HANDS ON 3 : CBMC

CS60030 Formal Systems

PALLAB DASGUPTA,
FNAE, FASc,
A K Singh Distinguished Professor in AI,
Dept of Computer Science & Engineering
Indian Institute of Technology Kharagpur
Email: pallab@cse.iitkgp.ac.in
Web: http://cse.iitkgp.ac.in/~pallab

INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

FMSAFE

FORMAL METHODS FOR SAFETY CRITICAL SYSTEMS

Introduction

1. **CBMC is a Bounded Model Checker for C and C++ programs.**
2. **CBMC verifies memory safety (which includes array bounds checks and checks for the safe use of pointers), checks for exceptions, checks for various variants of undefined behavior, and user-specified assertions**
3. **Download CBMC from the following link.**

   http://www.cprover.org/cbmc/

   Click on the os compatible version and install

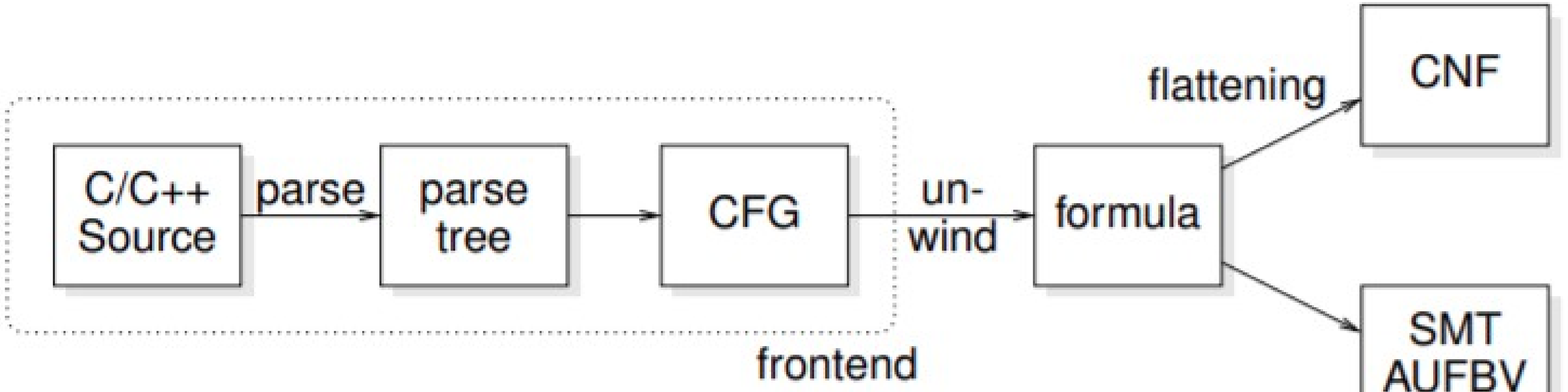
   Check by executing ./cbmc test_file.c

CBMC Tool flow : Summary

1. Parse, build CFG

2. Unwind CFG, form formula

3. Formula is solved by SAT/SMT



Checking Simple Programs

Example Problem

Set of Statements

```
x = 2*x;
z = x+1;
{z != 0 }
```

$\longrightarrow$

C Program

```
int main(){
    int x , y, z;
        x = 2*x;
        z = x+1;
        assert(z != 0)
}
```

$\downarrow$

Run using : ./cbmc file_name.c --trace

Write the C code for the following statement and check using CBMC. The second statement fails. Can you add a condition such that the assertion holds?

1.
```
x = y - 2;
z = x + y;
{z > 0 }
```

2.
```
y = 2 * x;
y = y + 2;
z = y/2;
assert (z > x)
```

Program Verification Problems

Verify the following programs discussed in tutorial 4.

```
L1 :      x = 1;
L2 :      if (y < = 10){
L3 :      y = 10;
     }
L3 :else{
L5 :      while (x < y){
L6 :           x = 2 * x;
L7 :           y = y - 1;
          }
     }
L8 :x = y + 1;
L9 :assert (x > 0);
```

```
L1 :      a = b = i = 0;
L2 :      while (a <= 10) {
L3 :           a = b + i;
L4 :           b = a + 1;
L5 :           i = i + 1;
L6 :      }
L7 :      if (b > 20) {
L8 :           error: exit(-1);
L9 :      }
```

```
./cbmc file_name.c --trace --unwind <trace_number>
```

(3n + 1) Conjecture

Apply the following operations on any positive integer *i*

1. if *i* is even, i = i/2
2. if *i* is odd, i = 3*i + 1
3. if (*i* ==1) break;
4. else goto 1

For any initial value of i, it will eventually converge to 1. This is known as the *Collatz conjecture* or *(3n+1) conjecture.*

The number of steps i takes to converge to one is called the total stopping time of i.

Prove that for i<20000, the total stopping time is always less than 280.