The notation `do P while b od` has been used as a loop construct that is executed as follows. First, `P` is executed; then `b` is evaluated, and if its value is `T` then execution is repeated, and if its value is `⊥` then execution is finished. Let the program variable be integer variable `x`. Prove that the specification

\[ \text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \]

is refined by the loop

\[ \text{do } x := x - 2 \text{ while } x \geq 2 \text{ od} \]

§ To prove `S` is refined by `do P while b od`, prove instead

\[ S \iff P. \text{ if } b \text{ then } S \text{ else } \text{ok fi} \]

So we prove

\[ (\text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \iff x := x - 2. \text{ if } x \geq 2 \text{ then } \text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \text{ else ok fi}) \]

replace `ok` by cases

\[ = (\text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \iff x := x - 2. \text{ if } x \geq 2 \text{ then } \text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \text{ else } x' = x - 2 \text{ fi}) \]

substitution

\[ = \text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \iff \text{if } x - 2 \geq 2 \text{ then } \text{mod } x' \cdot 2 = \text{mod } (x - 2) \cdot 2 \text{ else } x' = x - 2 \text{ fi} \]

specialization and

\[ = (\text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \iff x - 2 \geq 2 \land \text{mod } x' \cdot 2 = \text{mod } (x - 2) \cdot 2) \]

specialization again

\[ \land (\text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \iff x - 2 < 2 \land x' = x - 2) \]

context and

\[ = (\text{mod } x' \cdot 2 = \text{mod } x \cdot 2 \iff \text{mod } x' \cdot 2 = \text{mod } (x - 2) \cdot 2) \]

context again

\[ \land (\text{mod } (x - 2) \cdot 2 = \text{mod } x \cdot 2 \iff x' = x - 2) \]

\[ = \top \land \top \]

\[ = \top \]

1 From the construction axiom `0, 1, 2–few: few`

(a)[3] what elements are constructed?

§ `0, 1, 2`

(b)[3] give three solutions (considering `few` as the unknown).

§ solution `(0, 1, 2)` and solution `(0, 1/2, 1, 3/2, 2)` and solution `(0, 1/3, 2/3, 1, 4/3, 5/3, 2)` and solution `int` and solution `rat` and solution `real`

(c)[3] give the corresponding induction axiom.

§ `0, 1, 2–B: B \Rightarrow few: B`

(d)[3] state which solution is specified by construction and induction.

§ `0, 1, 2`
2 Let $x$ and $y$ be rational variables. Define program $zot$ by the fixed-point equation

$$zot = \begin{cases} y := 0 & \text{if } x = y \\ x := (x+y)/2. & \text{else} \end{cases}$$

where $+$ is rational addition and $/$ is rational division.

(a)[3] Add recursive time.

$zot = \begin{cases} y := 0 & \text{if } x = y \\ x := (x+y)/2. & t := t+1. \end{cases}$

(b)[9] Give two solutions to this equation (with recursive time added) (considering $zot$ as the unknown). (No proof needed.)

$x = y \Rightarrow x' = x \land y' = 0$

if $x = y$ then $x' = x \land y' = 0 \land t' = t$ else $x' = y' = 17 \land t' = \infty$

(c)[3] The definition of $zot$ makes it a solution (fixed-point) of an equation. What axiom is needed to make $zot$ the weakest solution (weakest fixed-point)?

$(\forall x, y, t, x', y', t'. Z = \begin{cases} y := 0 & \text{if } x = 0 \\ x := x/2. & \text{else} \end{cases} t := t+1. Z) \Rightarrow (\forall x, y, t, x', y', t'. Z \Rightarrow zot)$