Program Transformation and Analysis

Assignment 3

Benjamin Brandt Ohrt, zpn492

May 15, 2019

1 Introduction

This the fourth 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 partial evaluation.

2 Partial Evaluation

In partial evaluation the first example is a two-input program, which can be specialized into a one-input program, if we can make the other input static and create a new program of the two-input program one static input and produce a one-input program.

3 Online Partial Evaluation

Partial evaluation in three steps:

- Collect all reachable states

  see which blocks of the program we need and what data will arrive at each block.

- Program point specialization

  for each reachable state (l,σ), create a version of l, that is specialized wrt σ.

- Transition compression

  remove trivial jumps (gotos)

  'Definition 4.5 Let pp be a label occuring in program p, and consider a jump goto pp. The replacement of this goto by a copy of the basic block labelled pp is transition compression.'[1, p. 81]

3.1 Online Partial Evaluation algorithm

S $:= \emptyset$
P $:= (l_{init}, \sigma_{init})$

(0) while P is not empty do

$(l, \sigma) := remove(P)$
(1) $if (l, \sigma) \notin S$
$insert((l, \sigma), S)$

/* Let b' be the result */
/* of specializing the */
/* block labelled l */
/* and let S be */
/* the new */
/* configuration */
/* generated */

$insert((b', R)$

(2) $foreach (l', \sigma') \in S$
(3) if $l' \notin halt$
then $insert((l', \sigma'), P)$
(3e) $endif$
(2e) $endfor$
(1e) $endif$
(0e) $endwhile$

4 Assignment

Exercise 1

A

Show the main steps of specializing the Turing-machine interpreter with respect to the Turing-program into the target program. Use online or offline FCL-partial evaluator.

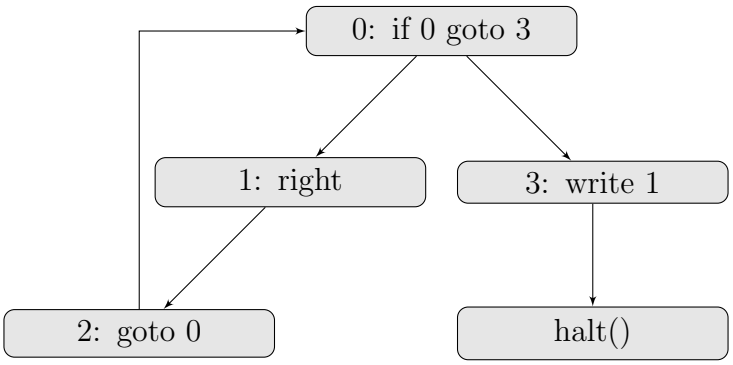
```
prg(String s):
        0: if 0 goto 3
        1: right
        2: goto 0
        3: write 1
```

We use the online FCL-partial evaluation method to specialize the program above. **First we find all reachable states.**

Table over states

	0 (goto loop/jump)	
1 (loop) 2 0 (goto loop/jump)		3 (jump) halt()

Above table shows which states are reachable from another. Ex. State 0, can reach 1 or 3. 1 can reach 2 and 2 can reach 0.



Second we specialize each block with respect to the program prg and the Turing-interpreter. The specialization is bluntly to replace dynamic values in the interpreter with static information known from program.

```
Original block:
  0: if 0 goto 3

Specialized wrt (0, [input->S, symbol->0, nextlabel->3]):
  (0, [input->S, symbol->0, nextlabel->3]) :
        if 0 = firstsym(Right) goto jump else loop;


Original block:
  1: right

Specialized wrt (1, [input->S]):
  (1, [input->S]) :
        Left := cons(firstsym(Right), Left);
        Right:= tl(Right);
        Goto loop;


Original block:
  2: goto 0

Specialized wrt (2, [input->S, nextlabel->0]):
  (2, [input->S, nextlabel->0]) :
        new_tail(0, Q);
        Goto loop;


Original block:
  3: write 1

Specialized wrt (3, [input->S, symbol->1, nextlabel->stop]):
  (3, [input->S, symbol->1]) :
        Right:= cons(1, tl(Right));
        return right;
```

As the third and last step, we use transition compression on trivial jumps. Its shown in above specialization that the only purpose of instruction 2 is to move the instruction pointer to state 0, and can therefore be removed. So instead of letting state 2 follow up on state 1 we extend state 1 with the combination of state 1 and 0.

```
Specialized wrt (1, [input->S]):
  (1, [input->S, symbol->0]) :
        Left := cons(firstsym(Right), Left);
        Right:= tl(Right);
        if 0 = firstsym(Right) goto jump else loop;
```

Resulting in the the following specialized program:

```
read(Right);
0:
        Left := '();
        if 0 = firstsym(Right) goto jump else loop;
loop:
        Left := cons(firstsym(Right), Left);
        Right:= tl(Right);
        if 0 = firstsym(Right) goto jump else loop;
jump:
        Right:= cons(1, tl(Right));
        return right;
```

B

The transformation in A, performs The First Futamura projection, which takes an S-interpreter $[int]$ written in a language L and s as a S-program and specializes the interpreter with respect to s.

$$\begin{aligned}[s] &= [int]_L[s] \\ &= [([mix]_L, [int, s])]_L \\ &= [target]_L\end{aligned}$$

