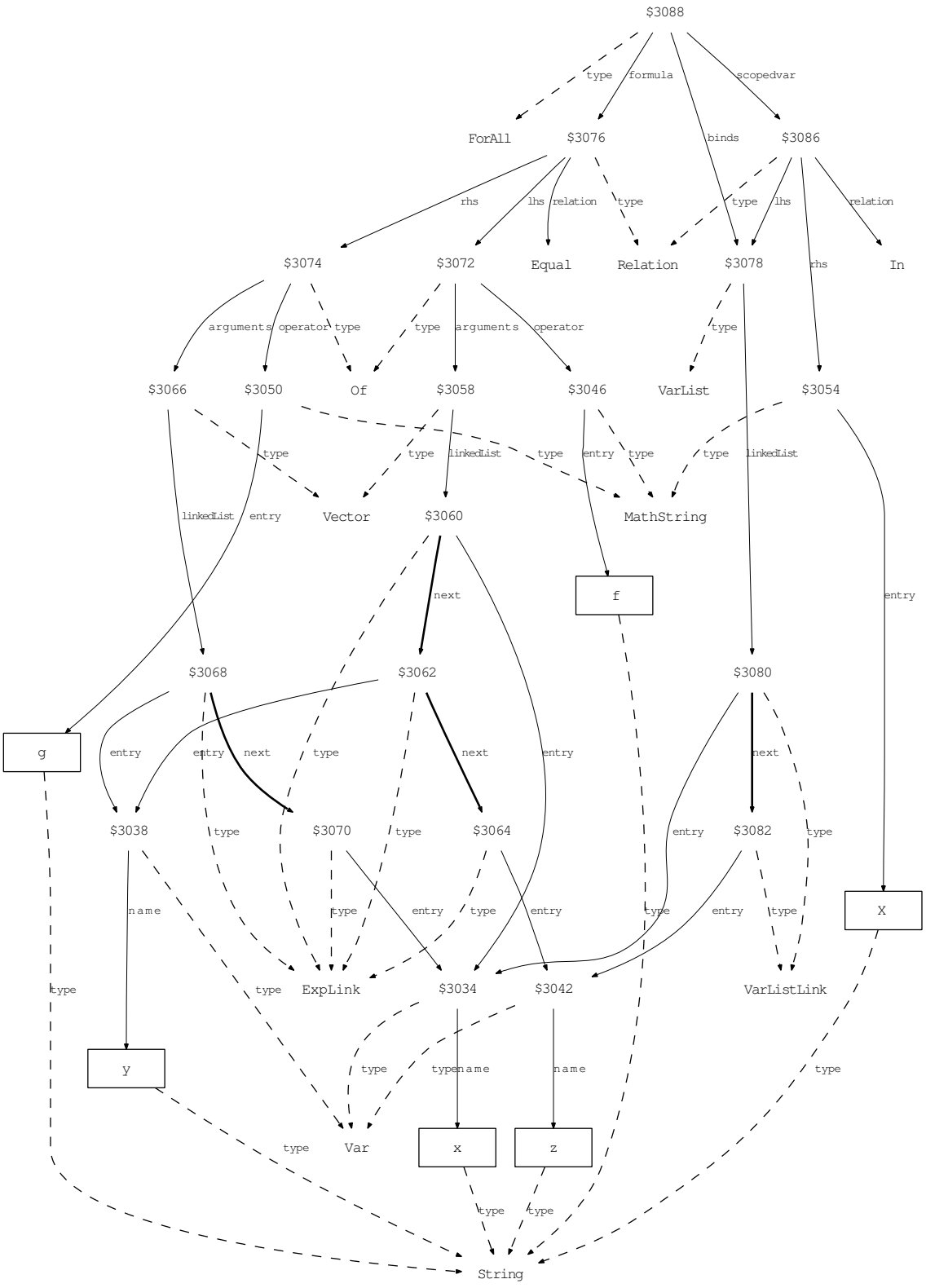

$3016
type
linkedList
Vector
$3018
next
entry
$3020
type
$3006
entry
type
type
$3008
ExpLink
Sqrt
linkedList
type
$3010
SignedSum
next
sign
radicand
$3012
InvisPlusSign
type
next
entry
entry
type
$3014
sign
type
entry
sign
$3002
SignedSumLink
$3004
PlusSign
$3000
formula
formula
sub
type
type
formula
sub
type
sub
$2990
2
Script
3
1
name
type
type
type
type
x
Var
Integer
type
String

Example 4. The variable y is the only free variable.

$$\forall x, z{\in}X : f\left(x, y, z\right) {=} g\left(y, x\right)$$

```
\forall x , z{ \in }X : f \left(x , y , z\right) {{=}}g \left(y
    , x\right)
```

```
$3088.type=ForAll
$3088.formula=$3076
$3088.scopedvar=$3086
$3088.binds=$3078
$3076.type=Relation
$3076.lhs=$3072
$3076.relation=Equal
$3076.rhs=$3074
$3072.type=Of
$3072.operator=$3046
$3072.arguments=$3058
$3046.type=MathString
$3046.entry=$3048
$3048.type=String
$3058.type=Vector
$3058.linkedList=$3060
$3060.type=ExpLink
$3060.next=$3062
$3060.entry=$3034
$3034.type=Var
$3034.name=$3036
$3036.type=String
$3062.type=ExpLink
$3062.next=$3064
$3062.entry=$3038
$3038.type=Var
$3038.name=$3040
$3040.type=String
$3064.type=ExpLink
$3064.entry=$3042
$3042.type=Var
$3042.name=$3044
$3044.type=String
$3074.type=Of
$3074.operator=$3050
$3074.arguments=$3066
$3050.type=MathString
$3050.entry=$3052
$3052.type=String
$3066.type=Vector
$3066.linkedList=$3068
$3068.type=ExpLink
$3068.next=$3070
$3068.entry=$3038
$3070.type=ExpLink
$3070.entry=$3034
$3078.type=VarList
$3078.linkedList=$3080
$3080.type=VarListLink
$3080.next=$3082
$3080.entry=$3034
$3082.type=VarListLink
$3082.entry=$3042
$3086.type=Relation
$3086.lhs=$3078
$3086.relation=In
$3086.rhs=$3054
$3054.type=MathString
$3054.entry=$3056
$3056.type=String
VALUE($3036) = x
VALUE($3040) = y
VALUE($3044) = z
VALUE($3048) = f
VALUE($3052) = g
VALUE($3056) = X
```

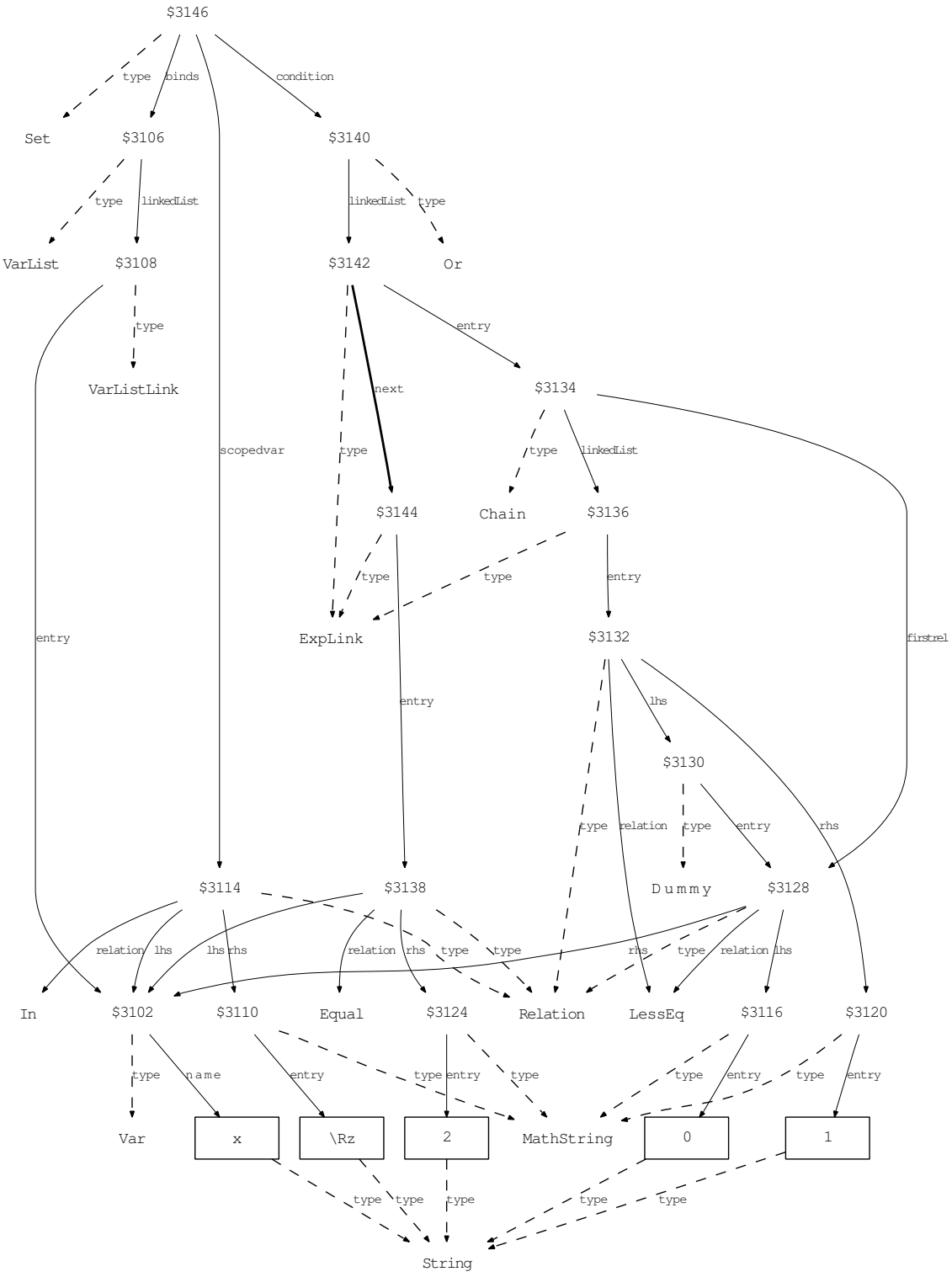

$3088
type
formula
scopedvar
ForAll
$3076
binds
$3086
rhs
lhs
relation
type
type
lhs
relation
$3074
$3072
Equal
Relation
$3078
rhs
In
arguments
operator
type
type
arguments
operator
type
$3066
$3050
Of
$3058
$3046
VarList
$3054
type
type
linkedList
type
entry
type
type
linkedList
linkedList
entry
Vector
$3060
MathString
next
f
entry
$3068
$3062
$3080
g
entry
entry
next
type
next
entry
next
$3038
type
$3070
type
$3064
entry
$3082
type
name
type
type
entry
type
entry
type
entry
type
X
type
type
ExpLink
$3034
$3042
VarListLink
y
type
type
name
name
type
type
Var
x
z
type
type
String

Example 5.

$$\{x{\in}\mathbb{R} \mid 0{\leq}x{\leq}1 \vee x{=}2\}$$

```
\left\{x{ \in }\Rz \mid 0{ \leq }x{ \leq }1 \vee x{{=}}2\right
    \}
```

```
$3146.type=Set
$3146.scopedvar=$3114
$3146.binds=$3106
$3146.condition=$3140
$3106.type=VarList
$3106.linkedList=$3108
$3108.type=VarListLink
$3108.entry=$3102
$3102.type=Var
$3102.name=$3104
$3104.type=String
$3114.type=Relation
$3114.lhs=$3102
$3114.relation=In
$3114.rhs=$3110
$3110.type=MathString
$3110.entry=$3112
$3112.type=String
$3140.type=Or
$3140.linkedList=$3142
$3142.type=ExpLink
$3142.next=$3144
$3142.entry=$3134
$3134.type=Chain
$3134.firstrel=$3128
$3134.linkedList=$3136
$3128.type=Relation
$3128.lhs=$3116
$3128.relation=LessEq
$3128.rhs=$3102
$3116.type=MathString
$3116.entry=$3118
$3118.type=String
$3136.type=ExpLink
$3136.entry=$3132
$3132.type=Relation
$3132.lhs=$3130
$3132.relation=LessEq
$3132.rhs=$3120
$3120.type=MathString
$3120.entry=$3122
$3122.type=String
$3130.type=Dummy
$3130.entry=$3128
$3144.type=ExpLink
$3144.entry=$3138
$3138.type=Relation
$3138.lhs=$3102
$3138.relation=Equal
$3138.rhs=$3124
$3124.type=MathString
$3124.entry=$3126
$3126.type=String
VALUE($3104) = x
VALUE($3112) = \Rz
VALUE($3118) = 0
VALUE($3122) = 1
VALUE($3126) = 2
```

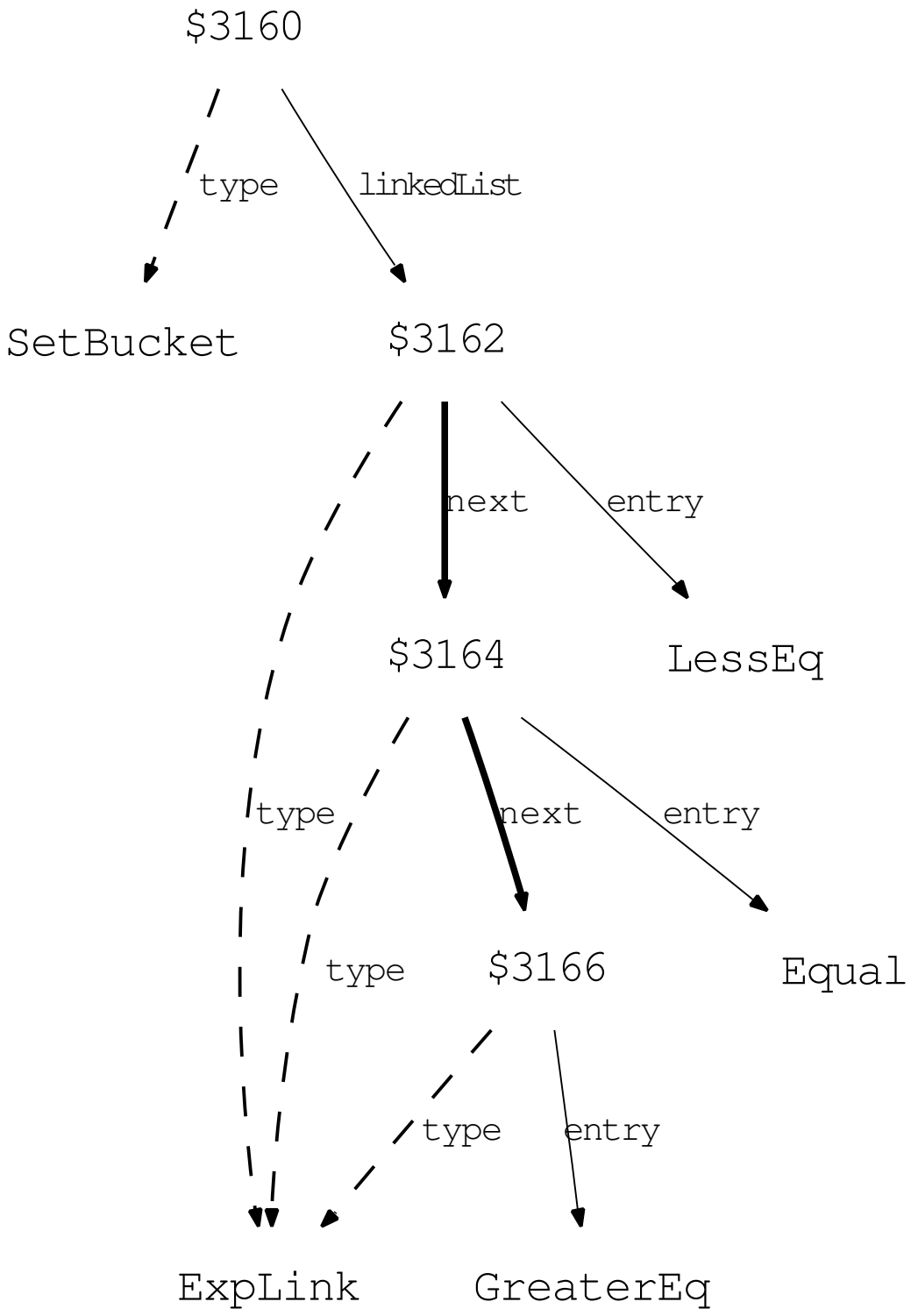

$3146
type
binds
condition
Set
$3106
$3140
type
linkedList
linkedList
type
VarList
$3108
$3142
Or
type
entry
VarListLink
next
$3134
scopedvar
type
type
linkedList
$3144
Chain
$3136
type
type
entry
entry
ExpLink
$3132
firstrel
entry
lhs
$3130
type
relation
type
entry
rhs
$3114
$3138
Dummy
$3128
relation
lhs
lhs
rhs
relation
rhs
type
type
rhs
type
relation
lhs
In
$3102
$3110
Equal
$3124
Relation
LessEq
$3116
$3120
type
name
entry
type
entry
type
type
entry
type
entry
Var
x
\Rz
2
MathString
0
1
type
type
type
type
type
String

Example 6. A set as list with characters as entries.

$$\{\leq, =, \geq\}$$

```
 \left\{ \leq , {=} , \geq \right\}
```

```
$3160.type=SetBucket
$3160.linkedList=$3162
$3162.type=ExpLink
$3162.next=$3164
$3162.entry=LessEq
$3164.type=ExpLink
$3164.next=$3166
$3164.entry=Equal
$3166.type=ExpLink
$3166.entry=GreaterEq
```

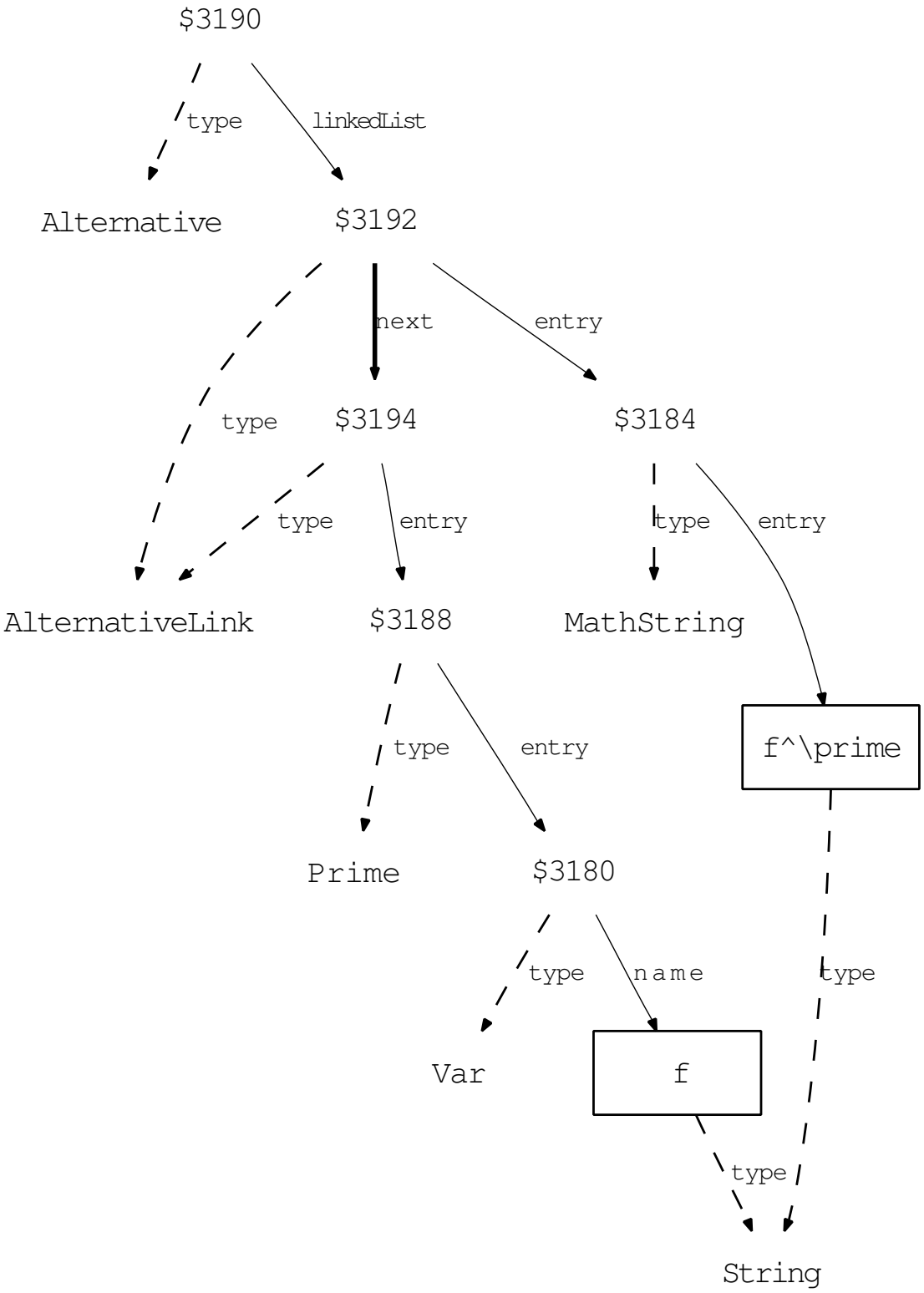

$3160
type
linkedList
SetBucket
$3162
next
entry
$3164
LessEq
type
next
entry
type
$3166
Equal
type
entry
ExpLink
GreaterEq

