Algebraic Methods in the Analysis of Attack Resistance in Decentralized Systems

Dr. Oleksandr Letychevsky

Glushkov Institute of Cybernetics NAS of Ukraine,

Our Algebraic Approach

- Glushkov's algebraic school is known from early 70-th: automated theorem proving;
- Algebraic Programing System (1990): symbolic computations, rewriting rules technique;
- Insertion Modeling (2000): generalization of transition systems interaction, theory of agents and environments;
- Behavior Algebra (2010) – symbolic modeling, predicate transformers theory, algebraic matching.

Behavior Algebra. Behavior

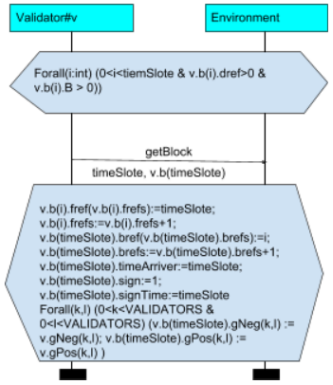
- Within the scope of the insertion modeling method we use the *behavior algebra* specifications for the formalization of the project introduced in 1997, by D. Gilbert(UK) and A. Letichevsky Senior (Ukraine).
- The operations are the *prefixing a.u* (where *a* is an action and *u* is a behavior) and *non-deterministic choice* of behaviors *u* + *v*. The terminal constants are successful termination Δ, deadlock 0, and unknown behavior ⊥. The approximation relation ⊑ is a partial order on the set of behaviors.
- *B0 = a1.a2.B1 + a3.B2, B1 = a4, B2 =…*
- The behavior algebra is also enriched by two operations: parallel (||) and sequential (;) compositions of behaviors.

Behavior Algebra. Actions.

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

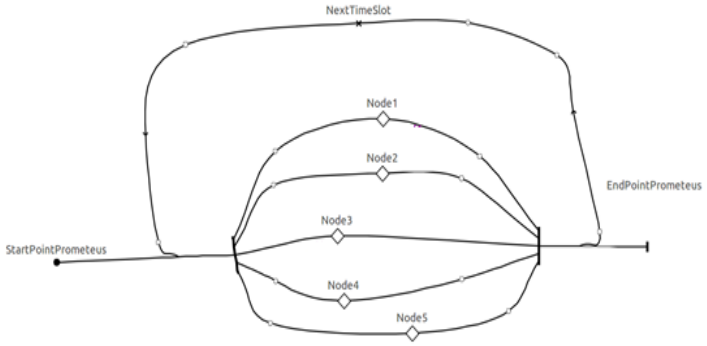
Action(A,B) = (A > B) && !(A == 0) -> B = (B + 1)/A

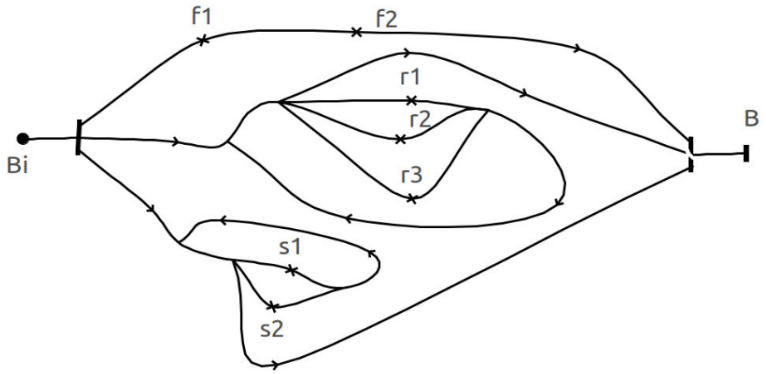
The semantic of the action presented in C-like syntax means that if precondition *(A > B) && !(A == 0)* is true for concrete values of *A* and *B* or is satisfiable for symbolic (arbitrary) values of *A* and *B*, then we can change attribute B by the assignment *B = (B + 1)/A*. The action can be parametrized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



Action presented as MSC diagram. The transition between two states can be illustrated by MSC statement (message sending, local actions).

Example of Consensus Protocol





- The nodes work concurrently and in terms of insertion modeling, their high-level behavior *B0* can be presented by the following behavior algebra expressions:

