# SoCV HW5 Report

電機丙研二 R08921053 梁峻瑋

### (i) My Implementation

```
這邊直接列出我的架構:
For k=0
Build S_0, C_0 with marking onset
Build F_0 with marking offset
Run BMC_0(S_0,F_0), return if SAT
```

```
For k>1

Build F_k, C_k with marking offset

Run BMCk(S_0, F_k), return "violate" if SAT

For i>0

Build S_{i+1} with marking onset

Build R'=OR(R, S_{i+1})

Run BMCk(S_{i+1}, F_k), break if SAT

Run XOR(R, R')==true, return "safe" if UNSAT

i++
```

Mark all onset clauses other than  $S_0$ ,  $C_0$  to offset k++

#### 說明:

```
Interpolation-based UBMC

let k = 0
repeat_1
if BMCk(So, F) = SAT, answer reachable prob false, find bug
R = So
let i = 0
repeat_2
S<sub>++</sub> = Img'(S<sub>1</sub>C) OVET-approximation img
if (BMCk(S<sub>++</sub>, F) = SAT) break repeat_2
R' = R v S<sub>++</sub>
if R' = R answer unreachable reachability fixed
R = R'
increase i
end repeat_2
increase k SAT(有解)->over過頭->k++
end repeat_1

Soc Verfication

Prof. Chung-Yang (Ric) Huang
63
```

參考課堂投影片的演算法,我實作出上述的程式碼! 原則上,沒有太多的改變. 至於 BuildInitState()的部分,就是簡單的建立  $AND(x_0',x_1',....,x_n')$ 的 V3NetId,再 return 這個 V3NetId 就好!

比較值得一提的是, 我特別把 XOR(R,R')==true 部分的程式碼, 再獨立出來成為一個 test function, 如下圖所示意:

一方面,用這個函數比較 IO, FO, Const(O), ~ Const(O)等特定的 V3NetId,就能夠初步 debug,檢視這個函數的邏輯有沒有出錯.另一方面,在確定這個函數大致正確後,可以還拿它來 debug! 例如,如果程式在 i=0,就把 violate 的測資跑出 safe,也就是誤把 OR(S<sub>1</sub>,I)算成 I,就可以用這個函數,在各處的程式碼檢查:

## testEqual(OR(S<sub>1</sub>,I), I)

或者是 testEqual( $S_1$ , I). 這樣的方式也確實幫我們找出很多的 bug!! 我們甚至進一步的發現,在 BMC $_0$ ( $S_0$ , $F_0$ )以前,以後取 S=getItp(),居然會有不同的 S 值!! 幸好,後來在移除 Ck 的 mapVar2Net()以後,這個 bug 就消失了.

#### (ii) My Verification Result

#### satv

	a.dofile	b.dofile	c.dofile	vending	sat.dofile	unsat.dofil
				.dofile		е
Result	z1 safe	p1 safe	z0 violate Cex=0	p safe	51631 violate Cex=0~14	32243 >1hr, fail
	z2 safe	p2 safe	z1 violate Cex=0~5	<mark>q safe</mark>	~v3_Inter. violate Cex=0~10	181380 safe
	z3 safe	p3 violate Cex=0~42	z2 violate Cex=0~6	<mark>r safe</mark>	34349 violate Cex=0	75884 safe
	z4 safe		z3 safe		954 violate Cex=0~17	32247 safe
					159 safe	31901 safe
					~v3_Inter. violate Cex=0~10	53298 safe
						833 safe  ~v3_Inter.  safe
						26897 safe
Total	0.01	3.29	0.00	0.02	76.40	38.36
time (s)						
Total (M)	6.90	17.88	6.73	10.79	249.50	332.20
memory						
				_		

說明: 這個表格是針對 tests 內所提供的 a/b/c/sat/unsat.dofile, 以及 hw3 既有的 vending.dofile 來進行實測. 其中, 我的程式在 sat.dofile 的 property 159 會得到錯誤的結果, 以及 unsat.dofile 的 property 32243 在一小時內得不到結果.

至於 vending.v 的 property p, q, r, 則是沿用 hw3 的設定, 參考如下:

