Program Transformation and Analysis

Assignment 2

Benjamin Brandt Ohrt, zpn492

May 1, 2019

1 Introduction

This the second of five weekly assignment in the course Program Transformation and Analysis (PAT) at Copenhagen University. The course professor is Robert Glück. The course is held in block 4, 2019.

In this assignment the focus is on Reversible Computing and the Janus programming language: `http://topps.diku.dk/pirc/?id=janus`.

2 Assignment

Exercise 1

The program encode in section 3 is a run-length encoder written in Janus (2007). The store is int i, j, text[10], code[20]. What is the store after running the encoder with the initial text[10] = {3,3,7,7,7,7,5,5,5,0}? Array indexing starts at 0. What are the main differences between this implementation and an implementation of the encoder in a conventional (C-like) imperative language?

The store is like: i = 0, j = 0, text[10] = [0,..,0] and code[20] = [3,2,7,4,5,3,0,..,0]

Janus have unlike a conventional (C-like) language both an entry and an exit condition in every fromloop. do statement is atleast runned once when the entry condition is true and then runned again every time the exit condition is false. the loop is only runned while the exit condition is false. In a conventional C-like language you would probably have some loop running while a condition is true.

You never assign values while inside a function you will instead increment, decrement or swap. Again a C-like language you probably just do an assignment.

State	condition	i	j	text	code
fst from	true	0	0	[3,3,7,7,7,7,5,5,5,0]	[0,..0]
fst from_loop_1	none	0	0	[0,3,7,7,7,7,5,5,5,0]	[3,0,..,0]
sec from_1	true	1	0	[0,3,7,7,7,7,5,5,5,0]	[3,1,0,..,0]
sec from_do_1	none	1	0	[0,3,7,7,7,7,5,5,5,0]	[3,1,0,..,0]
sec from_until_1	false	1	0	[0,3,7,7,7,7,5,5,5,0]	[3,1,0,..,0]
sec from_loop_1	none	1	0	[0,0,7,7,7,7,5,5,5,0]	[3,1,0,..,0]
sec from_do_2	none	2	0	[0,0,7,7,7,7,5,5,5,0]	[3,2,0,..,0]
sec from_until_2	true	2	2	[0,0,7,7,7,7,5,5,5,0]	[3,2,0,..,0]
fst from_until_1	false	2	2	[0,0,7,7,7,7,5,5,5,0]	[3,2,0,..,0]
fst from_loop_2	none	2	2	[0,0,0,7,7,7,5,5,5,0]	[3,2,7,0,..,0]
sec from_2	true	2	2	[0,0,0,7,7,7,5,5,5,0]	[3,2,7,0,..,0]
sec from_2_do_1	none	3	2	[0,0,0,7,7,7,5,5,5,0]	[3,2,7,1,0,..,0]
sec from_2_until_1	false	3	2	[0,0,0,7,7,7,5,5,5,0]	[3,2,7,1,0,..,0]
sec from_2_loop_1	none	3	2	[0,0,0,0,7,7,5,5,5,0]	[3,2,7,1,0,..,0]
sec from_2_do_2	none	4	2	[0,0,0,0,7,7,5,5,5,0]	[3,2,7,2,0,..,0]
sec from_2_until_2	false	4	2	[0,0,0,0,7,7,5,5,5,0]	[3,2,7,2,0,..,0]
sec from_2_loop_2	none	4	2	[0,0,0,0,0,7,5,5,5,0]	[3,2,7,2,0,..,0]
sec from_2_do_3	none	5	2	[0,0,0,0,0,7,5,5,5,0]	[3,2,7,3,0,..,0]
sec from_2_until_3	false	5	2	[0,0,0,0,0,7,5,5,5,0]	[3,2,7,3,0,..,0]
sec from_2_loop_3	none	5	2	[0,0,0,0,0,0,5,5,5,0]	[3,2,7,3,0,..,0]
sec from_2_do_3	none	6	2	[0,0,0,0,0,0,5,5,5,0]	[3,2,7,4,0,..,0]
sec from_2_until_3	true	6	4	[0,0,0,0,0,0,5,5,5,0]	[3,2,7,4,0,..,0]
fst from_until_2	false	6	4	[0,0,0,0,0,0,5,5,5,0]	[3,2,7,4,0,..,0]
fst from_loop_3	none	6	4	[0,0,0,0,0,0,0,5,5,0]	[3,2,7,4,5,0,..,0]
sec from_3	true	6	4	[0,0,0,0,0,0,0,5,5,0]	[3,2,7,4,5,0,..,0]
sec from_3_do_1	none	7	4	[0,0,0,0,0,0,0,5,5,0]	[3,2,7,4,5,1,0,..,0]
sec from_3_until_1	false	7	4	[0,0,0,0,0,0,0,5,5,0]	[3,2,7,4,5,1,0,..,0]
sec from_3_loop_1	none	7	4	[0,0,0,0,0,0,0,0,5,0]	[3,2,7,4,5,1,0,..,0]
sec from_3_do_2	none	8	4	[0,0,0,0,0,0,0,0,5,0]	[3,2,7,4,5,2,0,..,0]
sec from_3_until_2	false	8	4	[0,0,0,0,0,0,0,0,5,0]	[3,2,7,4,5,2,0,..,0]
sec from_3_loop_2	none	8	4	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,2,0,..,0]
sec from_3_do_3	none	9	4	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
sec from_3_until_3	false	9	6	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
fst from_until_3	true	9	6	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
thd from_1	true	9	6	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
thd from_1_do_1	none	6	4	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
thd from_1_until_1	false	6	4	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
thd from_1_do_2	none	2	2	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
thd from_1_until_2	false	2	2	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
thd from_1_do_3	none	0	0	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]
thd from_1_until_3	true	0	0	[0,0,0,0,0,0,0,0,0,0]	[3,2,7,4,5,3,0,..,0]

