Verification, Testing, Security Analysis

Dr. Oleksandr Letychevskyi

LitSoft

Dr. Oleksandr Letychevskyi

Dr. Volodymyr Peschanenko

Prof. Alexander Letichevsky

Letichevsky Criterion

First parallel OS with distributed memory

80 Automatic Theorem Prover

90 Algebraic Programming System

Insertion Modeling System

2000 Verification of Requirements Specifications

Verification of Parallel Programs

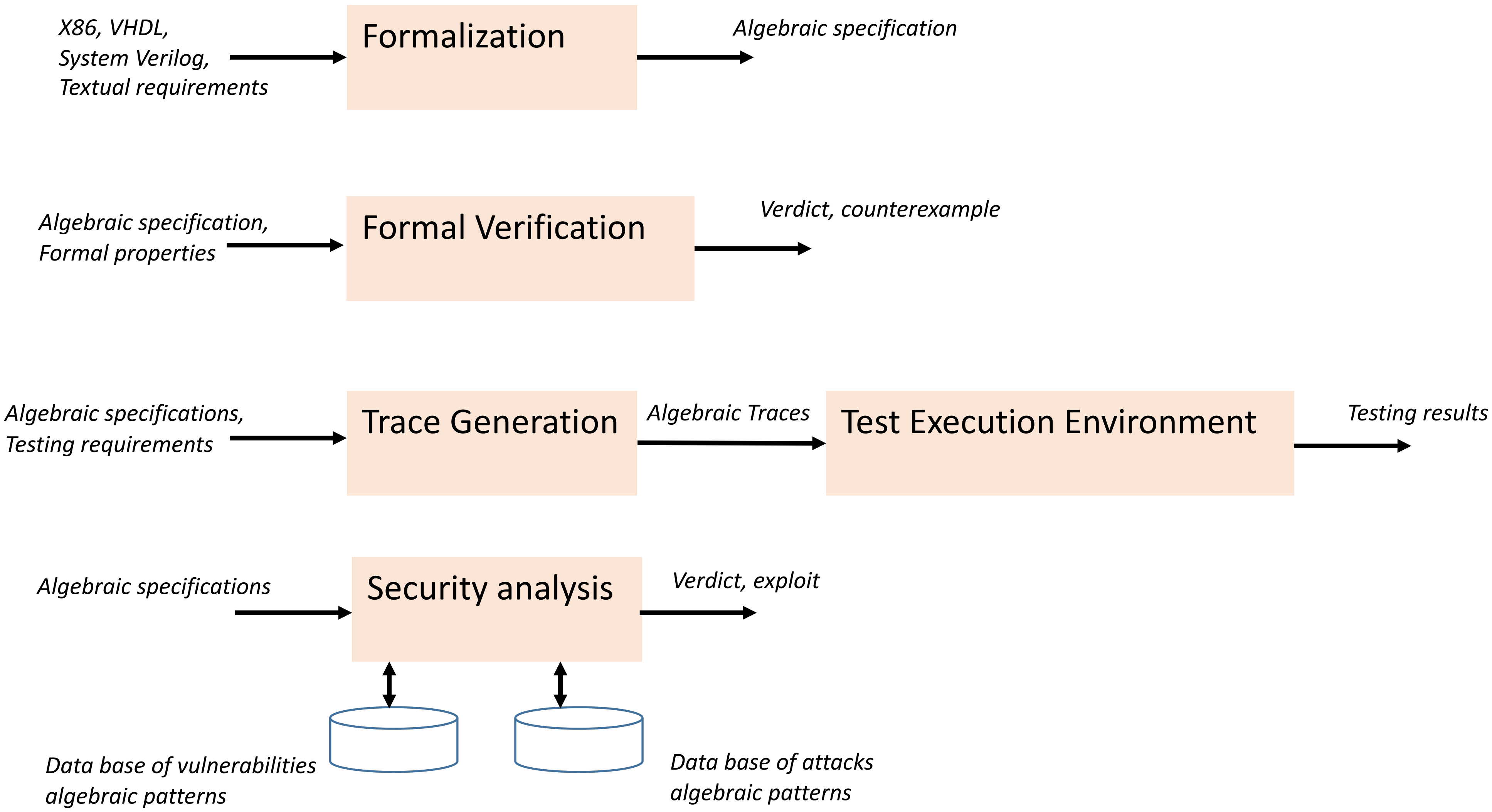
2010 Model-Based Testing

Reverse Engineering

Cyber Security

Our approach:

Create algebraic model of hardware and use the algebraic methods for verification, model-based testing, security analysis



Algebra of Behaviors

Algebra of behaviors was developed by D. Gilbert and A. Letichevsky (Senior) in 1997
It consider the operations over **actions** and **behaviors**.

