FuSeBMC_IA

FuSeBMC_IA : Interval Analysis and Methods for Test-case Generation

Author: Mohannad Aldughaim
Mohannad.aldughaim@manchester.ac.uk
Co-Authors: Kaled Alshmrany, Mikhail R. Gadelha, Rosiane de Freitas & Lucas C. Cordeiro
Presenter: Kaled Alshmrany

FuSeBMC IA Team

Mr. Mohannad Aldughaim

Dr. Kaled Alshmrany

Dr. Lucas Cordeiro

Dr. Mikhail R. Gadelha

Dr. Rosiane de Freitas

King Saud University

Institute of Public Administration

ESBMC team

Manchester University

Federal University of Amazonas

Motivation

Software testing is one of the most crucial phases in software development. Tests often expose critical bugs in software applications.

Abstract Interpretation

Bounded Model Checking

Fuzzing

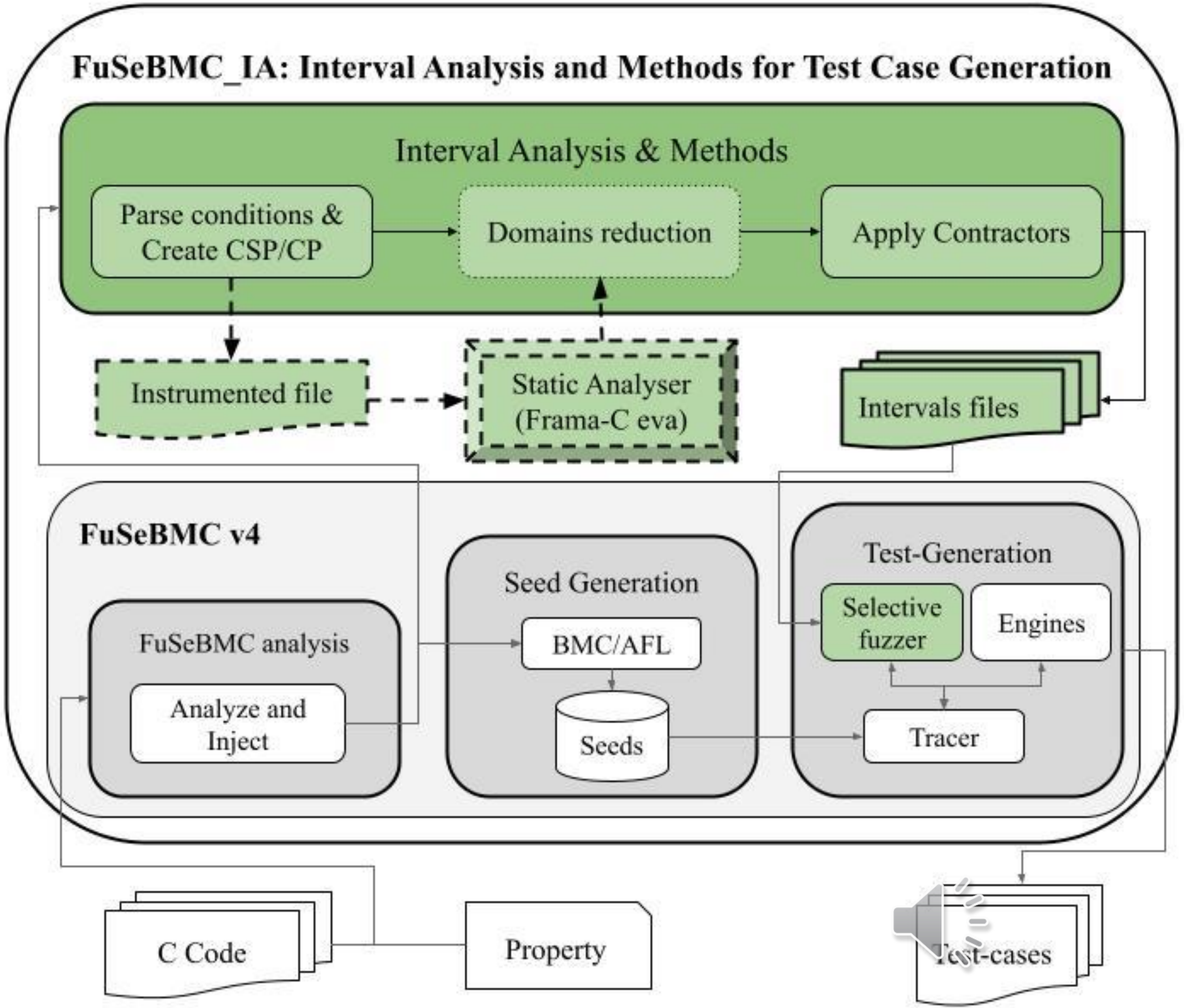
- Combine different techniques to produce various test-cases.
- Desire to optimize the current version of FuSeBMC in terms of time by pruning the search space.

FuSeBMC_IA

We propose FuSeBMC_IA, a test case generator that relies on Abstract Interpretation and Interval Analysis and Methods to improve the selective fuzzer by pruning the search space for the fuzzer.

FuSeBMC_IA Framework

- Built on top of FuSeBMC v4
- Utilizes Abstract interpretation to extract accurate intervals.
- Applies Interval Methods (Contractor) to prune the search space for the Selective fuzzer.



FuSeBMC_IA: Interval Analysis and Methods for Test Case Generation

Interval Analysis & Methods

Parse conditions & Create CSP/CP

Domains reduction

Apply Contractors

Instrumented file

Static Analyser (Frama-C eva)

