Contents

Lexical Anaylser

Design a lexical analyser to process floating point values

- `123.456E78`
- How is the value zero represented in floating point?
  - `0`
  - `.0`
  - `0.`
  - `0.0` - unambiguous
- Can define
  - `DIGIT = [0123456789].`
    - `[regex option]`
  - `TOKEN`
    - lexical tokens
    - `SIGN = "+"|"-".`
    - `DIGIT SEQUENCE = DIGIT {DIGIT}.`
      - `{0 or more}`
    - `EXPONENT = ("e"|"E")[SIGN]{DIGIT SEQUENCE}.`
      - `(groups)`
    - `FLOAT = [SIGN]{DIGIT SEQUENCE}("."{DIGIT SEQUENCE}[EXPONENT]|EXPONENT).`

	d	d	.	d	d	E	-	d	d
1	2	2	3	4	4	5	6	7	7

	-	d	d	.	d
1	8	2	2	3	4

	d	E	d
1	2	5	7

2. Digits before the decimal point
3. Decimal point
4. Digits after the decimal point
5. Exponent
6. Sign before exponent
7. Digits of the exponent
8. Sign before number

- *input alphabet* = `{d, s, e, .,⊣}`
  - d = digit
  - s = +/- sign
  - e = e/E = exponent
  - . = decimal
  - ⊣ = end marker
    - Any input that isn't in the input language
- *states* = `{1, 2, 3, 4, 5, 6, 7, 8}`

  All blank table entries represent exits to an error routine

	d	s	e	.	⊣
1	2	8			
2	2		5	3	
3	4				
4	4		5		"Yes"
5	7	6			
6	7				
7	7				"Yes"
8	2				

  This is a reduced table - no extrenuous states

- *12.34E5*

  - `12.34E5 = 1234E3`
    * NUMBER 1234
    * EXPONENT 3

- Floating point value is $number * 10^{exponent}$
  - $1234*10^{3}

2. Digits before the decimal point
3. Decimal point
4. Digits after the decimal point
5. Exponent
6. Sign before exponent
7. Digits of the exponent
8. Sign before number

	d	s	e	.	⊣
1	2A	8			
2	2B		5A	3	
3	4A				
4	4B		5B		“Yes1”
5	7A	6			
6	7B				
7	7C				“Yes2”
8	2C				

- Variables
  - NUMBER, EXPONENT, SIGN, EXPONENT SIGN, COUNT
    * COUNT keeps track of the number of times we've multipled by 10
      · How much the exponent needs adjusting when finished
    * (EXPONENT)SIGN; 1=PLUS; -1=MINUS;
- Before taking a transition from one state to another, perform the associated action routine and advance the input
  - `1 -> 2A(perform action routing) -> advance the input -> remove state 2A`
- `2A`

  - `SIGN <- 1`
  - `NUMBER <- value(digit)`

- `2B`
  - `NUMBER <- NUMBER*10 + value(digit)`
- `2C`
  - `NUMBER <- value(digit)`
- `3, 5A`
  - `COUNT <- 0`
- `5B`
  - NO ACTION

  This is already done at `3` or `5A` `5B` is after the decimal point, i.e. after `3`

- `4A, 4B`
  - `NUMBER <- NUMBER*10 + value(digit)`
  - `COUNT <- COUNT+1`

  States `4` are after the decimal point Exponents have to be decreased by one

- `6`
  - `if SIGN=+ then`
  - `EXPONENT SIGN <- 1 else`
  - `EXPONENT SIGN <- -1`
- `7A`
  - `EXPONENT SIGN <- 1`
  - `EXPONENT <- value(digit)`

  Got here *without* seeing the sign

- `7B`
  - `EXPONENT <- value(digit)`

  Got here *with* seeing the sign

- `7C`

  - EXPONENT SIGN <- 1
  - EXPONENT <- EXPONENT*10 + value(digit)

  nth digit, so fix sign

- 8
  - if SIGN=+ then
  - SIGN=1 else
  - SIGN=-1

  note: SIGN is SIGN=+ is the SIGN from the symbol table

- Yes1
  - NUMBER <- NUMBER*SIGN
  - EXPONEONT <- EXPONENT*EXPONENT SIGN
- Yes2
  - NUMBR <- NUMBER*SIGN
  - EXPONEONT <- EXPONENT*(EXPONENT SIGN-COUNT)