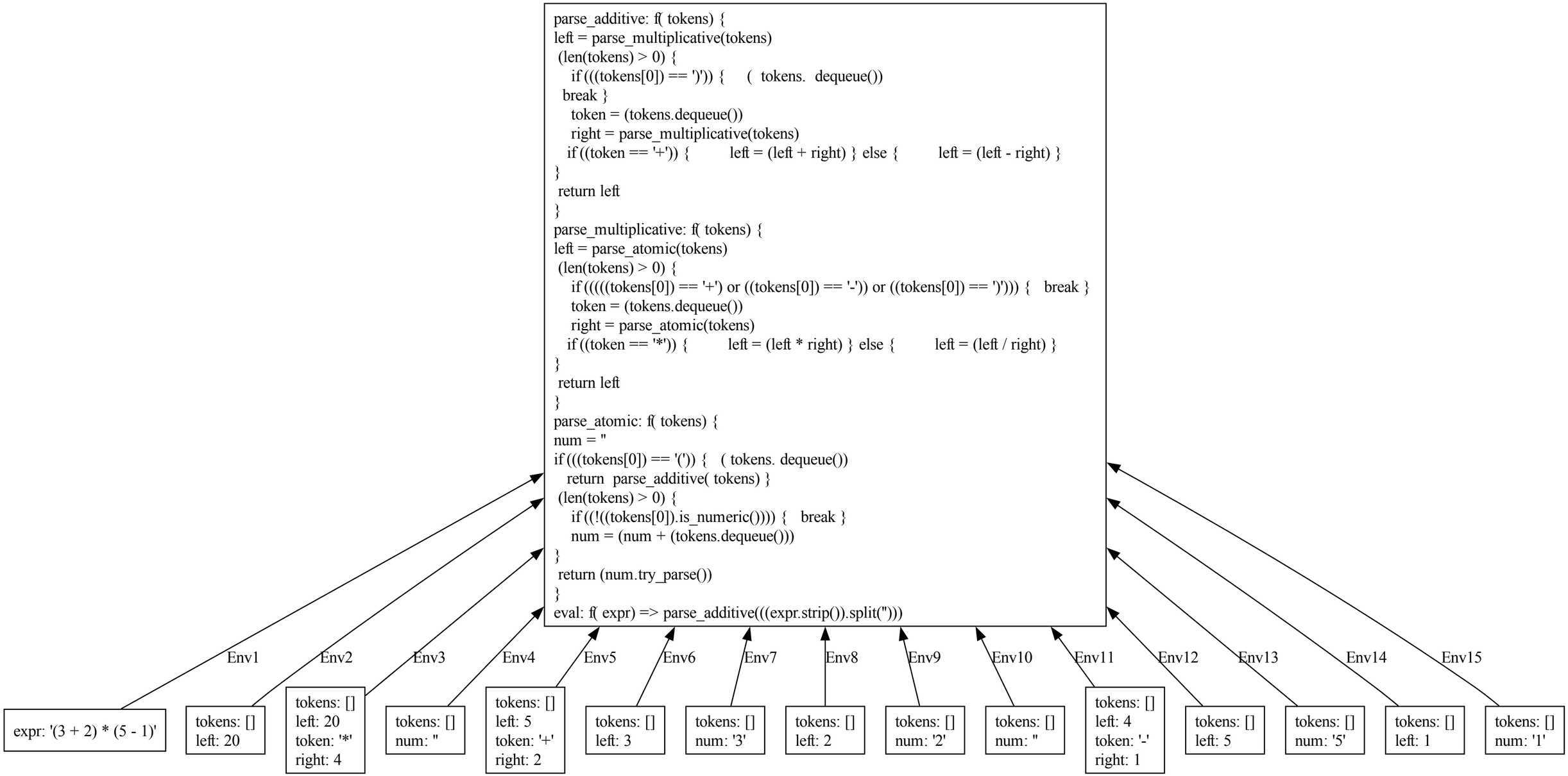
parse_additive: f(tokens) {
left = parse_multiplicative(tokens)
(len(tokens) > 0) {
if (((tokens[0]) == ')')) { (tokens. dequeue())
break }
token = (tokens.dequeue())
right = parse_multiplicative(tokens)
if ((token == '+')) { left = (left + right) } else { left = (left - right) }
}
return left
}
parse_multiplicative: f(tokens) {
left = parse_atomic(tokens)
(len(tokens) > 0) {
if (((((tokens[0]) == '+') or ((tokens[0]) == '-')) or ((tokens[0]) == ')'))) { break }
token = (tokens.dequeue())
right = parse_atomic(tokens)
if ((token == '*')) { left = (left * right) } else { left = (left / right) }
}
return left
}
parse_atomic: f(tokens) {
num = ''
if (((tokens[0]) == '(')) { (tokens. dequeue())
return parse_additive(tokens) }
(len(tokens) > 0) {
if (((!((tokens[0]).is_numeric())))) { break }
num = (num + (tokens.dequeue()))
}
return (num.try_parse())
}
eval: f(expr) => parse_additive(((expr.strip()).split('')))
Env1
Env2
Env3
Env4
Env5
Env6
Env7
Env8
Env9
Env10
Env11
Env12
Env13
Env14
Env15
expr: '(3 + 2) * (5 - 1)'
tokens: []
left: 20
tokens: []
left: 20
token: '*'
right: 4
tokens: []
num: ''
tokens: []
left: 5
token: '+'
right: 2
tokens: []
left: 3
tokens: []
num: '3'
tokens: []
left: 2
tokens: []
num: '2'
tokens: []
num: ''
tokens: []
left: 4
token: '-'
right: 1
tokens: []
left: 5
tokens: []
num: '5'
tokens: []
left: 1
tokens: []
num: '1'