BSSE System and Software Engineering

Potentials of Constraint-based Methods in Software Verification and Validation

R.Gerlich, R.Gerlich (BSSE)

Data Systems in Aerospace DASIA 2012

May 14th – 16th 2012

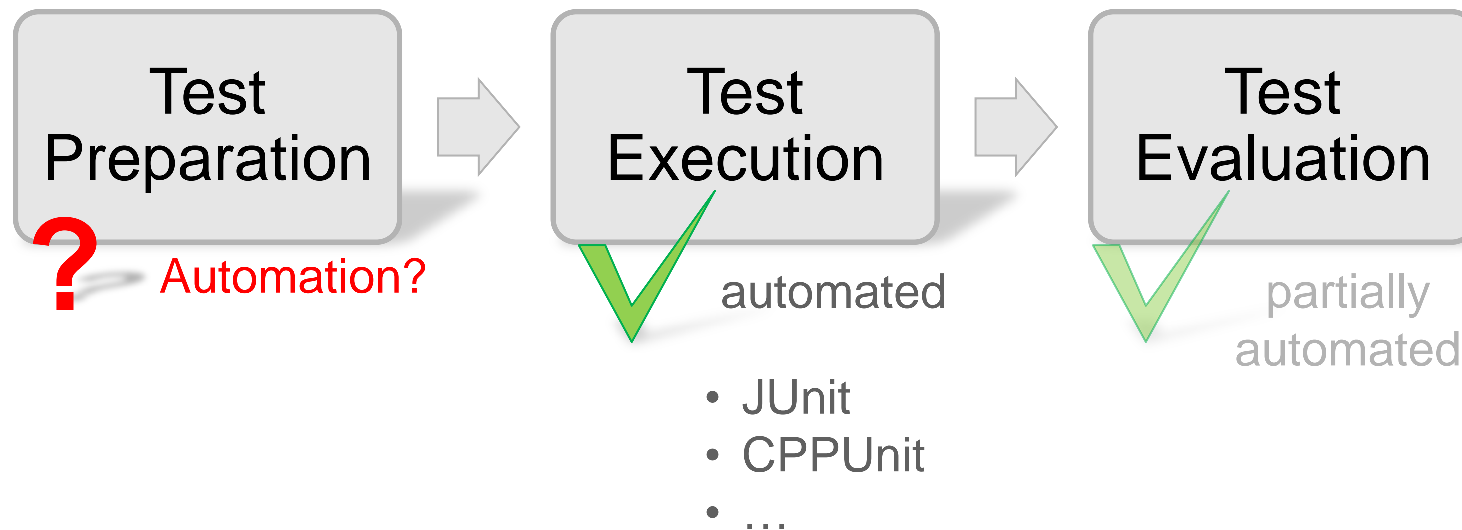
Dubrovnik, Croatia

Dr. Rainer Gerlich BSSE System and Software Engineering
Auf dem Ruhbühl 181
88090 Immenstaad
Germany

Tel. +49/7545/91.12.58
Fax +49/7545/91.12.40
Mobil +49/171/80.20.659
email Ralf.Gerlich@bsse.biz
Rainer.Gerlich@bsse.biz