Exercise 2

Invert the Janus-program. Show the resulting inverse program.

```
procedure encode^{-1}
        from i = 0 do
                i += code[j+1]
                j += 2
        until code[j] = 0
        from text[i] = 0 loop
                j -= 2
                from code[j] != text[i]
                do i -= 1
                        code[j+1] -= 1
                loop text[i] += code[j]
                until code[j+1] = 0
                code[j] <=> text[i]
        until i = 0 && j = 0
```

Exercise 3

Define concisely the program inversion rules that you used.

+= changes to -=
-= changes to +=
swap condition between from and until
reverse the order of the two outer loops
reverse the order of operations within each do and loop statement
!= is left untouched !=
= is left untouched =
&& is left untouched &&
substraction or addition on variables on the
left handside is left untouched.

Exercise 4

Implement a simple discrete physical simulation for free-falling objects in Janus. a. The simulation calculates the velocity v and the height h of a free-falling object in discrete time steps. The program contains four variables:

t:	time
v:	velocity
h:	height
tend:	duration of simulation

At each simulation step, t is incremented by 1 [sec]. The simulation runs for tend steps. For simplicity, we assume that the gravitational acceleration g is 10 [m/s^2]. The equation for calculating the calues of v and h at time step t are:

$$v\_t := v\_\{t-1\} + g \tag{1}$$

$$h\_t := h\_\{t-1\} - v\_t + g/2 \tag{2}$$

Write a Janus program fall that performs the simulation.

```
procedure fall(int t, int v, int h, int tend)
    local int g = 10
    from t = 0 do
        t += 1
        v += g
        h += g/2 - v
    until t = tend
    delocal int g = 10
```

b. Run the program by hand or interpreter. Start with the initial store (t=0, v=0, h=176, tend=3)

State	condition	t	v	h
from	true	0	0	176
do_1	none	1	10	171
until_1	false	1	10	171
do_2	none	2	20	156
until_2	false	2	20	156
do_3	none	3	30	131
until_3	true	3	30	131

Exercise 5

Solve the inverse problem: When an object is dropped from the top of a tower in Copenhagen, it takes four second for it to reach the ground and its velocity is 40 $[m/s]$. What is the height of the this tower? Use program fall to calculate the height. Show the initial store and how you ran the program to calculate the solution.

```
procedure fall^{-1}(int t, int v, int h, int tend)
    local int g = 10
    from t = tend do
        h -= g/2 - v
        v -= g
        t -= 1
    until t = 0
    delocal int g = 10
```

We run the program with a initial store (t=4, v=40, h=0, tend=4).

State	condition	t	v	h
from	true	4	40	0
do_1	none	3	30	35
until_1	false	3	30	35
do_2	none	2	20	50
until_2	false	2	20	60
do_3	none	1	10	65
until_3	false	1	10	75
do_4	none	0	0	70
until_4	true	0	0	80

The height h of the tower is 80.

3 Janus - encode

```
procedure encode
        from i = 0 && j = 0 loop
                code[j] <=> text[i]
                from code[j+1] = 0
                do code[j+1] += 1
                        i += 1
                loop text[i] -= code[j]
                until code[j] != text[i]
                j += 2
        until text[i] = 0
        from code[j] = 0 do
                j -= 2
                i -= code[j+1]
        until i = 0
```

References

[1] Yokoyama, T. and Glück, R. *A reversible programming language and its invertible self-interpreter.*. ACM, 2007.