import picobot._
import picobot.Picobot._

Picobot("empty.txt") {
    0 where * | * | x | * -> W goto 0
    0 where x | * | W | * -> N goto 1
    1 where x | * | * | * -> N goto 1
    1 where N | * | * | x -> S goto 2
    2 where * | * | * | x -> S goto 2
    2 where * | x | * | S -> E goto 3
    3 where x | * | * | * -> N goto 3
    3 where N | x | * | * -> E goto 2
}
rule ::= state where pattern -> direction goto state
pattern ::= ndirection | edirection | wdirection | sdirection
ndirection ::= N | odirection
edirection ::= E | odirection
wdirection ::= N | odirection
sdirection ::= S | odirection
odirection ::= * | x
direction ::= N | E | W | S | x
state \in Int
program ::= Picobot("filename")\{rule*\}
Syntactic Domain

```
2 where * | x | * | S -> E goto 3
```

(State PicoPattern edirection ndirection wdirection sdirection direction state)

```
(2.where(((.*|((x)).|(*)).|(S.->(E)))).goto(3)
```

### Operator & Precedence

<table>
<thead>
<tr>
<th>Operator &amp; Precedence</th>
<th>Associativity</th>
</tr>
</thead>
<tbody>
<tr>
<td>special characters</td>
<td>left, unless the right-most character is :</td>
</tr>
<tr>
<td>* / %</td>
<td>left</td>
</tr>
<tr>
<td>+ -</td>
<td>left</td>
</tr>
<tr>
<td>:</td>
<td>right</td>
</tr>
<tr>
<td>= !</td>
<td>left</td>
</tr>
<tr>
<td>&lt; &gt;</td>
<td>left</td>
</tr>
<tr>
<td>&amp;</td>
<td>left</td>
</tr>
<tr>
<td>^</td>
<td>left</td>
</tr>
<tr>
<td>all letters</td>
<td>left</td>
</tr>
</tbody>
</table>
Syntactic Domain

2 where * | x | * | S -> E goto 3

(2.where(((.*|(x)).|(*)).|(S.->(E)))).goto(3)

Picobot_Internal.scala

```scala
...  
type SD = (PicoPattern, PicoMove)
class SDHelper(p: PicoPattern) {
  def ->(m: PicoMove): SD = (p, m)
}
implicit def s2SDHelper(p: PicoPattern): SDHelper = new SDHelper(p)

type NE = (PicoPattern, PicoPattern)
class NEHelper(n: PicoPattern) {
  def |(e: PicoPattern): NE = (n, e)
}
implicit def n2NEHelper(n: PicoPattern): NEHelper = new NEHelper(n)

type NEW = (PicoPattern, PicoPattern, PicoPattern)
class NEWHelper(ne: NE) {
  def |(w: PicoPattern): NEW = (ne._1, ne._2, w)
}
implicit def w2NEWHelper(ne: NE): NEWHelper = new NEWHelper(ne)
...  
```
Syntactic Domain

Intermediate Representation

2 where * | x | * | S -> E goto 3