

# SMT Solvers, CBMC

<http://d3s.mff.cuni.cz>

Department of  
Distributed and  
Dependable  
Systems



*Pavel Parízek*



FACULTY  
OF MATHEMATICS  
AND PHYSICS  
Charles University

- SMT solver
  - theories: linear integer arithmetic, uninterpreted functions, arrays, bit vectors, ...
  - Input: SMT-LIB v2 (<http://smtlib.cs.uiowa.edu/>)
- Created by Microsoft Research
- Supports both Windows and Linux
- Source code & wiki: <https://github.com/Z3Prover/z3>
- Online interface: <https://jfmc.github.io/z3-play/>

# SMT-LIB: basics

; comment: after the semicolon until the end of a line  
;

; specify the logic to be used  
(set-logic QF\_UFLIA)

; condition that should hold  
(assert (= c (+ a 2))) ; c == a + 2

# SMT-LIB: expressions

(not P)

(and b1 b2)

(or ...)

(xor ...)

(+ a b c d)

(= a b)

(=> true false)

# SMT-LIB: predicates and functions

**; function symbol**

```
(declare-fun Plus (Int Int) Int)
```

**; predicate**

```
(declare-fun Odd (Int) Bool)
```

**; constant**

```
(declare-fun a () Int)
```

```
(declare-const c Int) ; syntactic sugar
```

# SMT-LIB: example

```
(declare-const a Int)
(declare-const b Int)
(declare-const c Int)

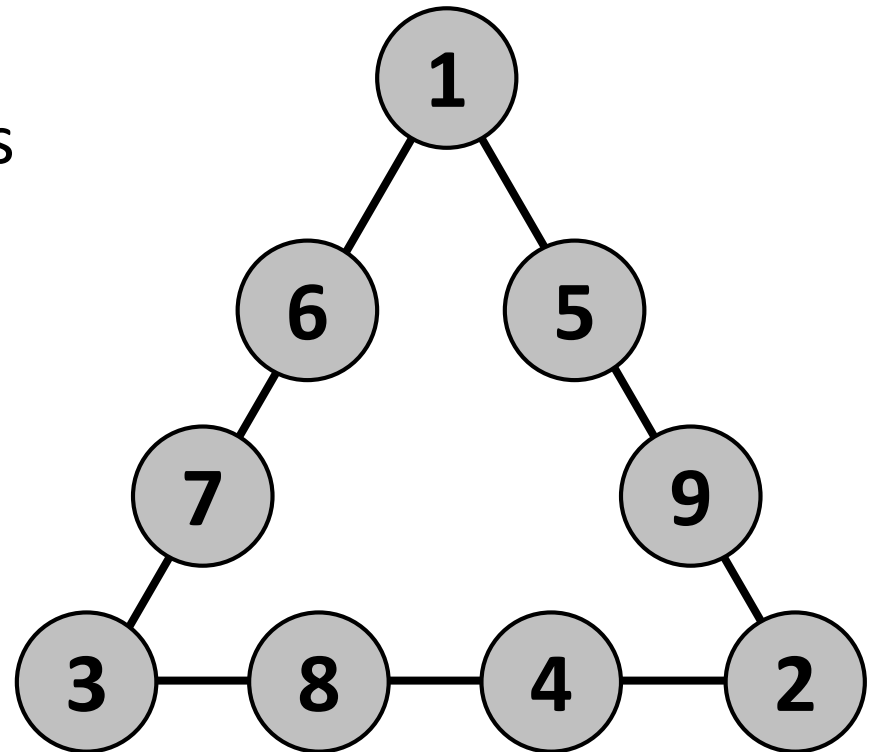
(assert (<= a b))
(assert (<= b c))
(assert (<= c a))
(assert (= 6 (+ a b c)))

; result: sat, unsat, unknown
(check-sat)

(get-model)
```

# Task 1: Triangle puzzle

- Goal
  - Fill the circles with different numbers from 1-9
  - Keep the sum of numbers on every side equal to 17



