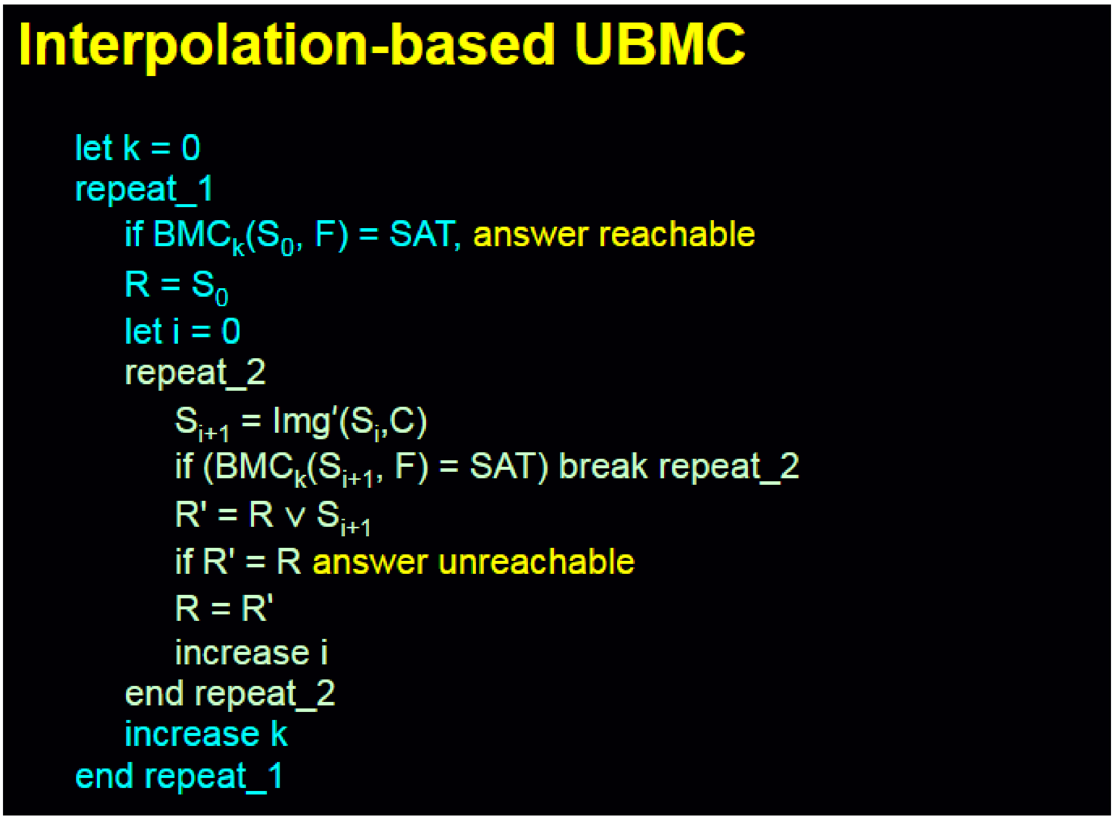
SoCV HW5

B10901085 李昀儒

1)Implementation

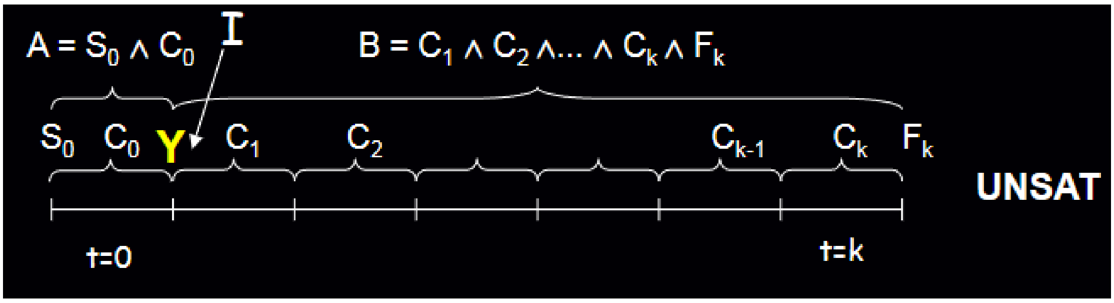
For the implementation of buildInitState, we simply return a gate whose value is AND(c0’,c1’…cn’) for all the state variable ci.

