Syntax

Sau P

20 April 2023

Contents

---

1 Syntax of our language

This language will be a domain specific language specialising in the manipulation of tiles.

1.1 Language specification in Backus-Naur form

```
<program> ::= {<statement>}

<statement> ::= <comment>
             | <tile-definition>
             | <variable-declaration>
             | <operation>
             | <iteration>
             | <print>

<comment> ::= "//" {<character>}

<tile-definition> ::= "tile" <identifier> <matrix>

<matrix> ::= "[" {<row>} "]"

<row> ::= "[" {<cell-value>} "]"

<cell-value> ::= "0" | "1"

<variable-declaration> ::= "let" <identifier> [":" <tile-type>] "=" <expression>

<tile-type> ::= "Tile" <positive-integer> "x" <positive-integer>

<operation> ::= <rotate>
```

```
             | <hjoin>
             | <vjoin>

<rotate> ::= "rotate(" <expression> "," <angle> ")"

<angle> ::= "90" | "180" | "270"

<hjoin> ::= "hjoin(" <expression> "," <expression> ")"

<vjoin> ::= "vjoin(" <expression> "," <expression> ")"

<iteration> ::= "for" <identifier> "in" <range> "{" {<statement>} "}"

<range> ::= <positive-integer> ".." <positive-integer>

<print> ::= "print(" <expression> ")"

<expression> ::= <identifier>
              | <operation>
              | <matrix>

<identifier> ::= <letter> {<letter> | <digit>}

<letter> ::= "a" | ... | "z" | "A" | ... | "Z"

<digit> ::= "0" | ... | "9"

<positive-integer> ::= <digit> {<digit>}
```

1.2 Examples

1.2.1 Defining tiles

```
tile T1 [
  [1, 0],
  [0, 1]
]
```

1.2.2 Variables

```
let myTile = T1
```

1.2.3 Types

There are two variations of the types of tiles you can use. You can use the type that was defined above, or you can use a fixed size tile:

```
let myTile : Tile2x2 = T1
```

This represents the size of the tile, so this one is 2 by 2.

1.2.4 Operations (rotation, vertical and horizontal joining)

```
let rotatedTile = rotate(T1, 90)
let combinedTile = hjoin(T1, rotatedTile)
let stackedTile = vjoin(T1, rotatedTile)
```

1.2.5 Iteration

```
for i in 1..4 {
  let newTile = rotate(myTile, i * 90)
  // Do something with newTile
}
```

1.2.6 Example dummy program

```
// Define a 2x2 tile
tile T1 [
  [1, 0],
  [0, 1]
]

// Define another 2x2 tile
tile T2 [
  [0, 1],
  [1, 0]
]

// Declare a variable and store T1 in it
let myTile: Tile2x2 = T1

// Rotate T1 by 90 degrees
let rotatedTile = rotate(T1, 90)

// Join T1 and rotatedTile horizontally
let combinedTile = hjoin(T1, rotatedTile)

// Join T1 and rotatedTile vertically
let stackedTile = vjoin(T1, rotatedTile)

// Iterate over rotations of T2 and join them horizontally
let finalTile = T2
for i in 1..3 {
  let newTile = rotate(T2, i * 90)
  finalTile = hjoin(finalTile, newTile)
}

// Print the final result
print(finalTile)
```