# Task 1: Triangle puzzle

```
(declare-const c1 Int)
...
```

; TODO you must add constraints on c1 ... c9 to get the solution

```
; all different
(assert (not (= c1 c2))) (assert (not (= c1 c3))) ...
(assert (not (= c2 c3))) ...
...
```

```
; get results
(check-sat)
(get-model)
```



# Task 1: Triangle puzzle

```
(declare-const c1 Int)
...

; sums for edges
(assert (= 17 (+ c1 c2 c3 c4)))
(assert (= 17 (+ c4 c5 c6 c7)))
(assert (= 17 (+ c7 c8 c9 c1)))

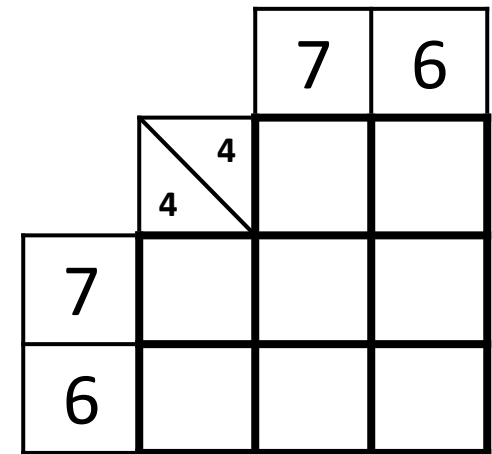
; in-range check
(assert (< 0 c1))
(assert (> 10 c1))
...

; all different
(assert (not (= c1 c2))) (assert (not (= c1 c3))) ...
(assert (not (= c2 c3))) ...
...

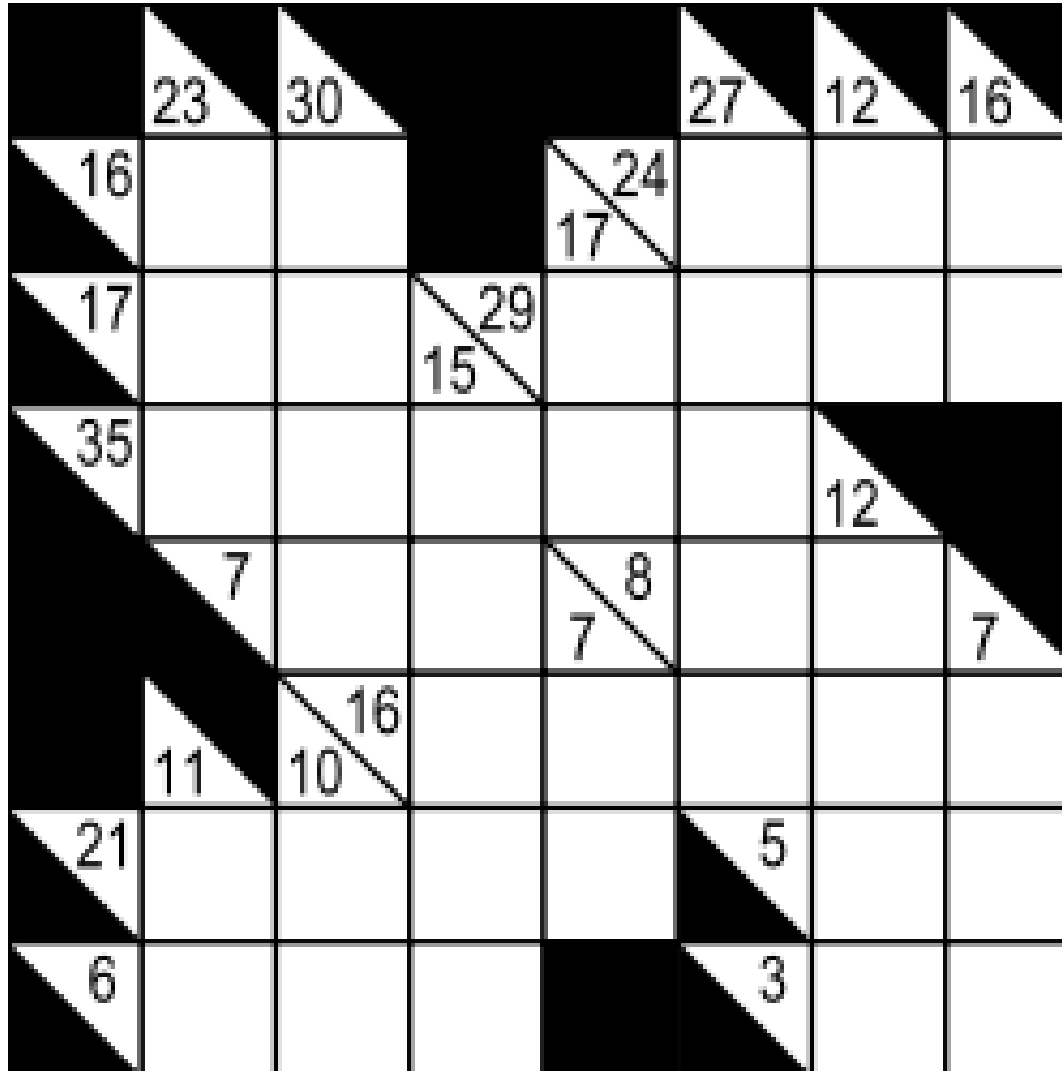
; get results
(check-sat)
(get-model)
```

