

## Review: Prolog Interpreter for ONE

A simple interpreter that computes a semantic value for syntactic constructs, the computation of this semantic value can be interpreted as the behavior: val1 / 2, AST input and semantic value as output.

```
val1(plus(X,Y),Value) :-  
    val1(X,XValue),  
    val1(Y,YValue),  
    Value is XValue + YValue.  
  
val1(times(X,Y),Value) :-  
    val1(X,XValue),  
    val1(Y,YValue),  
    Value is XValue * YValue.  
  
val1(const(X),X).
```

## Problems

- What is the value of a constant?
  - Interpreter says `val1(const(X),X)`.
  - This means that the value of a constant in Language One is whatever the value of that same constant is *in Prolog*
  - Unfortunately, different implementations of Prolog handle this differently

## Value Of A Constant

```
?- val1(const(2147483647), X) . %2^31 - 1
```

```
X = 2147483647
```

```
Yes
```

```
?- val1(const(2147483648), X) . %2^31
```

```
X = 2.14748e+009
```

```
Yes
```

- Our Prolog treats values greater than  $2^{31}-1$  as floating-point constants
- Did we mean Language One to do this?

## Value Of A Sum

```
?- val1(plus(const(2147483646), const(1)), X) .
```

```
X = 2147483647
```

```
Yes
```

```
?- val(plus(const(2147483647), const(1)), X) .
```

```
X = 2.14748e+009
```

```
Yes
```

- Our Prolog expresses sums greater than  $2^{31}-1$  as floating-point results
- Did we mean Language One to do this?

## Defining Semantics By Interpreter - Caveats

- Our `val1` is not satisfactory as a definition of the semantics of Language ONE
- “Language ONE programs behave the way this interpreter says they behave, *running under this implementation of Prolog on this computer system*”  
⇒ We need something more abstract

## Natural Semantics

- A mathematical view of interpretation
- A formal notation we can use to capture the same basic proof rules in `val1`
- We are trying to define the relation between an AST and the result of evaluating it
  - We will use the symbol  $\rightarrow$  for this relation, writing  $E \rightarrow v$  to mean that the AST  $E$  evaluates to the value  $v$
  - For example, our semantics should establish `times(const(2), const(3)) → 6`

## A Rule In Natural Semantics

$$\frac{E_1 \rightarrow v_1 \quad E_2 \rightarrow v_2}{\mathbf{times}(E_1, E_2) \rightarrow v_1 \times v_2}$$

- Conditions above the line, conclusion below
- The same idea as our Prolog rule:

```
val1(times(X,Y),Value) :-  
    val1(X,XValue),  
    val1(Y,YValue),  
    Value is XValue * YValue.
```

## ONE: Natural Semantics

$$\frac{E_1 \rightarrow v_1 \quad E_2 \rightarrow v_2}{\mathbf{plus}(E_1, E_2) \rightarrow v_1 + v_2}$$

$$\frac{E_1 \rightarrow v_1 \quad E_2 \rightarrow v_2}{\mathbf{times}(E_1, E_2) \rightarrow v_1 \times v_2}$$

$$\mathbf{const}(n) \rightarrow eval(n)$$

```
val1(plus(X,Y),Value) :-  
    val1(X,XValue),  
    val1(Y,YValue),  
    Value is XValue + YValue.  
val1(times(X,Y),Value) :-  
    val1(X,XValue),  
    val1(Y,YValue),  
    Value is XValue * YValue.  
val1(const(X),X).
```

- Of course, this still needs definitions for +, × and eval, but at least it won't accidentally use Prolog's
- Examples: 1+2\*3...

## Natural Semantics, Note

- There may be more than one rule for a particular kind of AST node
- For instance, for an ML-style if-then-else we might use something like this:

$$\frac{E_1 \rightarrow true \quad E_2 \rightarrow v_2}{\mathbf{if}(E_1, E_2, E_3) \rightarrow v_2}$$

$$\frac{E_1 \rightarrow false \quad E_3 \rightarrow v_3}{\mathbf{if}(E_1, E_2, E_3) \rightarrow v_3}$$