For the function itpUbmc, it is implemented basically the same as on the lecture slide of Topic 9.



Interpolation-based UBMC

```
let k = 0
repeat_1
    if $BMC_k(S_0, F)$ = SAT, answer reachable
    $R = S_0$
    let i = 0
    repeat_2
        $S_{i+1} = Img'(S_i,C)$
        if ($BMC_k(S_{i+1}, F)$ = SAT) break repeat_2
        $R' = R \vee S_{i+1}$
        if R' = R answer unreachable
        R = R'
        increase i
    end repeat_2
    increase k
end repeat_1
```



In the implementation, the clauses of A(S0 ∧ C0) and the interpolation S’ is marked as onset and B(C1∧C2∧...∧Fk) is marked as offset.

To test if the reachable states is fixed, we create a cirGate g with value is XOR(S, OR(S,S’)) and check if (g == 0) is UNSAT or not.

2)Result

	gv	gv-ref
a.v	0.09s 43.12M	0.11s 42.8M
b.v	5.87s 50.91M	4.19s 49.8M
c.v	0.02s 41.87M	0.03s 41.7M

	gv	gv-ref
sat.v	6.15s 107.2M	6.75s 103.5M
unsat.v	3.62s 53.86M	4.28s 56.88M

Testcases of vending machine

	gv	gv-ref
vending-abs.v	0.16s 44.9M	0.2s 44.82M
Vending-fixed.v	Failed to proof	Failed to proof

Testcases in HWMCC:
UNSAT

	gv	gv-ref
6s6.aig monitor"5369"is safe.	51.33s 79.93M	50.35s 80.7M
6s136.aig monitor "25378" is safe.	248.1s 206.9M	211.3s 198.6M
6s206rb025.aig monitor "141223" is safe.	4.62s 167.8M	3.68s 151.2M
6s221rb18.aig monitor "201417" is safe.	6.14s 211.2M	8.37s 197.8M
6s327rb10.aig monitor "25050" is safe.	1.27s 58.14M	1.17s 57.99M

6s380b129.aig monitor "43668" is safe.	2.92s 79.38M	3.67s 76.6M
6s388b07.aig monitor "34359" is safe.	0.06s 51.67M	0.07s 50.25M
pdtpmsfpmult.aig monitor "1348" is safe.	3.98s 53.86M	4.6s 57.07M
pj2018.aig monitor "26898" is safe.	102.6s 147.6M	8.5s 81.3M

SAT

bob9234spec7neg.aig monitor "813" is violated.	31.54s 51M	31.9s 51.24M
6s307rb06.aig monitor "37108" is violated.	6.32s 107.2M	6.19s 103.7M
6s326rb02.aig monitor "25376" is violated.	110.4s 171.6M	1124s 353.1M
abp4pold.aig monitor "955" is violated.	52.18s 129.5M	38.14s 125.9M
6s347b029.aig monitor "326273" is violated.	88.93s 312.1M	114.9s 315.3M
bobpci215.aig monitor "4469" is violated.	19.1s 68.03M	26.86s 71.28M

The performance of gv and gv-ref varies under different test cases. And in both program there exists some test case that makes one of the program’s performance very poor compared to another one.

Compared the performance in HW5 and in HW3(the three monitors are the same)

	UBMC	BDD-based(without restrict)	BDD-based(with restrict)
a.v	0.09s 43.12M	0.75s 72.07M	0.63s 72.08M
b.v	5.87s 50.91M	0.3s 43.47M	0.5s 43.47M
c.v	0.02s 41.87M	0s 41.88M	0s 41.88M
Vending-abs.v	0.16s 44.9M	19.15s 179.9M	10.83s 180M
Vending-fixed.v	Failed to proof	Falied to proof	Failed to proof

For the smaller test cases like b.v and c.v, the UBMC is slower than BDD-based method, but for the test cases like a.v or large design like vending.v, the UBMC method could outperform the BDD-based method in both time and space.