Example 7. f' is ambiguous: It could be the application of the operation ' to the mapping f or the name of fuzzy numbers f, f', f''.

$$f'$$

```
f^\prime
```

```
$3190.type=Alternative
$3190.linkedList=$3192
$3192.type=AlternativeLink
$3192.next=$3194
$3192.entry=$3184
$3184.type=MathString
$3184.entry=$3186
$3186.type=String
$3194.type=AlternativeLink
$3194.entry=$3188
$3188.type=Prime
$3188.entry=$3180
$3180.type=Var
$3180.name=$3182
$3182.type=String
VALUE($3182) = f
VALUE($3186) = f^\prime
```

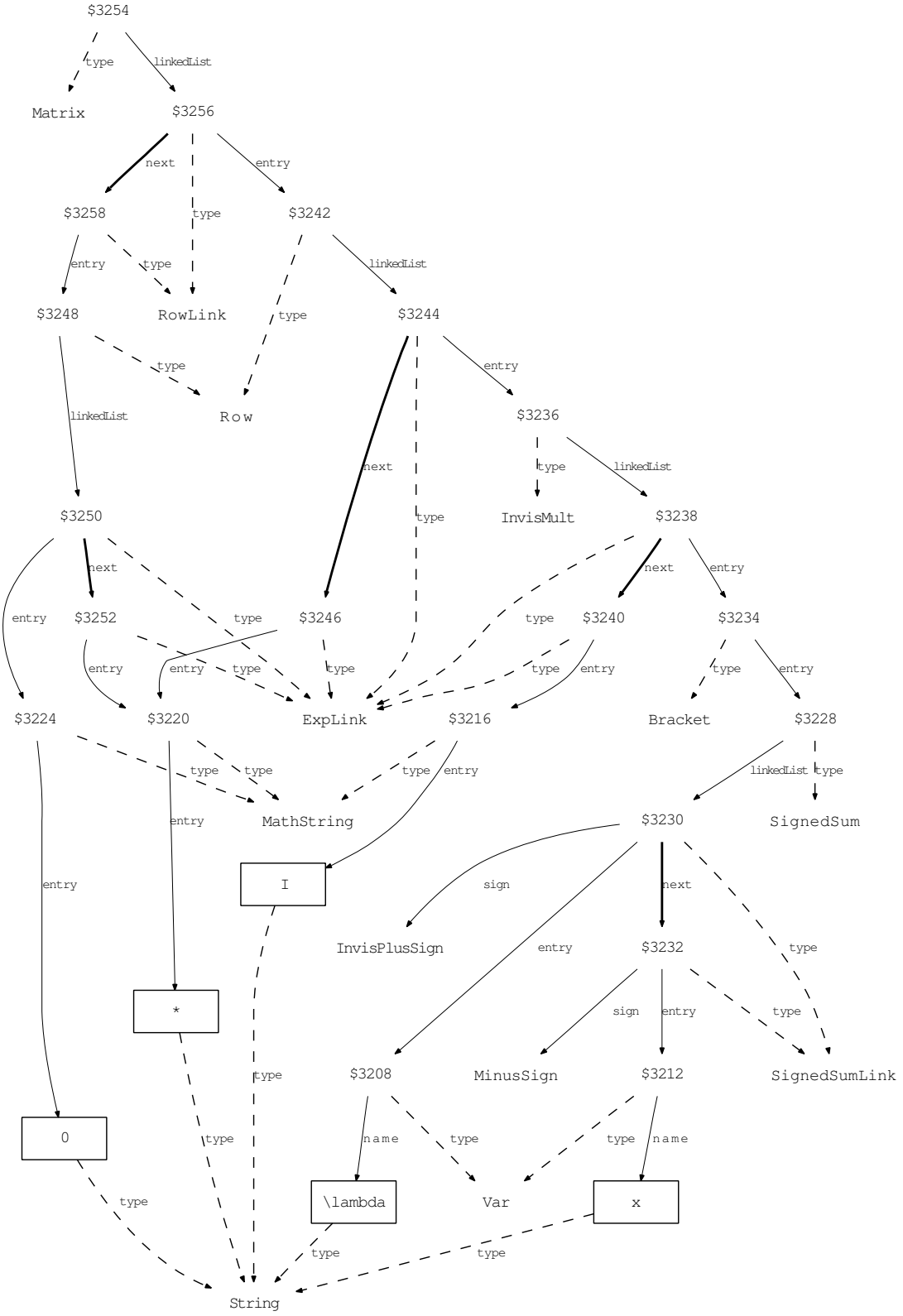

$3190
type
linkedList
Alternative
$3192
next
entry
type
$3194
$3184
type
entry
type
entry
AlternativeLink
$3188
MathString
type
entry
f^\prime
Prime
$3180
type
name
type
Var
f
type
String

Example 8. A matrix with wildcard characters, denoted by $*$.

$$\begin{pmatrix} (\lambda - x)\, I & * \\ 0 & * \end{pmatrix}$$

```
\left(\begin{array}{cc} \left( \lambda - x \right) I & * \\ 0 &
   * \\ \end{array} \right)
```

```
$3254.type=Matrix
$3254.linkedList=$3256
$3256.type=RowLink
$3256.next=$3258
$3256.entry=$3242
$3242.type=Row
$3242.linkedList=$3244
$3244.type=ExpLink
$3244.next=$3246
$3244.entry=$3236
$3236.type=InvisMult
$3236.linkedList=$3238
$3238.type=ExpLink
$3238.next=$3240
$3238.entry=$3234
$3234.type=Bracket
$3234.entry=$3228
$3228.type=SignedSum
$3228.linkedList=$3230
$3230.type=SignedSumLink
$3230.next=$3232
$3230.sign=InvisPlusSign
$3230.entry=$3208
$3208.type=Var
$3208.name=$3210
$3210.type=String
$3232.type=SignedSumLink
$3232.sign=MinusSign
$3232.entry=$3212
$3212.type=Var
$3212.name=$3214
$3214.type=String
$3240.type=ExpLink
$3240.entry=$3216
$3216.type=MathString
$3216.entry=$3218
$3218.type=String
$3246.type=ExpLink
$3246.entry=$3220
$3220.type=MathString
$3220.entry=$3222
$3222.type=String
$3258.type=RowLink
$3258.entry=$3248
$3248.type=Row
$3248.linkedList=$3250
$3250.type=ExpLink
$3250.next=$3252
$3250.entry=$3224
$3224.type=MathString
$3224.entry=$3226
$3226.type=String
$3252.type=ExpLink
$3252.entry=$3220
VALUE($3210) = \lambda
VALUE($3214) = x
VALUE($3218) = I
VALUE($3222) = *
VALUE($3226) = 0
```

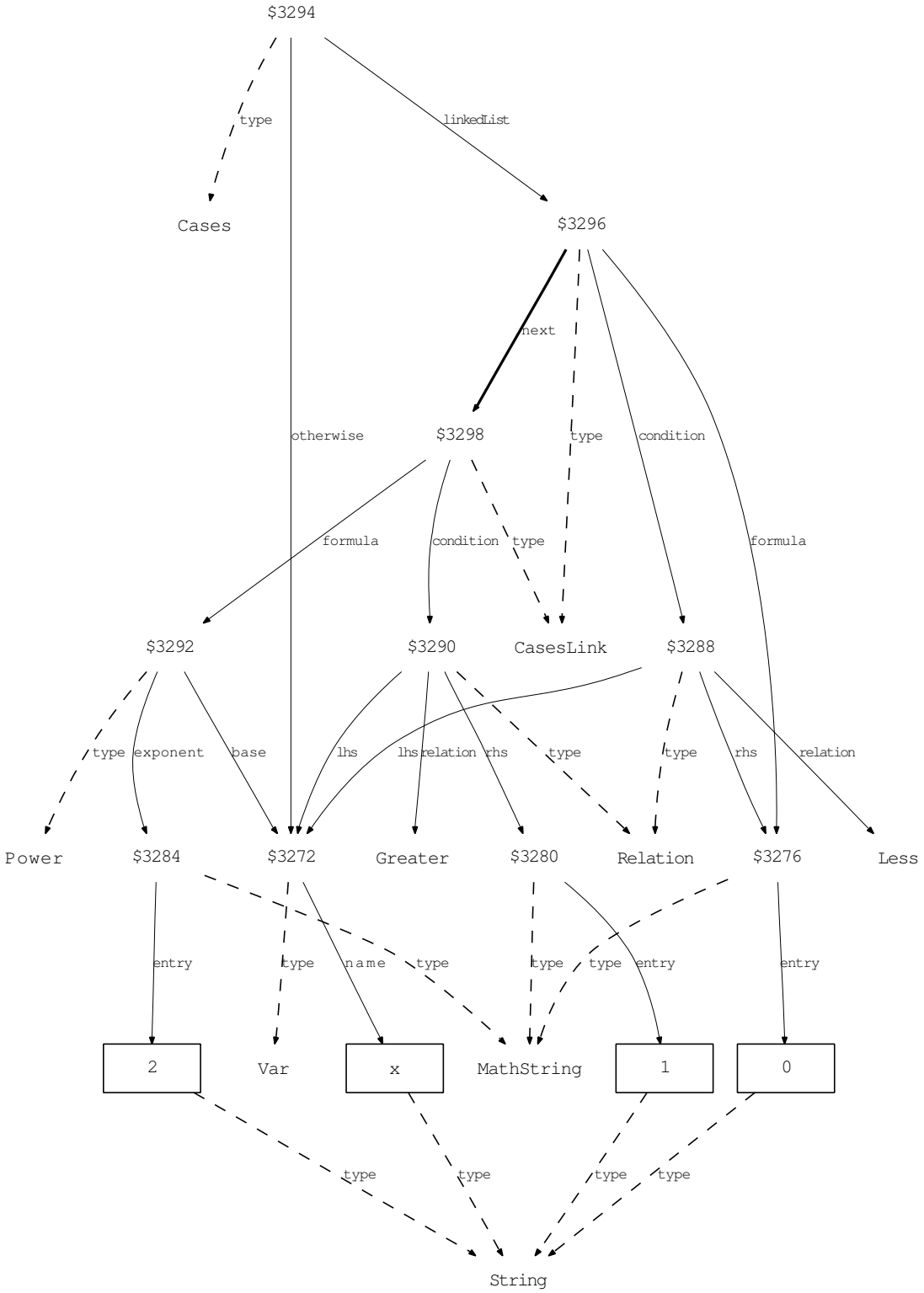

$3254
type
linkedList
Matrix
$3256
next
entry
$3258
type
$3242
entry
type
linkedList
$3248
RowLink
type
$3244
type
type
entry
linkedList
Row
$3236
$3250
next
type
type
linkedList
InvisMult
$3238
entry
next
next
entry
$3252
type
$3246
type
$3240
$3234
entry
entry
type
type
type
type
type
entry
$3224
$3220
ExpLink
$3216
Bracket
$3228
type
type
type
entry
linkedList
type
MathString
$3230
SignedSum
entry
entry
I
sign
next
InvisPlusSign
entry
$3232
type
*
sign
entry
type
type
$3208
MinusSign
$3212
SignedSumLink
0
type
name
type
type
name
\lambda
Var
x
type
type
String

Example 9.

$$\begin{cases} 0 & \text{if } x<0 \\ x^2 & \text{if } x>1 \\ x & \text{otherwise} \end{cases}$$

```
\begin{cases} 0 & \text{if }x{ < }0 \\{x}^{2} & \text{if }x{ >
   }1 \\x & \text{otherwise}\end{cases}
```

```
$3294.type=Cases
$3294.otherwise=$3272
$3294.linkedList=$3296
$3272.type=Var
$3272.name=$3274
$3274.type=String
$3296.type=CasesLink
$3296.next=$3298
$3296.formula=$3276
$3296.condition=$3288
$3276.type=MathString
$3276.entry=$3278
$3278.type=String
$3288.type=Relation
$3288.lhs=$3272
$3288.relation=Less
$3288.rhs=$3276
$3298.type=CasesLink
$3298.formula=$3292
$3298.condition=$3290
$3290.type=Relation
$3290.lhs=$3272
$3290.relation=Greater
$3290.rhs=$3280
$3280.type=MathString
$3280.entry=$3282
$3282.type=String
$3292.type=Power
$3292.base=$3272
$3292.exponent=$3284
$3284.type=MathString
$3284.entry=$3286
$3286.type=String
VALUE($3274) = x
VALUE($3278) = 0
VALUE($3282) = 1
VALUE($3286) = 2
```

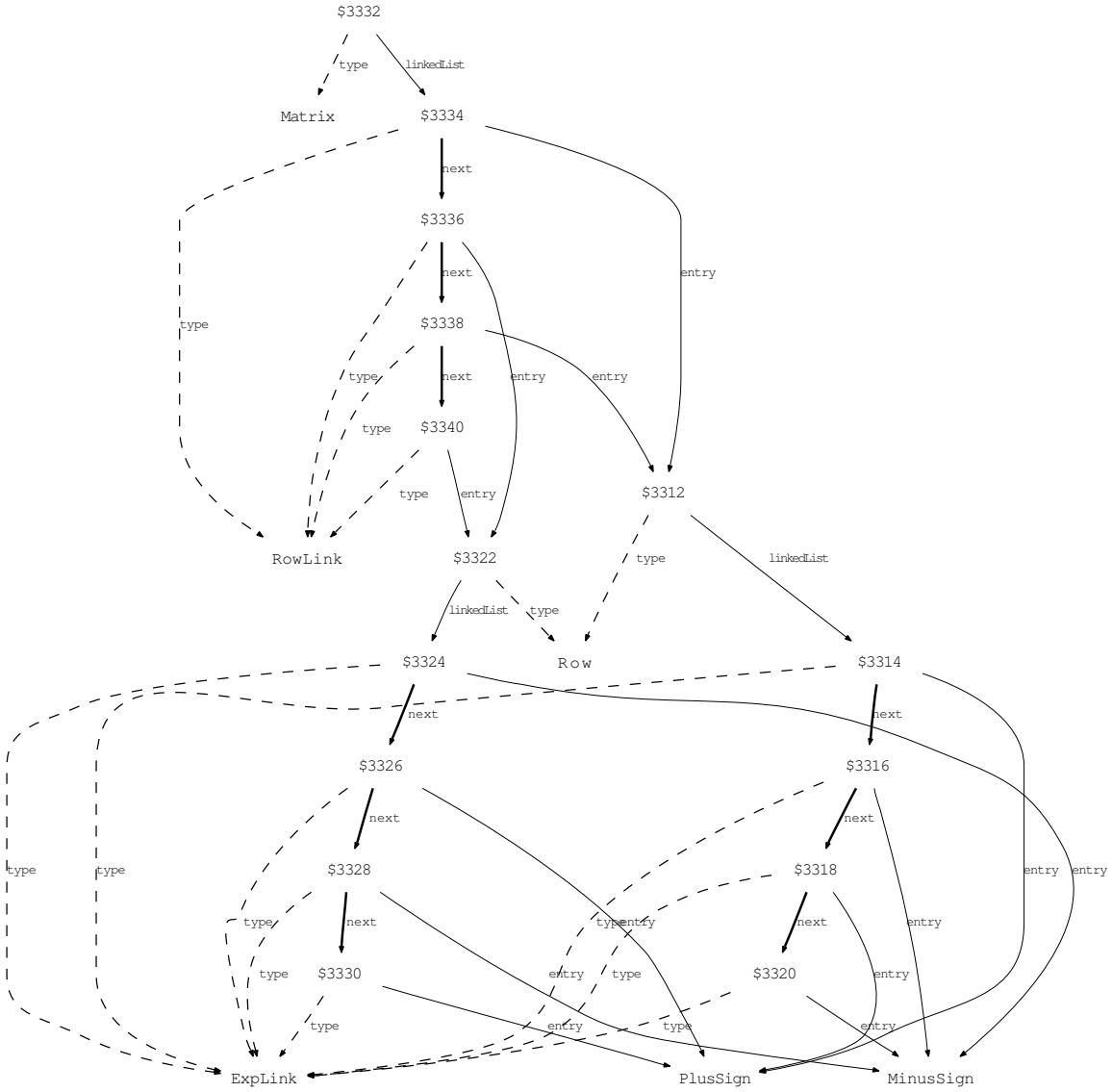

$3294
type
linkedList
Cases
$3296
next
otherwise
$3298
type
condition
formula
condition
type
formula
$3292
$3290
CasesLink
$3288
type
exponent
base
lhs
lhs
relation
rhs
type
type
rhs
relation
Power
$3284
$3272
Greater
$3280
Relation
$3276
Less
entry
type
name
type
type
type
entry
entry
2
Var
x
MathString
1
0
type
type
type
type
String

Example 10.

$$\left(\begin{array}{cccc} + & - & + & - \\ - & + & - & + \\ + & - & + & - \\ - & + & - & + \end{array} \right)$$

```
\left(\begin{array}{cccc} + & - & + & - \\ - & + & - & + \\ + &
   - & + & - \\ - & + & - & + \\ \end{array} \right)
```

```
$3332.type=Matrix
$3332.linkedList=$3334
$3334.type=RowLink
$3334.next=$3336
$3334.entry=$3312
$3312.type=Row
$3312.linkedList=$3314
$3314.type=ExpLink
$3314.next=$3316
$3314.entry=PlusSign
$3316.type=ExpLink
$3316.next=$3318
$3316.entry=MinusSign
$3318.type=ExpLink
$3318.next=$3320
$3318.entry=PlusSign
$3320.type=ExpLink
$3320.entry=MinusSign
$3336.type=RowLink
$3336.next=$3338
$3336.entry=$3322
$3322.type=Row
$3322.linkedList=$3324
$3324.type=ExpLink
$3324.next=$3326
$3324.entry=MinusSign
$3326.type=ExpLink
$3326.next=$3328
$3326.entry=PlusSign
$3328.type=ExpLink
$3328.next=$3330
$3328.entry=MinusSign
$3330.type=ExpLink
$3330.entry=PlusSign
$3338.type=RowLink
$3338.next=$3340
$3338.entry=$3312
$3340.type=RowLink
$3340.entry=$3322
```


$3332
type
linkedList
Matrix
$3334
next
$3336
next
entry
type
$3338
type
next
entry
entry
type
$3340
type
entry
$3312
RowLink
$3322
type
linkedList
linkedList
type
Row
$3324
$3314
next
next
$3326
$3316
next
next
type
type
$3328
$3318
entry
entry
type
next
type
entry
next
entry
type
$3330
entry
type
$3320
entry
type
entry
type
entry
ExpLink
PlusSign
MinusSign

