R. Hinze, S. Franck Summer Semester 2005

Software Engineering

http://proglang.informatik.uni-freiburg.de/teaching/swt/2005/

Exercise Sheet 9

Deadline: June 21st, 2005

Exercise 1 – Design by Contract: (4 points)
Download the Java-Program "Buffer.java" which models a buffer of objects. Give reasonable pre- and postconditions for all methods and the constructor in class Buffer. Are there any class invariants?
Try to write the pre- and postconditions as correct java-expressions, if possible. If it is not possible, try to describe them in a semi-formal way.

Solution:

Constructor

pre `anzahl >= 0`

post `buf.length == anzahl`

empty

pre *true*

post *true*

full()

pre *true*

post *true*

add(Object o)

pre o != null; !full()

post Old.in == in - 1;

remove()

pre `!this.empty()`

post `Old.in == in + 1; result != null;`

contains(Object o)

pre `o != null`

post *true*

The class invariants are:

range `0 <= in - out && in - out <= buf.length;`

valid content `(forall i : out%buf.length .. in%buf.length-1 . buf[i] != null);`

Exercise 2 – Hoare-Calculus: (6 points)
Consider the program:

```
s := 0;
a := x;
b := y;
while (a > 0)
  if (a mod 2 == 1) then
    a = a - 1;
    s = s + b;
  else
    b = b * 2;
    a = a / 2;
  end if
end while
```

Prove that the postcondition {`s == x * y`} holds after execution of this program if the precondition is {`x >= 0`}, and that the program terminates.

Solution:
We need some abbreviations here: % is the modulo operator. P_1 is $a = a - 1; s = s + b$; and P_2 is $b = b * 2; a = a/2;$. Finally the $=$ in conditions expresses logic equality ($==$).
To prove on while loops we need a loop invariant (LI).
An easy way to find the LI is to look for counters or placeholders. In this case the calculation is carried out with a representing x and b representing y. So we first try to replace x and y in the postcondition with a and b. This yields our first maybe-LI:

$$s = ab$$

This LI is used three times:

1. The LI has to be true when we reach the loop.
2. The loop body has the LI as a postcondition, if LI and the loop condition (in this case `a > 0`) are it's precondition.
3. The LI and the negated loop condition (in this case `!(a > 0)`) imply the postcondition of the loop.

Items 1 and 3 are fairly easy to see, so we verify our LI $s = ab$ against these conditions:
Does it hold on entry to the loop? Definitly no, because s is zero and x and y may hold any value, but our loop invariant demands $0 = xy$.
So here comes the creative part: We can correct our LI by simply subtracting xy. This gives us another maybe LI:

$$s = ab - xy$$

Now the first item holds, because $0 = xy - xy$. How about the third item?
Does the LI and the negated loop condition imply the postcondition? We have $s = ab - xy \wedge \neg(a > 0)$ which is $s = ab - xy \wedge a \leq 0$. We can't derive anything from here. We must have an equality conditon on a to proceed. What can we do?
We use a little trick here: To get our desired $a = 0$ with which we can continue, we have to add the condition $a \geq 0$, since $a \leq 0 \wedge a \geq 0 \Rightarrow a = 0$. We can't change the loop condition (this is the program, which we cannot change), but we may add this constraint to our loop invariant:

$$s = ab - xy \wedge a \geq 0$$

Item 1 still holds, since $x \geq 0$ is the precondition of the whole program.
How about the third item now?

$$\begin{aligned} s = ab - xy \wedge a \geq 0 \wedge \neg(a > 0) \quad &\Rightarrow \quad s = ab - xy \wedge a = 0 \\ &\Rightarrow \quad s = -xy \end{aligned}$$

Almost there! We just have to get rid of the minus, which can be done by swapping ab and xy, so our LI is:

$$s = xy - ab \wedge a \geq 0$$

Don't be scared by now, this was a rather complicated example. This approach is not guaranteed to give you a loop invariant, but it works quite well with most examples, and normally you don't need as many steps as in this example to get the LI.
We know by now, that this LI will work on items 1 and 3, and we hope that it will work on the second item. But we still have to prove all three of them:

1. The Invariant holds on entry to the loop:

$$\dfrac{\dfrac{\{0 = 0 \wedge x \geq 0\}s = 0;\{s = 0 \wedge x \geq 0\}}{\{s = xy - xy \wedge x \geq 0\}a = x;\{s = ay - xy \wedge a \geq 0\}} \qquad \{s = ay - xy \wedge a \geq 0\}b = y\{LI\}}{\{x \geq 0\}s = 0; a = x; b = y;\{LI\}}$$

2. If the condition of the loop and the invariant hold, than the invariant holds after one execution of the while body. This body is and if-then-else, so let B_1 be the if-condition $a\%2 = 1$ and apply the if-rule:

$$\dfrac{\{B_1 \wedge LI \wedge a > 0\}P_1\{LI\} \qquad \{\neg B_1 \wedge LI \wedge a > 0\}P_2\{LI\}}{\{LI \wedge a > 0\}\text{if } (B_1) \text{ then } P_1 \text{ else } P_2\{LI\}}$$

   Now we can proof the parts P_1 and P_2 separately:

$$\dfrac{\dfrac{\{s + b = xy - (a-1)b \wedge a - 1 \geq 0\}a = a - 1;\{s + b = xy - ab \wedge a \geq 0\}}{\{s + b = xy - ab \wedge a \geq 0\}s = s + b;\{LI\}}}{\{s = xy - ab \wedge a - 1 \geq 0\}a = a - 1; s = s + b;\{LI\}}$$

   Since $s = xy - ab \wedge a - 1 \geq 0 \Rightarrow s = xy - ab \wedge a \geq 0 \wedge a > 0 \wedge a\%2 = 1$ we can deduce that $\{LI \wedge B_1 \wedge a > 0\}P_1\{LI\}$ The second part of the if can be proved in the same way:

$$\dfrac{\dfrac{\{s = xy - 2\frac{a}{2}b \wedge \frac{a}{2} \geq 0\}b = b * 2;\{s = xy - \frac{a}{2}b \wedge \frac{a}{2} \geq 0\}}{\{s = xy - \frac{a}{2}b \wedge \frac{a}{2} \geq 0\}a = a/2;\{LI\}}}{\{s = xy - 2\frac{a}{2}b \wedge \frac{a}{2} \geq 0\}b = b * 2; a = a/2;\{LI\}}$$

   With $s = xy - 2\frac{a}{2}b \wedge \frac{a}{2} \geq 0 \Rightarrow s = xy - ab \wedge a \geq 0 \wedge a > 0 \wedge a\%2 = 0$ we get $\{LI \wedge B_1 \wedge a > 0\}P_1\{LI\}$.

3. The loop invariant and the negated condition of the while loop imply the postcondition.

$$\begin{aligned} LI \wedge \neg a > 0 &= s = xy - ab \wedge a \geq 0 \wedge a \leq 0 \\ &\Rightarrow s = xy - ab \wedge a = 0 \\ &\Rightarrow s = xy \end{aligned}$$

Our termination term t will be a, and our well-founded ordering is $(\mathbb{N}, <)$.
First we have to prove that $a \in \mathbb{N}$. a cannot become a friction, since it is only divided by 2 if even, and the loop invariant ensures that $a \geq 0$. Therefore $a \in \mathbb{N}$.
No we have to show that $t_{after} < t_{before}$. We have to consider both cases:

`a mod 2 == 1` In this case $t_{after} = t_{before} - 1$ and therefore $t_{after} < t_{before}$.

`a mod 2 == 0` In this case $t_{after} = t_{before}/2$ and since $a \geq 2$ $t_{after} < t_{before}$.

Thus, our function terminates.
But how fast is it? (This analysis was not part of the exercise)
With every iteration of the loop a is either halfed or reduced by 1. Since a can only be halfed if $a \geq 2$, the division results in a reduction of at least 1. Hence, the loop has to terminate after a maximum of at least a executions.
But we can give a smaller maximum of executions:
Consider a in binary format. If a ends with a 0, it is divided by 2, so the length of a's binary representation is reduced by one. If a ends with a 1, one is subtracted, and the number of 1s in the binary represesantation is reduced by one.
Hence, the maximum number of execution steps is the number of digits plus the number of 1s of the binary representation of a.
The worstcase therefore is $2 \cdot \log_2(a)$, the average case $1.5 \cdot \log_2(a)$.