Prefixing operation $a.B$ means that action a follows behaviour B. The operation of *nondeterministic choice* of behaviours $u + v$ establishes alternative behaviours. The algebra has three terminal constants: successful termination Δ, deadlock 0, and unknown behaviour $\perp$. The parallel and sequential composition are defined on the behaviors.

Example:

B0 = a1.a2.B1 + a3.B2,
B1 = a4.Δ,
B2 =…

The example define the order of event. The behavior B0 has two alternative - first is two actions a1 and a2 and then the rest behavior B1 or action a3 and rest behavior B2. Behavior B1 is action a4 and end of behavior etc.

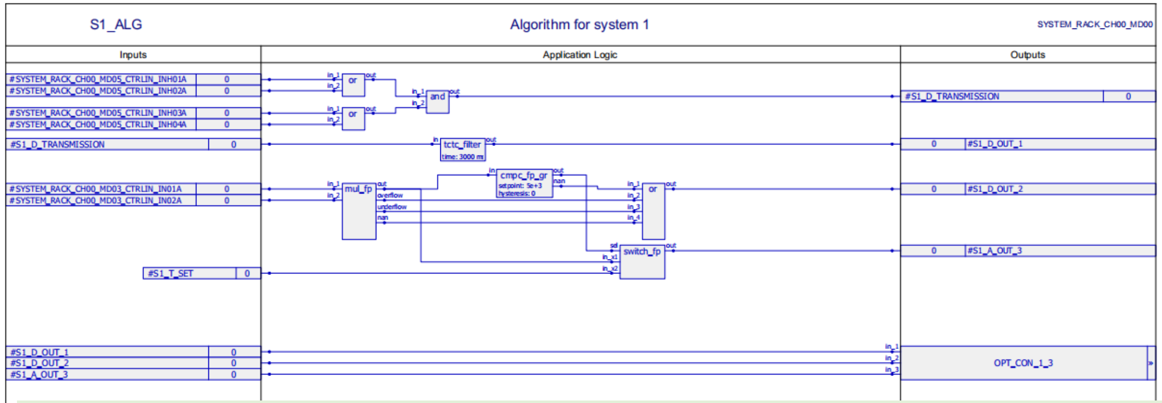
Algebra of Behaviors

Every action is also defined by a couple, namely, the **precondition** and **postcondition** of an action, given as an expression in some formal theory.

Action(A,B) = (A > B) && !(A == 0) -> B = (B + 1)/A

The semantic of the action presented in C-like syntax means that if precondition *(A > B) && !(A == 0)* is true for concrete values of *A* and *B* or is satisfiable for symbolic (arbitrary) values of *A* and *B*, then we can change attribute B by the assignment *B = (B + 1)/A*. The action can be parametrized by the attributes used in the action's conditions.

Algebraic Specifications of FPGA code




```
Aor2(s, x1, x2, x3) = (in_1, in_2, out : bool) 1->
<receive(x1: signal (in_1), receive(x2: signal (in_2), send(y : signal(out))>
out = in_1 || in_2,
Aand(s, x1, x2, y) = (in_1,in_2,out : bool) 1->
<receive(x1: signal (in_1), receive(x2: signal (in_2),send(y: signal(out))>
out = in_1 && in_2,
Acmp_fp_eq(s,x1,x2,y0,y1,y2,y3,y4,deadband:real) = (in:real, set:real,
out,overflow,underflow,nan:bool)
deadband >= 0 ->
<receive(x1 : signal(in)), receive(x2 : signal(set)),
send(y0:signal(out)),send(y1:signal(out)), send(y2:signal(overflow)),
send(y3:signal(underflow)), send(y4:signal(nan))>
out = (in >= set) && (in – set <= deadband/2) ||(in <= set) && (set – in <=
deadband/2);
overflow = (in >= set) && (in – set > maxReal32) || (in <= set) && (set – in >
maxReal32);
underflow = (in >= set) && (in – set < minReal32) || (in <= set) && (set – in <
minReal32)
```

```
S1_ALG = P11 || P12 || P13,
P11 = (
(A_from_MATS2(or2_11)|| A_from_MATS2(or2_12));
  (
    Aor2(or2_11, MATS, MATS, and1) ||
    Aor2(or2_12, MATS, MATS, and1)
  );
  Aand (and1, or2_11, or2_12, tctc_filter_1)
 ),
P12 = Btctc_filter(tctc_filter_1, and1, xor_1, 3000),
P13 = (
(A_from_MATS2(mul_fp_1)|| A_from_MATS2(switch_fp_1));
  Amul_fp(mul_fp_1, MATS, MATS, cmpc_fp_gr_1, switch_fp_1, or4_1, or4_1, or4_1);
  Acmpc_fp_gr (cmpc_fp_gr_1, mul_fp_1, switch_fp_1, or4_1, 5000, 0);
  (
    Aor4(or4_1, cmpc_fp_gr_1, mul_fp_1, mul_fp_1, mul_fp_1, xor_2) ||
    Aswitch_fp(switch_fp_1, cmpc_fp_gr_1, mul_fp_1, MATS, cmp_fp_eq)
  )
 )
```