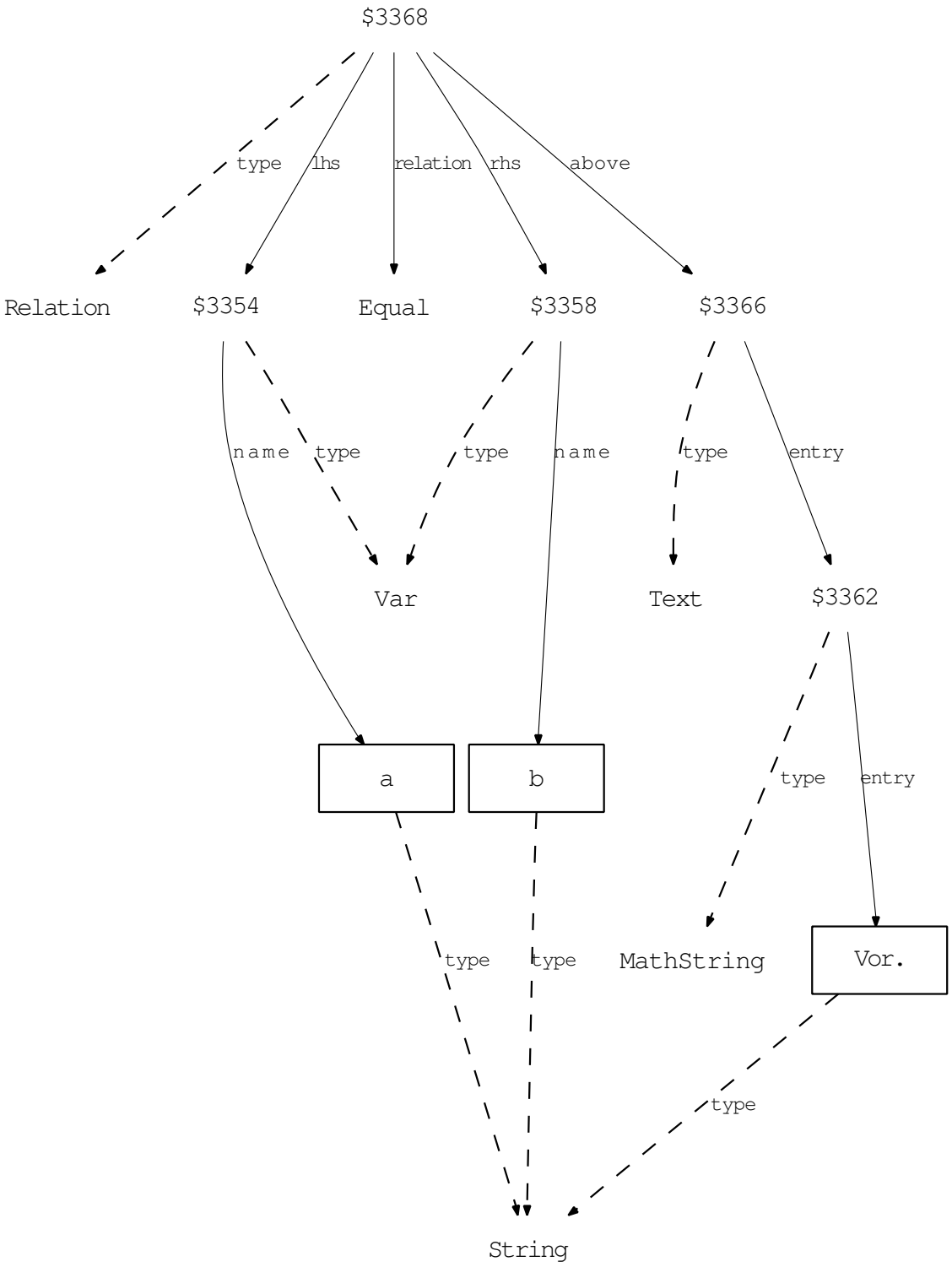
Example 11. Equality with auxiliary information, stored in the node `above`.

$$a \overset{\text{Vor.}}{=} b$$

```
a\stackrel{\text{Vor.}}{=}b
```

```
$3368.type=Relation
$3368.lhs=$3354
$3368.relation=Equal
$3368.rhs=$3358
$3368.above=$3366
$3354.type=Var
$3354.name=$3356
$3356.type=String
$3358.type=Var
$3358.name=$3360
```

```
$3360.type=String
$3366.type=Text
$3366.entry=$3362
$3362.type=MathString
$3362.entry=$3364
$3364.type=String
VALUE($3356) = a
VALUE($3360) = b
VALUE($3364) = Vor.
```

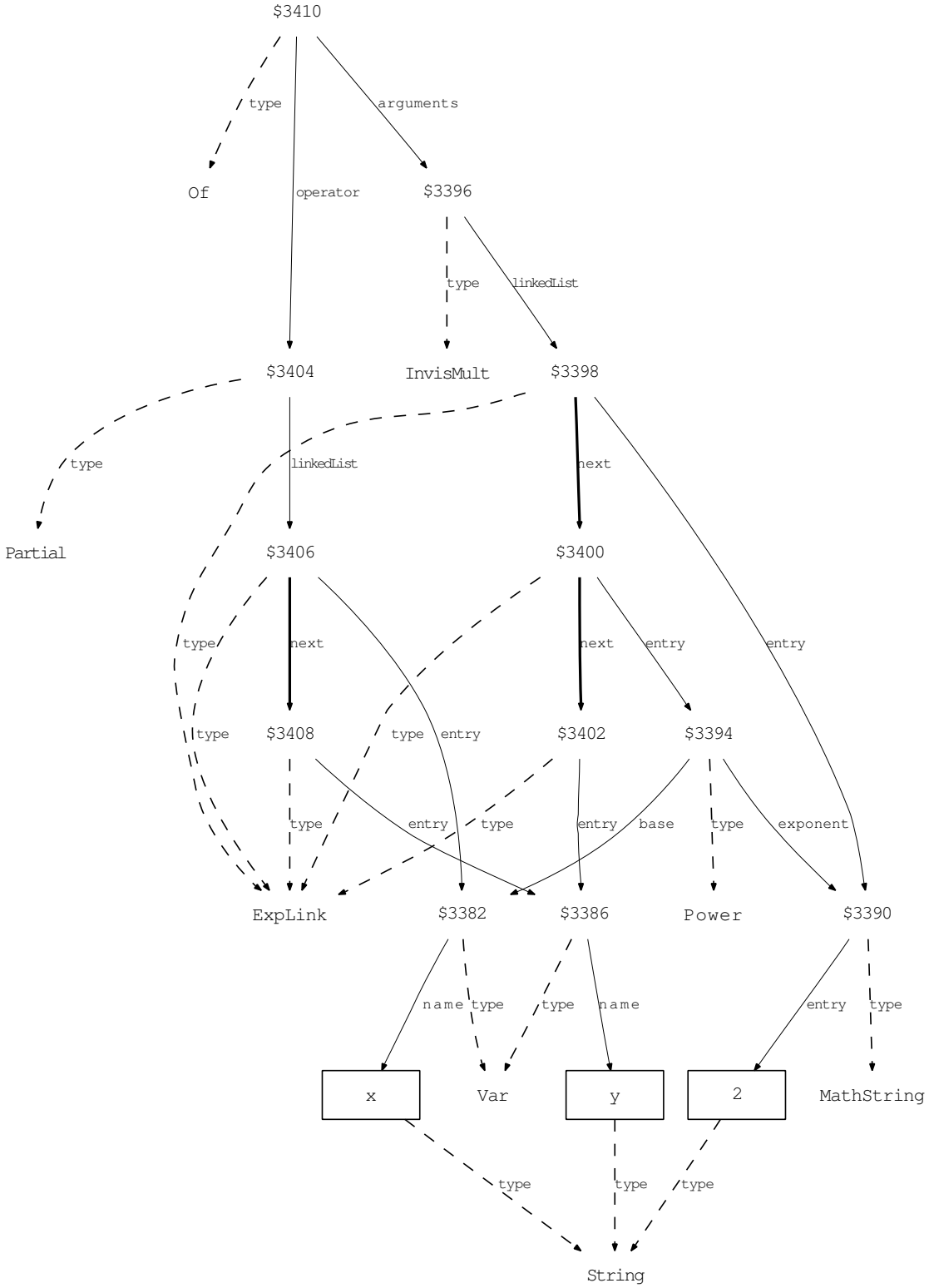

$3368
type
lhs
relation
rhs
above
Relation
$3354
Equal
$3358
$3366
name
type
type
name
type
entry
Var
Text
$3362
a
b
type
entry
type
type
MathString
Vor.
type
String

Example 12.

$$\frac{\partial^2}{\partial x \partial y} 2x^2 y$$

```
\frac{\partial^{2}}{ \partial x \partial y}2 {x}^{2} y
```

```
$3410.type=Of
$3410.operator=$3404
$3410.arguments=$3396
$3396.type=InvisMult
$3396.linkedList=$3398
$3398.type=ExpLink
$3398.next=$3400
$3398.entry=$3390
$3390.type=MathString
$3390.entry=$3392
$3392.type=String
$3400.type=ExpLink
$3400.next=$3402
$3400.entry=$3394
$3394.type=Power
$3394.base=$3382
$3394.exponent=$3390
$3382.type=Var
$3382.name=$3384
$3384.type=String
$3402.type=ExpLink
$3402.entry=$3386
$3386.type=Var
$3386.name=$3388
$3388.type=String
$3404.type=Partial
$3404.linkedList=$3406
$3406.type=ExpLink
$3406.next=$3408
$3406.entry=$3382
$3408.type=ExpLink
$3408.entry=$3386
VALUE($3384) = x
VALUE($3388) = y
VALUE($3392) = 2
```

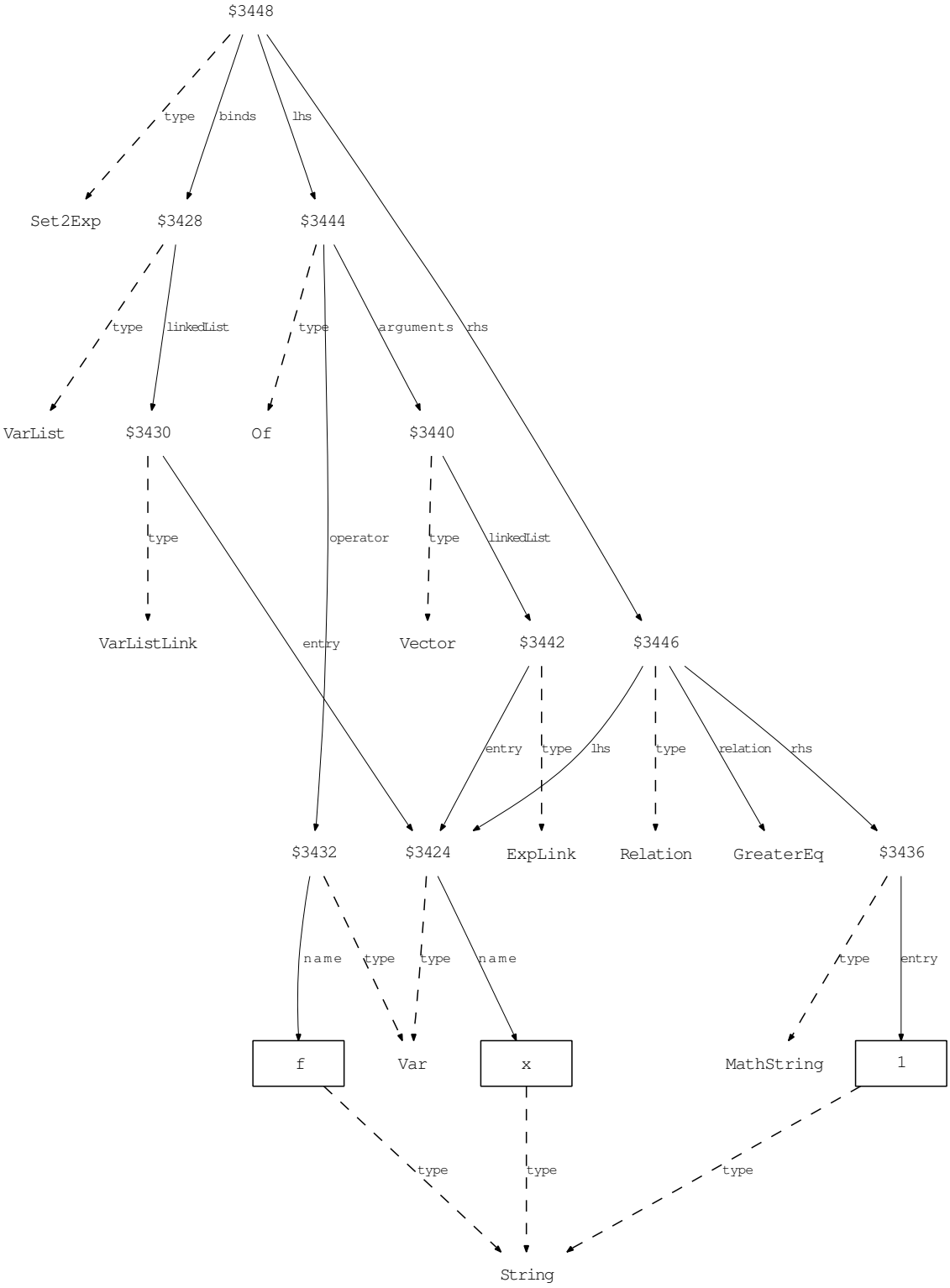

$3410
type
arguments
Of
operator
$3396
type
linkedList
$3404
InvisMult
$3398
type
linkedList
next
Partial
$3406
$3400
type
next
next
entry
entry
type
$3408
type
entry
$3402
$3394
type
entry
type
entry
base
type
exponent
ExpLink
$3382
$3386
Power
$3390
name
type
type
name
entry
type
x
Var
y
2
MathString
type
type
type
String

Example 13. Another typical way of denoting a set: expressions left and right.

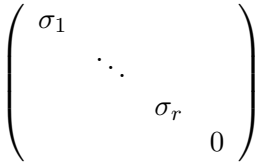
$$\{f\,(x) \mid x{\geq}1\}$$

```
\left\{ f \left(x\right) \mid x{ \geq }1 \right\}
```

```
$3448.type=Set2Exp
$3448.binds=$3428
$3448.lhs=$3444
$3448.rhs=$3446
$3428.type=VarList
$3428.linkedList=$3430
$3430.type=VarListLink
$3430.entry=$3424
$3424.type=Var
$3424.name=$3426
$3426.type=String
$3444.type=Of
$3444.operator=$3432
$3444.arguments=$3440
$3432.type=Var
$3432.name=$3434
$3434.type=String
$3440.type=Vector
$3440.linkedList=$3442
$3442.type=ExpLink
$3442.entry=$3424
$3446.type=Relation
$3446.lhs=$3424
$3446.relation=GreaterEq
$3446.rhs=$3436
$3436.type=MathString
$3436.entry=$3438
$3438.type=String
VALUE($3426) = x
VALUE($3434) = f
VALUE($3438) = 1
```

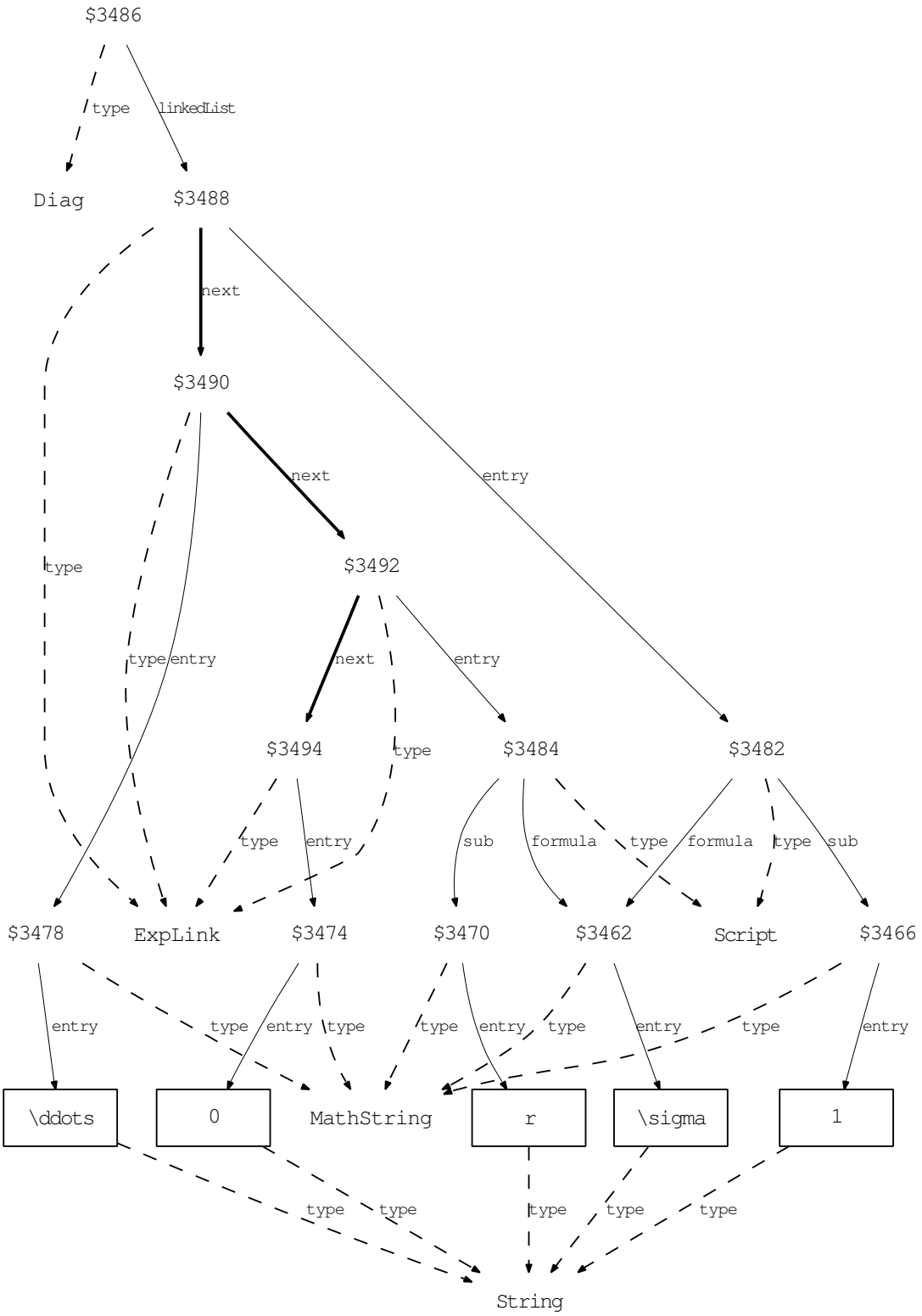

$3448
type
binds
lhs
Set2Exp
$3428
$3444
type
linkedList
type
arguments
rhs
VarList
$3430
Of
$3440
type
operator
type
linkedList
VarListLink
entry
Vector
$3442
$3446
entry
type
lhs
type
relation
rhs
$3432
$3424
ExpLink
Relation
GreaterEq
$3436
name
type
type
name
type
entry
f
Var
x
MathString
1
type
type
type
String

Example 14.



```
\left(\begin{array}{cccc} {}{\sigma}_{1}\\&\ddots\\&&{}{\sigma}_
   {r}\\&&&0\\ \end{array} \right)
```

```
$3486.type=Diag
$3486.linkedList=$3488
$3488.type=ExpLink
$3488.next=$3490
$3488.entry=$3482
$3482.type=Script
$3482.formula=$3462
$3482.sub=$3466
$3462.type=MathString
$3462.entry=$3464
$3464.type=String
$3466.type=MathString
$3466.entry=$3468
$3468.type=String
$3490.type=ExpLink
$3490.next=$3492
$3490.entry=$3478
$3478.type=MathString
$3478.entry=$3480
$3480.type=String
$3492.type=ExpLink
$3492.next=$3494
$3492.entry=$3484
$3484.type=Script
$3484.formula=$3462
$3484.sub=$3470
$3470.type=MathString
$3470.entry=$3472
$3472.type=String
$3494.type=ExpLink
$3494.entry=$3474
$3474.type=MathString
$3474.entry=$3476
$3476.type=String
VALUE($3464) = \sigma
VALUE($3468) = 1
VALUE($3472) = r
VALUE($3476) = 0
VALUE($3480) = \ddots
```

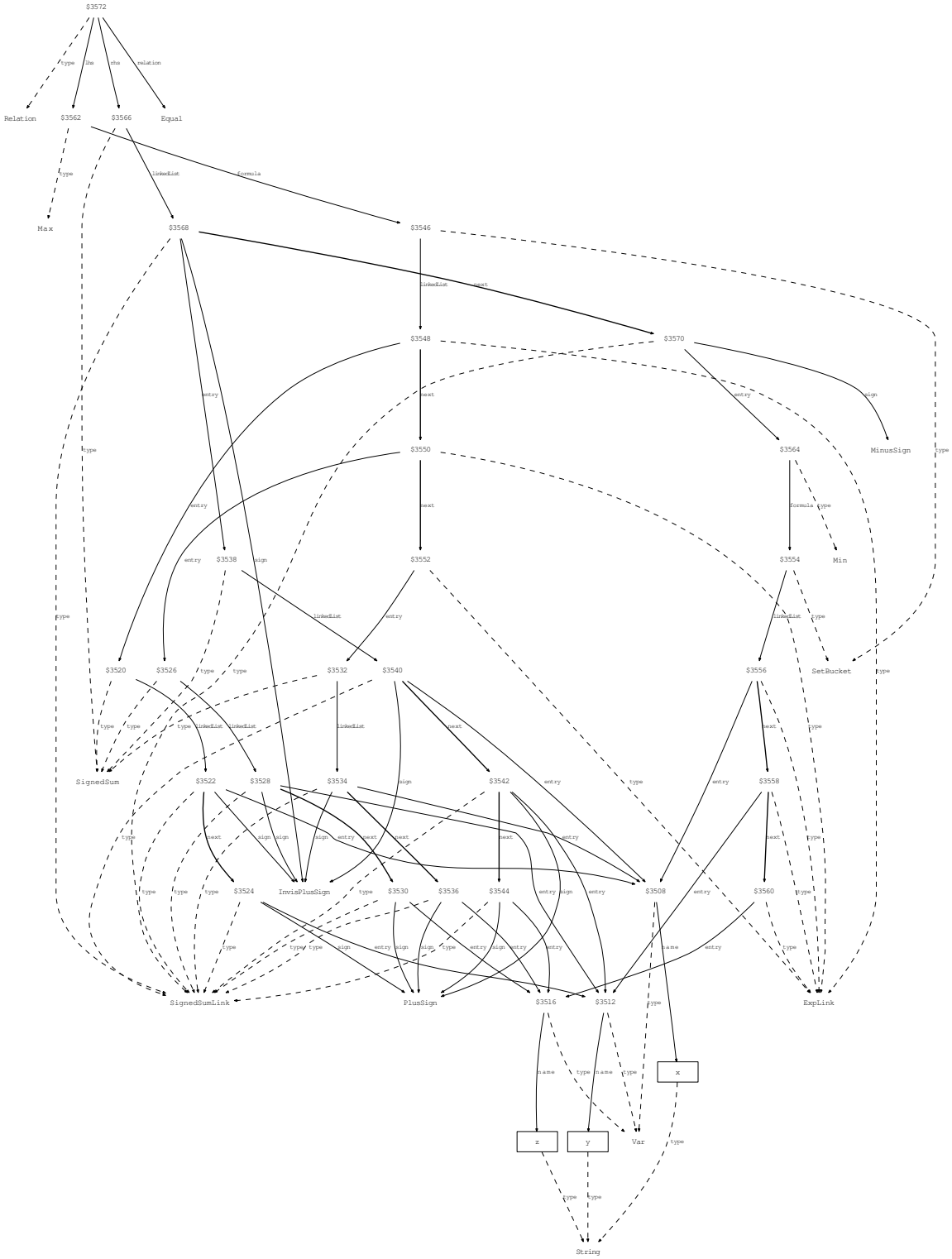

$3486
type
linkedList
Diag
$3488
next
$3490
next
entry
type
$3492
type
entry
next
entry
$3494
type
$3484
$3482
type
entry
sub
formula
type
formula
type
sub
$3478
ExpLink
$3474
$3470
$3462
Script
$3466
entry
type
entry
type
type
entry
type
entry
type
entry
\ddots
0
MathString
r
\sigma
1
type
type
type
type
type
String