# Kakuro puzzle

- Goal
  - Fill fields with numbers from 1-9
  - In a single row/column, the numbers may not repeat
  - Sums in the rows/columns must equal the specified number



# Kakuro puzzle



# How to encode programs using SMT

```
int max(int a, int b) {  
    int r;  
    if (a < b) {  
        r = b;  
    }  
    else { // a >= b  
        r = a;  
    }  
    assert (a <= r && b <= r);  
    return r;  
}
```

# How to encode programs using SMT

```
int max(int a, int b) {  
    int r;  
    if (a < b) {  
        r = b;  
    }  
    else { // a >= b  
        r = a;  
    }  
    assert (a <= r && b <= r);  
    return r;  
}
```



```
a = *, b = *  
r = *  
if (a < b) {  
    r = b;  
}  
else { // a >= b  
    r = a;  
}  
  
assert (a <= r && b <= r);
```

# How to encode programs using SMT

```
a = *, b = *  
r = *  
if (a < b) {  
    r = b;  
}  
else { // a >= b  
    r = a;  
}  
  
assert (a <= r && b <= r);
```

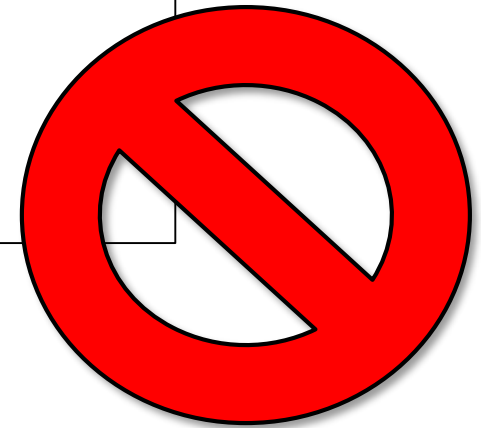


```
(declare-const a Int)  
(declare-const b Int)  
(declare-const r Int)  
  
(assert (or  
  (and (< a b) (= r b))  
  (and (>= a b) (= r a))  
)  
)  
  
(assert (and (<= a r) (<= b r)))  
  
(check-sat)  
(get-model)
```

# Variable assignments

```
int x = 0;  
  
x = x + 1;  
  
assert (x == 1);
```

```
(declare-const x1 Int)  
  
(assert (= x1 0))  
  
(assert (= x1 (+ x1 1)))  
  
(assert (= x1 1))  
  
(check-sat)  
(get-model)
```