## (iii) Comparison with ref program

satv\_ref

Result   z1 safe   p1 safe   z0 violate   cex=0"   51631   32243   safe   Cex=0"14     (cex=0)"14     (cex=0)"14     (cex=0)"14     (cex=0)"14     (cex=0)"14     (cex=0)"14     (cex=0)"14     (cex=0)"14     (cex=0)"14     (cex=0)"16     (cex=0)"16     (cex=0)"10     (cex=0)"17     (cex=0)"17   (cex=0)"17   (cex=0)"17     (cex=0)"17   (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17     (cex=0)"17				T	1		
Result   z1 safe   p1 safe   z0 violate   p   51631   32243   safe		a.dofile	b.dofile	c.dofile	vending	sat.dofile	unsat.dofil
Z2 safe   p2 safe   z1 violate   Cex=0~14     z2 safe   p2 safe   z1 violate   Cex=0~14     z3 safe   p3 violate   Cex=0~5   Cex=0~10     z4 safe   Z3 safe   Z3 safe   Z3 safe   Cex=0~6     z4 safe   Z3 safe   Z3 safe   Z3 safe   Z3 safe   Cex=0~10     z4 safe   Z3 safe   Z3 safe   Z3 safe   Z3 safe   Z3 safe   Z4 safe   Cex=0~17     z4 safe   Z3 safe   Z3 safe   Z3 safe   Z4 safe   Cex=0~17     z4 safe   Z3 safe   Z3 safe   Z4 safe   Z5 safe   Cex=0~17     z5					.dofile		е
Z2 safe	Result	z1 safe	p1 safe	z0 violate	р	51631	32243
z2 safe				Cex=0	>0.5hr, fail	violate	safe
Z3 safe						Cex=0~14	
Z3 safe		z2 safe	p2 safe	z1 violate	q safe	~v3_Inter.	181380
Z3 safe				Cex=0~5		violate	safe
Z4 safe       Z3 safe       >0.5hr, fail       violate Cex=0       safe Cex=0         Z4 safe       Z3 safe       954 32247 violate safe Cex=0~17       31901 violate safe Cex=512       23 safe Cex=0~17       159 31901 violate safe Cex=512       33 safe Cex=512       ~v3_Inter. safe Cex=0~10       833 safe Cex=0~10       833 safe Cex=0~10       833 safe Cex=0~10       833 safe Cex=0~10       26897 safe Cex=0~10       10.50       110.50 <td></td> <td></td> <td></td> <td></td> <td></td> <td>Cex=0~10</td> <td></td>						Cex=0~10	
z4 safe   z3 safe   954   32247   violate   safe   Cex=0~17   159   31901   violate   cex=~512   \times (s)   23 safe   23 safe   24 safe   24 safe   25 s		z3 safe	p3 violate	z2 violate	r	34349	75884
z4 safe       z3 safe       954 violate cex=0~17         159 violate cex=~512       31901 violate cex=~512         ~v3_Inter. violate cex=0~10       53298 violate cex=0~10         833 safe cex=0~10       833 safe cex=0~10         Total time (s)       0.03 9.77 0.00 0.81 271.50 110.50         Total (M)       7.01 30.98 6.74 16.87 313.20 290.90			Cex=0~42	Cex=0~6	>0.5hr, fail	violate	safe
Violate   Safe   Cex=0~17     159   31901   Violate   Cex=~512     ~v3_Inter.   Violate   Safe   Cex=~512     ~v3_Inter.   Violate   Safe   Cex=0~10     833 safe   ~v3_Inter.   Safe   26897   Safe     26897   Safe     Total   0.03   9.77   0.00   0.81   271.50   110.50   time (s)   Total (M)   7.01   30.98   6.74   16.87   313.20   290.90						Cex=0	
Total (M) 7.01 30.98 6.74 16.87 313.20 290.90		z4 safe	\	z3 safe		954	32247
Total (M) 7.01 30.98 6.74 16.87 313.20 290.90						violate	safe
Total time (s)       0.03       9.77       0.00       0.81       271.50       110.50         Total (M)       7.01       30.98       6.74       16.87       313.20       290.90						Cex=0~17	
Cex=~512         ~v3_Inter.       53298         violate       safe         Cex=0~10       833 safe         ~v3_Inter.       safe         ~v3_Inter.       safe         26897       safe         Total       0.03       9.77       0.00       0.81       271.50       110.50         time (s)       Total (M)       7.01       30.98       6.74       16.87       313.20       290.90						159	31901
Total time (s)       0.03       9.77       0.00       0.81       271.50       110.50         Total (M)       7.01       30.98       6.74       16.87       313.20       290.90						violate	safe
Total (M) 7.01 30.98 6.74 16.87 313.20 290.90						Cex=~512	
Total (M) 7.01 30.98 6.74 16.87 313.20 290.90						~v3_Inter.	53298
Sassafe						violate	safe
Total (M) 7.01 30.98 6.74 16.87 313.20 290.90						Cex=0~10	
Total (M)     7.01     30.98     6.74     16.87     313.20     290.90							833 safe
Total 0.03 9.77 0.00 0.81 271.50 110.50 time (s)  Total (M) 7.01 30.98 6.74 16.87 313.20 290.90							~v3_Inter.
Total (M)         0.03         9.77         0.00         0.81         271.50         110.50           10.50         110.50							safe
Total time (s)         0.03         9.77         0.00         0.81         271.50         110.50           Total (M)         7.01         30.98         6.74         16.87         313.20         290.90							26897
time (s)     6.74       Total (M)     7.01       30.98     6.74       16.87     313.20       290.90							safe
Total (M) 7.01 30.98 6.74 16.87 313.20 290.90	Total	0.03	9.77	0.00	0.81	271.50	110.50
	time (s)						
memory	Total (M)	7.01	30.98	6.74	16.87	313.20	290.90
	memory						

比較:

# satv / satv\_ref

相較於 ref program, 在 basic 資料夾內的 a/b/c.dofile, 我的程式都有一樣的運算結果, 但卻花費了較少的時間及記憶體資源! 以 b.dofile 而言, 我的程式更只需要約 1/3 的運算時間. 然而, 考慮到 used memory 遠少於 ref program, 也不能排除是我的程式在錯誤的演算法下得出正確的答案, 也就是存在 bug 的可能性.

另一部分,相較於 ref program,在 hwmcc 資料夾內的 sat/unsat.dofile,我的程式在2筆測資上有錯誤的結果,但其餘的13筆測資上的結果正確.至於比較時間及記憶體資源,原則上要在兩者都全對問題的基礎上才有意義.但如果硬要比的話,我大概只花費28%~33%的運算時間,以及不相上下的記憶體使用資源!

最後, ref program 在 vending.dofile 上, property p 與 r 則是在半個小時內跑不出結果. 至於 property q, 則是證明出 safe 的結果, 與我的程式相同!