1 Error corrections

- 5.6 on p57: It is said that the C99 standard does not enforce short-circuit evaluation. This is wrong. The logical AND "&&", logical OR "||", and the conditional operator "?:" have short-circuit evaluation semantics. E.g. if the left operand of the logical OR is true then the right operand is not evaluated anymore. Therefore the example $1||(x/0)$ is safe, because the subexpression $x/0$ is never evaluated. Nevertheless, it is true that except for these three operators the order of evaluation of subexpressions and the order in which side effects take place are both unspecified.

2 Additional notes

- C32SAT assumes two's complement semantics. As a result of this the expression $INT\_MIN/-1$ leads to an overflow which would not be the case if C32SAT used one's complement representation. Note that the C99 standard does not guarantee two's complement semantics. Nevertheless, nearly all modern compilers use two's complement semantics so this design decision has been useful.

- In the thesis it is said that dividing by zero leads to a terminating trap in real programs. Although this is nearly on every system the case, it is not enforced by the C99 standard. If the right operand of the division or modulo operator is zero then the behaviour is simply undefined. Undefined behaviour ranges from ignoring the situation to terminating execution. In this sense dividing by zero is as dangerous as shifting by a negative integer which also leads to undefined behaviour.