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Motivation



- JUnit
- CPPUnit
- …

→Automatic Test Data Generation

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Other applications

Static Analysis and Verification
- Model-Checking
- Abstract Interpretation

Autonomy
- Continuous replanning
- Reconfiguration

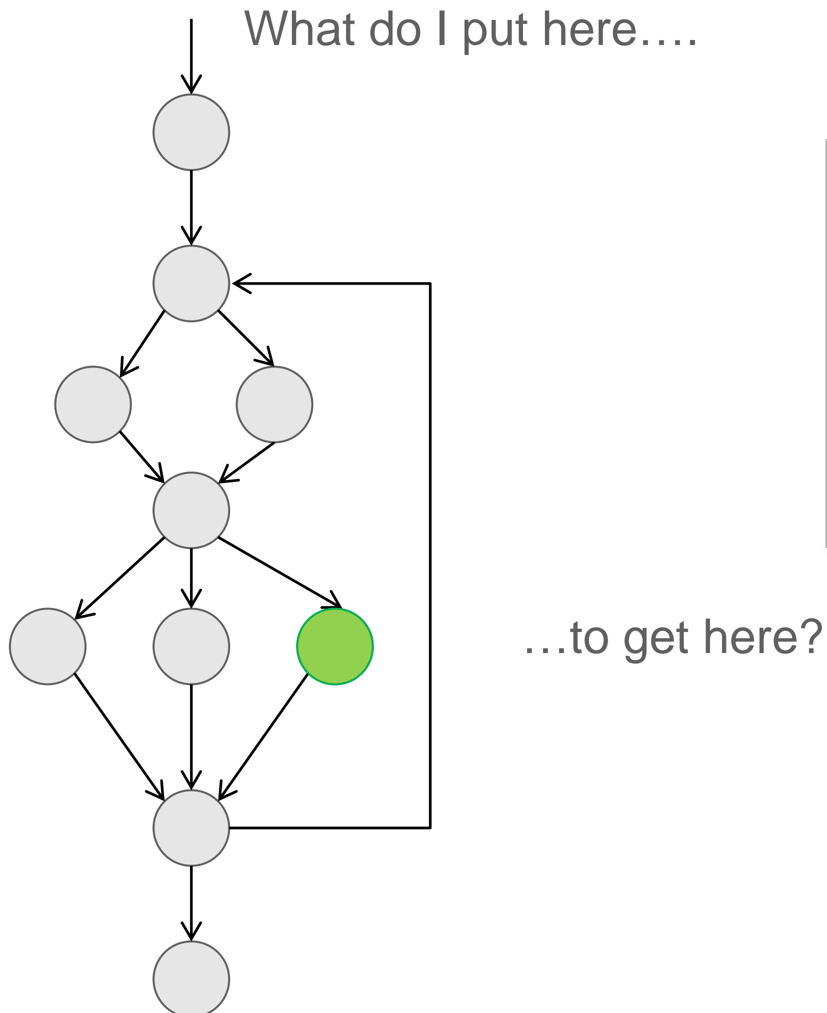
Model-based approaches
- Automatic design decisions
- Optimisation
- Model-based Testing

Automatic/Automated Parallelisation (e.g. for multiprocessors or multicores)

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Coverage Testing

What input is required to reach a specific point in the code?



Constraint-based Test Data Generation (CBTDG):

1. Construct a path to the target
2. Execute path symbolically
3. Solve the constraint system

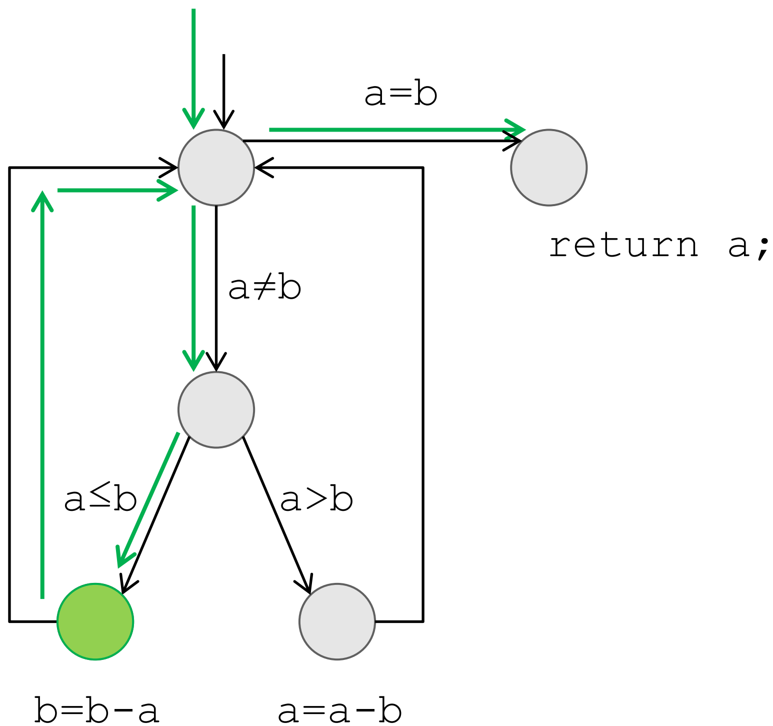
© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Symbolic Execution of Paths (Basic Idea)