# Variable assignments: versions & SSA

- Variables have multiple versions
- Static single assignment (SSA)

```
int x = 0;  
  
x = x + 1;  
  
assert (x == 1);
```

```
(declare-const x1 Int)  
(declare-const x2 Int)  
  
(assert (= x1 0))  
  
(assert (= x2 (+ x1 1)))  
  
(assert (= x2 1))  
  
(check-sat)  
(get-model)
```



# Arrays

```
(declare-const c Int)
```

```
(declare-const a1 (Array Int Int))
```

```
(declare-const a2 (Array Int Int))
```

```
(assert (= c (select a1 10)))
```

```
(assert (= a2 (store a1 1 20)))
```

# Bounded model checking

- Often used with SAT (propositional logic)
  - Limited expressiveness: no “+” and “-”

# Integer addition in HW/CPU

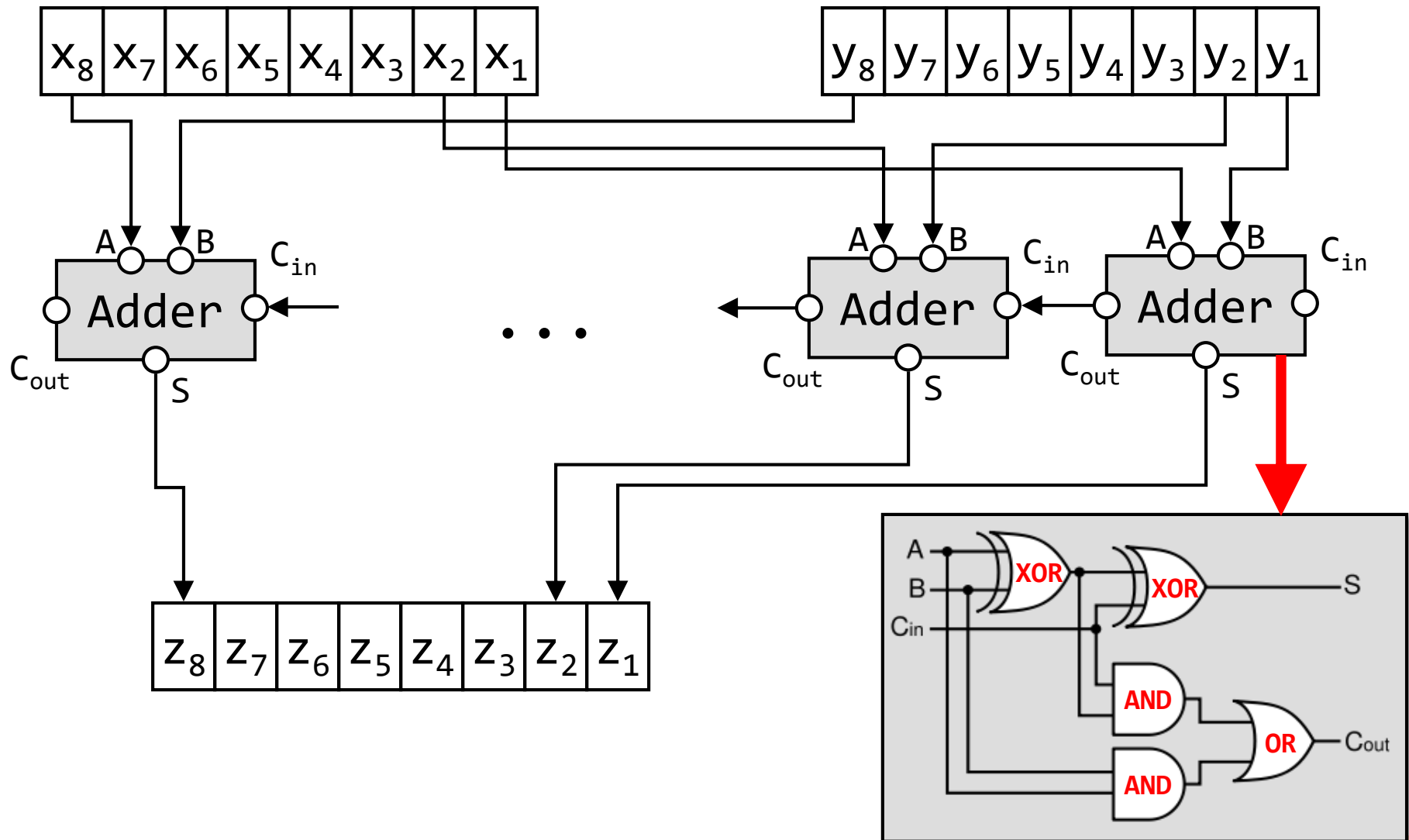


Image taken from Wikipedia

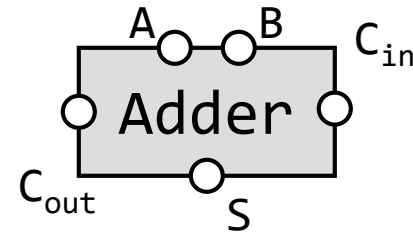
# Encoding “Adder” as a SAT instance

0x!  $((A \ \& \ B \ \& \ C_{in}) \Rightarrow (S \ \& \ C_{out})) \ \&$

1x! {  $((!A \ \& \ B \ \& \ C_{in}) \Rightarrow (!S \ \& \ C_{out})) \ \&$   
 $((A \ \& \ !B \ \& \ C_{in}) \Rightarrow (!S \ \& \ C_{out})) \ \&$   
 $((A \ \& \ B \ \& \ !C_{in}) \Rightarrow (!S \ \& \ C_{out})) \ \&$

2x! {  $((!A \ \& \ !B \ \& \ C_{in}) \Rightarrow (S \ \& \ !C_{out})) \ \&$   
 $((!A \ \& \ B \ \& \ !C_{in}) \Rightarrow (S \ \& \ !C_{out})) \ \&$   
 $((A \ \& \ !B \ \& \ !C_{in}) \Rightarrow (S \ \& \ !C_{out})) \ \&$