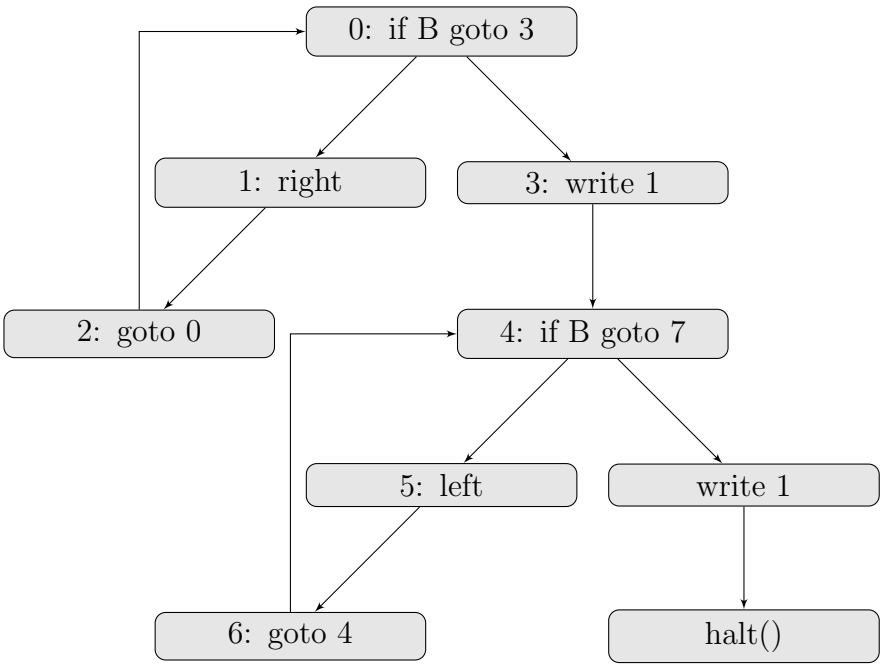
As shown above, the specialized program is written the same language as the interpreter.

Exercise 2

Answer exercise 4.5[1]: Specialize the Turing interpreter with respect to the following program:

```
0: if B goto 3
1: right
2: goto 0
3: write 1
4: if B goto 7
5: left
6: goto 4
7: write 1
```

	0 (goto loop/jump)		
1 (loop) 2 0 (goto loop/jump)	 5 (loop1) 6 4 (goto loop1/jump1)	3 (jump) 4 (goto loop1/jump1)	 7 (jump1) halt()



```
Original block:
  0: if B goto 3

Specialized wrt (1, [input->S, symbol->B, nextlabel->3]):
        if B = firstsym(Right) goto jump else loop;


Original block:
  1: right

Specialized wrt (1, [input->S]):
  (1, [input->S]) :
        Left := cons(firstsym(Right), Left);
        Right:= tl(Right);
        Goto loop;


Original block:
  2: goto 0

Specialized wrt (2, [input->S, nextlabel->0]):
  (2, [input->S, nextlabel->0]) :
        new_tail(0, Q);
        Goto loop;


Original block:
  3: write 1

Specialized wrt (3, [input->S, symbol->1, nextlabel->loop1]):
  (3, [input->S, symbol->1]) :
        Right:= cons(1, tl(Right));
        Goto loop1


Original block:
  4: if B goto 7

Specialized wrt (4, [input->S, symbol->B, nextlabel->7]):
```

```
        if B = firstsym(Right) goto jump1 else loop1;


Original block:
  5: left

Specialized wrt (5, [input->S]):
  (5, [input->S]) :
        Right := cons(firstsym(Right), Right);
        Left:= tl(Left);
        Goto loop1;


Original block:
  6: goto 4

Specialized wrt (4, [input->S, nextlabel->4]):
  (4, [input->S, nextlabel->4]) :
        new_tail(4, Q);
        Goto loop1;


Original block:
  7: write 1

Specialized wrt (7, [input->S, symbol->1, nextlabel->stop]):
  (7, [input->S, symbol->1]) :
        Right:= cons(1, tl(Right));
        return right;
```

Again state 2 can be replaced with the combination of state 1 + 0. And state 6 can be replaced by state 5 + 4. We see that state 3 and 4 can be merged as an initial state before the second loop. Which makes the specialization of state 1, 3 and 5 the following:

```
Original block:
  1: right

Specialized wrt (1, [input->S, symbol->B]):
```

```
  (1, [input->S, symbol->B]) :
        Left := cons(firstsym(Right), Left);
        Right:= tl(Right);
        if B = firstsym(Right) then jump else loop


Original block:
  3: write 1

Specialized wrt (3, [input->S, symbol->1, symbol'->B,
nextlabel->loop1]):
  (3, [input->S, symbol->1]) :
        Right:= cons(1, tl(Right));
        if B = firstsym(Right) then jump1 else loop1;


Original block:
  5: left

Specialized wrt (5, [input->S, symbol->B]):
  (5, [input->S, symbol->B]) :
        Right := cons(firstsym(Right), Right);
        Left:= tl(Left);
        if B = firstsym(Right) then jump1 else loop1
```

A merge of above specialization where states with only goto is removed, the specialization can be shown as the following:

```
read(Right)
0:
        Left := '();
        if B = firstsym(Right) then jump else loop;
loop:
        Left := cons(firstsym(Right), Left);
        Right:= tl(Right)
        if B = firstsym(Right) then jump else loop;
jump:
        Right := cons('1, tl(Right));
```

```
        if B = firstsym(Right) then jump1 else loop1;
loop1:
        Right := cons(firstsym(Right), Right);
        Left:= tl(Left);
        if B = firstsym(Right) then jump1 else loop1
jump1:
        Right:= cons(1, tl(Right));
        return right;
```

Excercise 3

Answer exercise 4.7[1]: Will specialization of the Turing interpreter (Figure 4.4[1]) with respect to a program p terminate for all Turing programs p?

References

[1] Jones N.D., Gomard C.K., Sestoft P., *Partial Evaluation and Automatic Program Generation.* Prentice Hall 1993. `https://www.itu.dk/people/sestoft/pebook/pebook.html` (Links to an external site.)