Intervals files

FuSeBMC v4

FuSeBMC analysis

Analyze and Inject

Seed Generation

BMC/AFL

Seeds

Test-Generation

Selective fuzzer

Engines

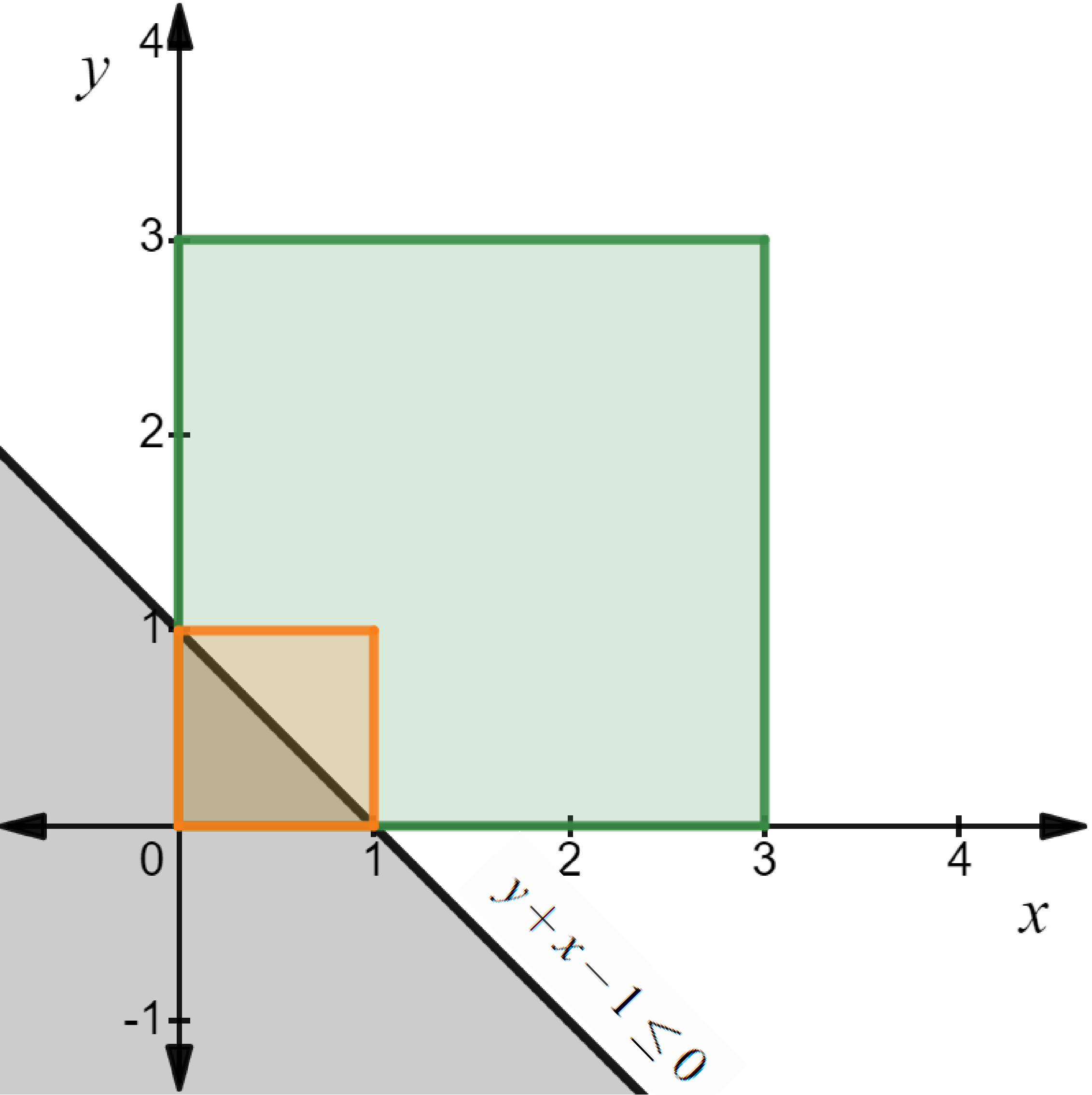
Tracer

C Code

Property

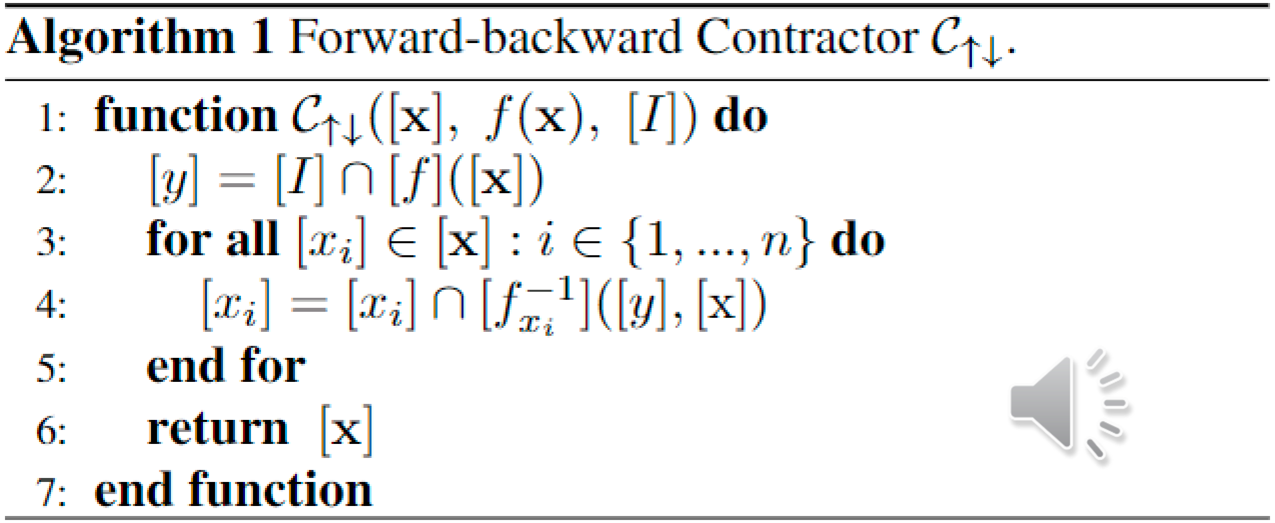
Test-cases

Forward-Backward Contractors



Forward-backward Contractor is an interval method that is applied to a Constraint Satisfaction Problem with one single constraint; it contracts in two steps: forward evaluation and backward propagation.

$$C([X]) \subseteq [X]$$

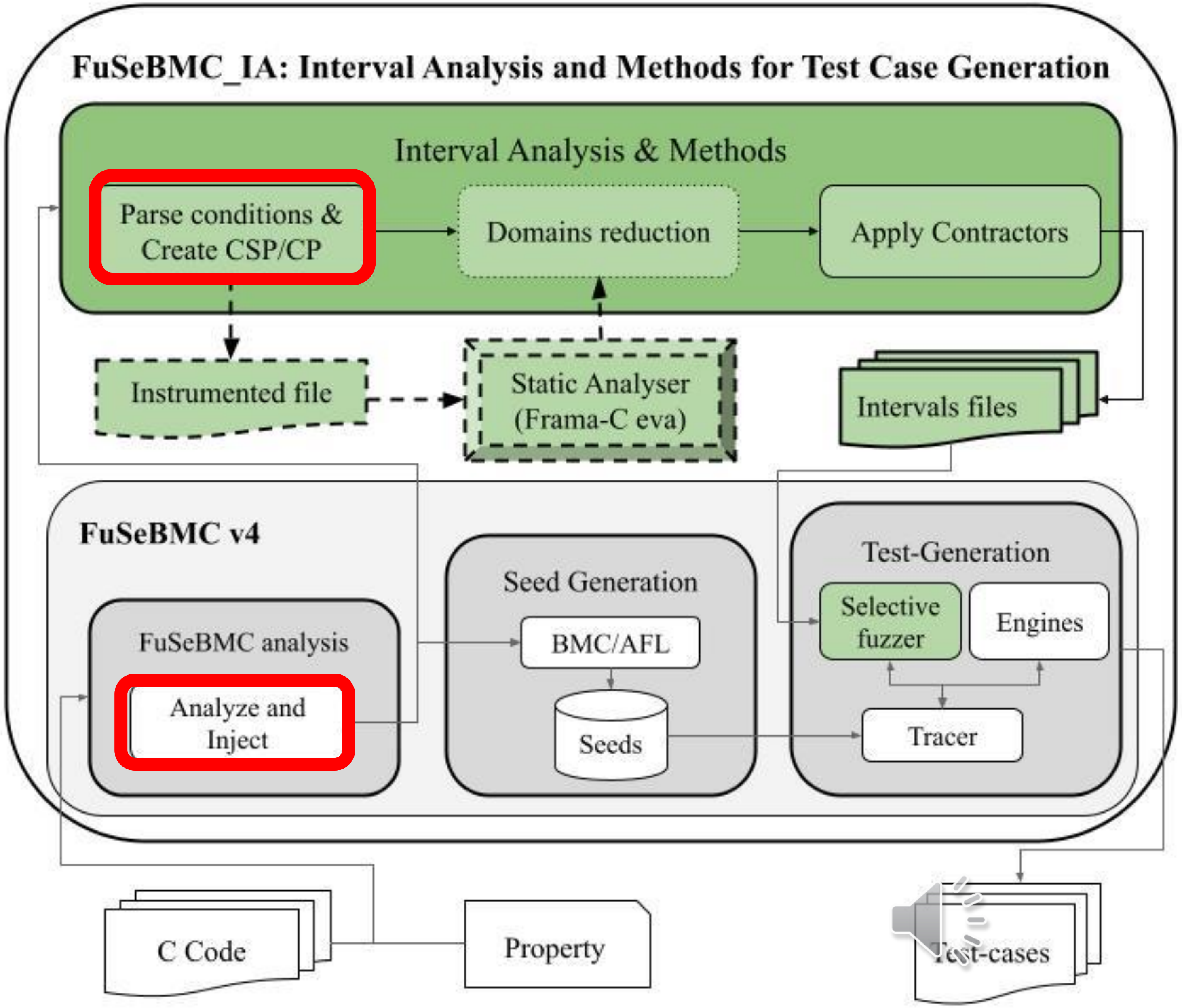


Algorithm 1 Forward-backward Contractor $\mathcal{C}_{\uparrow\downarrow}$.