Example 15.

$$\max\{x+y, y+z, x+z\}=x+y+z-\min\{x,y,z\}$$

```
\max{ \left\{ x + y , y + z , x + z \right\} }{{=}} x + y + z -
   \min{ \left\{ x , y , z \right\} }
```

```
$3572.type=Relation
$3572.lhs=$3562
$3572.relation=Equal
$3572.rhs=$3566
$3562.type=Max
$3562.formula=$3546
$3546.type=SetBucket
$3546.linkedList=$3548
$3548.type=ExpLink
$3548.next=$3550
$3548.entry=$3520
$3520.type=SignedSum
$3520.linkedList=$3522
$3522.type=SignedSumLink
$3522.next=$3524
$3522.sign=InvisPlusSign
$3522.entry=$3508
$3508.type=Var
$3508.name=$3510
$3510.type=String
$3524.type=SignedSumLink
$3524.sign=PlusSign
$3524.entry=$3512
$3512.type=Var
$3512.name=$3514
$3514.type=String
$3550.type=ExpLink
$3550.next=$3552
$3550.entry=$3526
$3526.type=SignedSum
$3526.linkedList=$3528
$3528.type=SignedSumLink
$3528.next=$3530
$3528.sign=InvisPlusSign
$3528.entry=$3512
$3530.type=SignedSumLink
$3530.sign=PlusSign
$3530.entry=$3516
$3516.type=Var
$3516.name=$3518
$3518.type=String
$3552.type=ExpLink
$3552.entry=$3532
$3532.type=SignedSum
$3532.linkedList=$3534
$3534.type=SignedSumLink
$3534.next=$3536
$3534.sign=InvisPlusSign
$3534.entry=$3508
$3536.type=SignedSumLink
$3536.sign=PlusSign
$3536.entry=$3516
$3566.type=SignedSum
$3566.linkedList=$3568
$3568.type=SignedSumLink
$3568.next=$3570
$3568.sign=InvisPlusSign
$3568.entry=$3538
$3538.type=SignedSum
$3538.linkedList=$3540
$3540.type=SignedSumLink
$3540.next=$3542
$3540.sign=InvisPlusSign
$3540.entry=$3508
$3542.type=SignedSumLink
$3542.next=$3544
$3542.sign=PlusSign
$3542.entry=$3512
$3544.type=SignedSumLink
$3544.sign=PlusSign
$3544.entry=$3516
$3570.type=SignedSumLink
$3570.sign=MinusSign
$3570.entry=$3564
$3564.type=Min
$3564.formula=$3554
$3554.type=SetBucket
$3554.linkedList=$3556
$3556.type=ExpLink
$3556.next=$3558
$3556.entry=$3508
$3558.type=ExpLink
$3558.next=$3560
$3558.entry=$3512
$3560.type=ExpLink
$3560.entry=$3516
VALUE($3510) = x
VALUE($3514) = y
VALUE($3518) = z
```

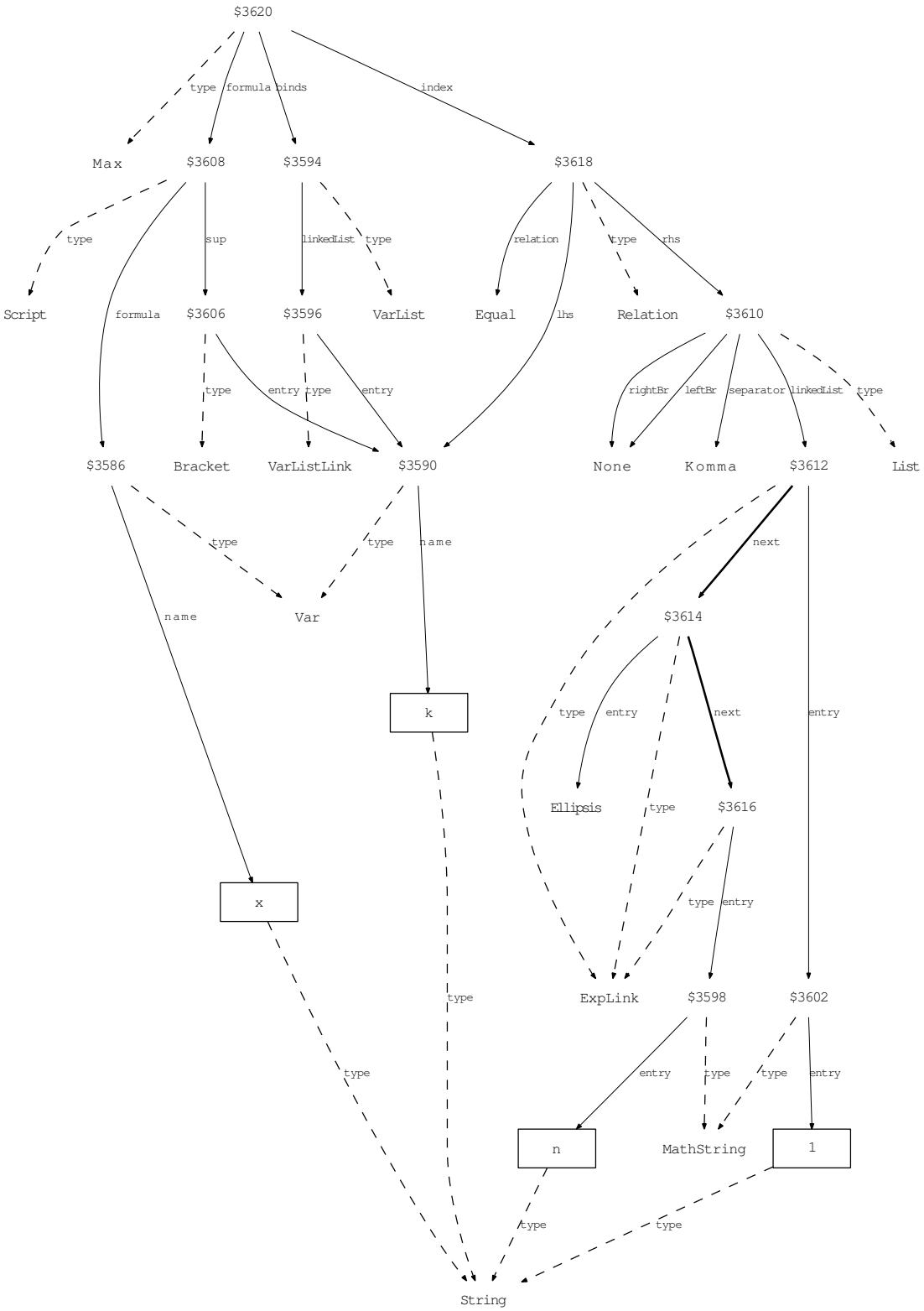

$3572
type
lhs
rhs
relation
Relation
$3562
$3566
Equal
type
linkedList
formula
Max
$3568
$3546
linkedList
next
$3548
$3570
entry
next
entry
sign
$3550
$3564
MinusSign
type
entry
next
formula
type
entry
$3538
sign
$3552
$3554
Min
linkedList
entry
linkedList
type
type
$3520
$3526
type
type
$3532
$3540
$3556
SetBucket
type
type
type
type
linkedList
linkedList
linkedList
next
next
type
SignedSum
$3522
$3528
$3534
sign
$3542
entry
type
entry
$3558
type
next
sign
sign
sign
entry
next
next
next
entry
next
type
type
type
$3524
InvisPlusSign
type
$3530
$3536
$3544
entry
sign
entry
$3508
entry
$3560
type
type
type
type
type
sign
entry
sign
sign
type
entry
sign
entry
entry
name
entry
type
SignedSumLink
PlusSign
$3516
$3512
ExpLink
type
x
name
type
name
type
z
y
Var
type
type
type
String

Example 16.

$$\max_{k=1,\ldots,n} x^{(k)}$$

```
\max_{k{=}1 , \ldots , n}{{}{x}^{\left(k\right)}}
```

```
$3620.type=Max
$3620.formula=$3608
$3620.binds=$3594
$3620.index=$3618
$3594.type=VarList
$3594.linkedList=$3596
$3596.type=VarListLink
$3596.entry=$3590
$3590.type=Var
$3590.name=$3592
$3592.type=String
$3608.type=Script
$3608.formula=$3586
$3608.sup=$3606
$3586.type=Var
$3586.name=$3588
$3588.type=String
$3606.type=Bracket
$3606.entry=$3590
$3618.type=Relation
$3618.lhs=$3590
$3618.relation=Equal
$3618.rhs=$3610
$3610.type=List
$3610.leftBr=None
$3610.separator=Komma
$3610.rightBr=None
$3610.linkedList=$3612
$3612.type=ExpLink
$3612.next=$3614
$3612.entry=$3602
$3602.type=MathString
$3602.entry=$3604
$3604.type=String
$3614.type=ExpLink
$3614.next=$3616
$3614.entry=Ellipsis
$3616.type=ExpLink
$3616.entry=$3598
$3598.type=MathString
$3598.entry=$3600
$3600.type=String
VALUE($3588) = x
VALUE($3592) = k
VALUE($3600) = n
VALUE($3604) = 1
```

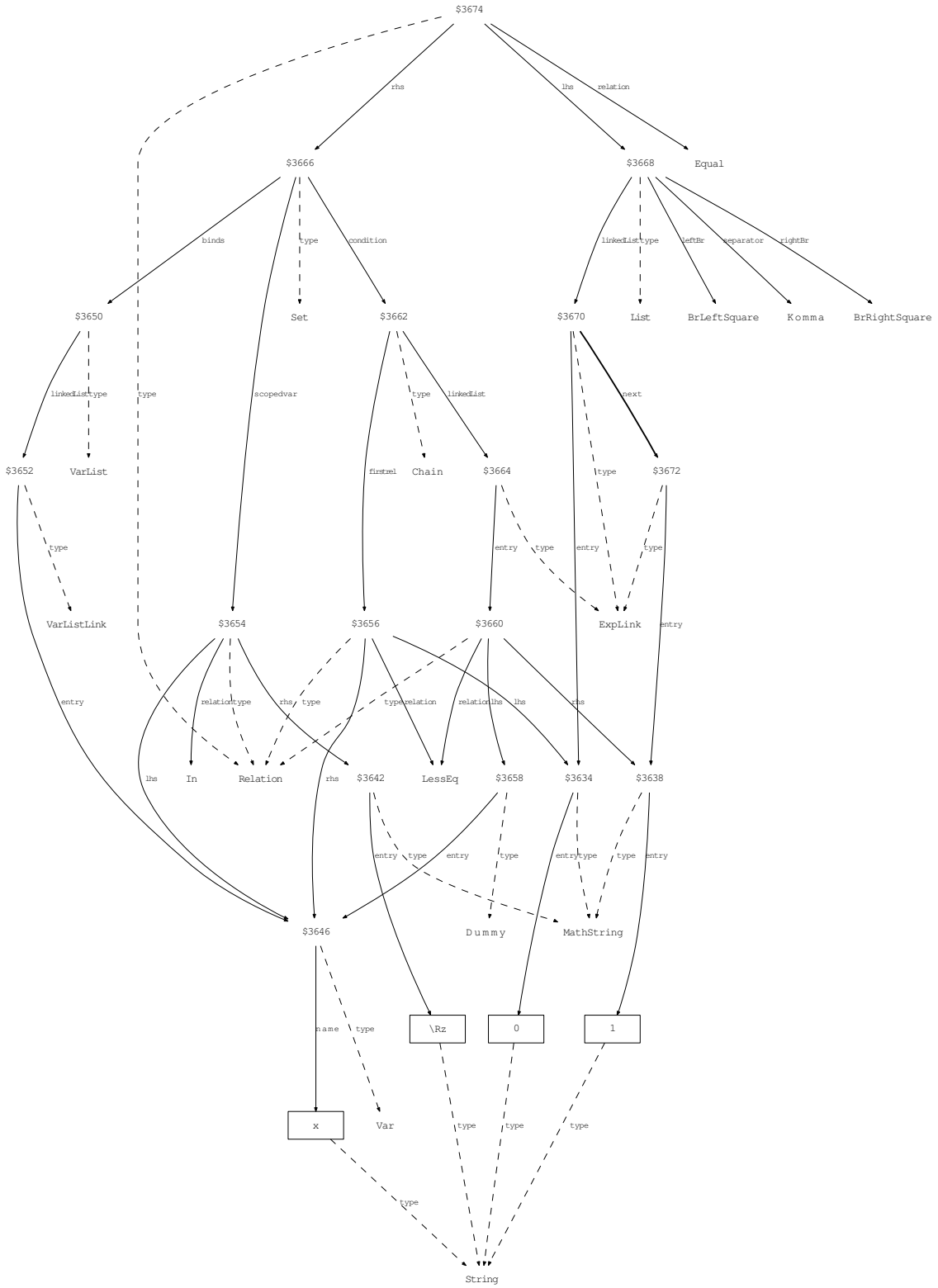

$3620
type
formula
binds
index
Max
$3608
$3594
$3618
type
sup
linkedList
type
relation
type
rhs
Script
formula
$3606
$3596
VarList
Equal
lhs
Relation
$3610
type
entry
type
entry
rightBr
leftBr
separator
linkedList
type
$3586
Bracket
VarListLink
$3590
None
Komma
$3612
List
type
type
name
next
name
Var
$3614
k
type
entry
next
entry
Ellipsis
type
$3616
x
type
entry
type
ExpLink
$3598
$3602
type
entry
type
type
entry
n
MathString
1
type
type
String

Example 17. The interval on the LHS is a list with layout options for the parentheses.

$$[0,1]=\{x\in\mathbb{R}\mid 0\leq x\leq 1\}$$

```
[0 , 1]{{=}} \left\{x{ \in }\Rz \mid 0{ \leq }x{ \leq }1\right\}
```

```
$3674.type=Relation
$3674.lhs=$3668
$3674.relation=Equal
$3674.rhs=$3666
$3666.type=Set
$3666.scopedvar=$3654
$3666.binds=$3650
$3666.condition=$3662
$3650.type=VarList
$3650.linkedList=$3652
$3652.type=VarListLink
$3652.entry=$3646
$3646.type=Var
$3646.name=$3648
$3648.type=String
$3654.type=Relation
$3654.lhs=$3646
$3654.relation=In
$3654.rhs=$3642
$3642.type=MathString
$3642.entry=$3644
$3644.type=String
$3662.type=Chain
$3662.firstrel=$3656
$3662.linkedList=$3664
$3656.type=Relation
$3656.lhs=$3634
$3656.relation=LessEq
$3656.rhs=$3646
$3634.type=MathString
$3634.entry=$3636
$3636.type=String
$3664.type=ExpLink
$3664.entry=$3660
$3660.type=Relation
$3660.lhs=$3658
$3660.relation=LessEq
$3660.rhs=$3638
$3638.type=MathString
$3638.entry=$3640
$3640.type=String
$3658.type=Dummy
$3658.entry=$3646
$3668.type=List
$3668.leftBr=BrLeftSquare
$3668.separator=Komma
$3668.rightBr=BrRightSquare
$3668.linkedList=$3670
$3670.type=ExpLink
$3670.next=$3672
$3670.entry=$3634
$3672.type=ExpLink
$3672.entry=$3638
VALUE($3636) = 0
VALUE($3640) = 1
VALUE($3644) = \Rz
VALUE($3648) = x
```

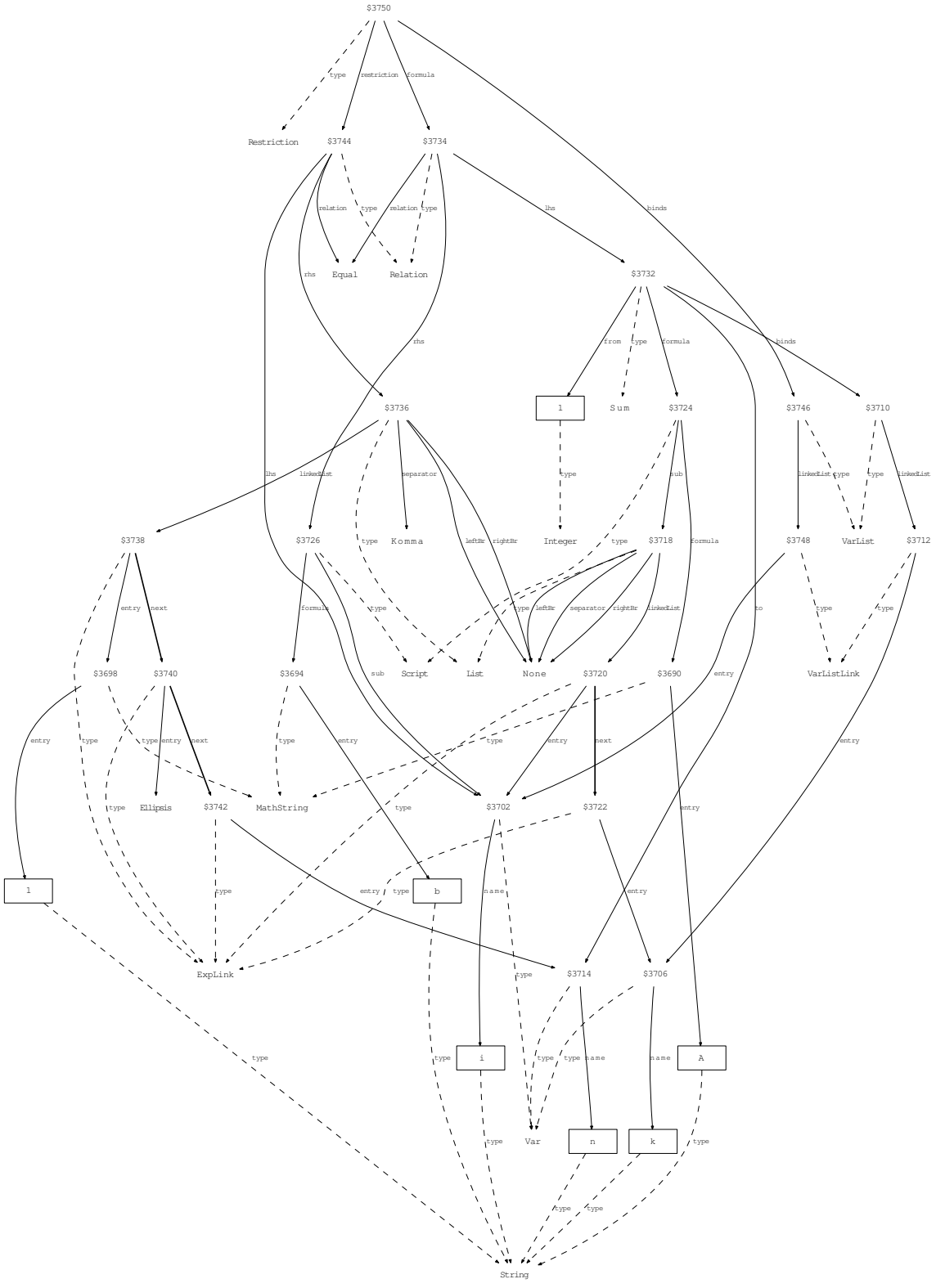

$3674
rhs
lhs
relation
$3666
$3668
Equal
binds
type
condition
linkedList
type
leftBr
separator
rightBr
$3650
Set
$3662
$3670
List
BrLeftSquare
Komma
BrRightSquare
linkedList
type
type
scopedvar
type
linkedList
next
$3652
VarList
firstnel
Chain
$3664
type
$3672
type
type
entry
type
entry
type
VarListLink
$3654
$3656
$3660
ExpLink
entry
entry
relation
type
rhs
type
type
relation
relation
lhs
lhs
rhs
lhs
In
Relation
rhs
$3642
LessEq
$3658
$3634
$3638
entry
type
entry
type
entry
type
type
entry
$3646
Dummy
MathString
name
type
\Rz
0
1
x
Var
type
type
type
type
String