BSSE System and Software Engineering

Constructed Path

```
unsigned int gcd(
        unsigned int a,
        unsigned int b)
```



Symbolic Execution

$a_1 \neq b_1$

Index to indicate „version“ of the variable

$a_1 \leq b_1$

$b_2 = b_1 - a_1$

New Index to indicate assignment

$a_1 = b_2$

Always use the latest index

Solve for a_1 and b_1: $b_1 = 2a_1,\ a_1 \leq b_1$

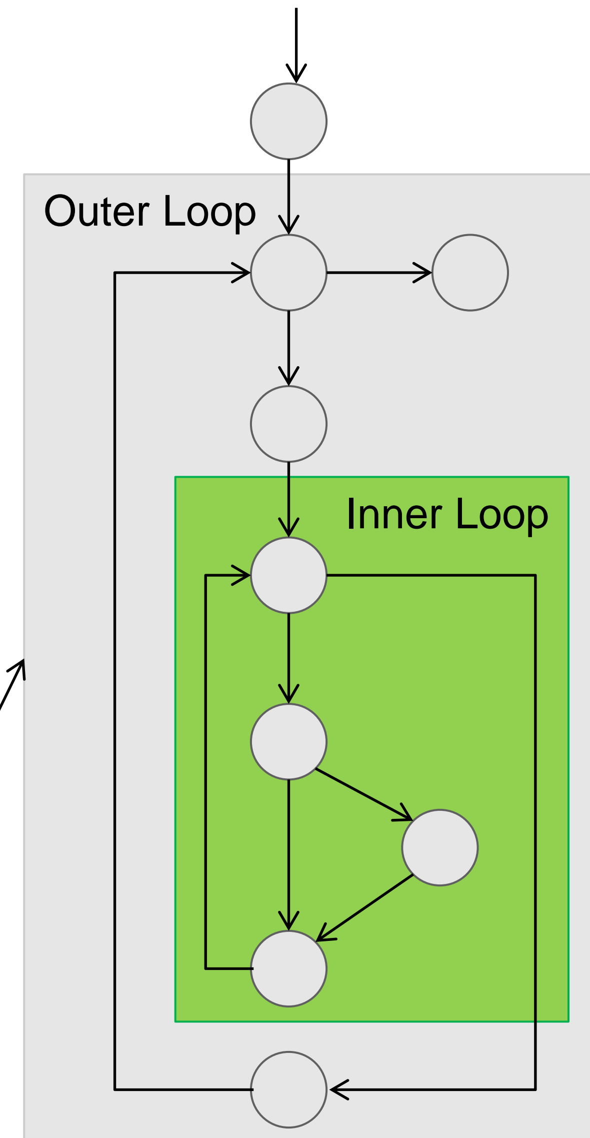
© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Infeasible Paths

BSSE System and Software Engineering

```
void sort(int a[], unsigned int n) {
    unsigned int i,j,min;
    int tmp;

    for (i=0;i<n;i++) {                              n times
        min=i;
        for (j=i+1;j<n;j++) {              n(n-1)/2 times
            if (a[j]<a[min]) {
                min=j;
            }
        }
        tmp = a[i];
        a[i] = a[min];
        a[min] = tmp;
    }
}
```

