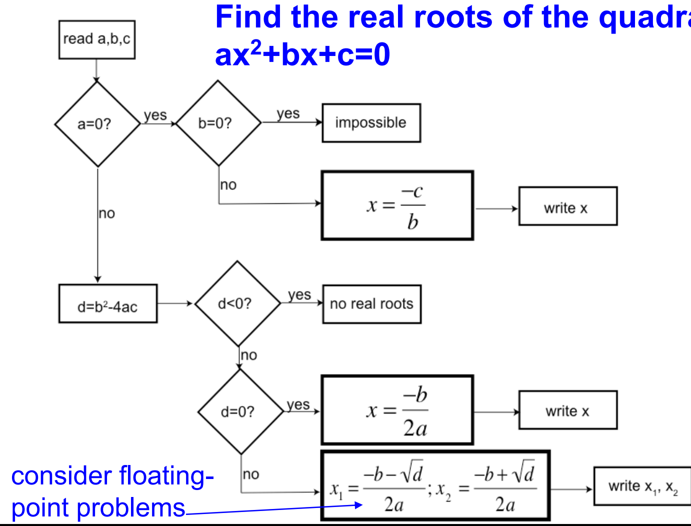
A (necessary) digression
Formal Specifications

- Specifications are central to many aspects of software engineering
- A specification is just a precise description of behaviour
- Requirements specifications describe required behaviour
- Design specifications describe the behaviour prescribed in a design

43

Specifications - 1

Find the real roots of the quadratic $ax^2+bx+c=0$



44

Specifications - 2

if ($a=0$ & $b=0$) then solution_caption := "Ill-conditioned"
else if ($b \neq 0$) then
    solution_caption := "One root"; $x_1 := -c/b$
else
    $d := b^2 - 4ac$
    if ($d<0$) then solution_caption := "No real roots"
    else if ($d=0$) then
        solution_caption := "One real root"; $x_1 = -b/(2a)$
    else
        solution_caption := "Two roots"
        if ($b \geq 0$) then $x_1 := (-b-\sqrt{d})/(2a)$
        else $x_1 := (-b+\sqrt{d})/(2a)$
        $x_2 := c/(ax_1)$

45

Specifications - 2

if ($a=0$ & $b=0$) then solution_caption := "Ill-conditioned"
else if ($b \neq 0$) then
    solution_caption := "One root"; $x_1 := -c/b$
else
    $d := b^2 - 4ac$
    if ($d<0$) then solution_caption := "No real roots"
    else if ($d=0$) then
        solution_caption := "One real root"; $x_1 = -b/(2a)$
    else
        solution_caption := "Two roots"
        if ($b \geq 0$) then $x_1 := (-b-\sqrt{d})/(2a)$
        else $x_1 := (-b+\sqrt{d})/(2a)$
        $x_2 := c/(ax_1)$

Is this a "good" specification?

46

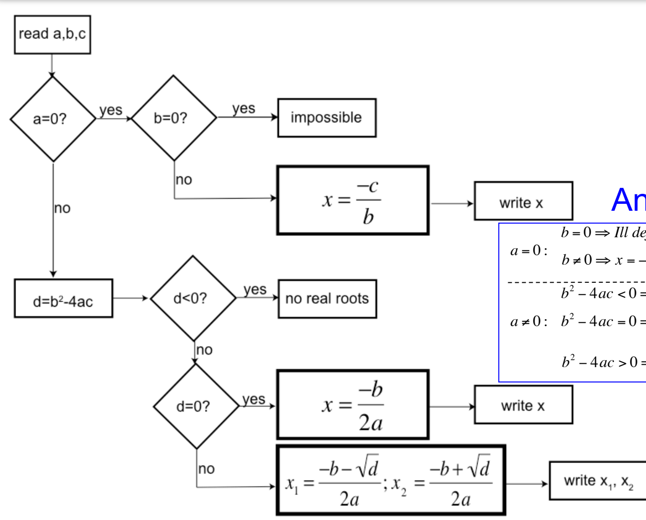
Specifications - 2

```
if (a=0 & b=0) then solution_caption := “Ill-conditioned”
else if (b≠0) then
    solution_caption := “One root”; x1 := -c/b
else
    d := b2-4ac
    if (d<0) then solution_caption := “No real roots”
    else if (d=0) then
        solution_caption := “One real root”; x1 = -b/(2a)
    else
        solution_caption := “Two roots”
        if (b≥0) then x1 := (-b-√d)/(2a)
        else x1 := (-b+√d)/(2a)
        x2 := c/(ax1)
```

Is this a “good” specification?