Example 18.

$$\sum_{k=1}^{n} A_{ik}=b_i \qquad (i=1,\ldots,n)$$

```
\sum_{k = 1 }^{n} {}{A}_{ik} {{=}}{}{b}_{i} \qquad ( i{{=}}1 , \
   ldots , n )
```

```
$3750.type=Restriction
$3750.formula=$3734
$3750.binds=$3746
$3750.restriction=$3744
$3734.type=Relation
$3734.lhs=$3732
$3734.relation=Equal
$3734.rhs=$3726
$3726.type=Script
$3726.formula=$3694
$3726.sub=$3702
$3694.type=MathString
$3694.entry=$3696
$3696.type=String
$3702.type=Var
$3702.name=$3704
$3704.type=String
$3732.type=Sum
$3732.formula=$3724
$3732.binds=$3710
$3732.from=$3688
$3732.to=$3714
$3688.type=Integer
$3710.type=VarList
$3710.linkedList=$3712
$3712.type=VarListLink
$3712.entry=$3706
$3706.type=Var
$3706.name=$3708
$3708.type=String
$3714.type=Var
$3714.name=$3716
$3716.type=String
$3724.type=Script
$3724.formula=$3690
$3724.sub=$3718
$3690.type=MathString
$3690.entry=$3692
$3692.type=String
$3718.type=List
$3718.leftBr=None
$3718.separator=None
$3718.rightBr=None
$3718.linkedList=$3720
$3720.type=ExpLink
$3720.next=$3722
$3720.entry=$3702
$3722.type=ExpLink
$3722.entry=$3706
$3744.type=Relation
$3744.lhs=$3702
$3744.relation=Equal
$3744.rhs=$3736
$3736.type=List
$3736.leftBr=None
$3736.separator=Komma
$3736.rightBr=None
$3736.linkedList=$3738
$3738.type=ExpLink
$3738.next=$3740
$3738.entry=$3698
$3698.type=MathString
$3698.entry=$3700
$3700.type=String
$3740.type=ExpLink
$3740.next=$3742
$3740.entry=Ellipsis
$3742.type=ExpLink
$3742.entry=$3714
$3746.type=VarList
$3746.linkedList=$3748
$3748.type=VarListLink
$3748.entry=$3702
VALUE($3688) = 1
VALUE($3692) = A
VALUE($3696) = b
VALUE($3700) = 1
VALUE($3704) = i
VALUE($3708) = k
VALUE($3716) = n
```

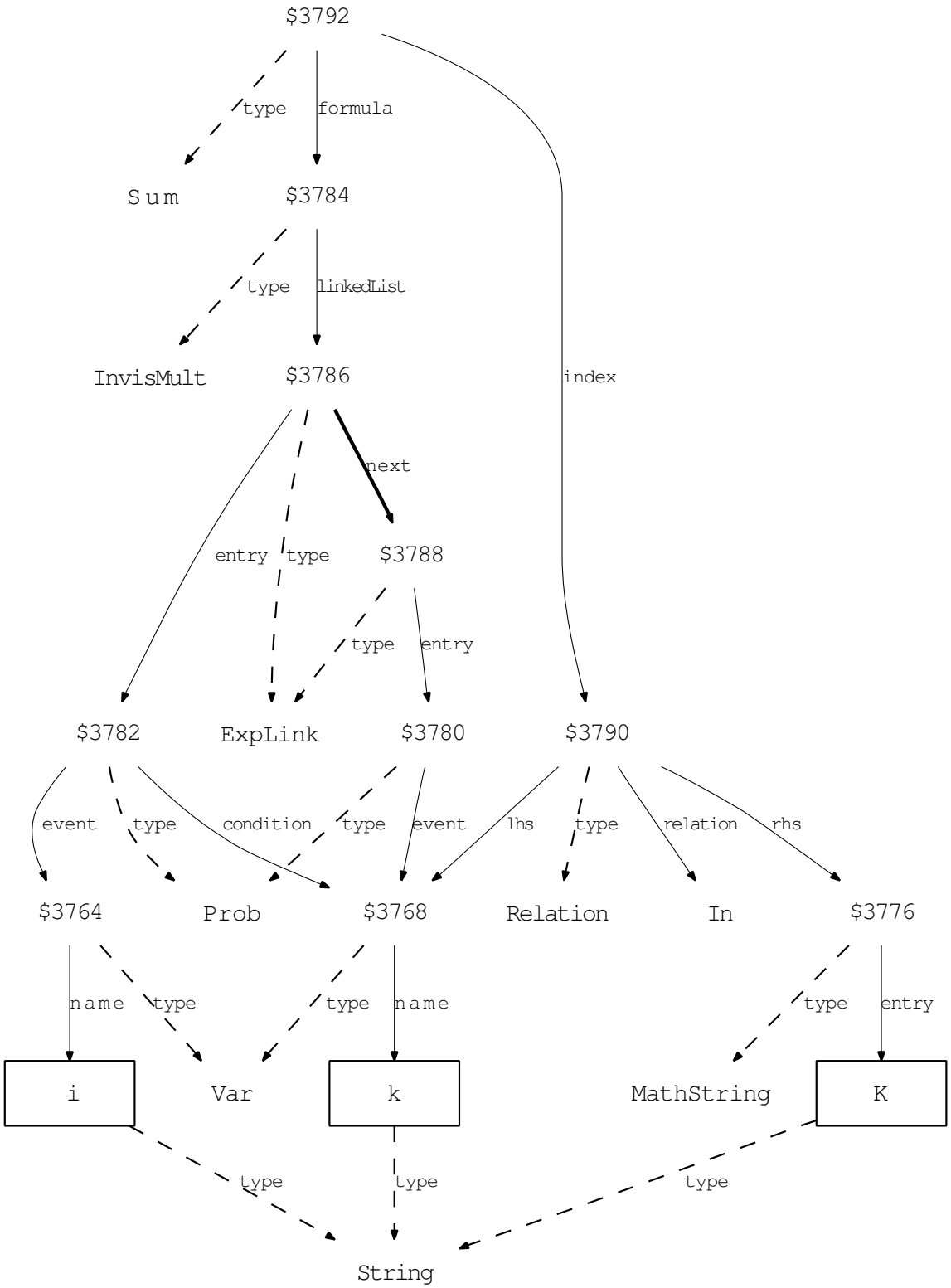

$3750
type
restriction
formula
Restriction
$3744
$3734
relation
type
relation
type
lhs
binds
rhs
Equal
Relation
$3732
rhs
from
type
formula
binds
$3736
1
Sum
$3724
$3746
$3710
lhs
linkedList
separator
type
sub
linkedList
type
type
linkedList
$3738
$3726
type
Komma
leftBr
rightBr
Integer
type
$3718
formula
$3748
VarList
$3712
entry
next
formula
type
type
type
leftBr
separator
rightBr
linkedList
to
type
type
$3698
$3740
$3694
sub
Script
List
None
$3720
$3690
entry
VarListLink
entry
type
type
entry
next
type
type
entry
type
entry
next
entry
type
Ellipsis
$3742
MathString
type
$3702
$3722
entry
1
type
entry
type
b
name
entry
ExpLink
type
$3714
$3706
type
type
i
type
type
name
name
A
type
Var
n
k
type
type
type
String

Example 19.

$$\sum_{k \in K} \Pr(i|k)\Pr(k)$$

```
\sum_{k{ \in }K} \text{Pr}(i|k) \text{Pr}(k)
```

```
$3792.type=Sum
$3792.formula=$3784
$3792.index=$3790
$3784.type=InvisMult
$3784.linkedList=$3786
$3786.type=ExpLink
$3786.next=$3788
$3786.entry=$3782
$3782.type=Prob
$3782.condition=$3768
$3782.event=$3764
$3764.type=Var
$3764.name=$3766
$3766.type=String
$3768.type=Var
$3768.name=$3770
$3770.type=String
$3788.type=ExpLink
$3788.entry=$3780
$3780.type=Prob
$3780.event=$3768
$3790.type=Relation
$3790.lhs=$3768
$3790.relation=In
$3790.rhs=$3776
$3776.type=MathString
$3776.entry=$3778
$3778.type=String
VALUE($3766) = i
VALUE($3770) = k
VALUE($3778) = K
```

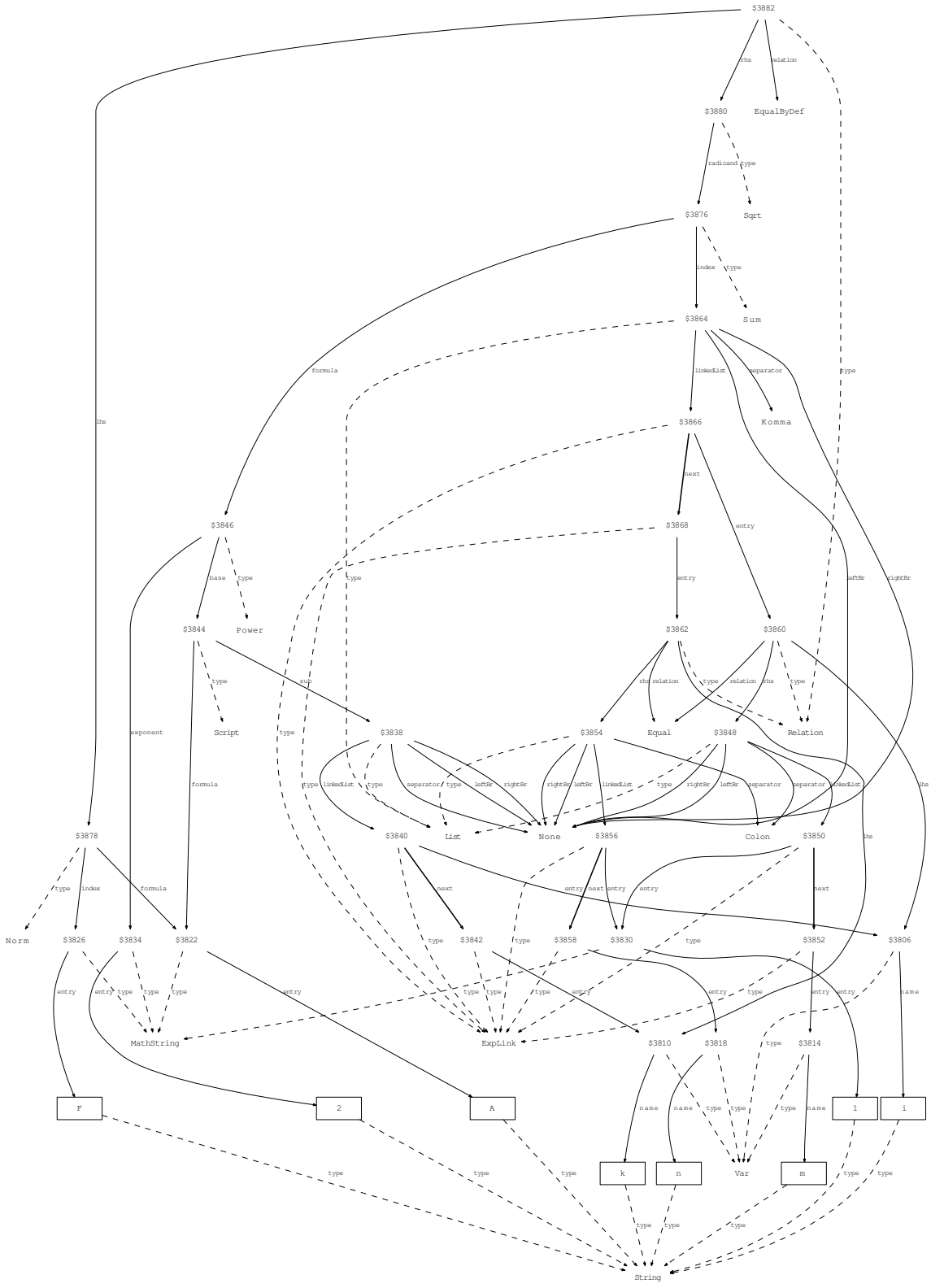

$3792
type
formula
Sum
$3784
type
linkedList
InvisMult
$3786
index
next
entry
type
$3788
type
entry
$3782
ExpLink
$3780
$3790
event
type
condition
type
event
lhs
type
relation
rhs
$3764
Prob
$3768
Relation
In
$3776
name
type
type
name
type
entry
i
Var
k
MathString
K
type
type
type
String

Example 20.

$$\|A\|_F := \sqrt{\sum_{i=1:m,k=1:n} {A_{ik}}^2}$$

```
\|A\|_{F}{ := }\sqrt{ \sum_{i{{=}}1 : m , k{{=}}1 : n} {{}{A}_{
    ik}}^{2} }
```

```
$3882.type=Relation
$3882.lhs=$3878
$3882.relation=EqualByDef
$3882.rhs=$3880
$3878.type=Norm
$3878.formula=$3822
$3878.index=$3826
$3822.type=MathString
$3822.entry=$3824
$3824.type=String
$3826.type=MathString
$3826.entry=$3828
$3828.type=String
$3880.type=Sqrt
$3880.radicand=$3876
$3876.type=Sum
$3876.formula=$3846
$3876.index=$3864
$3846.type=Power
$3846.base=$3844
$3846.exponent=$3834
$3834.type=MathString
$3834.entry=$3836
$3836.type=String
$3844.type=Script
$3844.formula=$3822
$3844.sub=$3838
$3838.type=List
$3838.leftBr=None
$3838.separator=None
$3838.rightBr=None
$3838.linkedList=$3840
$3840.type=ExpLink
$3840.next=$3842
$3840.entry=$3806
$3806.type=Var
$3806.name=$3808
$3808.type=String
$3842.type=ExpLink
$3842.entry=$3810
$3810.type=Var
$3810.name=$3812
$3812.type=String
$3864.type=List
$3864.leftBr=None
$3864.separator=Komma
$3864.rightBr=None
$3864.linkedList=$3866
$3866.type=ExpLink
$3866.next=$3868
$3866.entry=$3860
$3860.type=Relation
$3860.lhs=$3806
$3860.relation=Equal
$3860.rhs=$3848
$3848.type=List
$3848.leftBr=None
$3848.separator=Colon
$3848.rightBr=None
$3848.linkedList=$3850
$3850.type=ExpLink
$3850.next=$3852
$3850.entry=$3830
$3830.type=MathString
$3830.entry=$3832
$3832.type=String
$3852.type=ExpLink
$3852.entry=$3814
$3814.type=Var
$3814.name=$3816
$3816.type=String
$3868.type=ExpLink
$3868.entry=$3862
$3862.type=Relation
$3862.lhs=$3810
$3862.relation=Equal
$3862.rhs=$3854
$3854.type=List
$3854.leftBr=None
$3854.separator=Colon
$3854.rightBr=None
$3854.linkedList=$3856
$3856.type=ExpLink
$3856.next=$3858
$3856.entry=$3830
$3858.type=ExpLink
$3858.entry=$3818
$3818.type=Var
$3818.name=$3820
$3820.type=String
VALUE($3808) = i
VALUE($3812) = k
VALUE($3816) = m
VALUE($3820) = n
VALUE($3824) = A
VALUE($3828) = F
VALUE($3832) = 1
VALUE($3836) = 2
```

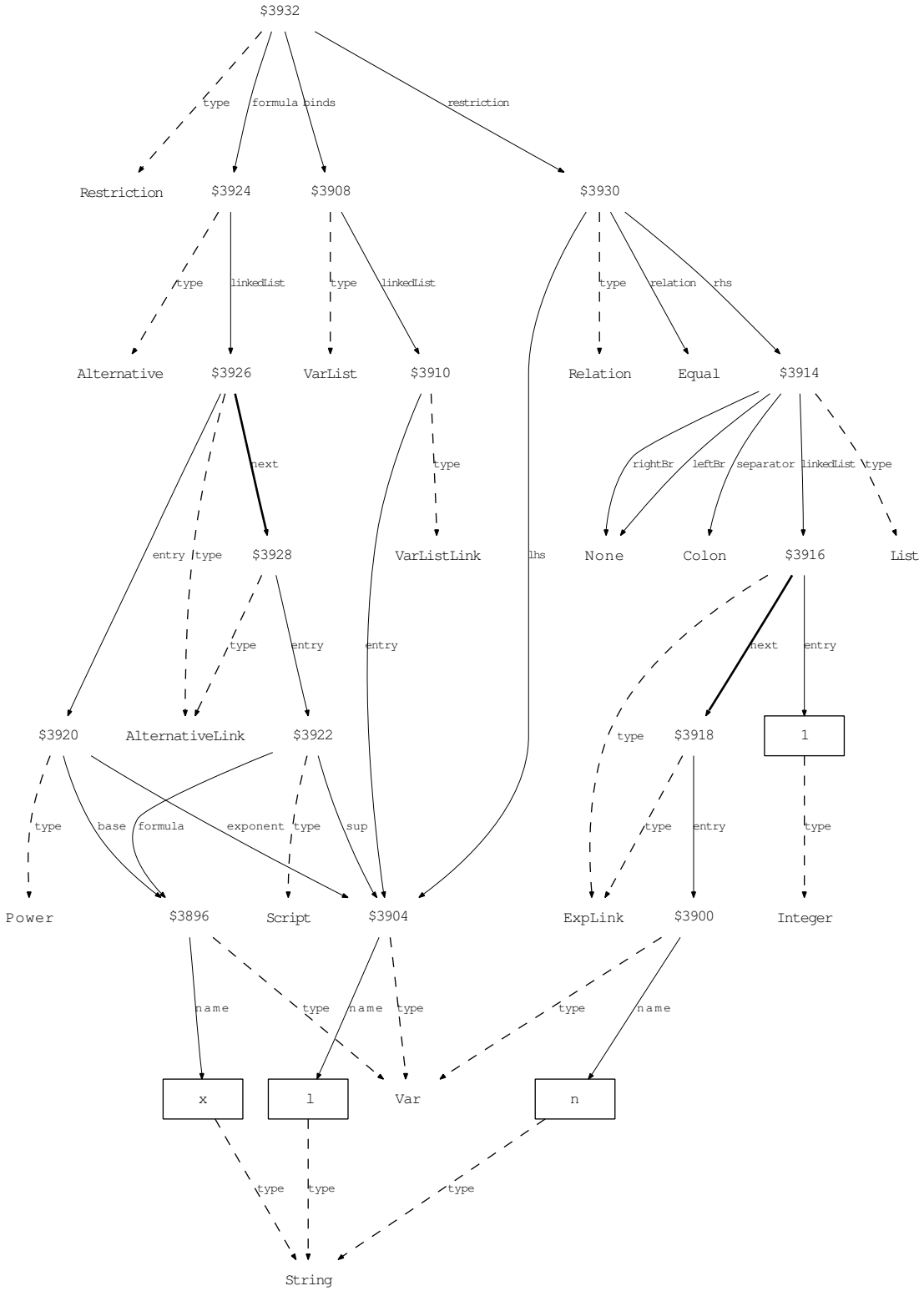

$3882
lhs
relation
$3880
EqualByDef
radicand
type
$3876
Sqrt
index
type
$3864
Sum
formula
linkedList
separator
type
$3866
Komma
lhs
next
$3846
$3868
entry
base
type
$3844
Power
type
$3862
$3860
leftBr
rightBr
exponent
Script
type
$3838
rhs
relation
type
$3854
Equal
$3848
Relation
formula
type
linkedList
separator
leftBr
rightBr
$3878
List
None
$3856
Colon
$3850
$3840
lhs
type
index
formula
next
entry
Norm
$3826
$3834
$3822
$3842
$3858
$3830
$3852
$3806
MathString
ExpLink
$3810
$3818
$3814
name
F
2
A
1
i
k
n
Var
m
String

Example 21.