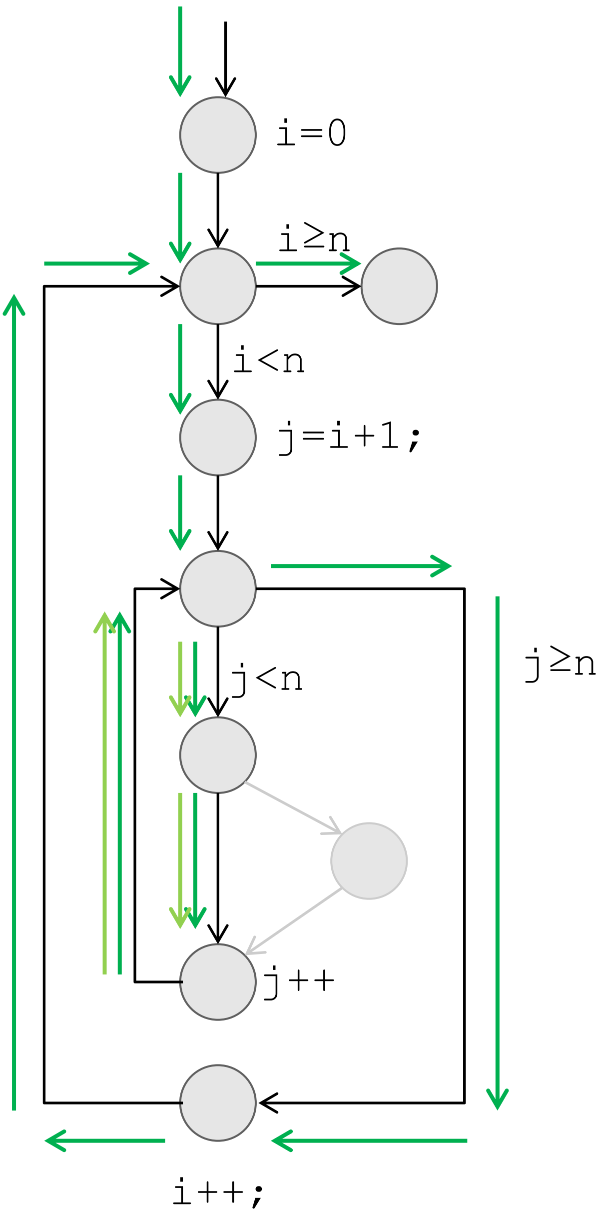


Inner and outer loop seem to be independent of each other!

Leads to construction of paths that are not possible in actual program!
→ infeasible paths

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Detecting Infeasible Paths



Contradiction!

$i_1=0$	
$i_1<n$	0<n
$j_1=i_1+1$	
$j_1<n$	1<n
$j_2=j_1+1$	
$j_2<n$	2<n
$j_3=j_2+1$	
$j_3\geq n$	3≥n
$i_2=i_1+1$	
$i_2\geq n$	1≥n

```
void sort(int a[], unsigned int n) {
    unsigned int i,j,min;
    int tmp;

    for (i=0;i<n;i++) {
        min=i;
        for (j=i+1;j<n;j++) {
            if (a[j]<a[min]) {
                min=j;
            }
        }
        tmp = a[i];
        a[i] = a[min];
        a[min] = tmp;
    }
}
```

Cannot be solved by brute force trial-and-error!

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

BSSE System and Software Engineering

Solution of systems of linear inequations by Joseph Fourier (1768-1830) and Theodore Motzkin (1908-1970).

1. Transform into upper and lower bounds for x:

$$\begin{array}{c} 1 < 4x + 3y < 5 \\ -1 < 2x - y < 2 \end{array} \Rightarrow \begin{array}{c} 1 - 3y < 4x < 5 - 3y \\ -1 + y < 2x < 2 + y \end{array} \Rightarrow \begin{array}{c} 1 - 3y < 4x < 5 - 3y \\ -2 + 2y < 4x < 4 + 2y \end{array}$$

2. Recombine upper and lower bounds:

$$\begin{array}{c} 1 - 3y < 4 + 2y \\ -2 + 2y < 5 - 3y \end{array} \Rightarrow \begin{array}{c} -3 < 5y \\ 5y < 7 \end{array} \Rightarrow -\frac{3}{5} < y < \frac{7}{5}$$

3. Find a solution by selecting a value for y:

$$y = 0 \Rightarrow \begin{array}{c} 1 < 4x < 5 \\ -1 < 2x < 2 \end{array} \Rightarrow \begin{array}{c} \frac{1}{4} < x < \frac{5}{4} \\ -\frac{1}{2} < x < 1 \end{array} \Rightarrow \frac{1}{4} < x < 1 \Rightarrow x = \frac{1}{2}$$

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

All issues solved?

So existing solvers for rational numbers should be sufficient for CBTDG, right?