Is it easy to understand?
Is it a requirement spec or is it a design spec?
What’s the difference?

Specifications - 3



Analysis

$a = 0:$ $b = 0 \Rightarrow \textit{Ill defined}$; $b \neq 0 \Rightarrow x = -\frac{c}{b}$

$a \neq 0:$ $b^2 - 4ac < 0 \Rightarrow \textit{No real roots}$; $b^2 - 4ac = 0 \Rightarrow x = -\frac{b}{2a}$; $b^2 - 4ac > 0 \Rightarrow x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

Specifications - 4

Tabular expression

			Solution caption	x_1	x_2
$a = 0$	$b = 0$		Ill defined	—	—
	$b \neq 0$		One root	$-\frac{c}{b}$	—
$a \neq 0$	$b^2 - 4ac < 0$		No real roots	—	—
	$b^2 - 4ac = 0$		One root	$-\frac{b}{2a}$	—
	$b^2 - 4ac > 0$	$b \geq 0$	Two roots	$\frac{-b-\sqrt{b^2-4ac}}{2a}$	$\frac{c}{ax_1}$
		$b < 0$	Two roots	$\frac{-b+\sqrt{b^2-4ac}}{2a}$	$\frac{c}{ax_1}$

Specifications - 4

Tabular expression

Think about why this is a great way to specify behaviour …

			Solution caption	x_1	x_2
$a = 0$	$b = 0$		Ill defined	—	—
	$b \neq 0$		One root	$-\frac{c}{b}$	—
$a \neq 0$	$b^2 - 4ac < 0$		No real roots	—	—
	$b^2 - 4ac = 0$		One root	$-\frac{b}{2a}$	—
	$b^2 - 4ac > 0$	$b \geq 0$	Two roots	$\frac{-b-\sqrt{b^2-4ac}}{2a}$	$\frac{c}{ax_1}$
		$b < 0$	Two roots	$\frac{-b+\sqrt{b^2-4ac}}{2a}$	$\frac{c}{ax_1}$

Specifications - 5

Another example

- The operator in a real-time application requests or cancels a setpoint, c_sp, by pressing momentary pushbuttons. There are two pushbuttons, called m_pbON and m_pbOFF. They can take on the values e_pressed and e_notPressed. We specify the required behaviour as follows:

      if (m_pbON = e_pressed ^ m_pbOFF = e_notPressed)
            c_sp = e_requested
      else if (m_pbON = e_notPressed ^ m_pbOFF = e_pressed)
            c_sp = e_cancelled

- This is clearly a simple example. Any problems with this?

51

Specifications - 5

Another example

- The operator in a real-time application requests or cancels a setpoint, c_sp, by pressing momentary pushbuttons. There are two pushbuttons, called m_pbON and m_pbOFF. They can take on the values e_pressed and e_notPressed. We specify the required behaviour as follows:

      if (m_pbON = e_pressed ^ m_pbOFF = e_notPressed)
            c_sp = e_requested
      else if (m_pbON = e_notPressed ^ m_pbOFF = e_pressed)
            c_sp = e_cancelled

- This is clearly a simple example. Any problems with this?
- What if both pushbuttons are not pressed?
- What if both pushbuttons are pressed?

52

Specifications - 5

Another example

- The operator in a real-time application requests or cancels a setpoint, c_sp, by pressing momentary pushbuttons. There are two pushbuttons, called m_pbON and m_pbOFF. They can take on the values e_pressed and e_notPressed. We specify the required behaviour as follows:

  if (m_pbON = e_pressed ^ m_pbOFF = e_notPressed)

  c_sp = e_requested

  else if (m_pbON = e_notPressed ^ m_pbOFF = e_pressed)

  c_sp = e_cancelled

- This is clearly a simple example. Any problems with this?
- What if both pushbuttons are not pressed?
- What if both pushbuttons are pressed?