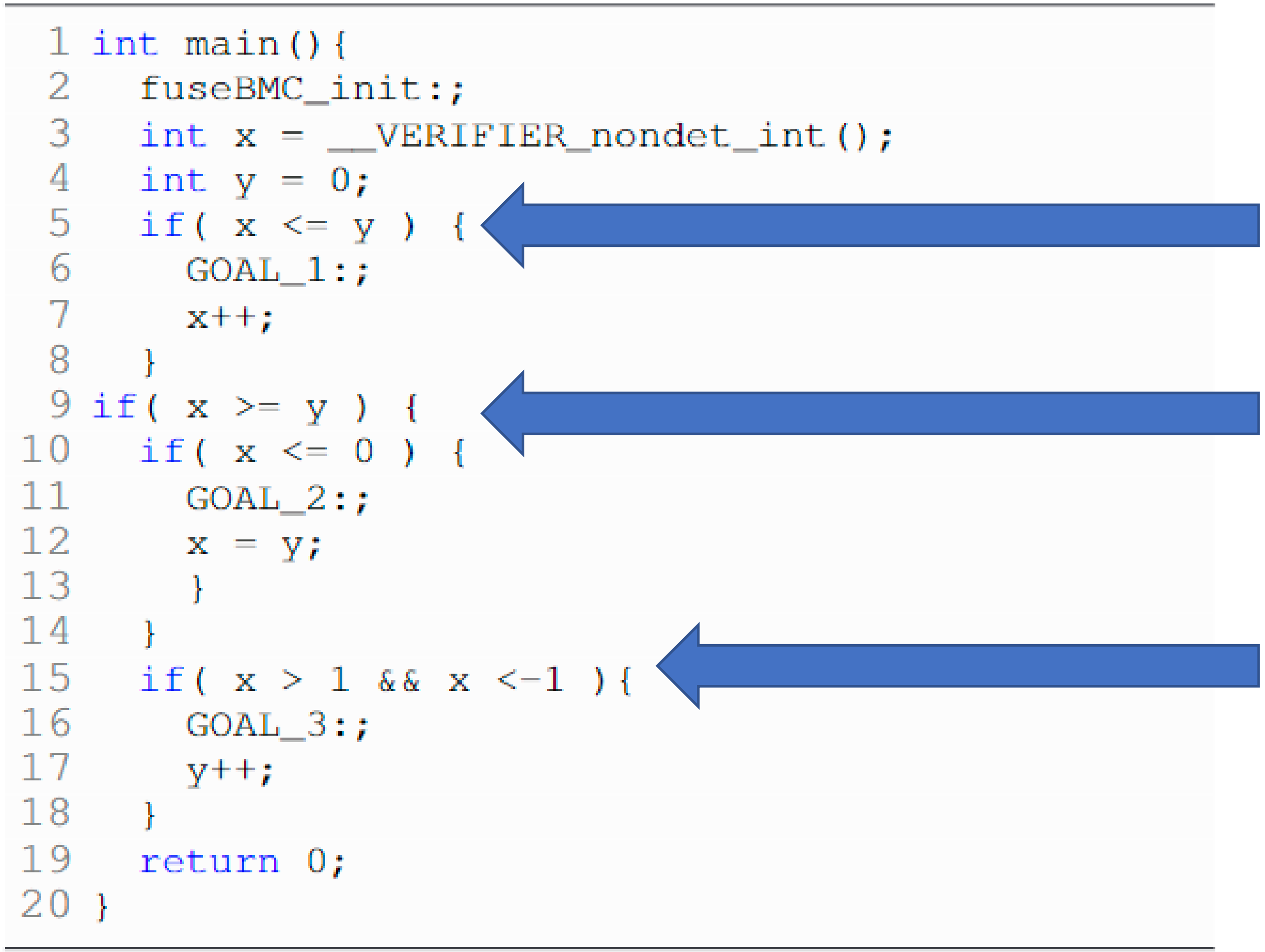
```
function $\mathcal{C}_{\uparrow\downarrow}([\mathrm{x}],\ f(\mathrm{x}),\ [I])$ do
    $[y] = [I] \cap [f]([\mathrm{x}])$
    for all $[x_i] \in [\mathrm{x}] : i \in \{1, ..., n\}$ do
        $[x_i] = [x_i] \cap [f_{x_i}^{-1}]([y], [\mathrm{x}])$
    end for
    return $[\mathrm{x}]$
end function
```

FuSeBMC_IA Framework

- Starting with FuSeBMC analysis. (Goals Instrumented)
- We parse the conditions leading to each goal to create a Constraint Satisfaction Problem.
- Produce an instrumented file to Frama-C



Creating Contractors



```
int main(){
  fuseBMC_init:;
  int x = __VERIFIER_nondet_int();
  int y = 0;
  if( x <= y ) {
    GOAL_1:;
    x++;
  }
if( x >= y ) {
  if( x <= 0 ) {
    GOAL_2:;
    x = y;
    }
  }
  if( x > 1 && x <-1 ){
    GOAL_3:;
    y++;
  }
  return 0;
}
```

File Instrumented by FuSeBMC analysis

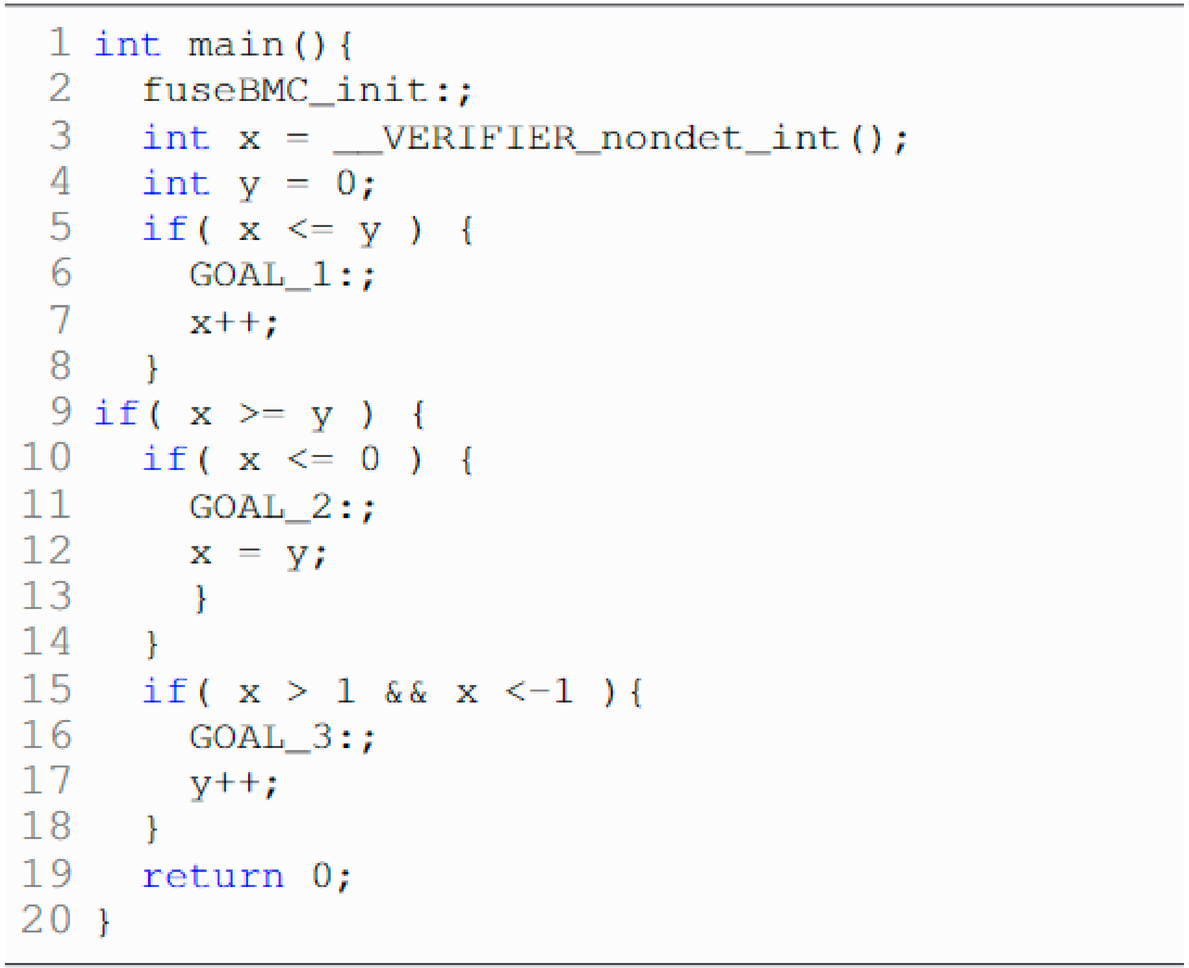
$X$is an interval vector or a box

$C_1(X)$constraint is $x - y \leq 0$

$C_2(X)$ is a composition of two contractors:
$C_{2.1}(X)$ with constraint $y - x < 0$
$C_{2.2}(X)$ with constraint $x \leq 0$
$C_2(X) = C_{2.1}(X) \cap C_{2.2}(X)$

