Piacente Cristian 866020

Homework H4 – Data flow analysis

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

Exercise 1) Consider the problem of identifying live (and dead) variables, that is, definitions of variables that may (and may not) be used on any path.

					init		PASS 1		PASS 2		PASS 3	
STATEMENT ID	STATEMENT	NEXT STATEMENTS	GEN	KILL	LIVE	LIVE-OUT	LIVE	LIVE-OUT	LIVE	LIVE-OUT	LIVE	LIVE-OUT
A	public int compute(int n) {	B	∅	n	∅	∅	n	∅	n	∅	n	∅
B	int x, y, z;	C	∅	x, y, z	∅	∅	n	n	n	n	n	n
C	if (n <= 0) {	D, E	n	∅	∅	∅	n	n	n	n	n	n
D	return 0; }		∅	∅	∅	∅	∅	∅	∅	∅	∅	∅
E	else if (n == 1) {	F, G	n	∅	∅	∅	n	n	n	n	n	n
F	x = 1; }	P	∅	x	∅	∅	x	∅	x	∅	x	∅
G	else {	H	∅	∅	∅	∅	n	n	n	n	n	n
H	x = 1;	I	∅	x	∅	∅	x, n	n	x, n	n	x, n	n
I	y = 2;	J	∅	y	∅	∅	x, y, n	x, n	x, y, n	x, n	x, y, n	x, n
J	while (x <= n) {	K, P	x, n	∅	∅	∅	x, y	x, y, n	x, y, n	x, y, n	x, y, n	x, y, n
K	z = x;	L	x	z	∅	∅	z, y	x, y	z, y, n	x, y, n	z, y, n	x, y, n
L	x = z * y;	M	z, y	x	∅	∅	x, z	z, y	x, z, n	z, y, n	x, z, n	z, y, n
M	y = z;	N	z	y	∅	∅	x, y, z	x, z	x, y, z, n	x, z, n	x, y, z, n	x, z, n
N	if (x == y * z) {	O, J	x, y, z	∅	∅	∅	x	x, y, z	x, y, n	x, y, z, n	x, y, n	x, y, z, n
O	return x; } }	∅	x	∅	∅	∅	∅	x	∅	x	∅	x
P	} return x; }	∅	x	∅	∅	∅	∅	x	∅	x	∅	x

- Identify an appropriate type of data flow analysis, indicating its characteristics (forward/backward, all/any path)
  - **Live variables**, since we need to consider the problem of identifying live variables.
    It is a **backward**, **any-path** analysis.
- Define kill and gen operations
  - **Kill set**: variables whose values are replaced.
  - **Gen set**: variables used at a node.
- Identify the kill and gen sets for the code
  - See the table above.
- Report the results of the analysis for each statement in the code
  - See the table above.

Piacente Cristian 866020

Homework H4 – Data flow analysis

Software Quality, Academic Year 2023-2024, University of Milan – Bicocca

Exercise 2) Consider the problem of identifying the availability of expressions, that is, the areas of the code where the value of an expression is valid.

Two tables were used for solving this exercise: the first one maps the expressions to the identifiers E1, E2, …, E11, while the second one is used for analyzing each code statement.

EXPRESSION ID	EXPRESSION
E1	number1
E2	number2
E3	number1 == 1 || number2 == 2
E4	number1 + number2
E5	number1 + tmp1
E6	tmp1 <= 100
E7	number2 + 1
E8	number2 * number1
E9	number1 + 2
E10	number2 * tmp3 * number1
E11	number1 + number2

- Identify an appropriate type of data flow analysis, indicating its characteristics (forward/backward, all/any path)
  - **Available expressions**, since we need to consider the problem of identifying the availability of expressions. It is a **forward**, **all-paths** analysis.
- Define kill and gen operations
  - **Kill set**: expressions that change and cease to be available.
  - **Gen set**: expressions that become available at each node.
- Identify the kill and gen sets for the code
  - See the table below.
- Report the results of the analysis for each statement in the code
  - See the table below.

					init		PASS 1		PASS 2	
ID	STATEMENT	PREV	GEN	KILL	AVAIL	AVAIL-OUT	AVAIL	AVAIL-OUT	AVAIL	AVAIL-OUT
A	public static int foo(int number1, int number2) {	∅	E1, E2	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	∅	E1, E2	∅	E1, E2
B	if (number1 == 1 || number2 == 2) {	A	E3	∅	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2	E1, E2, E3	E1, E2	E1, E2, E3
C	return 1; }	B	∅	∅	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3	E1, E2, E3	E1, E2, E3	E1, E2, E3
D	int tmp1 = number1 + number2;	B	E4	E5, E6	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3	E1, E2, E3, E4	E1, E2, E3	E1, E2, E3, E4
E	int tmp2 = number1 + tmp1;	D	E5	∅	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4	E1, E2, E3, E4, E5	E1, E2, E3, E4	E1, E2, E3, E4, E5
F	if (tmp1 <= 100) {	E	E6	∅	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5	E1, E2, E3, E4, E5, E6	E1, E2, E3, E4, E5	E1, E2, E3, E4, E5, E6
G	number2 = number2 + 1;	F	E7	E2, E3, E4, E7, E8, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6	E1, E5, E6, E7	E1, E2, E3, E4, E5, E6	E1, E5, E6, E7
H	int tmp3 = number2 * number1;	G	E8	E10	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E5, E6, E7	E1, E5, E6, E7, E8	E1, E5, E6, E7	E1, E5, E6, E7, E8
I	number1 = number1 + 2;	H	E9	E1, E3, E4, E5, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E5, E6, E7, E8	E6, E7, E9	E1, E5, E6, E7, E8	E6, E7, E9
J	int tmp = number2 * tmp3 * number1;	I	E10	∅	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E6, E7, E9	E6, E7, E9, E10	E6, E7, E9	E6, E7, E9, E10
K	return tmp; }	J	∅	∅	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E6, E7, E9, E10	E6, E7, E9, E10	E6, E7, E9, E10	E6, E7, E9, E10
L	return number1 + number2; }	F	E11	∅	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6, E7, E8, E9, E10, E11	E1, E2, E3, E4, E5, E6	E1, E2, E3, E4, E5, E6, E11	E1, E2, E3, E4, E5, E6	E1, E2, E3, E4, E5, E6, E11