Minimally Working Version

Danny Klein, Austin Osborn

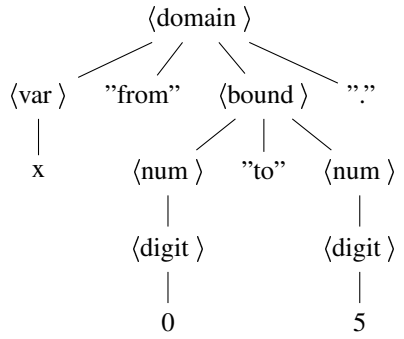
0.1 Minimally Working Interpreter

Our AST represnts multiple types of data: Numbers (ints), and Variables (characters) are the primitives. Our AST also has multiple combining forms, Bound is a tuple of two Numbers represnting the lower and upper bounds of the domain. Domain is a tuple of a Variable and a Bound, which is itself a tuple.
Our parser recognizes data inputed in the form "x from 0 to 10." as an example of a valid domain statement, and produces an AST with x as the variable, 0 as the lower bound, and 10 as the upper bound. The evaluator takes this Domain expression and turns it into an svg file that draws the domain on the axis of our variable.

0.2 Minimal Formal Grammar

```
<domian>   ::= <var> from <bound>.
<bound>    ::= <num> to <num>
<var>      ::= a | ... | z
<num>      ::= <digit>^+ | -<digit>^+
<digit>    ::= 0 | ... | 9
```

The domain: "x from 0 to 5." would produce the following tree:



0.3 Minimal Semantics

Sytnax	Abstract Syntax	Type	Prec./ Assoc.	Meaning
x	Var of Char	Char	N/A	x is a primitive, and will represent our independent variable
n	Num of int	int	N/A	n is a primitive. We represent integers using the 32-bit integer data type (Int32).
n to m	Bound of {lower: Num; upper: Num}	record of int*int	N/A	n to m is a combining form of two Nums (ints) that represent the lower and upper bound of our domain. It is saved as a record with the first int as lower and the second int as upper.
x from n to m	Domain of {var: Var; bounds: Bound}	record of char* Bound	N/A	x from n to m is a combining form of a Var (char) and two Nums (ints) that represent the variable, lower bound, and upper bound of our domain. It is saved as a record with the Var as var and the two ints as a Bound record.