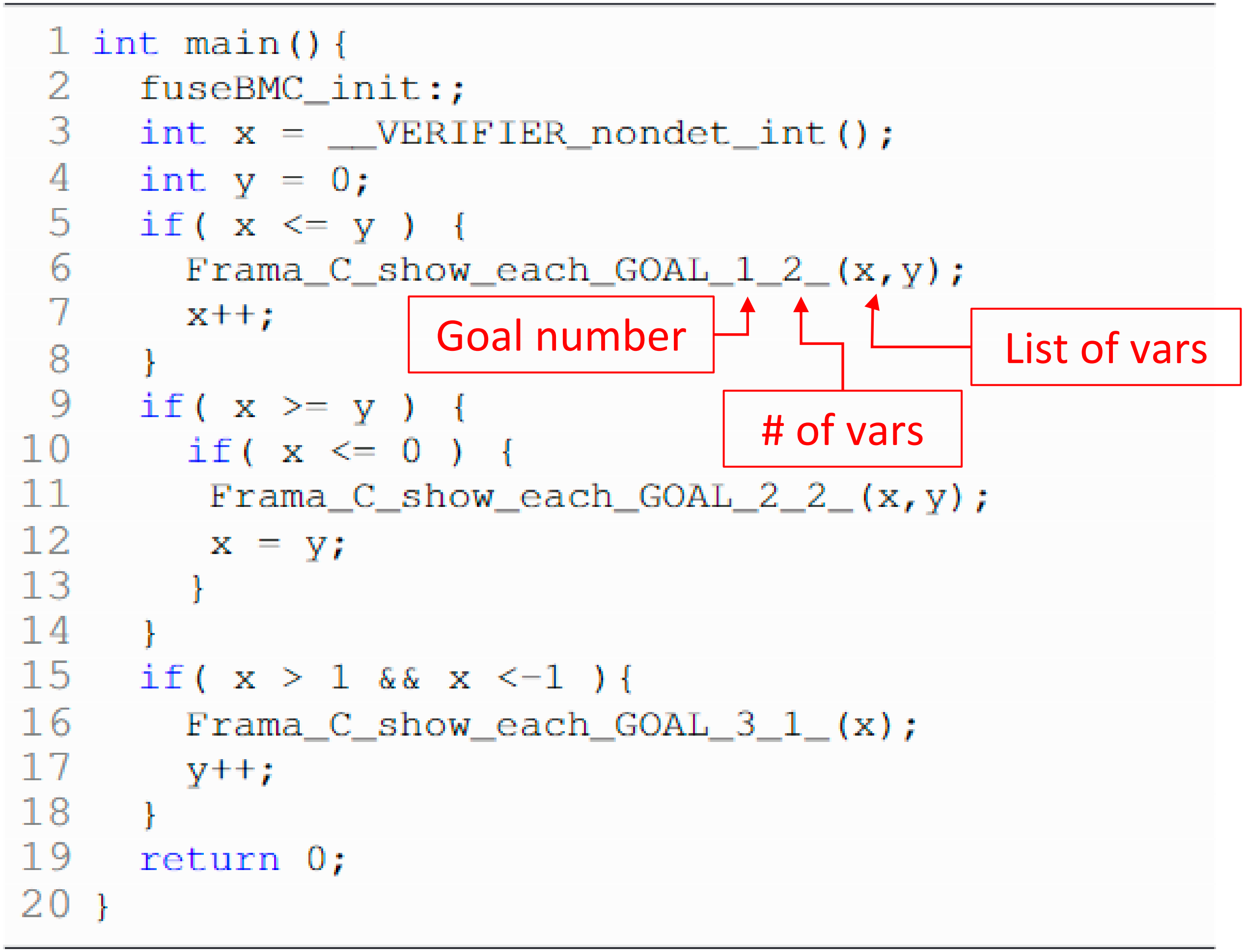
$C_3(X)$ is also a composition of two contractors:
$C_{3.1}(X)$ with constraint $1 - x \leq 0$
$C_{3.2}(X)$ with constraint $x + 1 < 0$
$C_3(X) = C_{3.1}(X) \cap C_{3.2}(X)$

Frama-C instrumentation



```
int main(){
  fuseBMC_init:;
  int x = __VERIFIER_nondet_int();
  int y = 0;
  if( x <= y ) {
    GOAL_1:;
    x++;
  }
if( x >= y ) {
  if( x <= 0 ) {
    GOAL_2:;
    x = y;
    }
  }
  if( x > 1 && x <-1 ){
    GOAL_3:;
    y++;
  }
  return 0;
}
```

File Instrumented by FuSeBMC analysis

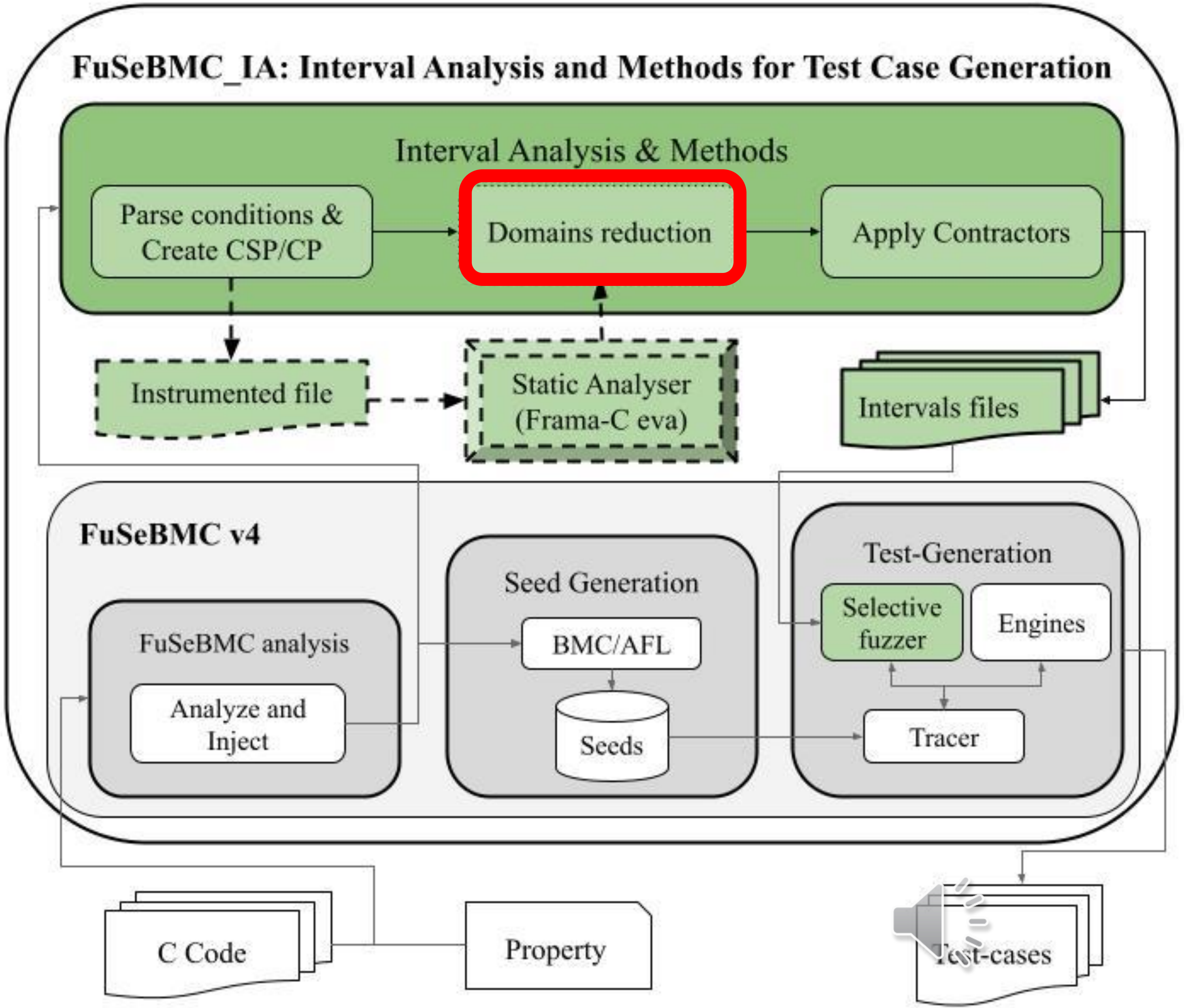


```
int main(){
  fuseBMC_init:;
  int x = __VERIFIER_nondet_int();
  int y = 0;
  if( x <= y ) {
    Frama_C_show_each_GOAL_1_2_(x,y);
    x++;
  }
  if( x >= y ) {
    if( x <= 0 ) {
     Frama_C_show_each_GOAL_2_2_(x,y);
     x = y;
    }
  }
  if( x > 1 && x <-1 ){
    Frama_C_show_each_GOAL_3_1_(x);
    y++;
  }
  return 0;
}
```

