16-124-836
Marcel Matti ZAUDER

7.1 Calculator Language Extension Subtraction and Division

7.1.1 Abstract Syntax

```
Prog ::= 'ON' Stmt
Stmt ::= Expr 'TOTAL' Stmt
      |   Expr 'TOTAL' 'OFF'
Expr ::= Expr1 '+' Expr2
      |   Expr1 '-' Expr2
      |   Expr1 '*' Expr2
      |   Expr1 '/' Expr2
      |   'IF' Expr1 ',' Expr2 ',' Expr3
      |   'LASTANSWER'
      |   '(' Expr ')'
      |   Num
```

7.1.2 Semantic Functions

In order to be able to output `"NOT A NUMBER"` the domain must be extended so the program can also return a *string*. Furthermoe I make the assumption that when the calculator is returnig `"NOT A NUMBER"` the following expressions shall be disregarded.

Programs

P: Program → (Int*|String)
P[ON S] = **S** [S] (0)

Statements

S: ExprSequence → (Int|String) → (Int*|String)
S[E TOTAL S](n) = let n' = **E**(n) in *cons*(n', **S**[S](n'))
S[E TOTAL OFF](n) = [**E**[E](n)]

Expressions

E: Expression → (Int|String) → (Int|String)
E[E](`"NOT A NUMBER"`) → `"NOT A NUMBER"`
E[E1 + E2](n) = **E**[E1](n) + **E**[E2](n)
E[E1 - E2](n) = **E**[E1](n) - **E**[E2](n)
E[E1 * E2](n) = **E**[E1](n) × **E**[E2](n)
E[E1 / E2](n) = E[IF E2, `"NOT A NUMBER"`, **E**[E1](n) : **E**[E2](n)](n)
E[IF E1, E2, E3](n) = if E[E1](n) = 0 then **E**[E2](n) else **E**[E3](n)
E[LASTANSWER](n) = n
E[(E)](n) = **E**[E](n)
E[N](n) = N
E[String](n) = String
E[E anyOperator `"NOT A NUMBER"`] = `"NOT A NUMBER"`

7.2 Language of Binary Numbers

My solution is based on the **Denotational Semantics** by *D. A. Schmidt*.

7.2.1 Abstract Syntax

B denotes the binary numeral and D the binary digit. A binary numeral is a sequence of binary digits. The binary number should be mapped to its corresponding decimal number:

```
B ::= BD | D
D ::= 0 | 1
```

7.2.2 Semantic Functions

B: Binary-Numeral → Int
B[BD] = ((**B**[B] * 2) + **D**[D]
B[D] = **D**[D]

D: Binary-Numeral → Int
D[0] = 0
D[1] = 1

7.2.3 Domain

The domain of this language would be:
`Binary-Numeral` → `Binary-Numeral` → `Int` → `Int`

7.2.4 Test

We want to test our function with the input '10100':

$$\begin{aligned}
\mathbf{B}['10100'] &= ((\mathbf{B}[1010] * 2) + \mathbf{D}[0]) \\
&= ((((\mathbf{B}[101] * 2) + \mathbf{D}[0]) * 2) + \mathbf{D}[0]) \\
&= (((((\mathbf{B}[10] * 2) + \mathbf{D}[1]) * 2) + \mathbf{D}[0]) * 2) + \mathbf{D}[0]) \\
&= ((((((((\mathbf{B}[1] * 2) + \mathbf{D}[0]) * 2) + \mathbf{D}[1]) * 2) + \mathbf{D}[0]) * 2) + \mathbf{D}[0]) \\
&= ((((((((\mathbf{D}[1] * 2) + \mathbf{D}[0]) * 2) + \mathbf{D}[1]) * 2) + \mathbf{D}[0]) * 2) + \mathbf{D}[0]) \\
&= ((((((((1 * 2) + 0) * 2) + 1) * 2) + 0) * 2) + 0) \\
&= (((((((2 + 0) * 2) + 1) * 2) + 0) * 2) + 0) \\
&= ((((((2 * 2) + 1) * 2) + 0) * 2) + 0) \\
&= (((((4 + 1) * 2) + 0) * 2) + 0) \\
&= ((((5 * 2) + 0) * 2) + 0) \\
&= (((10 + 0) * 2) + 0) \\
&= ((10 * 2) + 0) \\
&= (20 + 0) \\
&= 20
\end{aligned}$$