$$x^{l} \qquad (l{=}1 : n)$$

```
{x}^{l} \qquad ( l{{=}} 1 : n )
```

```
$3932.type=Restriction
$3932.formula=$3924
$3932.binds=$3908
$3932.restriction=$3930
$3908.type=VarList
$3908.linkedList=$3910
$3910.type=VarListLink
$3910.entry=$3904
$3904.type=Var
$3904.name=$3906
$3906.type=String
$3924.type=Alternative
$3924.linkedList=$3926
$3926.type=AlternativeLink
$3926.next=$3928
$3926.entry=$3920
$3920.type=Power
$3920.base=$3896
$3920.exponent=$3904
$3896.type=Var
$3896.name=$3898
$3898.type=String
$3928.type=AlternativeLink
$3928.entry=$3922
$3922.type=Script
$3922.formula=$3896
$3922.sup=$3904
$3930.type=Relation
$3930.lhs=$3904
$3930.relation=Equal
$3930.rhs=$3914
$3914.type=List
$3914.leftBr=None
$3914.separator=Colon
$3914.rightBr=None
$3914.linkedList=$3916
$3916.type=ExpLink
$3916.next=$3918
$3916.entry=$3912
$3912.type=Integer
$3918.type=ExpLink
$3918.entry=$3900
$3900.type=Var
$3900.name=$3902
$3902.type=String
VALUE($3898) = x
VALUE($3902) = n
VALUE($3906) = l
VALUE($3912) = 1
```

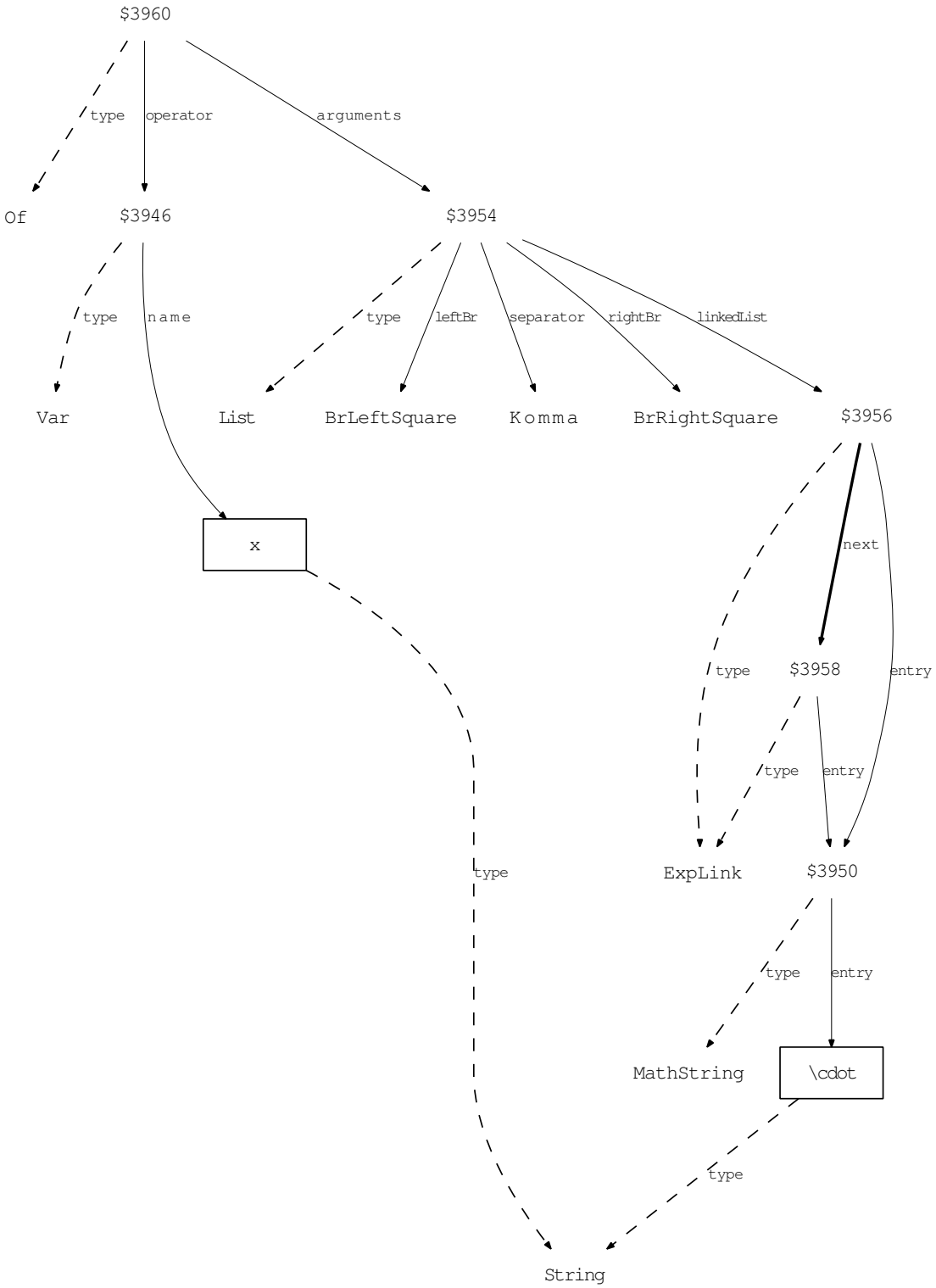

$3932
type
formula
binds
restriction
Restriction
$3924
$3908
$3930
type
linkedList
type
linkedList
type
relation
rhs
Alternative
$3926
VarList
$3910
Relation
Equal
$3914
next
type
rightBr
leftBr
separator
linkedList
type
entry
type
$3928
VarListLink
lhs
None
Colon
$3916
List
type
entry
entry
next
entry
$3920
AlternativeLink
$3922
type
$3918
1
type
base
formula
exponent
type
sup
type
entry
type
Power
$3896
Script
$3904
ExpLink
$3900
Integer
name
type
name
type
type
name
x
1
Var
n
type
type
type
String

Example 22.

$$x[\cdot, \cdot]$$

```
x[\cdot , \cdot]
```

```
$3960.type=Of
$3960.operator=$3946
$3960.arguments=$3954
$3946.type=Var
$3946.name=$3948
$3948.type=String
$3954.type=List
$3954.leftBr=BrLeftSquare
$3954.separator=Komma
$3954.rightBr=BrRightSquare
$3954.linkedList=$3956
$3956.type=ExpLink
$3956.next=$3958
$3956.entry=$3950
$3950.type=MathString
$3950.entry=$3952
$3952.type=String
$3958.type=ExpLink
$3958.entry=$3950
VALUE($3948) = x
VALUE($3952) = \cdot
```

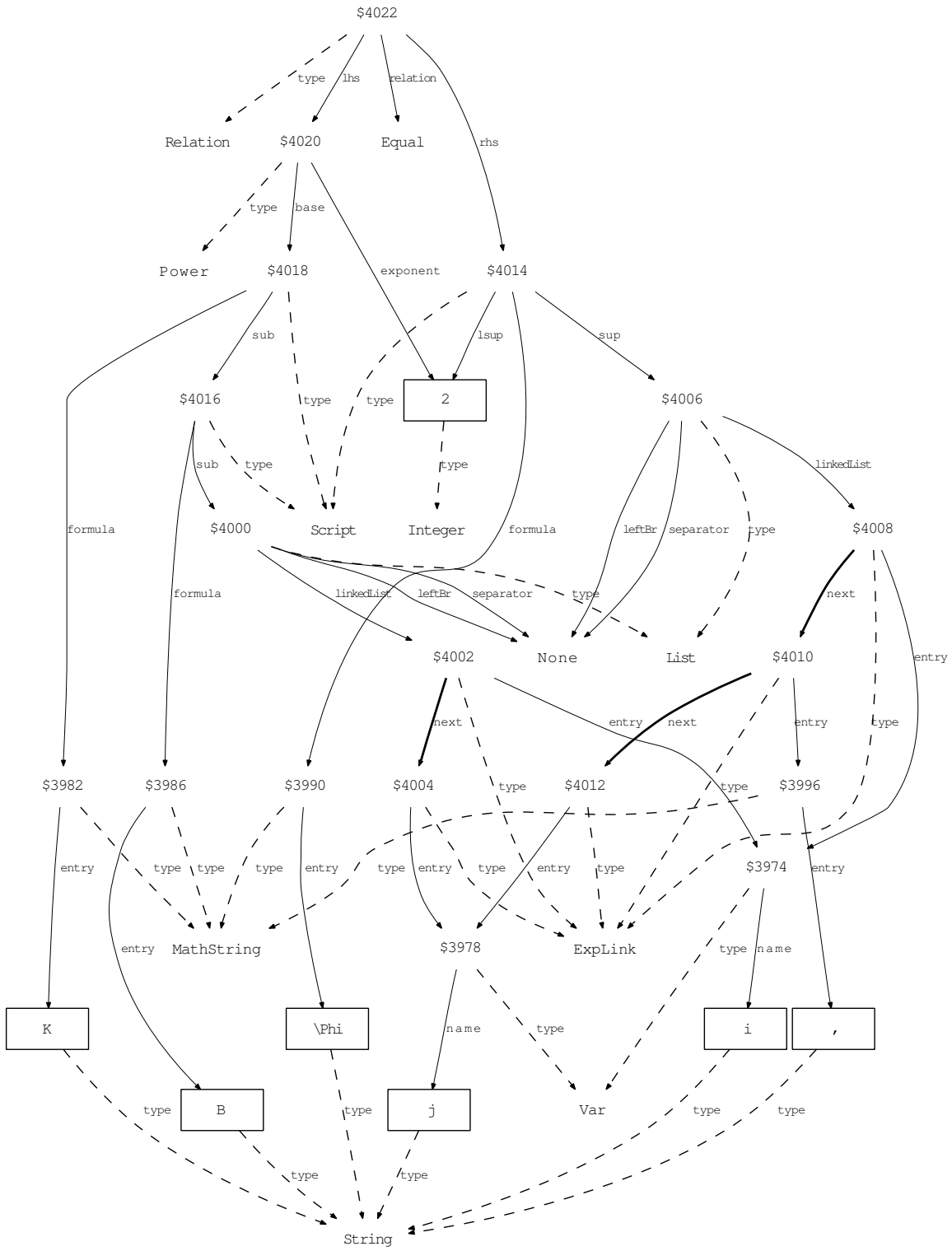

$3960
type
operator
arguments
Of
$3946
$3954
type
name
type
leftBr
separator
rightBr
linkedList
Var
List
BrLeftSquare
Komma
BrRightSquare
$3956
x
next
type
$3958
entry
type
entry
type
ExpLink
$3950
type
entry
MathString
\cdot
type
String

Example 23.

$$K_{B_{ij}}{}^{2}{=}^{2}\Phi^{i,j}$$

```
{{}{K}_{{}{B}_{ij}}}^{ 2 }{{=}}{}^{ 2 }{\Phi}^{i,j}
```

```
$4022.type=Relation
$4022.lhs=$4020
$4022.relation=Equal
$4022.rhs=$4014
$4014.type=Script
$4014.formula=$3990
$4014.sup=$4006
$4014.lsup=$3994
$3990.type=MathString
$3990.entry=$3992
$3992.type=String
$3994.type=Integer
$4006.type=List
$4006.leftBr=None
$4006.separator=None
$4006.linkedList=$4008
$4008.type=ExpLink
$4008.next=$4010
$4008.entry=$3974
$3974.type=Var
$3974.name=$3976
$3976.type=String
$4010.type=ExpLink
$4010.next=$4012
$4010.entry=$3996
$3996.type=MathString
$3996.entry=$3998
$3998.type=String
$4012.type=ExpLink
$4012.entry=$3978
$3978.type=Var
$3978.name=$3980
$3980.type=String
$4020.type=Power
$4020.base=$4018
$4020.exponent=$3994
$4018.type=Script
$4018.formula=$3982
$4018.sub=$4016
$3982.type=MathString
$3982.entry=$3984
$3984.type=String
$4016.type=Script
$4016.formula=$3986
$4016.sub=$4000
$3986.type=MathString
$3986.entry=$3988
$3988.type=String
$4000.type=List
$4000.leftBr=None
$4000.separator=None
$4000.linkedList=$4002
$4002.type=ExpLink
$4002.next=$4004
$4002.entry=$3974
$4004.type=ExpLink
$4004.entry=$3978
VALUE($3976) = i
VALUE($3980) = j
VALUE($3984) = K
VALUE($3988) = B
VALUE($3992) = \Phi
VALUE($3994) = 2
VALUE($3998) = ,
```

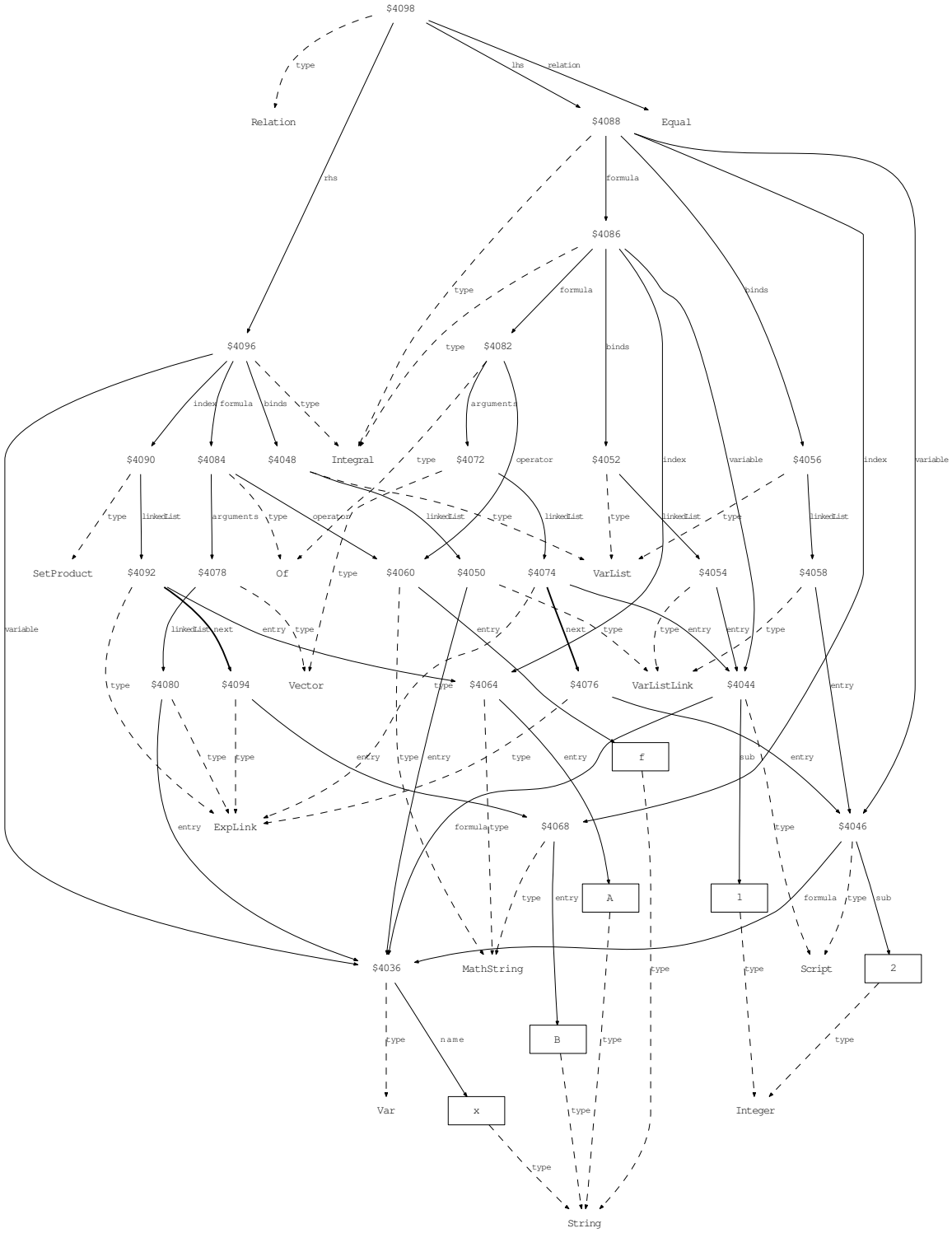

$4022
type
lhs
relation
rhs
Relation
$4020
Equal
type
base
Power
$4018
exponent
$4014
sub
lsup
sup
$4016
type
type
2
$4006
sub
type
type
linkedList
formula
$4000
Script
Integer
formula
leftBr
separator
type
$4008
formula
linkedList
leftBr
separator
type
next
$4002
None
List
$4010
entry
next
entry
next
entry
type
$3982
$3986
$3990
$4004
type
$4012
type
$3996
entry
type
type
type
entry
type
entry
type
entry
type
$3974
entry
entry
MathString
$3978
ExpLink
type
name
K
\Phi
name
type
i
,
type
B
type
j
Var
type
type
type
type
String

Example 24.

$$\int_B \int_A f\left(x_1, x_2\right)\ \mathrm{d}x_1\ \mathrm{d}x_2 = \int_{A \times B} f\left(x\right)\ \mathrm{d}x$$

```
\int_{B} \int_{A} f \left({}{x}_{ 1 } , {}{x}_{ 2 }\right) ~ \
   mathrm{d}{}{x}_{ 1 } ~ \mathrm{d}{}{x}_{ 2 }{{=}}\int_{A \
   times B} f \left(x\right) ~ \mathrm{d}x
```

```
$4098.type=Relation
$4098.lhs=$4088
$4098.relation=Equal
$4098.rhs=$4096
$4088.type=Integral
$4088.formula=$4086
$4088.binds=$4056
$4088.variable=$4046
$4088.index=$4068
$4046.type=Script
$4046.formula=$4036
$4046.sub=$4042
$4036.type=Var
$4036.name=$4038
$4038.type=String
$4042.type=Integer
$4056.type=VarList
$4056.linkedList=$4058
$4058.type=VarListLink
$4058.entry=$4046
$4068.type=MathString
$4068.entry=$4070
$4070.type=String
$4086.type=Integral
$4086.formula=$4082
$4086.binds=$4052
$4086.variable=$4044
$4086.index=$4064
$4044.type=Script
$4044.formula=$4036
$4044.sub=$4040
$4040.type=Integer
$4052.type=VarList
$4052.linkedList=$4054
$4054.type=VarListLink
$4054.entry=$4044
$4064.type=MathString
$4064.entry=$4066
$4066.type=String
$4082.type=Of
$4082.operator=$4060
$4082.arguments=$4072
$4060.type=MathString
$4060.entry=$4062
$4062.type=String
$4072.type=Vector
$4072.linkedList=$4074
$4074.type=ExpLink
$4074.next=$4076
$4074.entry=$4044
$4076.type=ExpLink
$4076.entry=$4046
$4096.type=Integral
$4096.formula=$4084
$4096.binds=$4048
$4096.variable=$4036
$4096.index=$4090
$4048.type=VarList
$4048.linkedList=$4050
$4050.type=VarListLink
$4050.entry=$4036
$4084.type=Of
$4084.operator=$4060
$4084.arguments=$4078
$4078.type=Vector
$4078.linkedList=$4080
$4080.type=ExpLink
$4080.entry=$4036
$4090.type=SetProduct
$4090.linkedList=$4092
$4092.type=ExpLink
$4092.next=$4094
$4092.entry=$4064
$4094.type=ExpLink
$4094.entry=$4068
VALUE($4038) = x
VALUE($4040) = 1
VALUE($4042) = 2
VALUE($4062) = f
VALUE($4066) = A
VALUE($4070) = B
```