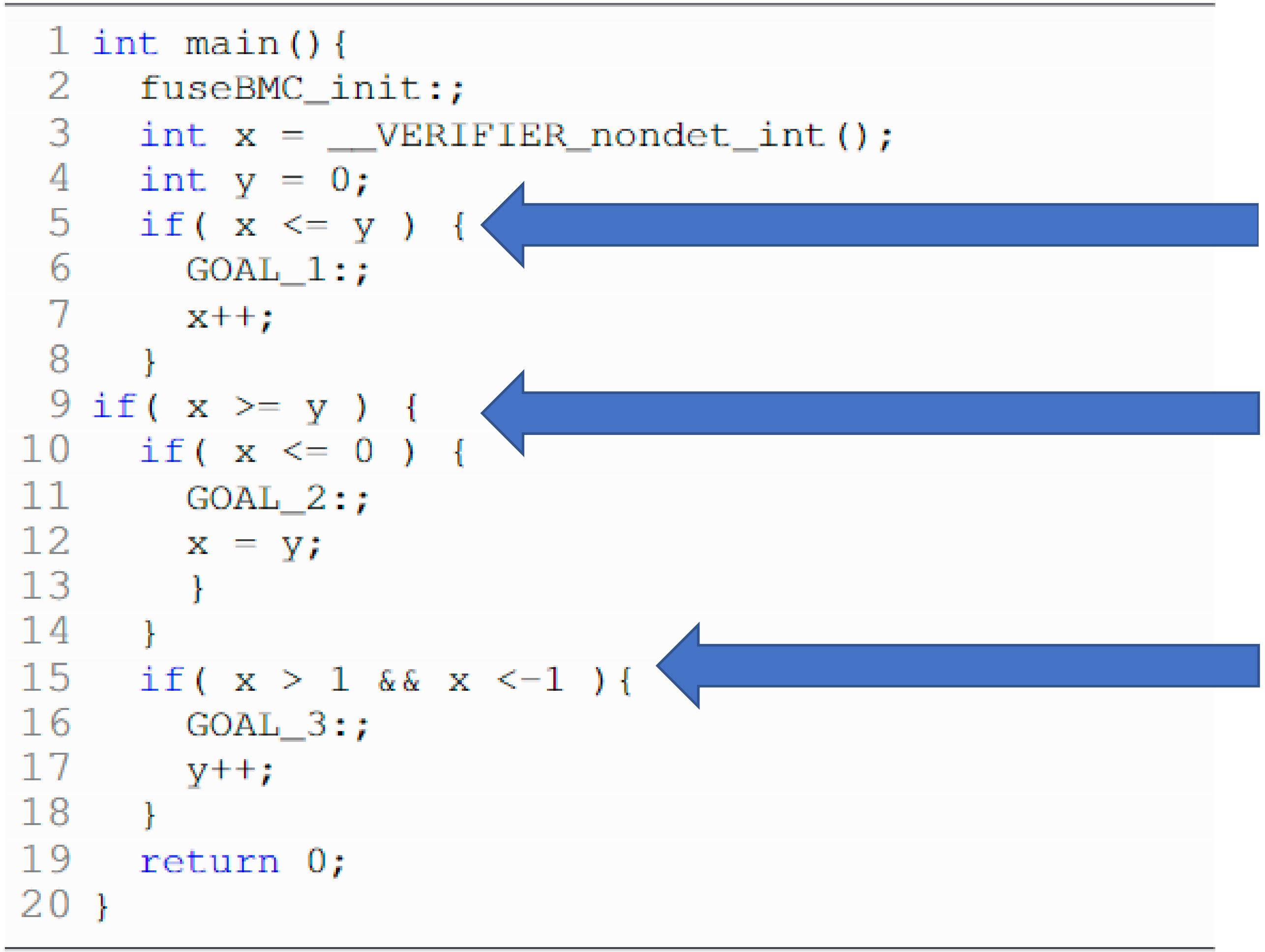
File Instrumented to be run in Frama-C

FuSeBMC_IA Framework

- Run Frama-C with the instrumented file
- Parse the output of Frama-C and update intervals



Frama-C instrumentation



```
int main(){
  fuseBMC_init:;
  int x = __VERIFIER_nondet_int();
  int y = 0;
  if( x <= y ) {
    GOAL_1:;
    x++;
  }
if( x >= y ) {
  if( x <= 0 ) {
    GOAL_2:;
    x = y;
    }
  }
  if( x > 1 && x <-1 ){
    GOAL_3:;
    y++;
  }
  return 0;
}
```

File Instrumented by FuSeBMC analysis

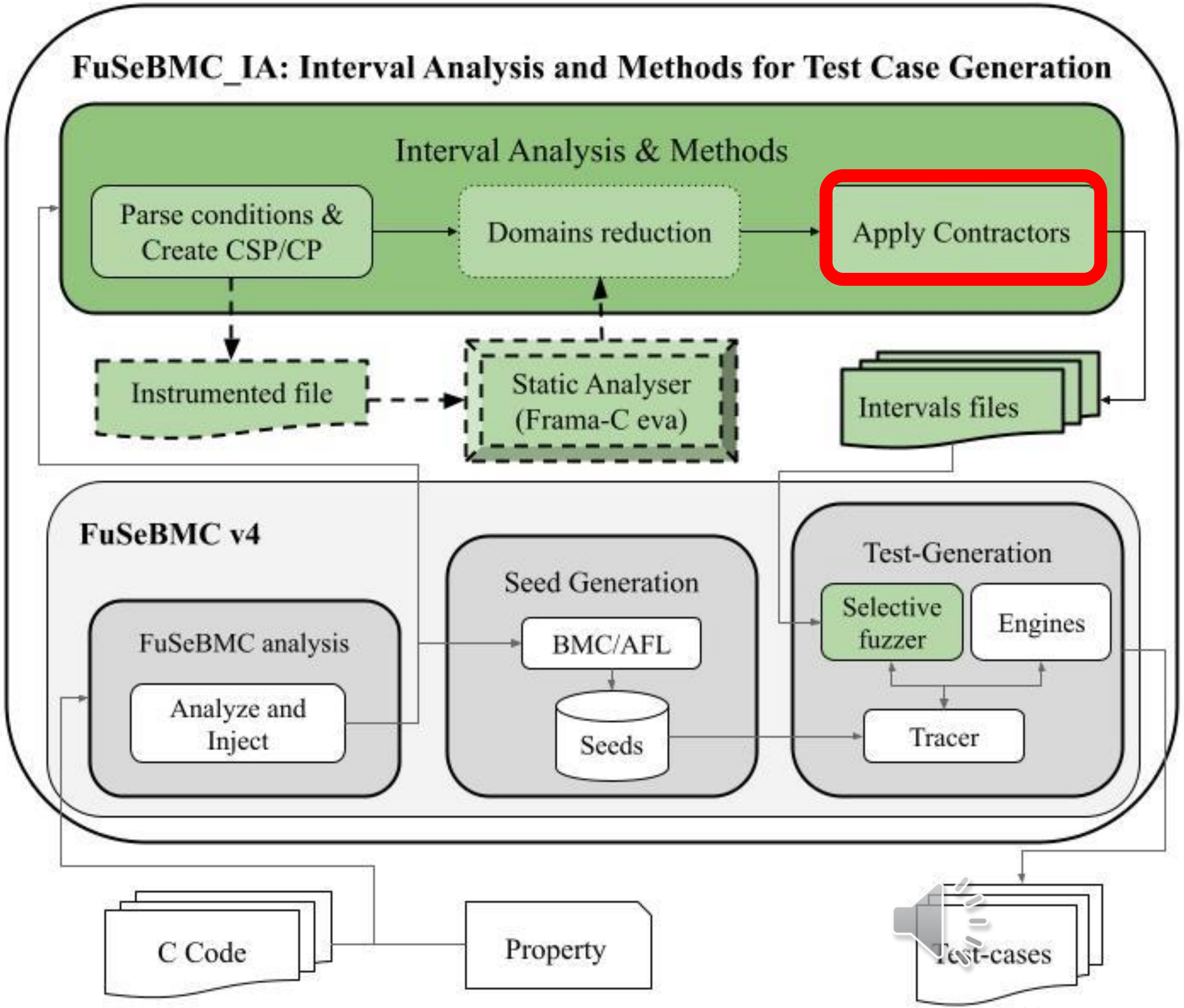
$[x] = (-\infty, \infty)$ reduced to $[-2147483648, 2147483647]$
$[y] = (-\infty, \infty)$ reduced to $[0, 0]$

