Brandan Smith Oct. 15, 2019

ECSE 429: Software Validation

Q1:

			F			$F^{g/0}$			$F^{e/1}$			
A	B	C	X	Y	Z	X	Y	Z	X	Y	Z	
0	0	0	1	1	1	1	1	1	1	1	1	
0	0	1	1	1	1	1	1	1	1	1	1	
0	1	0	1	1	1	1	1	1	1	1	1	
0	1	1	1	1	1	1	1	1	1	1	1	
1	0	0	1	1	1	1	1	1	1	1	1	
1	0	1	1	1	1	1	1	1	1	1	1	
1	1	0	0	0	1	0	0	1	0	1	1	←
1	1	1	0	0	1	0	0	1	0	1	0	←

a) Under the assumption that stuck-at only propogates forwards,
g stuck at 0 is <u>NOT</u> detectable by any input vector.
e stuck at 1 is detectable using vectors (1, 1, 0) and (1, 1, 1)

b) $F_x = \neg(\neg(\neg((A \wedge B) \oplus B) \oplus (A \wedge B)) \wedge (A \wedge B))$

$F_x^{g/0} = F_x \ , \quad F_x^{e/1} = F_x$

$F_y = \neg\neg((A \wedge B) \oplus B) \vee \neg(\neg(\neg((A \wedge B) \oplus B) \oplus (A \wedge B)))$

$F_y^{g/0} = F_y \ , \quad F_y^{e/1} = True \vee (\dots) = True = 1$

$F_z = \neg(C \wedge \neg((A \wedge B) \oplus B) \wedge \neg(\neg\neg(\neg((A \wedge B) \oplus B) \oplus (A \wedge B)) \oplus B))$

$F_z^{g/0} = \neg(C \wedge False \wedge \neg(\neg\neg(\neg((A \wedge B) \oplus B) \oplus (A \wedge B)) \oplus B))$

$F_z^{e/1} = \neg(C \wedge \neg((A \wedge B) \oplus B) \wedge \neg(True \oplus B))$

$\Rightarrow F_x^{g/0} = F_x \oplus F_x^{g/0} = F_x \oplus F_x = False \Rightarrow X$ cannot detect g/0

$\Rightarrow F_x^{e/1} = F_x \oplus F_x^{e/1} = F_x \oplus F_x = False \Rightarrow X$ cannot detect e/1

$\Rightarrow F_y^{g/0} = F_y \oplus F_y^{g/0} = F_y \oplus F_y = False \Rightarrow Y$ cannot detect g/0

$\Rightarrow F_y^{e/1} = F_y \oplus F_y^{e/1} = \neg\neg((A \wedge B) \oplus B) \vee \neg(\neg(\neg((A \wedge B) \oplus B) \oplus (A \wedge B))) \oplus True$

$= \neg\neg\neg((A \wedge B) \oplus B) \vee \neg(\neg(\neg((A \wedge B) \oplus B) \oplus (A \wedge B)))$

$= \neg(A \wedge B) \oplus B \wedge \neg(\neg((A \wedge B) \oplus B) \oplus (A \wedge B))$

$= \neg((A \wedge B) \wedge \neg B) \vee ((\neg A \vee \neg B) \wedge B)) \wedge \neg(\neg(((A \wedge B) \wedge \neg B) \vee ((\neg A \vee \neg B) \wedge B) \oplus (A \wedge B$

Changing Notation cause I'm going insane.

$= \neg((\neg A \vee \neg B) \wedge B) \wedge \neg(\neg(((\neg A \vee \neg B) \wedge B) \oplus (A \wedge B)))$

$= \ldots$

$= AB \Rightarrow$ Y detects e/1 if A=1 and B=1 so $\{\phi, 1, 0\}$ and $\{1, 1, 1\}$.

$F_z^{g/0} = F_z \oplus F_z^{g/0} = \overline{(C(\overline{(AB)} \oplus B)(((\overline{(AB)} \oplus B) \oplus (AB)) \oplus B)} \oplus \overline{(C(False)(((\overline{(AB)} \oplus B) \oplus (AB)) \oplus B))}$

$= (C(\overline{(AB)}\overline{B} + \overline{\overline{(AB)}}B)(((\overline{(AB)}\overline{B} + \overline{\overline{(AB)}}B) \oplus (AB)) \oplus B) \oplus \ldots$

$= (C(\overline{\overline{(AB)}B})(((\overline{\overline{(AB)}}B) \oplus (AB)) \oplus B)) \oplus \ldots$

$= (C(AB + \overline{B})(((\overline{\overline{(AB)}B})\overline{(AB)} + ((\overline{AB})B)(AB)) \oplus B)) \oplus \ldots$

$= (C(\overline{(\overline{AB})B})(((\overline{(AB)}B)(\overline{AB})) \oplus B)) \oplus \ldots$

$= (C(\overline{(\overline{AB})B})(((\overline{(\overline{AB})B})\overline{(AB)}\overline{B} + \overline{(\overline{AB})B}(\overline{AB})B)) \oplus \ldots$

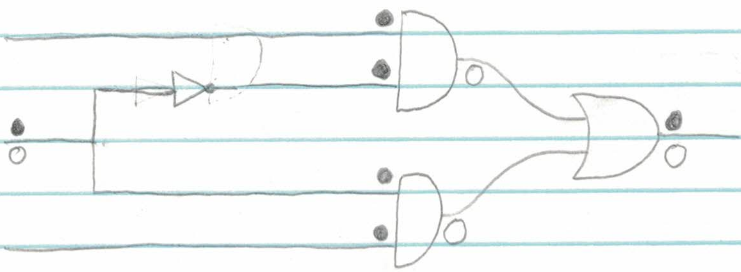
$\vdots$

$=$ False $\Rightarrow$ Z does not detect g/0.

$F_z^{e/1} = F_z \oplus F_z^{e/1} = \ldots$ Honestly I don't have the time to write it out but it should reduce to.

$= ABC \Rightarrow$ Z detects e/1 if A=1, B=1 and C=1 so $\{1,1,1\}$.

Q2:



As on diagram, Forks propagate no faults, nots don't really chang much cause s-a-1 is just = s-a-0 before the not, Ands propagate s-a-0 and or propagates s-a-1.

Q3: a) Unfortunately none exists because for any numbers the program is guaranteed to reach the for loop, and even a null will still reach the for loop before it throws the NullPointerException.

And most languages will execute from left to right for (①; ②; ③) so even for a null numbers we will still get i = 0+1 before the exception is thrown.

b) For the exact reasons as mentioned in a) the mutant infects before our first opportunity to "reach".

c) numbers = {0, 1}, val = 1

⇒ The program will enter the for-loop setting i = 0+1 on first iteration. it will then find numbers[1] == val to be true and then findval = 1. Thus returning the correct index. The state was infected, but no error was propogated.

d) numbers = {1}, val = 1 or numbers = {1, 0}, val = 1, etc...

⇒ The program will enter the for loop with i = 0+1, and then the for loop condition is immeditely false for numbers = {1} so it returns -1 For numbers = {1, 0} or {1, 0, 0} or {1, 0, 3, 4} etc, it will enter the loop but will never check numbers[0] so it will return -1, thus killing the mutant.