Homework report
Definition of the programming language with examples

Bigliazzi Pierfrancesco 550446

Data type for expressions with variables:

type expr =
| Eint of Int
| Ebool of Bool
| Den of ide
| Binop of ide * expr * expr
| Exec of expr
| LetPu of ide * expr * expr
| LetPr of ide * expr * expr
| If of expr * expr * expr
| Fun of ide * expr * perm list * label
| Call of expr * expr
| ReadData of expr
| WriteData of expr
| OpenFile of expr
| SendData of expr

The following programming language has been implemented with these commands:

- **Den(x)**: x is variable which is associated to a specific value stored in the environment.

- **LetPu(s,e1,e2)**: this primitive take an ide s (a string) ad associated it the expression e1. Then it evaluates the variable inside the expression e2. This specific assignment also associates the label Public to the variable.

- **LetPr(s,e1,e2)**: this primitive do the same thing as LetPu but the difference is that the label associated to the variable is Private.

- **Binop(ope,e1,e2)**: this primitive take a string ope which is a possible arithmetic operator (“+”, ”-“, “*”, “=”, “>”, “<”) and it executes the associated operations with e1 and e2 as operands.

- **If(e1,e2,e3)**: the following statement check a condition which an expression e1 and if it is true then it executes the expression e2 otherwise if it is false evaluates the expression e3.

- **Fun(param, body, perms, lab)**: this primitive returns the associated closure to the function with the name of the formal parameter (param), the code of the function (body), a list of permissions associated to the closure (perms) and a label which tells me if the function is Public of Private (lab).

- **Call(f,arg)**: this primitive evaluates the body of a function (f) and his argument (arg). Then it moves into the stack the list of permissions associated to the function, bind the associated closure to the environment and evaluates the body of the function.

- **Exec(e1)**: this primitive is used to check if a mobile code coming from the network can access to local variables or to local functions. To find if it can it checks the label associated to the expression e1 and if it is Private the primitive does not execute the mobile code otherwise it executes it. The expression e1 can be composed by a lot of subexpressions with associated labels. To decide if the expression e1 can be executed or not Exec checks all the labels associated to the subexpressions and if they are all Public then the mobile code can be executed otherwise the primitive stops the execution. To check the labels of all the subexpressions Exec uses a supporting recursive function called **show_label(e1,env).**

- **ReadData(e1)**: this primitive implements a Read operation.

- **WriteData(e1)**: this primitive implements a Write operation with e1 as expression to be written.

- **OpenFile(e1)**: this primitive opens a file whose name is expressed by the expression e1.

- **SendData(e1)**: this primitive sends the expression e1 outside in the network.

Examples:

let sumOpe = LetPu(“x”, Eint(9), Binop(“+”, Den(“x”), Eint(2)));;

let my_pin = LetPr(“pin”, Eint(12345), Den(“pin”));;

let a_test = LetPr(“a”, Eint(8), Den(“a));;

let testSendPu = LetPu(“x”, Fun(“y”, SendData(Eint(2)), [Send], Public), Call(Den(“x”), Eint(4)));;

let testWritePr = LetPr(“x”, Fun(“y”, WriteData(Eint(2)), [Write], Private), Call(Den(“x”), Eint(4)));;

let testExecutePr = Exec(LetPr(“x”, Fun(“y”, ReadData(Ebool(true)), [Read], Private), Call(Den(“x), a_test)));;