$[x] = (-\infty, \infty)$ reduced to $[-2147483648, 2147483647]$
$[y] = (-\infty, \infty)$ reduced to $[0, 0]$

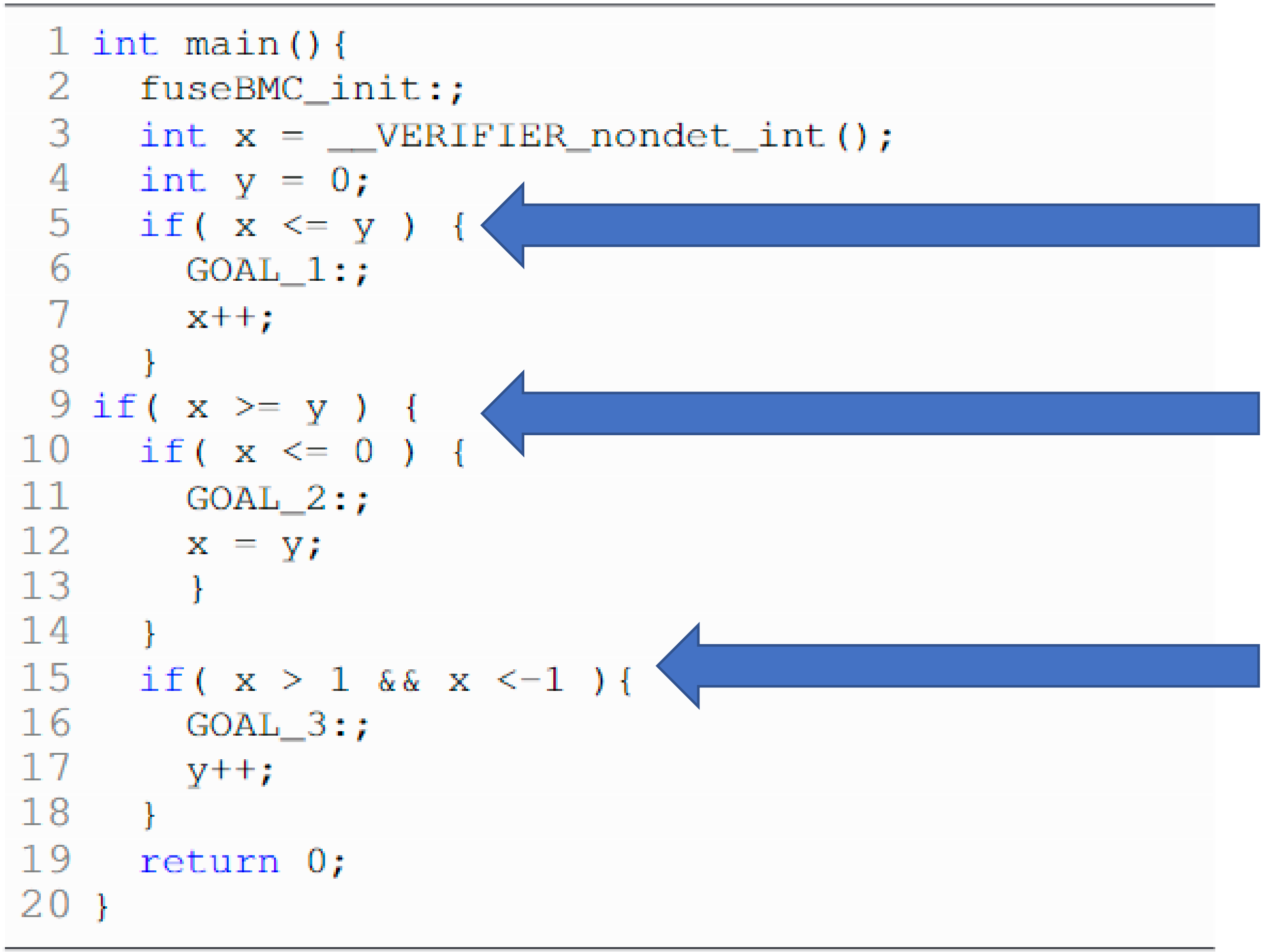
$[x] = (-\infty, \infty)$ reduced to $[-2147483648, 2147483647]$

FuSeBMC_IA Framework

- Apply Contractors for each goal.
- Produce a file with each goal and variables intervals



Apply Contractor



```
int main(){
  fuseBMC_init:;
  int x = __VERIFIER_nondet_int();
  int y = 0;
  if( x <= y ) {
    GOAL_1:;
    x++;
  }
if( x >= y ) {
  if( x <= 0 ) {
    GOAL_2:;
    x = y;
    }
  }
  if( x > 1 && x <-1 ){
    GOAL_3:;
    y++;
  }
  return 0;
}
```