Translation of x86 Assembler to Algebra of Behaviors

Set of Instructions is converted to

```
0000000000425060 <SSL_CTX_use_certificate_file>:
  425060:   41 55                   push   r13
  425062:   41 54                   push   r12
  425064:   49 89 f5                mov    r13,rsi
  425067:   55                      push   rbp
  425068:   53                      push   rbx
  425069:   49 89 fc                mov    r12,rdi
  42506c:   89 d5                   mov    ebp,edx
  42506e:   48 83 ec 08             sub    rsp,0x8
  425072:   e8 d9 24 fe ff          call   407550 <BIO_s_file@plt>
  425077:   48 89 c7                mov    rdi,rax
  42507a:   e8 a1 31 fe ff          call   408220 <BIO_new@plt>
  42507f:   48 85 c0                test   rax,rax
  425082:   0f 84 b0 00 00 00       je     425138
<SSL_CTX_use_certificate_file+0xd8>
  425088:   4c 89 e9                mov    rcx,r13
```

Set of Algebra Behavior Expressions

```
B425060 = a_push_33766.B425062,
B425062 = a_push_33767.B425064,
B425064 = a_mov_33768.B425067,
B425067 = a_push_33769.B425068,
B425068 = a_push_33770.B425069,
B425069 = a_mov_33771.B42506c,
B42506c = a_mov_33772.B42506e,
B42506e = a_sub_33773.B425072,
B425072 = a_call_33774.call B407550.B425077,
B425077 = a_mov_33775.B42507a,
B42507a = a_call_33776.call B408220.B42507f,
B42507f = a_test_33777.B425082,
B425082 = a_je_33778.B425138 + a_alt_je_33779.B425088,
B425088 = a_mov_33780.B42508b,
```

Set of Algebra Behavior Actions

```
a_push_33766 = Operator(1  -> ("x86: action 'push 425060';")
(rip := 4345954)),
a_push_33767 = Operator(1  -> ("x86: action 'push 425062';")
(rip := 4345956)),
a_mov_33768 = Operator(1  -> ("x86: action 'mov 425064';")
(rip := 4345959; r13 := rsi)),
a_push_33769 = Operator(1  -> ("x86: action 'push 425067';")
(rip := 4345960)),
a_push_33770 = Operator(1  -> ("x86: action 'push 425068';")
(rip := 4345961)),
a_mov_33771 = Operator(1  -> ("x86: action 'mov 425069';")
(rip := 4345964; r12 := rdi)),
a_mov_33772 = Operator(1  -> ("x86: action 'mov 42506c';")
(rip := 4345966; ebp := edx)),
a_sub_33773 = Operator(1  -> ("x86: action 'sub 42506e';")
(rip := 4345970; rsp := rsp - 8; ZF := (rsp - 8 = 0); PF :=
((rsp - 8) = 0); SF := (rsp - 8) < 0))),
a_call_33774 = Operator(1  -> ("x86: action 'call 425072';")
(rip := 4345975)),
a_mov_33775 = Operator(1  -> ("x86: action 'mov 425077';")
(rip := 4345978; rdi := rax)),
a_call_33776 = Operator(1  -> ("x86: action 'call 42507a';")
(rip := 4345983)),
a_test_33777 = Operator(1  -> ("x86: action 'test 42507f';")
(rip := 4345986)),
a_je_33778 = Operator((ZF = 1)  -> ("x86: action 'je
425082';")(rip := 4345992)),
a_alt_je_33779 = Operator(( ~(ZF = 1))  -> ("x86: action 'je
425082';")(rip := 4345992)),
a_mov_33780 = Operator(1  -> ("x86: action 'mov 425088';")
(rip := 4345995; rcx := r13)),
```

Formal Verification

Methods:

- Symbolic execution of model (forward, backward)
- Static proving
- Invariant generation
- Algebraic matching

Properties:

- Inconsistency (non-determinisms)
- Incompleteness (deadlocks)
- Timing properties
- Signal races
- Starvation
- Synchronization issues
- Safety
- Liveness
- …

$\bigcap a_i = 0$

Conjunction of precondition

$\bigcup a_i = 1$

Disjunction of precondition

T1 <= T(i) <= T2

Post(Env, A) = true

Reachability of starvation state

SECURITY ANALYSIS

Methods:
Algebraic matching,
Fuzzing,
Symbolic modeling,
Resolving of behavior algebra expressions,
Machine learning

Vulnerabilities:
SW: CVE/ CWE
Hardware: False switching-on
Overflow, underflow
…

MODEL-BASED TESTING

Methods:
Backward and Forward
Symbolic modeling

CHECKING OF EQUIVALENCY

Methods:
Backward and Forward
Symbolic modeling,
Transformations,
Behavior Reasoning