Progess Report 1

Xie Li

August 31, 2020

Overview of the Progress

- Survey of Tools: CBMC, NuSMV and CPAchecker
- Reading paper: things related to separation logic.
- Survey of Papers and Slides:
  - Slides from CBMC site.
  - Alessandro Cimatti et al. Integrating BDD-based and SAT-based Symbolic Model Checking.
  - E. Clarke et al. Symbolic Model Checking.
  - NuSMV 2.5 Tutorial and other related slides.
  - Slides from CPAchecker site.
  -
- Usage of the Tools and brief introduction to Algorithms.

Tool Survey: CBMC

CBMC is a bounded model checker for C and C++.

Functionalities:

```
Program instrumentation options:
 --bounds-check                 enable array bounds checks
 --pointer-check                enable pointer checks (always enabled for Java)
 --memory-leak-check            enable memory leak checks
 --div-by-zero-check            enable division by zero checks
 --signed-overflow-check        enable signed arithmetic over- and underflow check
s
 --unsigned-overflow-check      enable arithmetic over- and underflow checks
 --pointer-overflow-check       enable pointer arithmetic over- and underflow chec
ks
 --conversion-check             check whether values can be represented after type
 cast
 --undefined-shift-check        check shift greater than bit-width
 --float-overflow-check         check floating-point for +/-Inf
 --nan-check                    check floating-point for NaN
 --no-built-in-assertions       ignore assertions in built-in library
 --no-assertions                ignore user assertions
 --no-assumptions               ignore user assumptions
 --error-label label            check that label is unreachable
 --cover CC                     create test-suite with coverage criterion CC
 --mm MM                        memory consistency model for concurrent programs
 --reachability-slice           remove instructions that cannot appear on
                                a trace from entry point to a property
 --reachability-slice-fb        remove instructions that cannot appear on
                                a trace from entry point through a property
 --full-slice                   run full slicer (experimental)
```

Usage: `cbmc input.c --bounds-check --pointer-check`

Insight of Algorithm

```
clexma@clexma-ThinkPad-P52s:~/Desktop/Disk_D/gitRepos/Tex/TexBak/llvm_discussion1/cbmc$ cbmc buffer.c --bo
CBMC version 5.10 (cbmc-5.10) 64-bit x86_64 linux
Parsing buffer.c
Converting
Type-checking buffer
Generating GOTO Program
Adding CPROVER library (x86_64)
Removal of function pointers and virtual functions
Generic Property Instrumentation
Running with 8 object bits, 56 offset bits (default)
Starting Bounded Model Checking
size of program expression: 33 steps
simple slicing removed 6 assignments
Generated 1 VCC(s), 1 remaining after simplification
Passing problem to propositional reduction
converting SSA
Running propositional reduction
Post-processing
Solving with MiniSAT 2.2.1 with simplifier
328 variables, 11 clauses
SAT checker: instance is SATISFIABLE
Runtime decision procedure: 0.00503025s

** Results:
[main.array_bounds.1] array `buffer' upper bound in buffer[(signed long int)20]: FAILURE

** 1 of 1 failed (1 iteration)
```

Slides from the official website of CBMC.