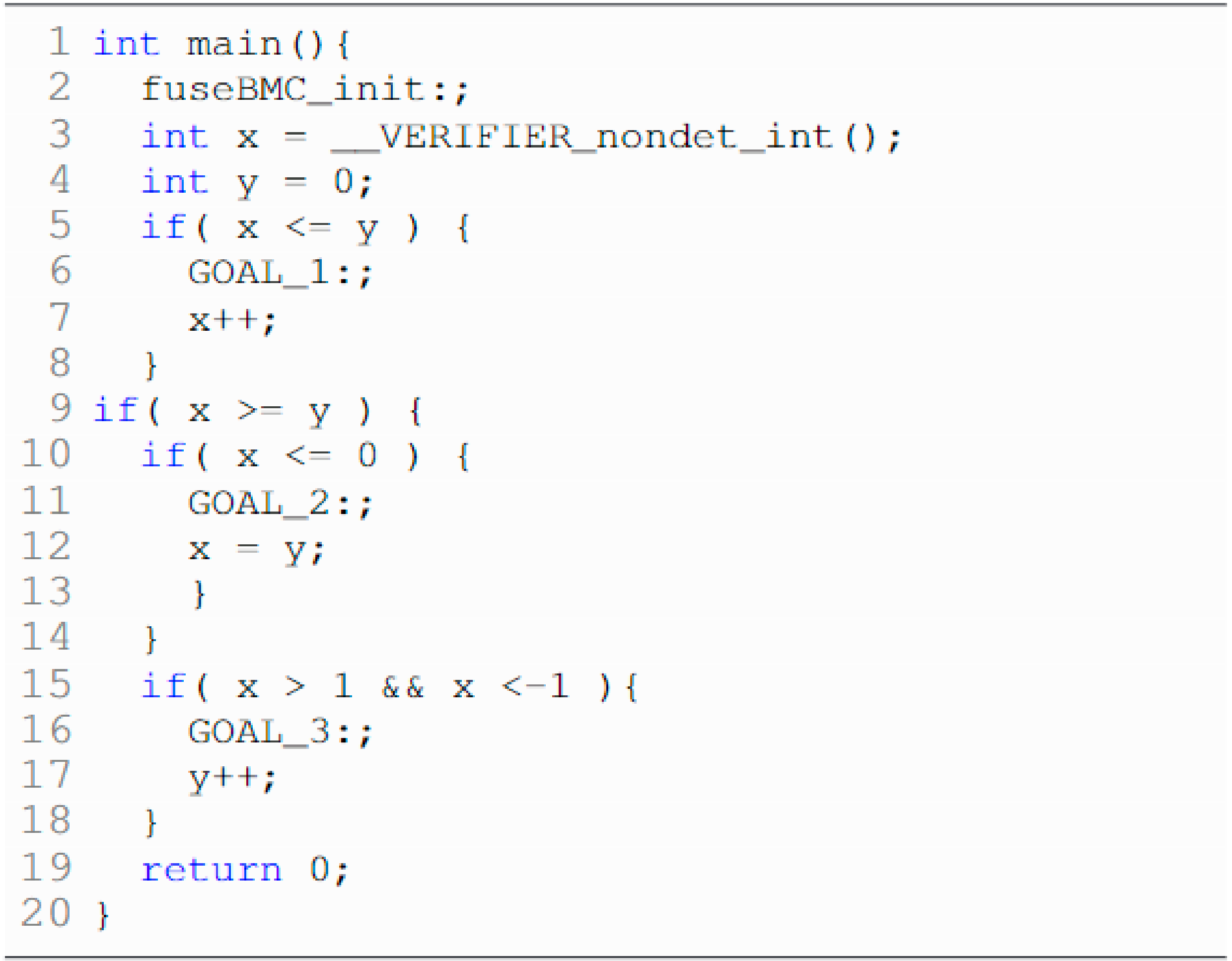
File Instrumented by FuSeBMC analysis

$$[X] = [[-2147483648, 2147483647], [0,0]]$$
$$C_1(X) = [[-2147483648, 0], [0,0]]$$

$$[X] = [[-2147483648, 2147483647], [0,0]]$$
$$C_2(X) = [[0,0], [0,0]]$$

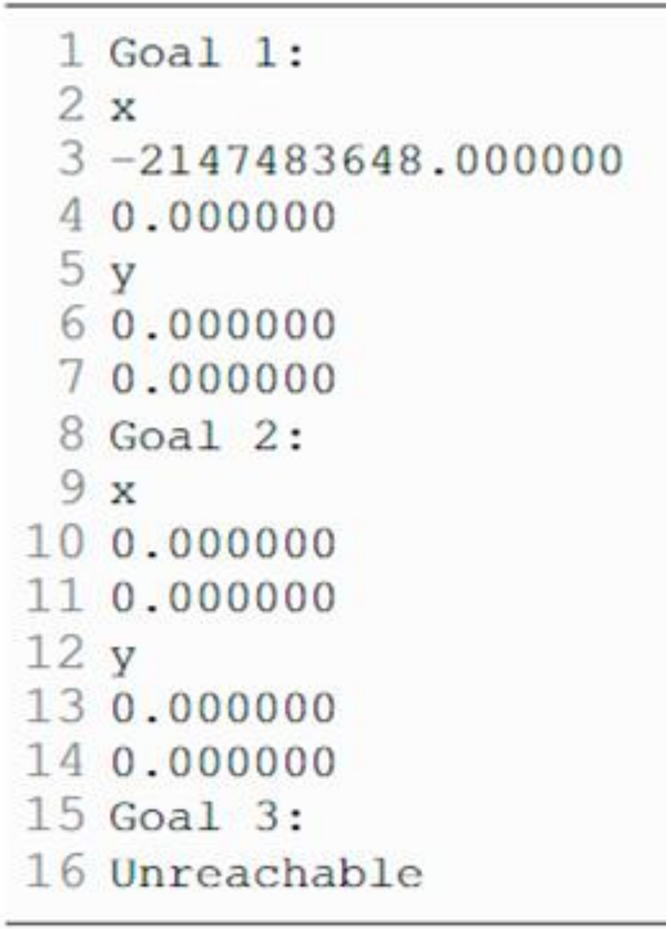
$$[X] = [-2147483648, 2147483647]$$
$$C_3(X) = \phi$$

Produced File



```
int main(){
  fuseBMC_init:;
  int x = __VERIFIER_nondet_int();
  int y = 0;
  if( x <= y ) {
    GOAL_1:;
    x++;
  }
if( x >= y ) {
  if( x <= 0 ) {
    GOAL_2:;
    x = y;
    }
  }
  if( x > 1 && x <-1 ){
    GOAL_3:;
    y++;
  }
  return 0;
}
```

File Instrumented by FuSeBMC analysis

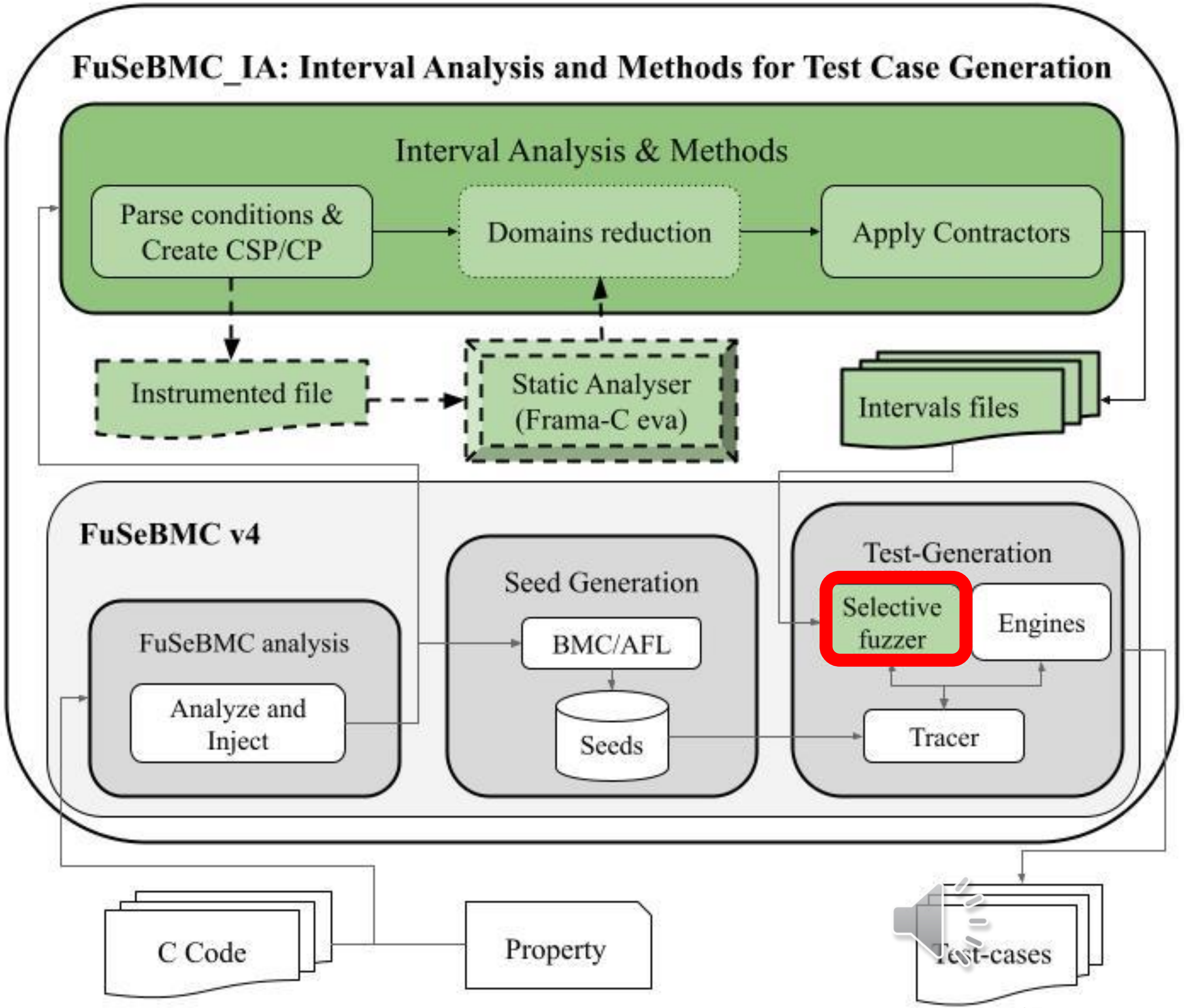


```
Goal 1:
x
-2147483648.000000
0.000000
y
0.000000
0.000000
Goal 2:
x
0.000000
0.000000
y
0.000000
0.000000
Goal 3:
Unreachable
```

