

## Masking Boundary Value Coverage: Effectiveness and Efficiency

P. Vijay Suman    Tukaram Muske    Prasad Bokil    Ulka Shrotri    R. Venkatesh

September 3rd, 2010

## Context/Ad (Work at TRDDC, Pune)

- ▶ Our Group: Verification and Validation of Embedded Systems
- ▶ Statechart Analysis, Program Analysis, Testing etc.
- ▶ Testing: Develop testing tools towards meeting international standards and customer requirements
- ▶ Boundary value coverage arised from one of our automotive domain customer requirement

# Overview

- ▶ Present two white-box coverage criteria for relational testing...
  1. Boundary Value Coverage (BVC)
  2. Masking Boundary Value Coverage (MBVC)

Both target relational bugs

- ▶ Argue that MBVC test data is (better) relatively more effective than BVC test data
- ▶ Experimental data supporting our claim/argument
- ▶ Conclusions

## Relational Testing

Masking of Conditions

MBVC Vs. BVC

Conclusion

# Boundary Values

## Given Code

```
int inp, op;  
void func()  
{  
    if(inp > 10)  
        op = 1;  
    else  
        op = 0;  
}
```

## Intended Code

```
int inp, op;  
void func()  
{  
    if(inp > 9)  
        op = 1;  
    else  
        op = 0;  
}
```

## Boundary Value Test Data

| inp | Actual O/P | Expected O/P |
|-----|------------|--------------|
| 9   | 0          | 0            |
| 10  | 0          | 1            |
| 11  | 1          | 1            |

# Errors Intended to be Detected

We consider detecting (exactly one at a time\*)...

1. Off-by-one errors
2. Incorrect operator errors

Let  $(a > 10)$  be the given relational condition

| Intended Code | Necessary Boundary Value |
|---------------|--------------------------|
| $a > 9$       | 10                       |
| $a > 11$      | 11                       |
| $a \geq 10$   | 10                       |
| $a < 10$      | 9, 11                    |
| $a \leq 10$   | 9, 10, 11                |
| $a = 10$      | 10                       |
| $a \neq 10$   | 9                        |

\*Combination of these errors: More test data required

Relational Testing

Masking of Conditions

MBVC Vs. BVC

Conclusion

# Masking - Independent Effect

## Function

```
int inp1, inp2, op;  
void func()  
{  
    if(inp1 && inp2)  
        op = 1;  
    else  
        op = 0;  
}
```

## Variant

```
int inp1, inp2, op;  
void func()  
{  
    if(inp1 || inp2)  
        op = 1;  
    else  
        op = 0;  
}
```

Condition Coverage Vs. Masking MC/DC:

| inp1 | inp2 | Function O/P | Variant O/P |
|------|------|--------------|-------------|
| 0    | 0    | 0            | 0           |
| 1    | 1    | 1            | 1           |

| inp1 | inp2 | Function O/P | Variant O/P |
|------|------|--------------|-------------|
| 1    | 1    | 1            | 1           |
| 1    | 0    | 0            | 1           |
| 0    | 1    | 0            | 1           |



# Masking BVC

## Function

```
int inp1, inp2, op;  
void func()  
{  
    if((inp1 > 10) && inp2)  
        op = 1;  
    else  
        op = 0;  
}
```

## Variant

```
int inp1, inp2, op;  
void func()  
{  
    if((inp1 > 9) && inp2)  
        op = 1;  
    else  
        op = 0;  
}
```

| inp1 | inp2 | Function O/P | Variant O/P |
|------|------|--------------|-------------|
| 9    | 0    | 0            | 0           |
| 10   | 0    | 0            | 0           |
| 11   | 0    | 0            | 0           |

| inp1 | inp2 | Function O/P | Variant O/P |
|------|------|--------------|-------------|
| 9    | 1    | 0            | 0           |
| 10   | 1    | 0            | 1           |
| 11   | 1    | 1            | 1           |

Relational Testing

Masking of Conditions

MBVC Vs. BVC

Conclusion

# Mutation

- ▶ We compare MBVC and BVC using mutation
- ▶ Mutation is purposely modifying code to simulate an error
- ▶ Example: Replace  $>$  by one of  $>=$ ,  $<$ ,  $<=$ ,  $==$  and  $!=$
- ▶ Mutation Adequacy of a test suite: Fraction of the given set of mutants it detects
- ▶ “Criterion  $C_1$  better than  $C_2$ ”  $\triangleq$  Mutation adequacy of test suite for  $C_1$  is greater than that of test suite for  $C_2$
- ▶ We implemented a mutation analysis tool, which introduces
  - ▶ Off-by-one errors
  - ▶ Incorrect operator errors

# Experimental Setup

- ▶ Application: An automobile battery controller code
- ▶ Randomly 12 modules out of 22 were selected
- ▶ For each function  $f$  do...
  1. Generate MBVC and BVC test data for  $f$  (our white-box tool AutoGen)
  2.  $f$  -(mutate at unit level)->  $\mathcal{M}$
  3. Run both on  $\mathcal{M}$  and compute adequacies

## MBVC Vs. BVC: Effectiveness

|   |             |                 |
|---|-------------|-----------------|
| <b>Funcs with Combinations of Rel Ops</b> | <b>74</b>   | <b>3646 LOC</b> |
| MBVC is better                            | 42          | <b>56%</b>      |
| BVC is better                             | 0           | <b>0%</b>       |
| Both equally good                         | 32          | <b>44%</b>      |
| <b>Total Mutants Generated</b>            | <b>4627</b> |                 |
| Mutants killed by MBVC                    | 3734        | 80.7%           |
| Mutants killed by BVC                     | 3253        | 70.3%           |
| <b>Detectable Mutants (Extrapolated)</b>  | <b>4026</b> | (Slide 15)      |
| Mutants killed by MBVC test data          | 3734        | <b>92.75%</b>   |
| Mutants killed by BVC test data           | 3253        | 80.8%           |

## MBVC Vs. BVC: Efficiency

| Parameter of Generation | Hike for MBVC |
|-------------------------|---------------|
| Time*                   | 9.38%         |
| Size**                  | 28.65%        |

\*The constraint that is generated by the model checker will be stronger when masking is used.

\*\*The probability of a test vector covering multiple states goes down in presence of masking.

# Propagation Problem

Limitation of Masking: Following kind of mutants might not be detected

```
int inp1, inp2, op;
void func()
{
    int local;
    if( inp1 > 10 && inp2) //if( inp1 > 11 && inp2)
        local = 1;
    else
        local = 0;
    if( local || inp3 )
        op = 1;
    else
        op = 0;
}
```

Two MBVC/BVC Test Data Entries for Decision 1

| inp1 | inp2 | inp3 | Actual O/P | Expected O/P |
|------|------|------|------------|--------------|
| 11   | 1    | 1    | 1          | 1            |
| 11   | 1    | 0    | 1          | 0            |

Relational Testing

Masking of Conditions

MBVC Vs. BVC

Conclusion



# Contributions

1. Analysis of boundary value analysis applied in the white-box setting (BVC)
2. Analysed the effect of extending BVC with masking of conditions
3. Gave formal argument and experimental evidence, both mutation based, that MBVC test data is better than BVC
4. Usage of a restricted mutation operator to compare criteria aiming specific kind of bugs

# Limitations

1. We consider only those functions for which the model checker terminated. A more fair analysis would require significant efforts towards loop abstraction
2. Sizes of the test sets being compared are not same. By padding more test data to the BVC test set we could have made it as big as that for MBVC

Thank you

Questions

...