Tool Survey: NuSMV

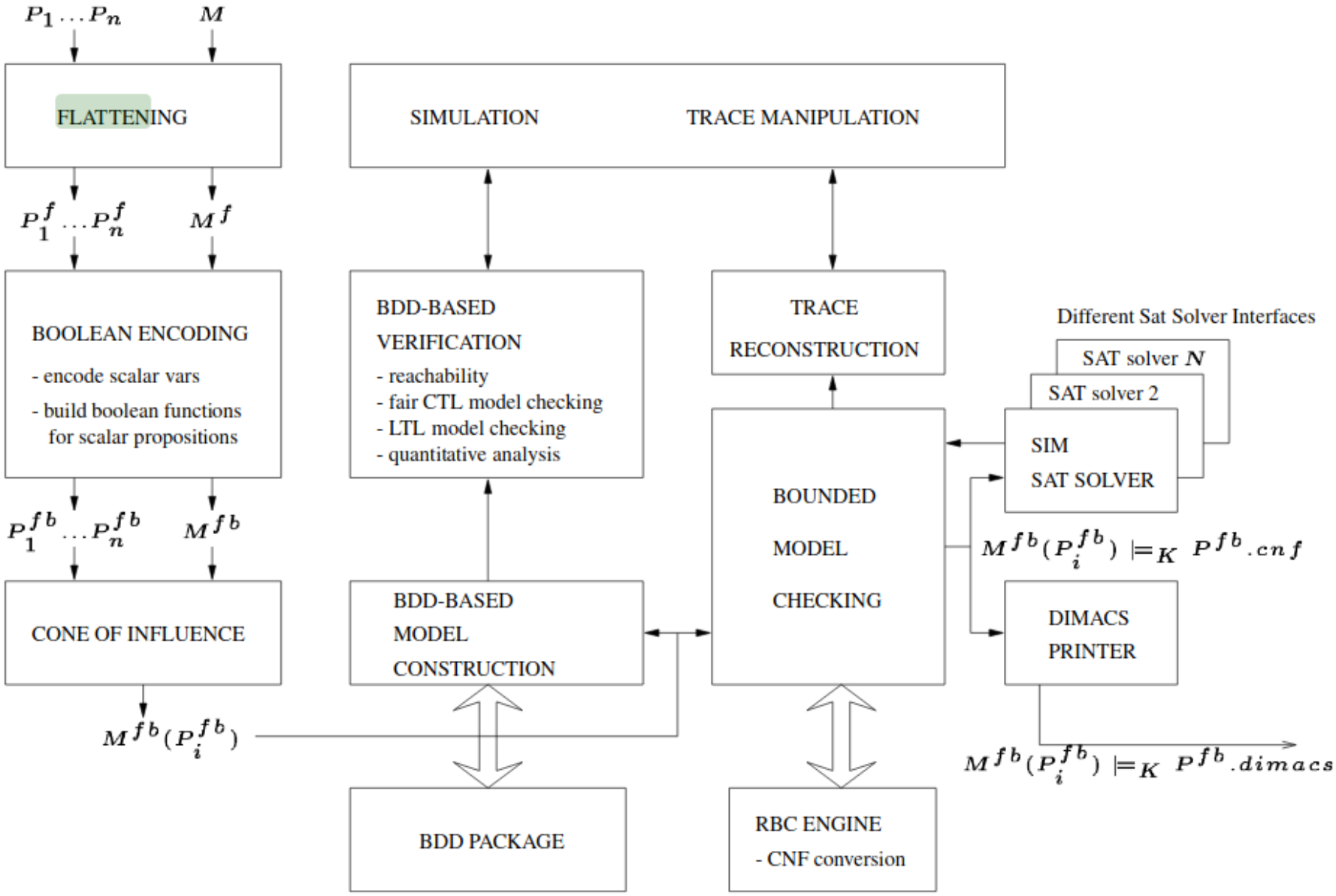
NuSVM is a NEW tool of Symbolic model checker for finite state systems.

Features:

- Analysis of invariants.
- LTL model checking.
- PSL model checking.
- SAT-based bounded model checking.

The tool is given a SMV language file as input and

Overview of NuSMV



Input Format of NuSMV

```
1 MODULE main
2 VAR
3   bit0 : counter_cell(TRUE);
4   bit1 : counter_cell(bit0.carry_out);
5   bit2 : counter_cell(bit1.carry_out);
6 SPEC
7   AG AF bit2.carry_out
8
9 SPEC AG(!bit2.carry_out)
10
11 MODULE counter_cell(carry_in)
12 VAR
13   value : boolean;
14 ASSIGN
15   init(value) := FALSE;
16   next(value) := value xor carry_in;
17 DEFINE
18   carry_out := value & carry_in;
```

