1)

Real division

var dividend, divisor, quotient : Real

pre divisor $\neq 0$

post (dividend = quotient’ * divisor)

Real division

var dividend, divisor, quotient : Real

pre **true**

post

```
    if divisor ≠ 0 then
      ( dividend = quotient’ * divisor )
    else quotient’ = 0
```

Real division

var dividend, divisor, quotient : Real

pre divisor $\neq$ 0

post (dividend = quotient’ * divisor)
dividend’ = dividend &
divisor’ = divisor

2)

```
Integer division
  var dividend, divisor, quotient, rest : Real

  pre   divisor $\neq 0$ & dividend $\in$ Nat & divisor $\in$ Nat

  post  ( dividend = (quotient’ * divisor) + rest’ ) &
        rest’ < divisor & quotient’ $\in$ Nat
```

```
Integer division
  var dividend, divisor, quotient, rest : Real

  pre  dividend $\in$ Nat & divisor $\in$ Nat

  post
       if divisor $\neq$ 0 then
         ( dividend = (quotient’ * divisor) + rest’ ) & rest’ < divisor  & quotient’ $\in$ Nat
       else quotient’ = 0
```

```
Integer division
  var dividend, divisor, quotient, rest : Real

  pre   divisor $\neq$ 0 & dividend $\in$ Nat & divisor $\in$ Nat

  post  ( dividend = (quotient’ * divisor) + rest’ ) &
        rest’ < divisor & dividend’ = dividend &
        divisor’ = divisor & quotient’ $\in$ Nat
```

3) The specifications in exercise 2) strengthen both the assumptions/pre-conditions and guarantees/post-conditions of the specifications in exercise 1) (by adding new conjuncts). They are therefore not refinements of the specifications in exercises 1).

4) The specifications in exercise 1) weaken both the assumptions/pre-conditions and the guarantees/post-conditions of the specifications in exercise 2). They are therefore not refinements of the specifications in exercise 2).

5) NOTE: In this exercise we only work with positive numbers

8-bit addition

var x,y,z : Nat

pre $x+y \leq 255$

post $z' = x+y$

8-bit subtraction

var x,y,z : Nat

pre $x \leq 255 \;\&\; y \leq x$

post $x = z'+y$

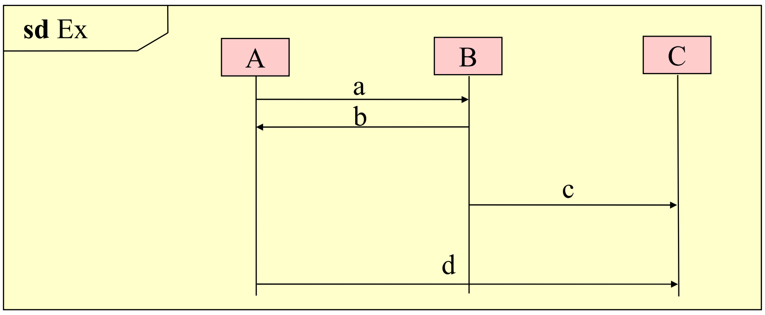
8-bit multiplication

var x,y,z : Nat

pre $x \times y \leq 255$

post $z' = x \times y$

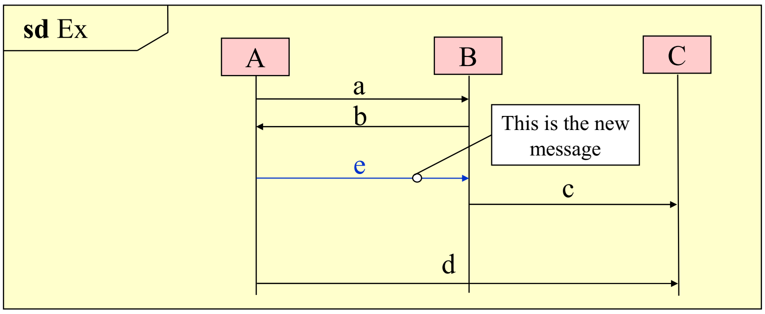
6)



In the original diagram there are six possible traces:

<!a, ?a, !b, ?b, !c, ?c, !d, ?d>
<!a, ?a, !b, ?b, !c, !d, ?c, ?d>
<!a, ?a, !b, ?b, !d, !c, ?c, ?d>
<!a, ?a, !b, !c, ?b, ?c, !d, ?d>
<!a, ?a, !b, !c, ?b, !d, ?c, ?d>
<!a, ?a, !b, !c, ?c, ?b, !d, ?d>

6)

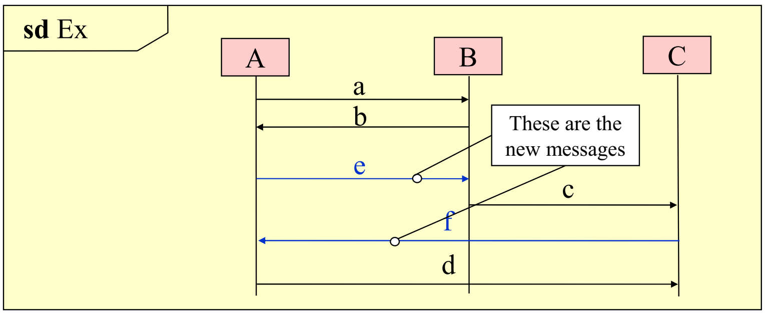


This is the new message

By adding one message we reduce the number of possible traces to four:

<!a, ?a, !b, ?b, !e, ?e, !c, ?c, !d, ?d>
<!a, ?a, !b, ?b, !e, ?e, !c, !d, ?c,?d>
<!a, ?a, !b, ?b, !e, !d, ?e, !c, ?c,?d>
<!a, ?a, !b, ?b, !e, ?e, !d, !c, ?c,?d>

7)

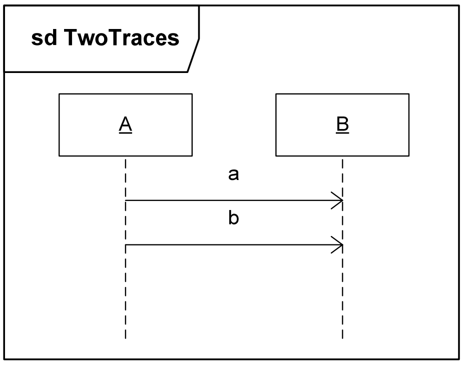


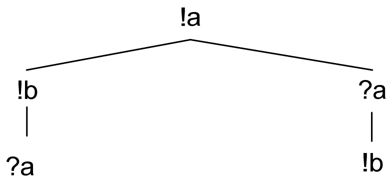
By adding two messages we reduce the number of possible traces to one:

<!a, ?a, !b, ?b, !e, ?e, !c, ?c, !f, ?f, !d, ?d>

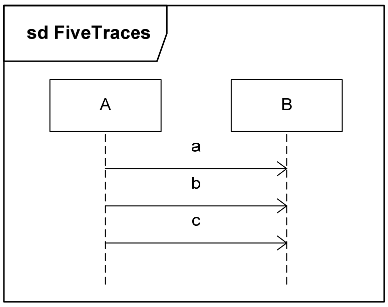
8) & 9) Traces

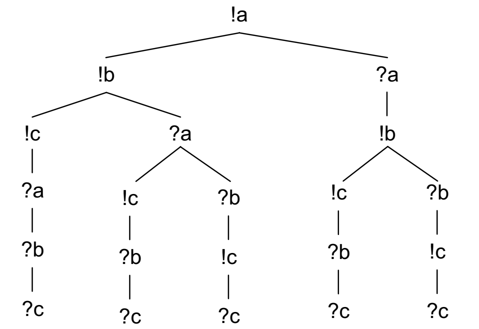
- **Weak sequencing:**
  - **events from the same lifeline are ordered in the trace in the same order as on the lifeline**
  - **events on different lifelines from different operands may come in any order**





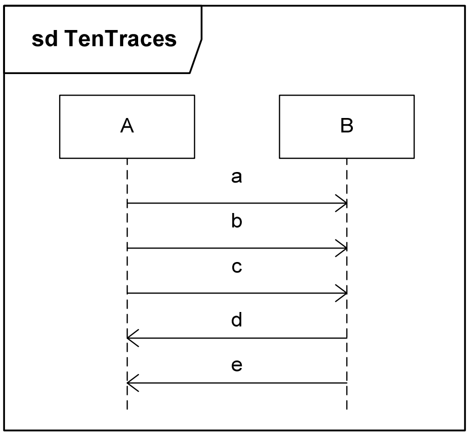
9) Obtain five traces using the seq-operator

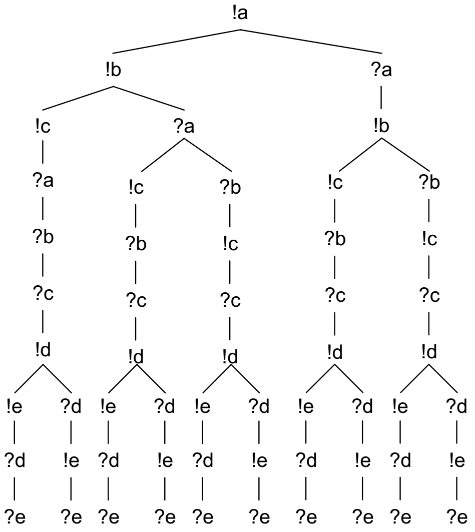




<!a, ?a>**seq**<!b, ?b>**seq**<!c, ?c>= {<!a, !b, !c, ?a, ?b, ?c>,
<!a, !b, ?a, !c,?b, ?c>,
<!a, !b, ?a,?b, !c, ?c>,
<!a, ?a, !b, !c, ?b, ?c>,
<!a, ?a, !b, ?b, !c, ?c>}

8) Ten traces = five times two


sd TenTraces
A
B
a
b
c
d
e


!a
!b
?a
!c
?a
!b
?a
!c
?b
!c
?b
?b
!b
?b
!c
?b
!c
?c
?c
?c
?c
?c
!d
!d
!d
!d
!d
!e
?d
!e
?d
!e
?d
!e
?d
!e
?d
?d
!e
?d
!e
?d
!e
?d
!e
?d
!e
?e
?e
?e
?e
?e
?e
?e
?e
?e
?e