3x!  $((!A \ \& \ !B \ \& \ !C_{in}) \Rightarrow (!S \ \& \ !C_{out}))$



- Bounded model checker for program in C/C++
- Developed at Oxford & Carnegie Mellon Uni
- <http://www.cprover.org/cbmc/>
- Source code and binaries freely available
  - Platforms: Windows, Linux, Mac OS

# CBMC: how to use it

- Download
  - <http://www.cprover.org/cbmc/download/cbmc-5-4-win.zip>
- Run from the Visual Studio Command Prompt
  - Why: correctly initialized environment
- Examples
  - <http://d3s.mff.cuni.cz/files/teaching/nswi132/files/bmc-examples.zip>

# CBMC: example 1

- Q: Exists an integer value  $x$  such that  $x \neq 0$  and  $x == -x$  ?
- Source code: `ex01-ints.c`
  - Q: Is the program safe or not ?
- Find the answer using CBMC
  - `cbmc64 bmc-examples\ex01-ints.c`

# CBMC: example 2

- Program `ex02-loops.c`
  - Loop with bounded number of iterations
- Command line argument “-function”
  - Specifies an entry point
- Usage
  - `cbmc64 ex02-loops.c -function sum`



# CBMC: example 3

- Program `ex03-fact.c`
  - Unbounded loop
  - Infinite unwinding
- Command line argument “`--unwind N`”
- Argument “`--unwinding-assertions`”

# CBMC: example 4

- Program `ex04-binsearch.c`
  - Loop bound cannot be determined statically
- Supported built-in properties
  - Checking bounds for accesses to array elements
    - Parameter “--bounds-check”
  - Checking null dereferences
    - Parameter “--pointer-check”