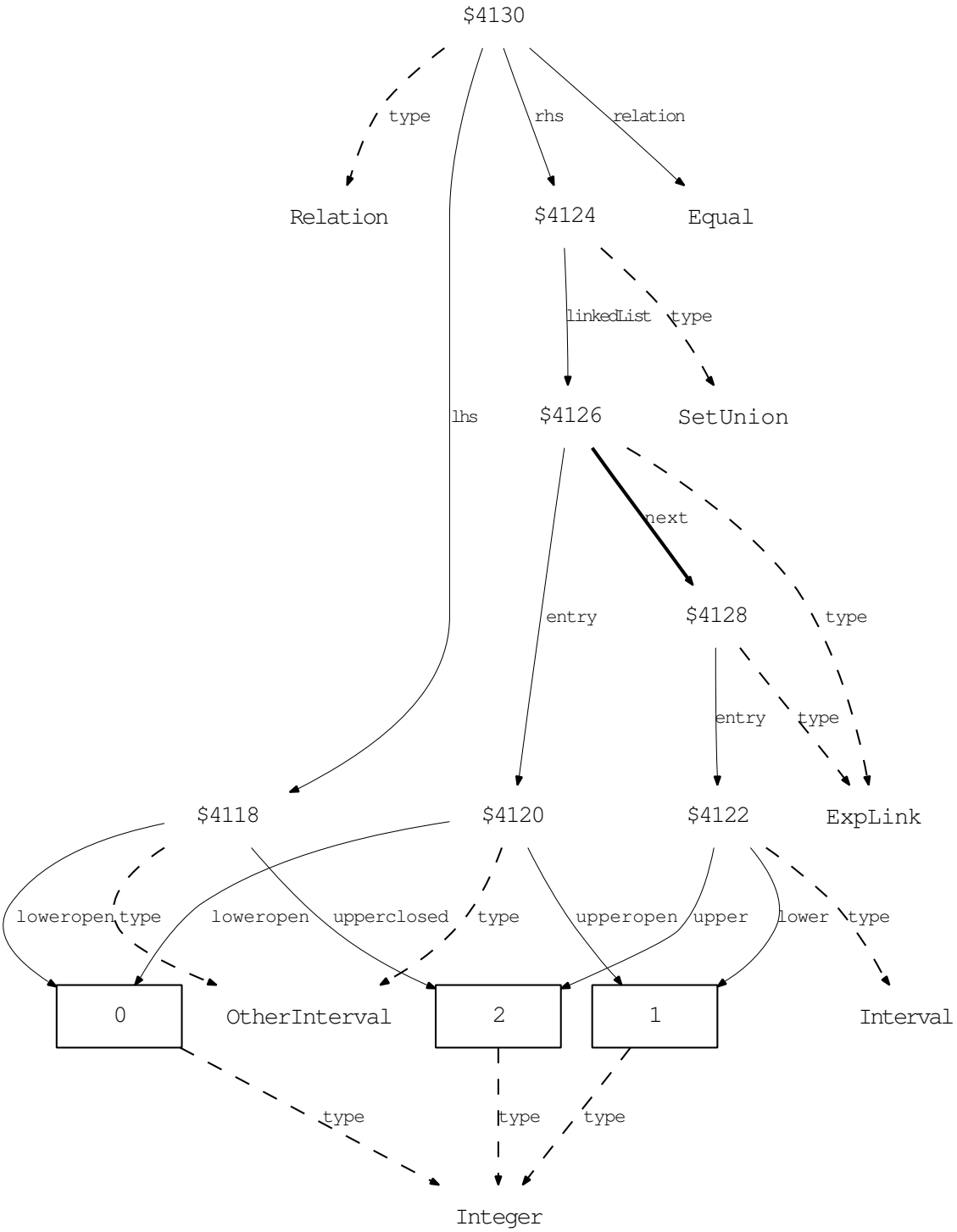

$4098
type
Relation
lhs
relation
$4088
Equal
rhs
formula
$4086
type
formula
binds
$4096
type
$4082
binds
index
formula
binds
type
arguments
$4090
$4084
$4048
Integral
type
$4072
operator
$4052
index
variable
$4056
index
variable
type
linkedList
arguments
type
operator
linkedList
type
linkedList
type
linkedList
type
linkedList
SetProduct
$4092
$4078
Of
type
$4060
$4050
$4074
VarList
$4054
$4058
variable
linkedList
next
entry
type
entry
next
type
type
entry
entry
type
type
$4080
$4094
Vector
type
$4064
$4076
VarListLink
$4044
entry
type
type
entry
type
entry
type
entry
f
sub
entry
entry
ExpLink
formula
type
$4068
type
$4046
type
entry
A
1
formula
type
sub
$4036
MathString
type
type
Script
2
type
name
B
type
type
Var
x
Integer
type
type
String

Example 25.

$$(0,2]{=}(0,1) \cup [1,2]$$

```
( 0 , 2 ]{{=}}( 0 , 1 ) \cup [ 1 , 2 ]
```

```
$4130.type=Relation
$4130.lhs=$4118
$4130.relation=Equal
$4130.rhs=$4124
$4118.type=OtherInterval
$4118.loweropen=$4112
$4118.upperclosed=$4116
$4112.type=Integer
$4116.type=Integer
$4124.type=SetUnion
$4124.linkedList=$4126
$4126.type=ExpLink
$4126.next=$4128
$4126.entry=$4120
$4120.type=OtherInterval
$4120.loweropen=$4112
$4120.upperopen=$4114
$4114.type=Integer
$4128.type=ExpLink
$4128.entry=$4122
$4122.type=Interval
$4122.lower=$4114
$4122.upper=$4116
VALUE($4112) = 0
VALUE($4114) = 1
VALUE($4116) = 2
```

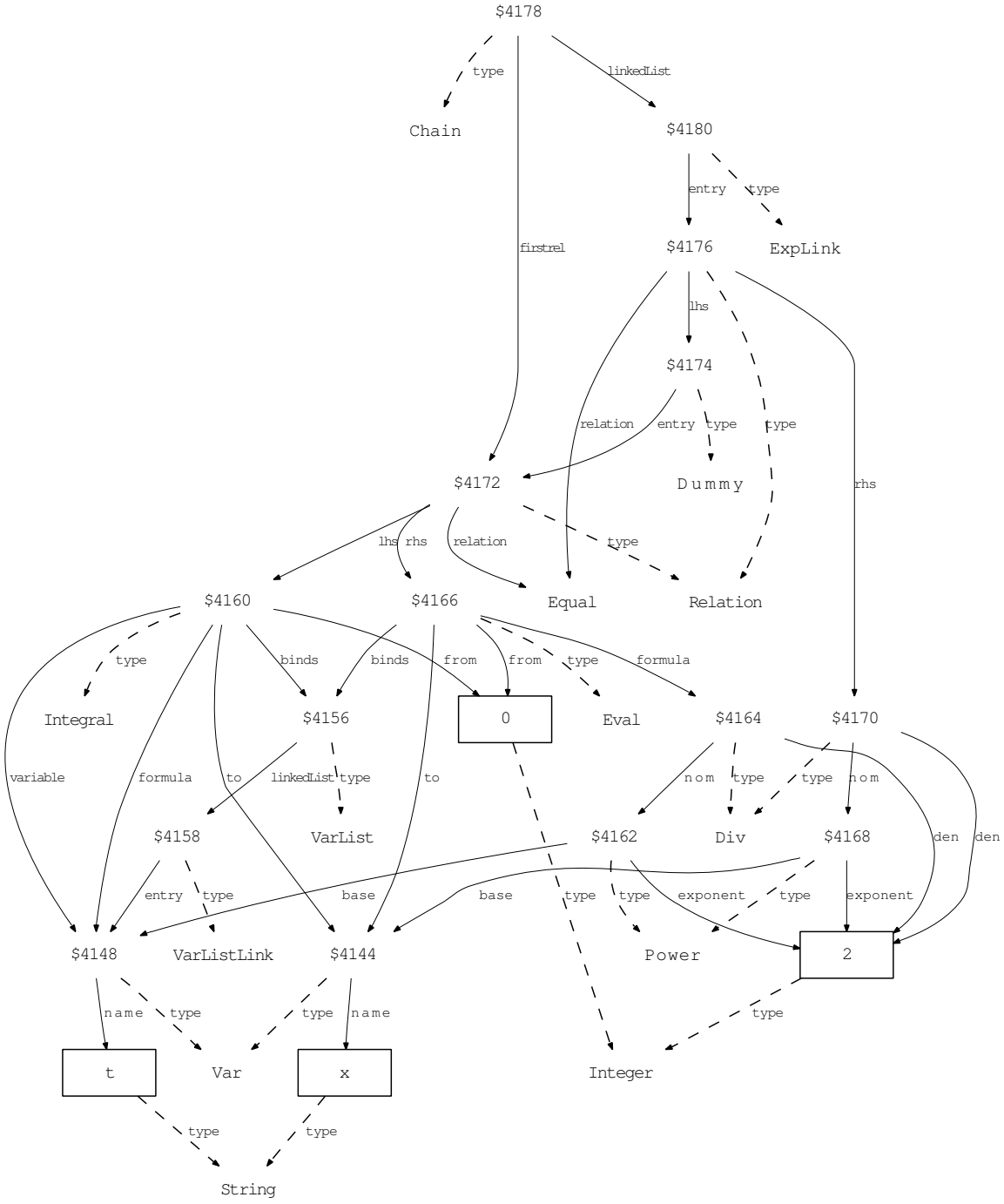

$4130
type
rhs
relation
Relation
$4124
Equal
linkedList
type
lhs
$4126
SetUnion
next
entry
$4128
type
entry
type
$4118
$4120
$4122
ExpLink
loweropen
type
loweropen
upperclosed
type
upperopen
upper
lower
type
0
OtherInterval
2
1
Interval
type
type
type
Integer

Example 26.

$$\int_0^x t \, \mathrm{d}t = \frac{t^2}{2}\Big|_0^x = \frac{x^2}{2}$$

```
\int_{ 0 }^{x} t ~ \mathrm{d}t{{=}}\left.\frac{{t}^{ 2 }}{ 2 }\
   right|_{ 0 }^{x}{{=}}\frac{{x}^{ 2 }}{ 2 }
```

```
$4178.type=Chain
$4178.firstrel=$4172
$4178.linkedList=$4180
$4172.type=Relation
$4172.lhs=$4160
$4172.relation=Equal
$4172.rhs=$4166
$4160.type=Integral
$4160.formula=$4148
$4160.binds=$4156
$4160.variable=$4148
$4160.from=$4152
$4160.to=$4144
$4144.type=Var
$4144.name=$4146
$4146.type=String
$4148.type=Var
$4148.name=$4150
$4150.type=String
$4152.type=Integer
$4156.type=VarList
$4156.linkedList=$4158
$4158.type=VarListLink
$4158.entry=$4148
$4166.type=Eval
$4166.formula=$4164
$4166.binds=$4156
$4166.from=$4152
$4166.to=$4144
$4164.type=Div
$4164.nom=$4162
$4164.den=$4154
$4154.type=Integer
$4162.type=Power
$4162.base=$4148
$4162.exponent=$4154
$4180.type=ExpLink
$4180.entry=$4176
$4176.type=Relation
$4176.lhs=$4174
$4176.relation=Equal
$4176.rhs=$4170
$4170.type=Div
$4170.nom=$4168
$4170.den=$4154
$4168.type=Power
$4168.base=$4144
$4168.exponent=$4154
$4174.type=Dummy
$4174.entry=$4172
VALUE($4146) = x
VALUE($4150) = t
VALUE($4152) = 0
VALUE($4154) = 2
```

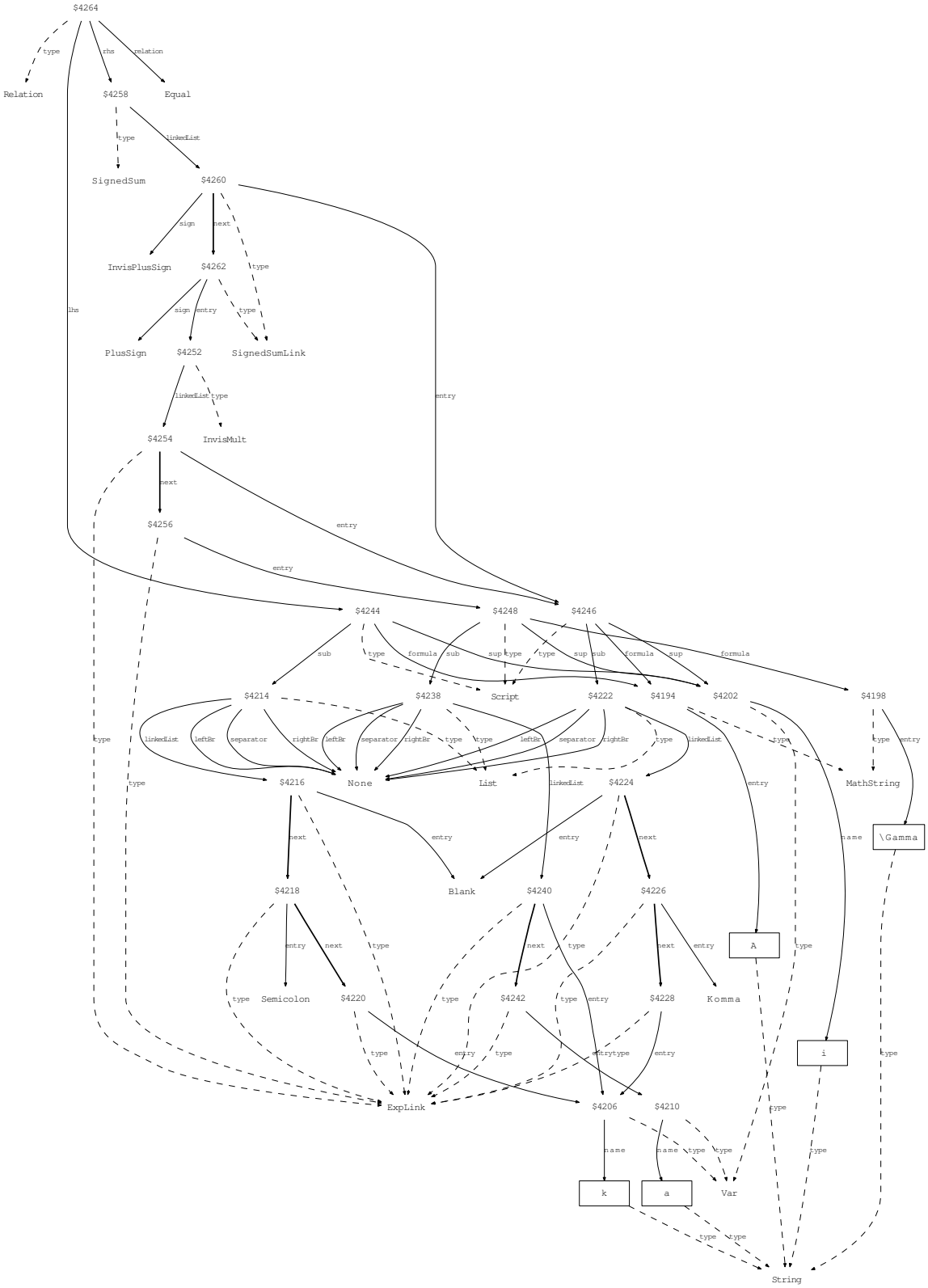

$4178
type
linkedList
Chain
$4180
entry
type
firstrel
$4176
ExpLink
lhs
$4174
relation
entry
type
type
$4172
Dummy
rhs
lhs
rhs
relation
type
$4160
$4166
Equal
Relation
type
binds
binds
from
from
type
formula
Integral
$4156
0
Eval
$4164
$4170
variable
formula
to
linkedList
type
to
nom
type
type
nom
$4158
VarList
$4162
Div
$4168
den
den
entry
type
base
base
type
type
exponent
type
exponent
$4148
VarListLink
$4144
Power
2
name
type
type
name
type
t
Var
x
Integer
type
type
String

Example 27.

$$A^{i}{}_{;k}=A^{i}{}_{,k}+A^{i}{}_{,k}\Gamma^{i}_{ka}$$

```
{}{A}_{ ~ ; k}^{i}{{=}} {}{A}_{ ~ , k}^{i} + {}{A}_{ ~ , k}^{i}
   {}{\Gamma}_{ka}^{i}
```

```
$4264.type=Relation
$4264.lhs=$4244
$4264.relation=Equal
$4264.rhs=$4258
$4244.type=Script
$4244.formula=$4194
$4244.sub=$4214
$4244.sup=$4202
$4194.type=MathString
$4194.entry=$4196
$4196.type=String
$4202.type=Var
$4202.name=$4204
$4204.type=String
$4214.type=List
$4214.leftBr=None
$4214.separator=None
$4214.rightBr=None
$4214.linkedList=$4216
$4216.type=ExpLink
$4216.next=$4218
$4216.entry=Blank
$4218.type=ExpLink
$4218.next=$4220
$4218.entry=Semicolon
$4220.type=ExpLink
$4220.entry=$4206
$4206.type=Var
$4206.name=$4208
$4208.type=String
$4258.type=SignedSum
$4258.linkedList=$4260
$4260.type=SignedSumLink
$4260.next=$4262
$4260.sign=InvisPlusSign
$4260.entry=$4246
$4246.type=Script
$4246.formula=$4194
$4246.sub=$4222
$4246.sup=$4202
$4222.type=List
$4222.leftBr=None
$4222.separator=None
$4222.rightBr=None
$4222.linkedList=$4224
$4224.type=ExpLink
$4224.next=$4226
$4224.entry=Blank
$4226.type=ExpLink
$4226.next=$4228
$4226.entry=Komma
$4228.type=ExpLink
$4228.entry=$4206
$4262.type=SignedSumLink
$4262.sign=PlusSign
$4262.entry=$4252
$4252.type=InvisMult
$4252.linkedList=$4254
$4254.type=ExpLink
$4254.next=$4256
$4254.entry=$4246
$4256.type=ExpLink
$4256.entry=$4248
$4248.type=Script
$4248.formula=$4198
$4248.sub=$4238
$4248.sup=$4202
$4198.type=MathString
$4198.entry=$4200
$4200.type=String
$4238.type=List
$4238.leftBr=None
$4238.separator=None
$4238.rightBr=None
$4238.linkedList=$4240
$4240.type=ExpLink
$4240.next=$4242
$4240.entry=$4206
$4242.type=ExpLink
$4242.entry=$4210
$4210.type=Var
$4210.name=$4212
$4212.type=String
VALUE($4196) = A
VALUE($4200) = \Gamma
VALUE($4204) = i
VALUE($4208) = k
VALUE($4212) = a
```

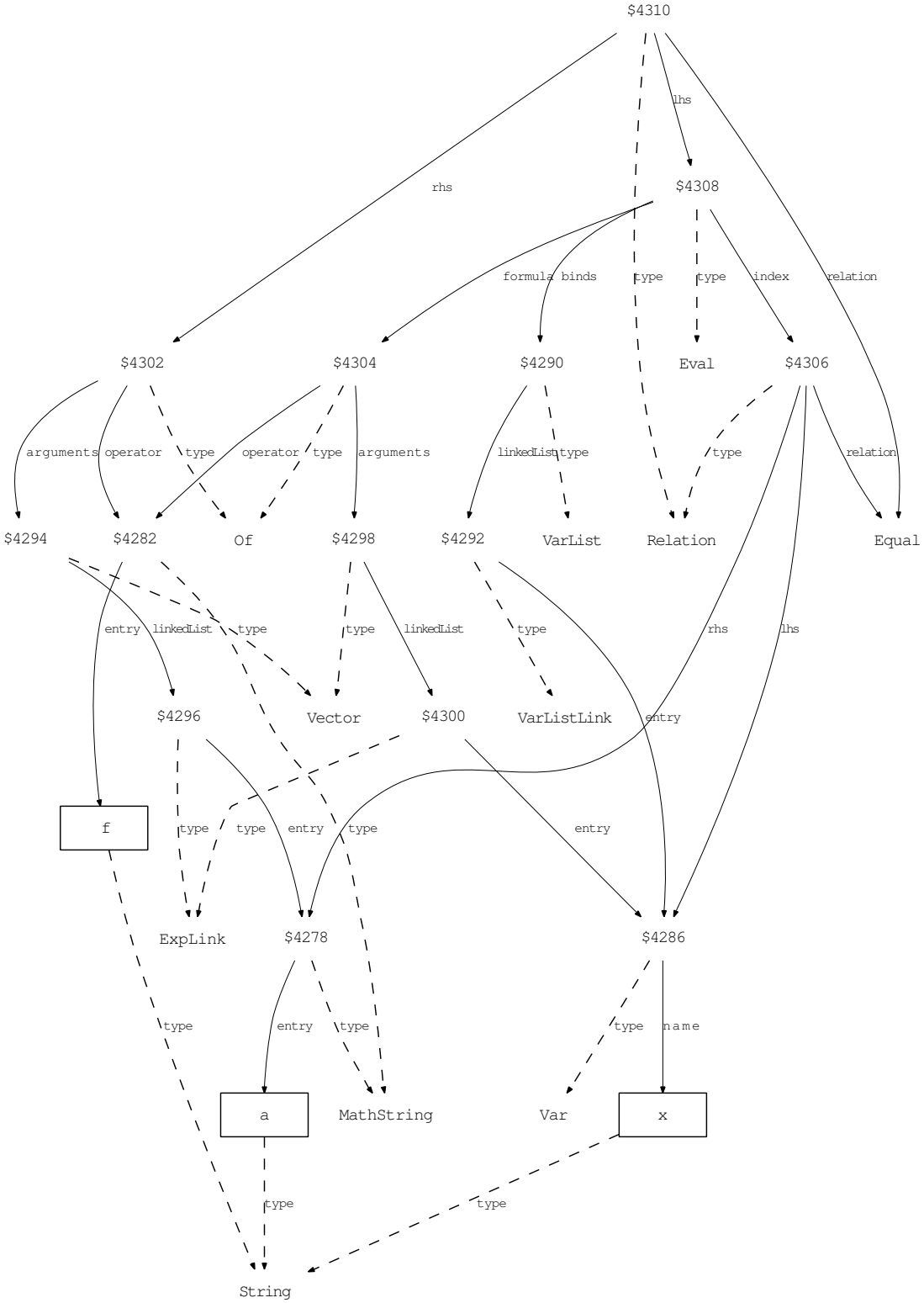

$4264
type
rhs
relation
Relation
$4258
Equal
type
linkedList
SignedSum
$4260
sign
next
InvisPlusSign
$4262
type
lhs
sign
entry
type
PlusSign
$4252
SignedSumLink
linkedList
type
entry
$4254
InvisMult
next
$4256
entry
entry
$4244
$4248
$4246
sub
type
formula
sub
sup
type
type
sup
sub
formula
sup
formula
$4214
$4238
Script
$4222
$4194
$4202
$4198
type
linkedList
leftBr
separator
rightBr
leftBr
separator
rightBr
type
type
leftBr
separator
rightBr
type
linkedList
type
type
entry
type
None
List
$4216
linkedList
$4224
MathString
\Gamma
type
next
entry
entry
next
name
$4218
Blank
$4240
$4226
entry
next
type
next
type
next
entry
A
type
type
Semicolon
$4220
type
$4242
type
entry
$4228
Komma
type
type
entry
type
entry
type
entry
type
i
type
ExpLink
$4206
$4210
type
name
name
type
type
type
k
a
Var
type
type
String