So I thought ~2 years ago, when I started implementing a CBTDG tool. Unfortunately, …

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Limitations: Integers are Rationals, but…

BSSE System and Software Engineering

$$1 < 4x + 3y < 5$$
$$-1 < 2x - y < 2$$

Fourier-Motzkin-Elimination tells us:

$$-\frac{3}{5} < y < \frac{7}{5}$$

So, y=0 and y=1 may be integer solutions…

…but none of them is!

$$y = 0 \Rightarrow \frac{1}{4} < x < 1$$
$$y = 1 \Rightarrow 0 < x < \frac{1}{2}$$

There is an algorithm (Omega Test by Pugh, 1991).
It may be computationally costly (exponentially with the number of variables).

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Limitations: Floats are Rationals, but…

We assume that…

$$\forall x, y: y > 0 \Rightarrow x + y > x$$

Not for float!

```
float x;
for (x=x0;x<limit;x+=inc) {
    /* … */
}
```

Seems like this loop will terminate in any case, …

…but if `x(<limit)` is sufficiently large compared to `inc`, we get…

$$x + inc = x$$

…although `inc` is greater than 0!

Floating-Point operations are fundamentally different from rational arithmetic due to rounding!

Current solver approaches (Botella et al, 2006) use domain filtering, which suffers from bad convergence.

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Linearisation of Bitwise Operations

BSSE System and Software Engineering

Basic approach: bit-vector

$$x = \sum_{i=0}^{n-1} x_i 2^i \qquad (0 \leq x_i \leq 1)$$

$$x \sqcap y = \sum_{i=0}^{n-1} (x_i \sqcap y_i) 2^i$$

Many additional variables

Nonlinear

But most of the time, one operand is constant!

$$y = x \& 12$$

$$y = 4x_2 + 8x_3$$

Bit 2 of x, Bit 3 of x

$$x = c_1 + 4c_2 + 16c_3$$

$$y = 4c_2$$

$$0 \leq c_1 < 4$$

$$0 \leq c_2 < 4$$

$$c_1 = x_0 + 2x_1$$

$$c_2 = x_2 + 2x_3$$

$$c_3 = \sum_{i=0}^{n-5} x_i 2^i$$

Exact linear representation!

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Integer arithmetic traps

BSSE System and Software Engineering

```
unsigned int a,b,c;
/* … */

c=a-b;
```

Is this classical arithmetics over integers?

No, it's not!

$$c = a - b + 2^{32}s$$

$$0 \le c \le 2^{32} - 1$$

```
void foo(int l, int u) {
        int i;
        if ((u-l)<100) {
                for (i=l;i<u;i++) {
                        /* … */
                }
        }
}
```

Is the iteration count of this loop really limited to 100?

No, it's not!

$$l = -2^{31} + 1$$

$$u = 1$$

$$u - l = 2^{31} \equiv -2^{31} < 100$$

Sometimes, even i+1=0, although i>0!

```
for (j=0;j<n;j+=i+1) { … }
```

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Avoidance of Bias

BSSE System and Software Engineering

Software test cannot prove absence of bugs, only their presence.

So software test is really a scientific experiment, testing the hypothesis: „The program is correct.“

Selection of test cases influences the validity of the result greatly.

When there is more than one test case fulfilling the same test objective, which one to use? (→Bias)

Simple algorithms are easier to rid of bias (e.g. in selecting paths to execute) than complex algorithms…

…and constraint solution is a pretty complex matter!

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Recommendations: Coding for CBTDG

BSSE System and Software Engineering

Recommendations to simplify constraint-based program analysis:

- Separate hardware interaction from logic (functional H/W I/F)
- Avoid memory overlays, `memcpy`, `memmove` and `union`
- Use small, well-specified functions for specific purposes
- Solver support for fixed-point is much better than for floating point

Recommendations to simplify automatic test-data generation:

- First test randomly, then complement by constraint-based methods
- Restrict parameter types as far as possible

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Conclusions and Outlook

BSSE System and Software Engineering

Not any solver is able to properly solve the problems coming up in static analysis of software.

In theory everything works, in practice…

Some static analysis tasks are still difficult to solve and the results may be surprising – or not, if they are masked by bias.

Ongoing research is expected to bring important advances in all areas, e.g.

- handling floating point
- parallelisation of constraint calculations
- parallelisation in software
- …

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved

Thank you for your Attention!

Any Questions?

© Dr. Rainer Gerlich BSSE System and Software Engineering, 2012 All Rights Reserved