No specified behaviour then no change in the current value of c_sp. So that makes sense.

Specifications - 5

Another example

- The operator in a real-time application requests or cancels a setpoint, c_sp, by pressing momentary pushbuttons. There are two pushbuttons, called m_pbON and m_pbOFF. They can take on the values e_pressed and e_notPressed. We specify the required behaviour as follows:

  if (m_pbON = e_pressed ^ m_pbOFF = e_notPressed)

  c_sp = e_requested

  else if (m_pbON = e_notPressed ^ m_pbOFF = e_pressed)

  c_sp = e_cancelled

- This is clearly a simple example. Any problems with this?
- What if both pushbuttons are not pressed?
- What if both pushbuttons are pressed?

No specified behaviour then no change in the current value of c_sp.
And it is wrong!

Specifications - 5

Another example

- The operator in a real-time application requests or cancels a setpoint, c_sp, by pressing momentary pushbuttons. There are two pushbuttons, called m_pbON and m_pbOFF. They can take on the values e_pressed and e_notPressed. We specify the required behaviour as follows:

  if (m_pbON = e_pressed ^ m_pbOFF = e_notPressed)

  c_sp = e_requested

  else if (m_pbON = e_notPressed ^ m_pbOFF = e_pressed)

  c_sp = e_cancelled

- It is not clear to readers whether the unstated cases are really "no action" cases, or oversights. In this example, if both pushbuttons are not pressed it is reasonable (and correct) to assume there should be no change to c_sp. If both pushbuttons are pressed, the domain experts (in the case I know about) wanted c_sp = e_requested!

55

Specifications - 6

Again, a tabular expression makes the specification clear

Condition	*Result* **c_sp**
(m_pbON = e_notPressed) & (m_pbOFF = e_notPressed)	No Change
(m_pbON = e_notPressed) & (m_pbOFF = e_pressed)	e_cancelled
(m_pbON = e_pressed) & (m_pbOFF = e_notPressed)	e_requested
(m_pbON = e_pressed) & (m_pbOFF = e_pressed)	e_requested

56

Specifications - 6

Again, a tabular expression makes the specification clear - and complete

Condition	Result: c_sp
(m_pbON = e_notPressed) & (m_pbOFF = e_notPressed)	No Change
(m_pbON = e_notPressed) & (m_pbOFF = e_pressed)	e_cancelled
(m_pbON = e_pressed) & (m_pbOFF = e_notPressed)	e_requested
(m_pbON = e_pressed) & (m_pbOFF = e_pressed)	e_requested

57

Specifications - 6

Again, a tabular expression makes the specification clear - and complete - and unambiguous

Condition	Result: c_sp
(m_pbON = e_notPressed) & (m_pbOFF = e_notPressed)	No Change
(m_pbON = e_notPressed) & (m_pbOFF = e_pressed)	e_cancelled
(m_pbON = e_pressed) & (m_pbOFF = e_notPressed)	e_requested
(m_pbON = e_pressed) & (m_pbOFF = e_pressed)	e_requested

58

Specifications - 6

Again, a tabular expression makes the specification clear - and complete - and unambiguous.
Let's see why …

Condition	Result: c_sp
(m_pbON = e_notPressed) & (m_pbOFF = e_notPressed)	No Change
(m_pbON = e_notPressed) & (m_pbOFF = e_pressed)	e_cancelled
(m_pbON = e_pressed) & (m_pbOFF = e_notPressed)	e_requested
(m_pbON = e_pressed) & (m_pbOFF = e_pressed)	e_requested

Specifications - 7

- General, very simple tabular expression:

Condition	Result: function name
condition 1	result 1
condition 2	result 2
…	…
condition n	result n

- Is equivalent to:
  - if (condition 1) then function name = result 1
  - else if (condition 2) then function name = result 2
  - else if …
  - else if (condition n) then function name = result 2
- Disjointness: (so it is unambiguous)
  condition i $\wedge$ condition j $\Leftrightarrow$ FALSE $\forall$i,j = 1,..,n, i $\neq$ j
- Completeness:
  condition 1 $\vee$ condition 2 $\vee$ … $\vee$ condition n $\Leftrightarrow$ TRUE

Requirements