Jupiter, The Julia Compiler

Gabriela Hecksher, Julia Noce

Data:26/06/2019

Universidade Federal Fluminense

Objetivos desta apresentação

Nessa apresentação iremos demonstrar a entrega do trabalho equivalente a p2 da disciplina.

O que foi feito

Especificar a semântica de declarações utilizando o Framework. OK

Implementar um parser para a linguagem Imp-1 estendendo Imp-0 com declarações de variáveis e constantes. OK

Implementar IR-mark1: (i) Interpreting Automata com ambientes, (ii) declarações de variáveis e constantes. OK

Implementar um compilador de Imp-1 para IR-mark1. OK

Op Code e operações com tipos

O que foi feito

```
function handle_Blk(element, control_stack, value_stack, env, store, locations)
    operands = element.val
    control_stack = push(control_stack, op_cmd)
    control_stack = push(control_stack, operands[2])

    control_stack = push(control_stack, op_blk)
    control_stack = push(control_stack, operands[1])

    value_stack = push(value_stack, locations)

    automaton(control_stack, value_stack, env, store, locations)
end


function handle_Ref(element, control_stack, value_stack, env, store, locations)
    control_stack = push(control_stack, op_ref)
    if typeof(element.val) <: Array{Any,1}
        control_stack = push(control_stack, element.val[1])
    else
        control_stack = push(control_stack, element.val)
    end

    automaton(control_stack, value_stack, env, store, locations)
end
```

Figura 1: Handle

O que foi feito

```
function handle_DeRef(element, control_stack, value_stack, env, store, locations)
    operand = element.val[1]
    id = operand.val # pega o identificador da variavel
    value_stack = push(value_stack, store[store[env[id]]]) # coloca a location associada a variavel na pilha de valores
    automaton(control_stack, value_stack, env, store, locations)
end

function handle_ValRef(element, control_stack, value_stack, env, store, locations)
    operand = element.val[1]
    id = operand.val # pega o identificador da variavel
    value_stack = push(value_stack, env[id]) # pega o valor apontado por um ponteiro
    automaton(control_stack, value_stack, env, store, locations)
    #ex x := &y  ValRef(Id(x))
    #ex env = [x->loc2, y->loc3] store = [loc2->loc3, loc3->8]
    #nesse caso receberia x e retornaria 8
end
```

```
function handle_DSeq(element, control_stack, value_stack, env, store, locations)
    operands = element.val

    control_stack = push(control_stack, operands[1]) #coloca as expressões a serem somadas na pilha de controle
    control_stack = push(control_stack, operands[2])

    automaton(control_stack, value_stack, env, store, locations)
end
```

O que foi feito

```
mutable struct Loc
    val :: Int
end
function calc(op, control_stack, value_stack, env, store, locations)
    if typeof(op) <: opCodeSum
        calc_sum(control_stack, value_stack, env, store, locations)
    elseif typeof(op) <: opCodeMul
        calc_mul(control_stack, value_stack, env, store, locations)
    elseif typeof(op) <: opCodeSub
        calc_sub(control_stack, value_stack, env, store, locations)
    elseif typeof(op) <: opCodeDiv
        calc_div(control_stack, value_stack, env, store, locations)
```

Figura 2: Calc

O que foi feito

```
function calc_cond(control_stack, value_stack, env, store, locations)
    condition = popfirst!(value_stack)
    command = popfirst!(value_stack)
    values = command.val

    control_stack = pop(control_stack)
    if condition
        control_stack = push(control_stack, values[2])
    else
        control_stack = push(control_stack, values[3])
    end
    automaton(control_stack, value_stack, env, store, locations)
end


function calc_bind(control_stack, value_stack, env, store, locations)
    loc = popfirst!(value_stack)
    identifier = popfirst!(value_stack)

    next = popfirst!(value_stack)


    if typeof(next) <: Dict # ja existe E'
        next[identifier] = loc
        value_stack = push(value_stack, next) # atualiza E' e coloca de volta na pilha de valores
    else # primeiro bind
        value_stack = push(value_stack, next) # coloca de volta o valor retirado
        new_env = Dict()
        new_env[identifier] = loc
        value_stack = push(value_stack, new_env)
    end

    automaton(pop(control_stack), value_stack, env, store, locations)
end
```

Figura 3: Calc

O que foi feito

```
function calc_ref(control_stack, value_stack, env, store, locations)
    value = popfirst!(value_stack)
    loc = Loc(length(store))
    store[loc] = value
    push(value_stack, loc)
    automaton(pop(control_stack), value_stack, env, store, locations)
end

function calc_blkdec(control_stack, value_stack, env, store, locations)
    result_env = copy_dict(env)
    blk_env = popfirst!(value_stack)
    exists = false
    for (key_blk, value_blk) in blk_env
        for (key, value) in result_env
            if key === key_blk
                result_env[key] = blk_env[key_blk]
                exists = true
            end
        end
        if !exists
            result_env[key_blk] = value_blk
        else
            exists = false
        end
    end
    value_stack = push(value_stack, env)
    automaton(pop(control_stack), value_stack, result_env, store, locations)
end

function calc_blkcmd(control_stack, value_stack, env, store, locations)
    env = copy_dict(popfirst!(value_stack))
    locations = copy_array(popfirst!(value_stack))
    automaton(pop(control_stack), value_stack, env, store, locations)
end
```

Figura 4: Calc

O que foi feito

```
abstract type opCode end

mutable struct opCodeSum <: opCode
    val :: String
end

mutable struct opCodeMul <: opCode
    val :: String
end

mutable struct opCodeSub <: opCode
    val :: String
end
```

Figura 5: OpCodes

```
op_sum = opCodeSum("#SUM")
op_mul = opCodeMul("#MUL")
op_sub = opCodeSub("#SUB")
op_div = opCodeDiv("#DIV")
op_eq = opCodeEq("#EQ")
op_lt = opCodeLt("#LT")
op_gt = opCodeGt("#GT")
op_le = opCodeLt("#LE")
```

O que não foi feito e porque

Quebrar em blocos de espacos.

O print do OpCode. Esta imprimindo "OpCodeSum(#SUM)"ao inves de "#SUM"(mas o automato le e entende o modo correto de #SUM)

A lista de Locations não esta completa.

Dúvidas

Como é o estado o final do BLK?

Avaliação da evolução do trabalho

Houve melhoria no desenvolvimento do trabalho em relacao a P1

Dificuldade em criar tipos abstratos em Julia e mutable structs

Dificuldades de implementar ValRef e DeRef na gramatica(definições pouco claras)

Dificuldades de implementar BLK no automato