Introduction to Computer Science
Constructor University
Dr. Jürgen Schönwälder

Module: CH-232
Date: 2023-11-24
Due: 2023-12-01

Problem Sheet #12

Problem 12.1: *bnf grammar for boolean expressions* (2+1 = 3 points)

We define the syntax of boolean expressions as follows:

(1) The two Boolean constants 0 and 1 are Boolean formulas.

(2) Every Boolean variable `Xn` is a Boolean formula (where `n` is a natural number)

(3) If A and B are Boolean formulas, then $(A * B)$ is a Boolean formula.

(4) If A and B are Boolean formulas, then $(A + B)$ is a Boolean formula.

(5) If A is a Boolean formula, then A' is a Boolean formula.

For readability, we allow space characters around the binary operators + and * and around the parenthesis. Here are some examples of syntactically valid boolean expressions:

a) Define a grammar in Backus-Naur form (BNF) for the non-terminal start symbol `<EXP>`. Some expressions your grammar should accept:

```
X0
1
0'
X1'
(1*0)
(1+(X5*1))
```

b) Extend the grammer to allow space characters around the binary operators + and * and around the parenthesis, but not before the unary postfix operator `'`. Some expressions your grammar should accept:

```
X0
1
0'
X1'
( 1 * 0 )
(1 + (X5* 1))
```

You may use online tools such as the BNF Playground to test your BNF grammar. Submit your solution as a text file.

Problem 12.2: *operating system processes* (1+1+1 = 3 points)

Consider the following RISC-V assembler program. Assume that all system calls succeed at runtime (error handling code has been left out for brevity).

```
        .global main
        .text                           # text segment (holding machine code)
main:                                   # called by C library's startup code
        addi    sp, sp, -16             # allocate stack frame
        sd      ra, 8(sp)               # save return address
        sd      s0, 0(sp)               # save frame pointer
        addi    s0, sp, 16              # establish new frame point
        jal     fork
        bne     a0, zero, done
        jal     fork
        bne     a0, zero, done
        jal     fork
        la      a0, date
        mv      a1, zero
        jal     execvp
done:
        la      a0, bye
        jal     puts
        mv      a0, zero
        ld      ra, 8(sp)               # restore return address
        ld      s0, 0(sp)               # restore frame pointer
        addi    sp, sp, 16              # deallocate stack frame
        ret                             # return to C library code

        .data                           # data segment (holding data)
bye:
        .asciz  "bye"
date:
        .asciz  "date"
```

a) Annotate the assembler instructions without comments.

b) Draw a tree diagram showing the processes created during the execution of the program (include the initial process created when the program is started in your diagram).

c) What will the program print to the standard output? Explain.

Problem 12.3: *pre- and postconditions* (1+1+2 = 4 points)

Determine the weakest precondition (respectively strongest postcondition) for the following algorithms. Explain how you arrived at your result.

a) What is the strongest postcondition of algorithm 1? Explain.

Algorithm 1

```
Precondition: $X > 8$
$Z := X \cdot 2$
$Y := Z + 2$
$X := Y \cdot 3$
Postcondition: ...
```

b) What is the weakest precondition of algorithm 2? Explain.

Algorithm 2

```
Precondition: ...
$X := X + 2$
$Y := X \cdot 4$
$X := Y - 4$
Postcondition: $X > 10 \wedge Y < 20$
```

c) Determine the weakest precondition for algorithm 3. Explain.

Algorithm 3

```
Precondition: …
$X := 3 \cdot Y - 2$
if $X < 12$ then
    $Y := 3 \cdot X - 9$
else
    $Y := X + 6$
fi
$Y := Y - 2$
Postcondition: $7 \leq Y < 25$
```