(\* Andrius Andrijauskas (s121060@student.dtu.dk), Lars Bonnicshen (lfbo@dtu.dk) \*)

program

int array[10];

int a;

skip; (\* label 0, a in {0}, array in {0} \*)

a := -1; (\* label 1, a in {-}, array in {0}\*)

while (a <= 10) do (\* label 2, a in {-,0,+}, array in {0, +} \*)

array[a]:= a \* a; (\* label 3, a in {-,0,+}, array in {0, +} a can be -1 and 10 \*)

a := a+1; (\* label 4, a in {-,0,+}, array in {0, +} \*)

od

skip; (\* label 5, a in {+}, array in {0, +} \*)

end

Program graph:

(1,skip;,2), (2,a := -1;,3), (3,a<=10,4), (4,array[a] := a\*a;,5), (5,a := a+1;,3), (3,!a<=10,6), (6,skip;,7)

Detection of signs solutions table 17:

1: a={0} array={0}

2: a={0} array={0}

3: a={-,0,+} array={0,+}

4: a={-,0,+} array={0,+}

5: a={-,0,+} array={0,+}

6: a={+} array={0,+}

7: a={+} array={0,+}

Low boundary violations for array indexing:

(4,array[a] := a\*a;,5),

Good example since before we did not consider that ,array[a] := a\*a; both values ‘a’ are the same and if a={-,0,+} we would get array={-,0,+}. But now it is fixed for the cases where both sides expressions of multiplication or division operation are variables or arrays (e.g. a\*a) and ‘(a+1)\* a’ will already be considered as different disregard to the signs of ‘a’.