`./NuSMV couter.smv`

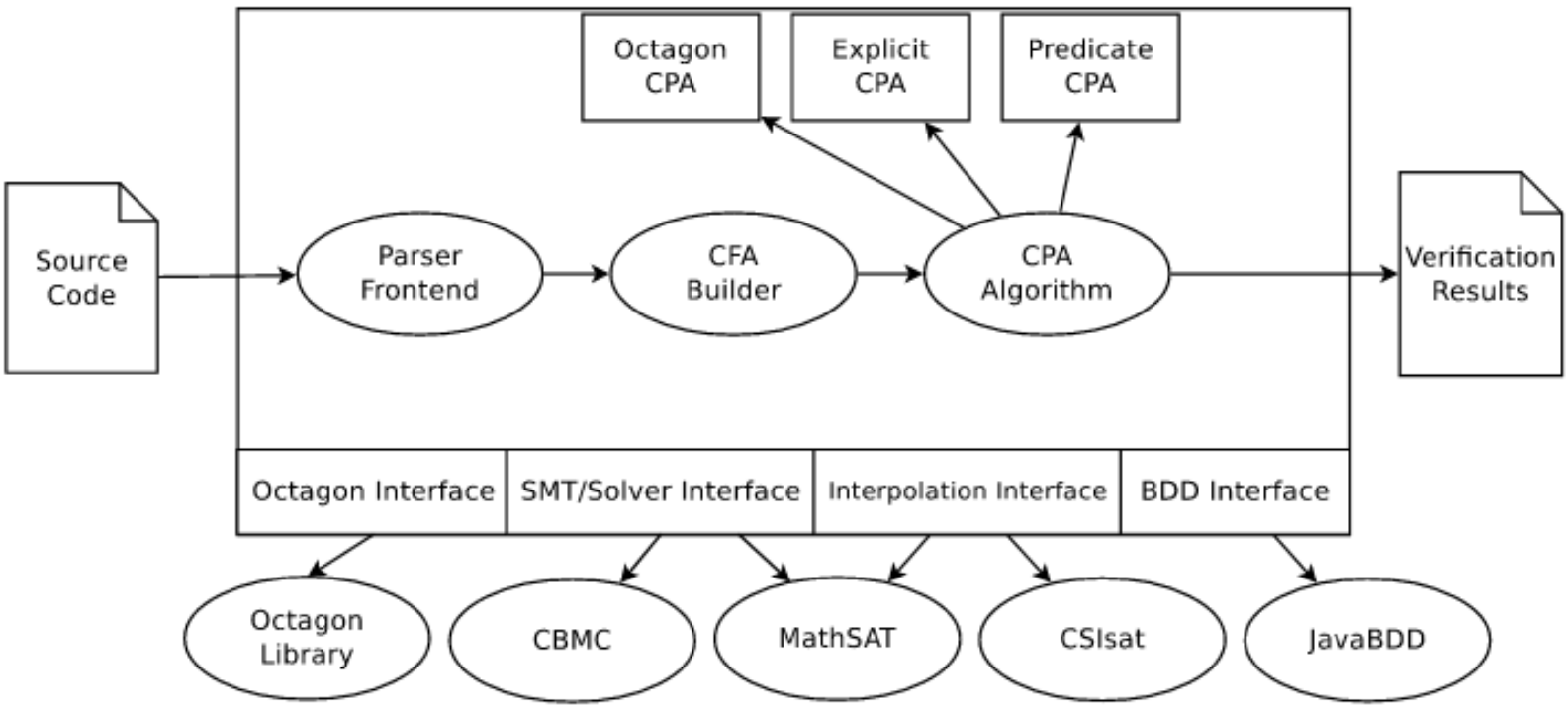
```
-- specification AG (AF bit2.carry_out)  is true
-- specification AG !bit2.carry_out  is false
-- as demonstrated by the following execution sequence
```

Tool Survey: CPAChecker

CPAChecker is a configurable software-verification platform that enables the parsing, analysing and verifying of the source program.

- Static Analysize: Data-Flow analysis etc.
- Invariant generation via over-approximation.
- Termination checking.
- ...

Architecture



Future Work

- A closer look into the source code of these tools especially CPAchecker.
- Better understanding of the underlying algorithm and techniques.
- Other tool survey: SPIN, Infer, KLEE, etc.