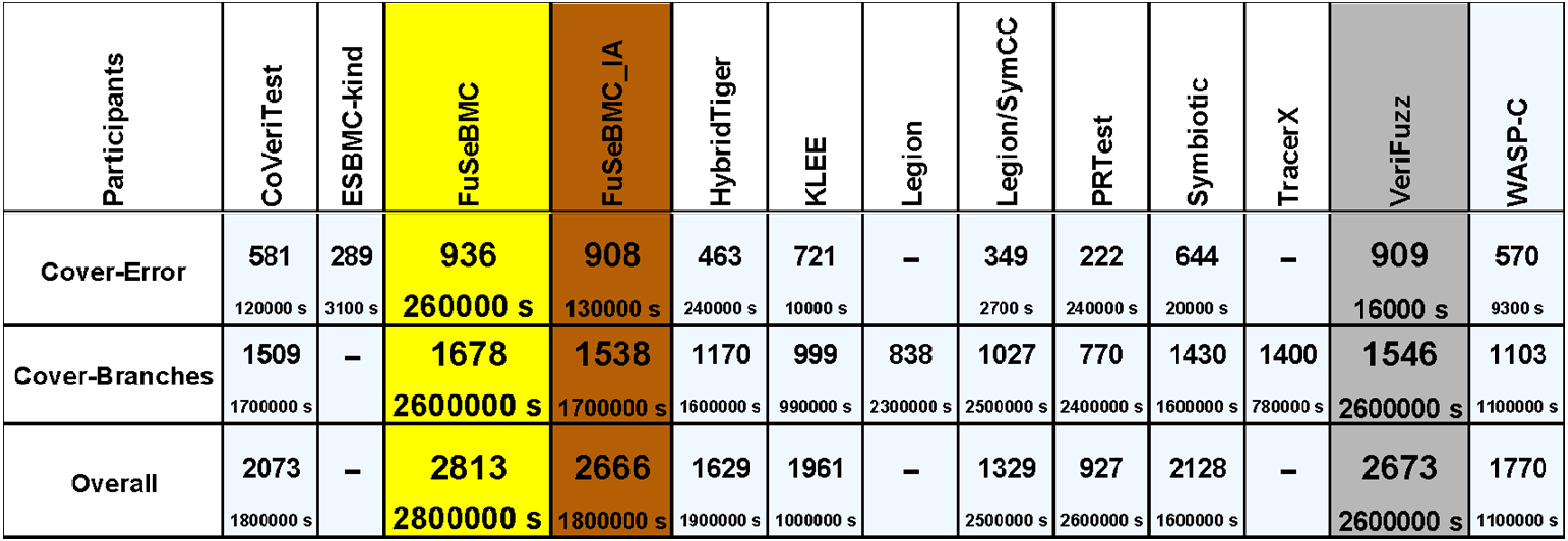
Produced Interval file

FuSeBMC_IA Framework

- With the intervals file as input, we fuzz the PUT with given intervals.
- If a goal is unreachable, we lower the priority for fuzzing it.



Evaluation of FuSeBMC_IA in Test-Comp 2023



Participants	CoVeriTest	ESBMC-kind	FuSeBMC	FuSeBMC_IA	HybridTiger	KLEE	Legion	Legion/SymCC	PRTest	Symbiotic	TracerX	VeriFuzz	WASP-C
Cover-Error	581 120000 s	289 3100 s	936 260000 s	908 130000 s	463 240000 s	721 10000 s	–	349 2700 s	222 240000 s	644 20000 s	–	909 16000 s	570 9300 s
Cover-Branches	1509 1700000 s	–	1678 2600000 s	1538 1700000 s	1170 1600000 s	999 990000 s	838 2300000 s	1027 2500000 s	770 2400000 s	1430 1600000 s	1400 780000 s	1546 2600000 s	1103 1100000 s
Overall	2073 1800000 s	–	2813 2800000 s	2666 1800000 s	1629 1900000 s	1961 1000000 s	–	1329 2500000 s	927 2600000 s	2128 1600000 s	–	2673 2600000 s	1770 1100000 s

The improvement achieved by FuSeBMC_IA in comparison to FuSeBMC v4 in terms of time

Participants	FuSeBMC	FuSeBMC_IA	Points decrease Time decrease
Cover-Error	936 260000	908 130000	-3% -50%
Cover-Branches	1678 2600000	1538 1700000	-8% -35%
Overall	2813 2800000	2666 1800000	-5% -36%
Points per minute	0.060278571	0.088866667	47% increase

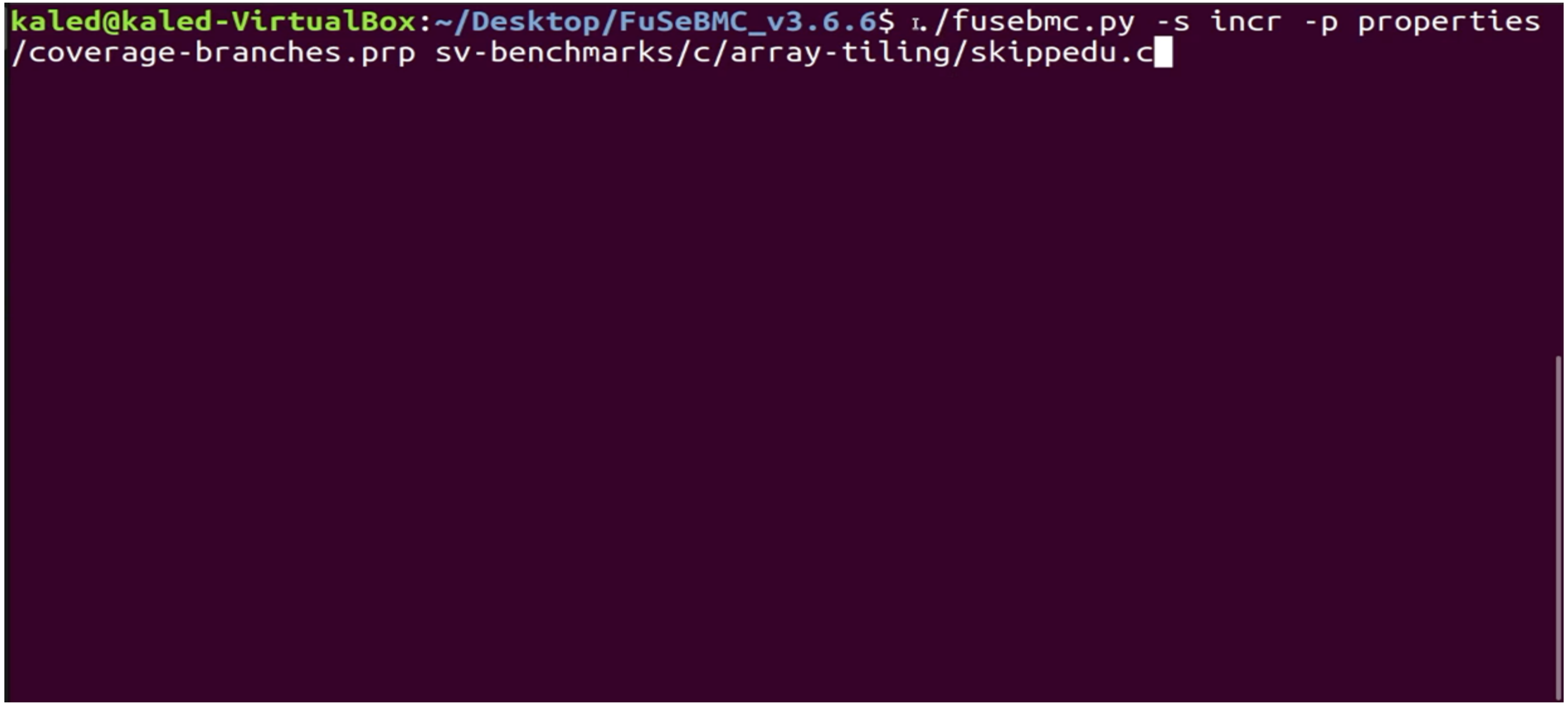
Awards

FuSeBMC_IA received three significant awards from the 5th International Competition on Software Testing (Test-Comp 2023) organized by the European Joint Conferences on Theory and Practice of Software (ETAPS).

- FuSeBMC_IA got third place in the most critical category of Test-Comp: **Cover-Error** (find a test that covers a bug).
- FuSeBMC_IA got third place in the category of Test-Comp: **Cover-Branches** (find a test that covers a branch).
- FuSeBMC_IA earned third place in Test-Comp's overall category.

Software Project

FuSeBMC_IA source code is written in C++ and Python; it is available for download on GitHub. Also, the instructions for using the tool FuSeBMC_IA are given in the file README.



```
kaled@kaled-VirtualBox:~/Desktop/FuSeBMC_v3.6.6$ ./fusebmc.py -s incr -p properties
/coverage-branches.prp sv-benchmarks/c/array-tiling/skippedu.c
```

Want to Try it?

Find out more about FuSeBMC_IA at :
https://github.com/Mohannad-aldughaim/FuSeBMC_IA

FuSeBMC_IA

Test-Comp’23 paper: “FuSeBMC_IA: Interval Analysis and Methods for Test Case Generation (Competition Contribution)”

Mohannad.aldughaim@manchester.ac.uk

Thank you…