Example 28.

$$f\left(x\right)\left|_{x=a}=f\left(a\right)\right.$$

```
\left.f \left(x\right) \right|_{x{{=}}a}{{=}}f \left(a\right)
```

```
$4310.type=Relation
$4310.lhs=$4308
$4310.relation=Equal
$4310.rhs=$4302
$4302.type=Of
$4302.operator=$4282
$4302.arguments=$4294
$4282.type=MathString
$4282.entry=$4284
$4284.type=String
$4294.type=Vector
$4294.linkedList=$4296
$4296.type=ExpLink
$4296.entry=$4278
$4278.type=MathString
$4278.entry=$4280
$4280.type=String
$4308.type=Eval
$4308.formula=$4304
$4308.binds=$4290
$4308.index=$4306
$4290.type=VarList
$4290.linkedList=$4292
$4292.type=VarListLink
$4292.entry=$4286
$4286.type=Var
$4286.name=$4288
$4288.type=String
$4304.type=Of
$4304.operator=$4282
$4304.arguments=$4298
$4298.type=Vector
$4298.linkedList=$4300
$4300.type=ExpLink
$4300.entry=$4286
$4306.type=Relation
$4306.lhs=$4286
$4306.relation=Equal
$4306.rhs=$4278
VALUE($4280) = a
VALUE($4284) = f
VALUE($4288) = x
```

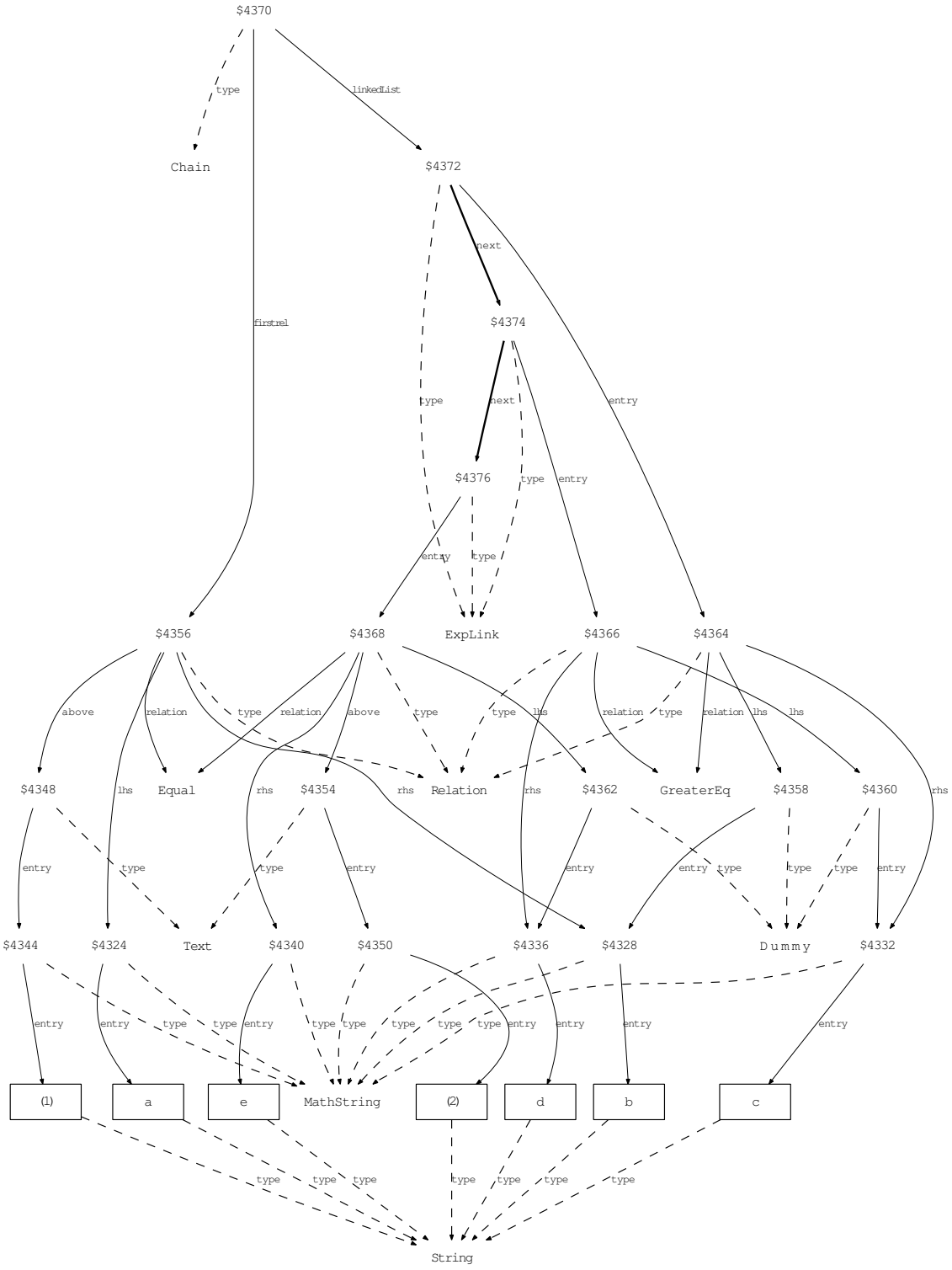

$4310
rhs
lhs
$4308
formula
binds
type
type
index
relation
$4302
$4304
$4290
Eval
$4306
arguments
operator
type
operator
type
arguments
linkedList
type
type
relation
$4294
$4282
Of
$4298
$4292
VarList
Relation
Equal
entry
linkedList
type
type
linkedList
type
rhs
lhs
$4296
Vector
$4300
VarListLink
entry
f
type
type
entry
type
entry
ExpLink
$4278
$4286
type
entry
type
type
name
a
MathString
Var
x
type
type
String

Example 29.

$$a \overset{(1)}{=} b \geq c \geq d \overset{(2)}{=} e$$

```
a\stackrel{\text{(1)}}{=}b{ \geq }c{ \geq }d\stackrel{\text
   {(2)}}{=}e
```

```
$4370.type=Chain
$4370.firstrel=$4356
$4370.linkedList=$4372
$4356.type=Relation
$4356.lhs=$4324
$4356.relation=Equal
$4356.rhs=$4328
$4356.above=$4348
$4324.type=MathString
$4324.entry=$4326
$4326.type=String
$4328.type=MathString
$4328.entry=$4330
$4330.type=String
$4348.type=Text
$4348.entry=$4344
$4344.type=MathString
$4344.entry=$4346
$4346.type=String
$4372.type=ExpLink
$4372.next=$4374
$4372.entry=$4364
$4364.type=Relation
$4364.lhs=$4358
$4364.relation=GreaterEq
$4364.rhs=$4332
$4332.type=MathString
$4332.entry=$4334
$4334.type=String
$4358.type=Dummy
$4358.entry=$4328
$4374.type=ExpLink
$4374.next=$4376
$4374.entry=$4366
$4366.type=Relation
$4366.lhs=$4360
$4366.relation=GreaterEq
$4366.rhs=$4336
$4336.type=MathString
$4336.entry=$4338
$4338.type=String
$4360.type=Dummy
$4360.entry=$4332
$4376.type=ExpLink
$4376.entry=$4368
$4368.type=Relation
$4368.lhs=$4362
$4368.relation=Equal
$4368.rhs=$4340
$4368.above=$4354
$4340.type=MathString
$4340.entry=$4342
$4342.type=String
$4354.type=Text
$4354.entry=$4350
$4350.type=MathString
$4350.entry=$4352
$4352.type=String
$4362.type=Dummy
$4362.entry=$4336
VALUE($4326) = a
VALUE($4330) = b
VALUE($4334) = c
VALUE($4338) = d
VALUE($4342) = e
VALUE($4346) = (1)
VALUE($4352) = (2)
```

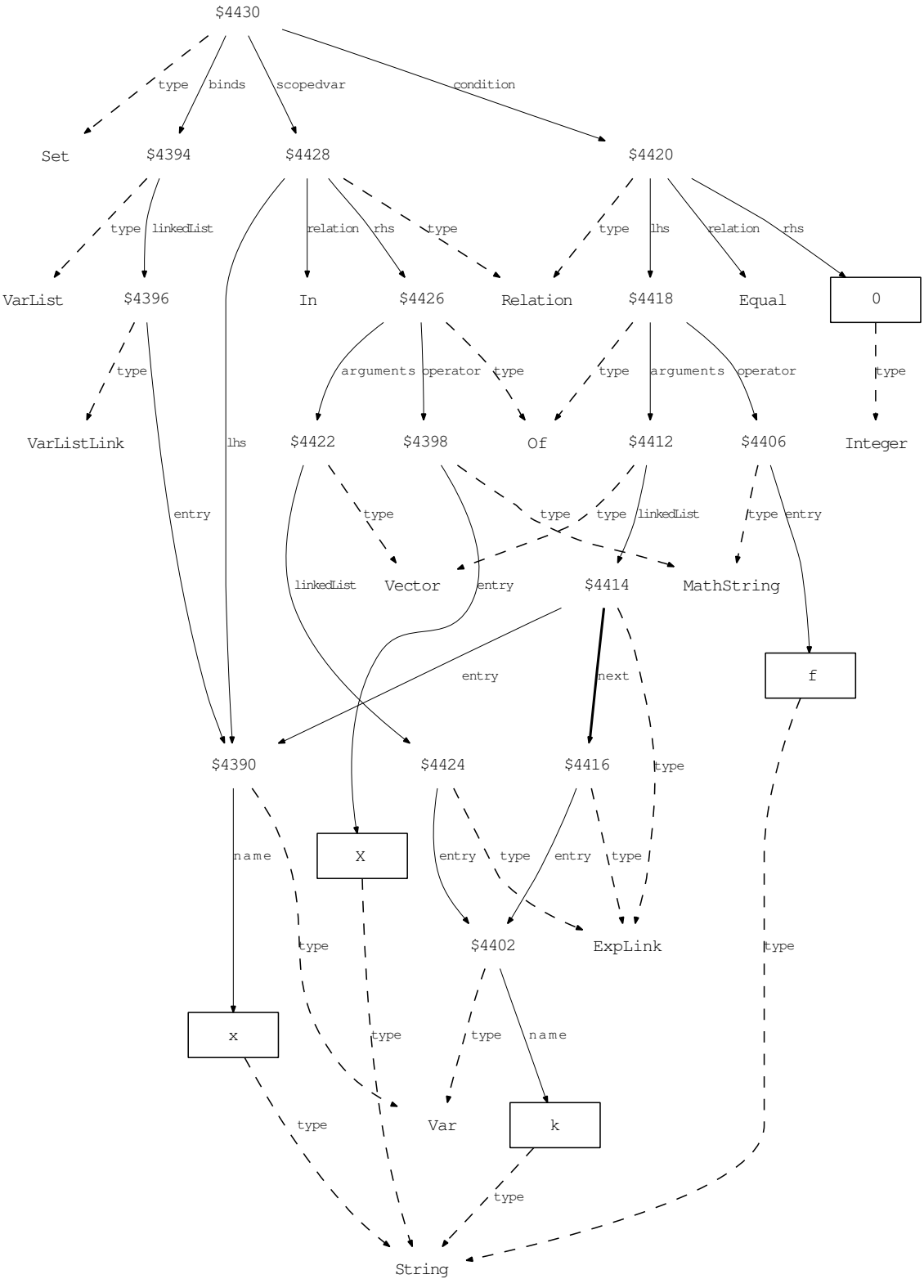

$4370
type
linkedList
Chain
$4372
next
firstrel
$4374
type
next
entry
$4376
type
entry
entry
type
$4356
$4368
ExpLink
$4366
$4364
above
relation
type
relation
above
type
type
lhs
relation
type
relation
lhs
lhs
$4348
lhs
Equal
rhs
$4354
rhs
Relation
rhs
$4362
GreaterEq
$4358
$4360
rhs
entry
type
type
entry
entry
entry
type
type
type
entry
$4344
$4324
Text
$4340
$4350
$4336
$4328
Dummy
$4332
entry
entry
type
type
entry
type
type
type
type
type
entry
entry
entry
entry
(1)
a
e
MathString
(2)
d
b
c
type
type
type
type
type
type
type
String

Example 30.

$$\{x \in X\,(k) \mid f\,(x,k) = 0\}$$

```
\left\{x{ \in }X \left(k\right) \mid f \left(x , k\right) {{=}}
    0 \right\}
```

```
$4430.type=Set
$4430.scopedvar=$4428
$4430.binds=$4394
$4430.condition=$4420
$4394.type=VarList
$4394.linkedList=$4396
$4396.type=VarListLink
$4396.entry=$4390
$4390.type=Var
$4390.name=$4392
$4392.type=String
$4420.type=Relation
$4420.lhs=$4418
$4420.relation=Equal
$4420.rhs=$4410
$4410.type=Integer
$4418.type=Of
$4418.operator=$4406
$4418.arguments=$4412
$4406.type=MathString
$4406.entry=$4408
$4408.type=String
$4412.type=Vector
$4412.linkedList=$4414
$4414.type=ExpLink
$4414.next=$4416
$4414.entry=$4390
$4416.type=ExpLink
$4416.entry=$4402
$4402.type=Var
$4402.name=$4404
$4404.type=String
$4428.type=Relation
$4428.lhs=$4390
$4428.relation=In
$4428.rhs=$4426
$4426.type=Of
$4426.operator=$4398
$4426.arguments=$4422
$4398.type=MathString
$4398.entry=$4400
$4400.type=String
$4422.type=Vector
$4422.linkedList=$4424
$4424.type=ExpLink
$4424.entry=$4402
VALUE($4392) = x
VALUE($4400) = X
VALUE($4404) = k
VALUE($4408) = f
VALUE($4410) = 0
```

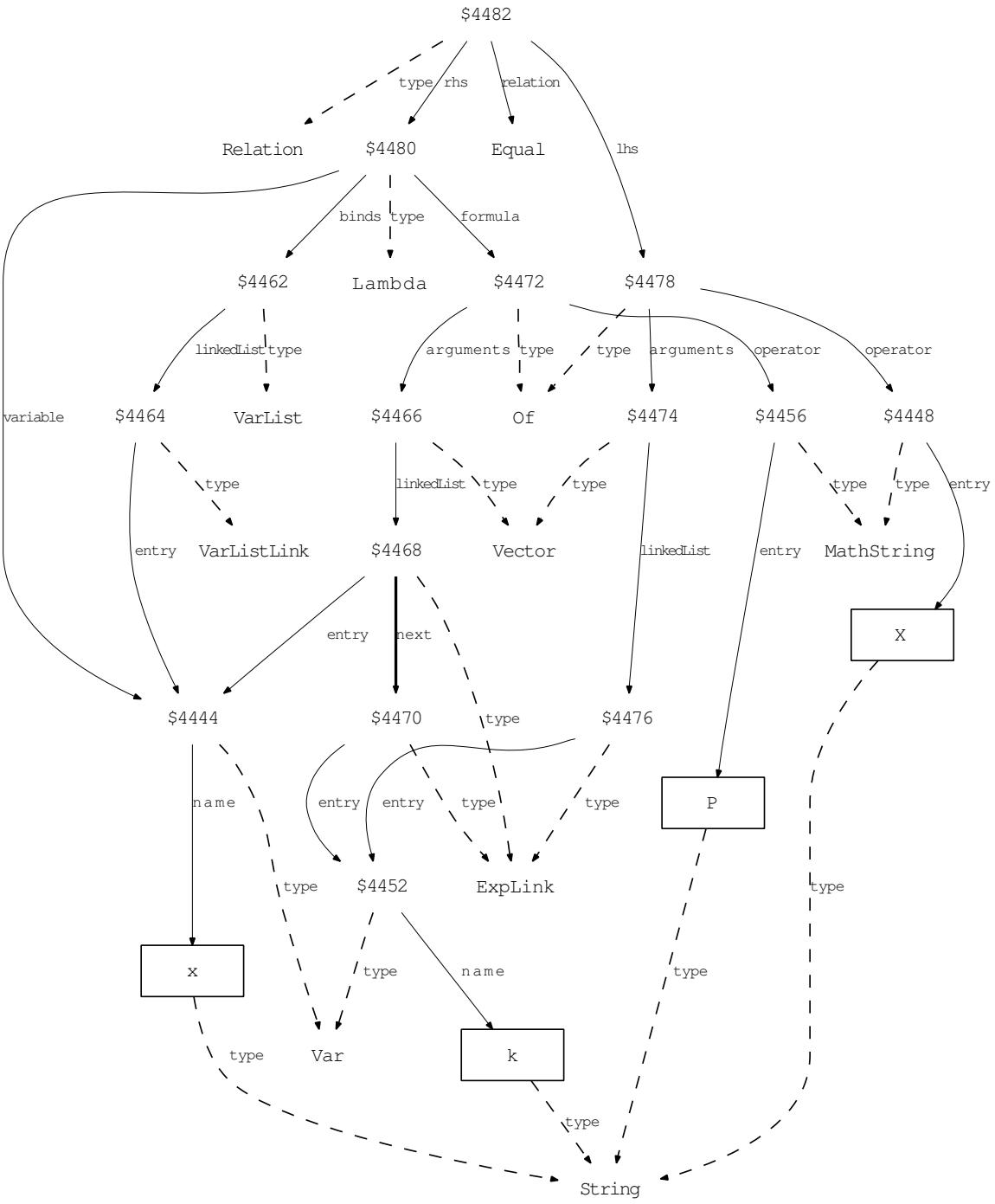

$4430
type
binds
scopedvar
condition
Set
$4394
$4428
$4420
type
linkedList
relation
rhs
type
type
lhs
relation
rhs
VarList
$4396
In
$4426
Relation
$4418
Equal
0
type
arguments
operator
type
type
arguments
operator
type
VarListLink
lhs
$4422
$4398
Of
$4412
$4406
Integer
entry
type
type
type
linkedList
type
entry
linkedList
Vector
entry
$4414
MathString
entry
next
f
$4390
$4424
$4416
type
name
X
entry
type
entry
type
type
$4402
ExpLink
type
x
type
type
name
type
Var
k
type
String

Example 31.

$$X\left(k\right) {=} \lambda x . P\left(x , k\right)$$

```
X \left(k\right) {{=}} \lambda x . P \left(x , k\right)
```

```
$4482.type=Relation
$4482.lhs=$4478
$4482.relation=Equal
$4482.rhs=$4480
$4478.type=Of
$4478.operator=$4448
$4478.arguments=$4474
$4448.type=MathString
$4448.entry=$4450
$4450.type=String
$4474.type=Vector
$4474.linkedList=$4476
$4476.type=ExpLink
$4476.entry=$4452
$4452.type=Var
$4452.name=$4454
$4454.type=String
$4480.type=Lambda
$4480.formula=$4472
$4480.binds=$4462
$4480.variable=$4444
$4444.type=Var
$4444.name=$4446
$4446.type=String
$4462.type=VarList
$4462.linkedList=$4464
$4464.type=VarListLink
$4464.entry=$4444
$4472.type=Of
$4472.operator=$4456
$4472.arguments=$4466
$4456.type=MathString
$4456.entry=$4458
$4458.type=String
$4466.type=Vector
$4466.linkedList=$4468
$4468.type=ExpLink
$4468.next=$4470
$4468.entry=$4444
$4470.type=ExpLink
$4470.entry=$4452
VALUE($4446) = x
VALUE($4450) = X
VALUE($4454) = k
VALUE($4458) = P
```


$4482
type
rhs
relation
Relation
$4480
Equal
lhs
binds
type
formula
$4462
Lambda
$4472
$4478
linkedList
type
arguments
type
type
arguments
operator
operator
variable
$4464
VarList
$4466
Of
$4474
$4456
$4448
type
linkedList
type
type
type
type
entry
entry
VarListLink
$4468
Vector
linkedList
entry
MathString
X
entry
next
$4444
$4470
type
$4476
name
entry
entry
type
type
P
type
$4452
ExpLink
type
x
type
name
type
type
Var
k
type
String

Acknowledgements. We thank the members of the FMathL seminar, in particular Hermann Schichl and Kevin Kofler for their contributions.

Support by the Austrian Science Fund (FWF) under contract number P20631 is gratefully acknowledged.

References

[1] P. Schodl and A. Neumaier. The FMathL type system. *Manuscript*, 2010.