$B0 = B;B0 + \Delta,$

$B = NextTimeSlot.(\; // \; Bi, \; 1<=i<=n \;)$

$Bi = F \; || \; R \; || \; S \; || \; t,$

$F = f1.f2.f3.B,$

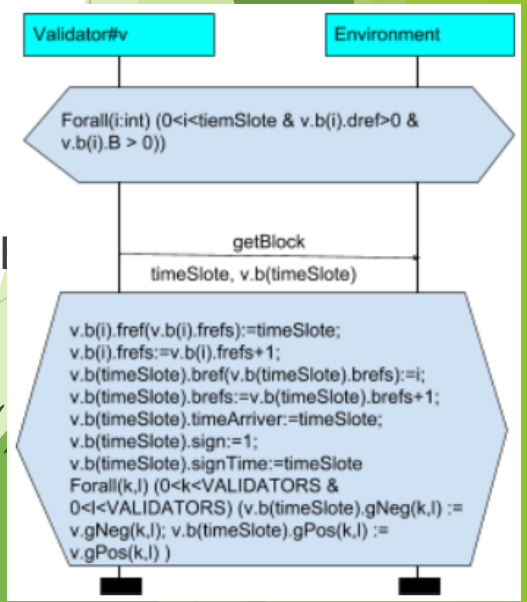
$R = r1.f2.R + r2.R + r3.R + B,$

$S = s1.S + s2.S + B$

Example of Consensus Protocol

- During time slot, each validator can perform the following concurrent actions:
- generates block during predefined time slot *(f1)*, creates references for other blocks*(f2)*, and sends block *(f3)*;
- receives blocks *(r1)* and sends/receives positive gossip messages about the receiving of missed blocks *(r2/r3)*;
- sends/receives negative gossip messages about missed blocks in the previous time slot *(s1/s2)*;
- sends transaction to the pool of unconfirmed transactions *(t)*

f1 = (timeslot == BN) -> p(f1) (BN = BN + 1, NODE(i).BB = BN+1),
f2 = 1 -> p(f2) (NODE(i).REF_BLOCK(BN) = NODE(i).HANG(NODE(i).MAX);),
f3 = 1-> p(f3) Forall (1<=j<=N && !(j==i)) NODE(j): send(b, NODE(i).REF_BLOCK(b)> 1),
r1 = ~(timeslot == BN) -> p(r1) NODE(i).REF_BLOCK(x) = y; NODE(i).BC = x; NODE(i).I
r2 = 1 -> p(r2) Forall (1<=j<=N && !(j==i)) NODE(j): *(SendPosGossip())*,
r3 = 1 -> p(r3) Forall (1<=j<=N && !(j==i)) NODE(j): *(ReceivePosGossip ())*,
s1 = BlockMissed() -> p(s1) Forall (1<=j<=N && !(j==i)) NODE(j): *SendNegGossip(*
s2 = 1 -> p(s2) Forall (1<=j<=N && !(j==i)) NODE(j): *ReceiveNegGossip ()*,
t(x) = 1-> p(t) putTransactionPool(x)



Double-Spending Attack

- Pattern of attack

Behavior:

B = Y;(t(A).X1);(t(A).C), //creation of two transactions with the same ID = A

C = X2;(f1(A).f2(r).X3.BB(1) || f1(A).f2(r).X4.BB(2)), //creation of alternative chain

BB(x) = (f1.f2.Z);BB(x) //continuation of the corresponding chain

X1,X2,X3,X4,Y,Z – arbitrary behaviors

Actions of patterns contain information on the attack possibility f. e. length of alternate chain exceed the critical value.

Resolving of Behavior Equations

- Behavior Equation:

B = NextTimeSlot.(|| Bi, 1<=i<=n)

Bi = F || R || S || t,

F = f1.f2(y).f3.B,

R = r1.f2.R + r2.R + r3.R + B,

S = s1.S + s2.S + B

- Unknown behavior XX:

XX = Y;(t(A).X1);(t(A).C),

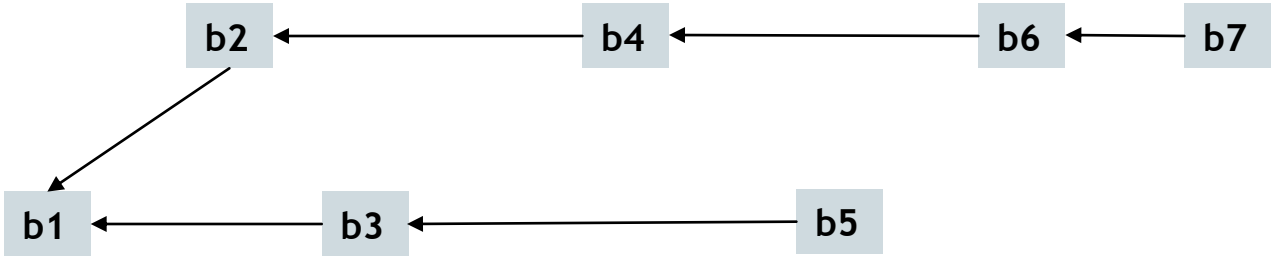
C = X2;(f1(A).f2(r). X3.BB(1) || f1(A).f2(r). X4.BB(2)),

BB(x) = (f1.f2.Z);BB(x)

- Resolution:

(yes (the concrete scenario or behavior tree)/no/unknown)

Resolving of Behavior Equations



Case: *t(A).t(A).f1(A).f2(0).f3. f1(A).f2(1). f1.f2(1).f3.s1.s2. f1.f3. [r1.f2(1).r2].f2(2).f3. f1.f2(3).f3. f1.f2(4).f3.f1.f2(6).f3*

Timeslot 1: t(A).t(A).f1(A).f2(0).f3.

Timeslot 2: f1(A).f2(1). // Block b2 was not sent in its time-slot

Timeslot 3: f1.f2(1).f3.s1.s2. //b3 created forking (f2(1))

Timeslot 4: f1.f3. [r1.f2(1).r2].f2(2).f3. // Block b2 was sent in timeslot 4 (f3) and received by other nodes [r1.f2(1).r2]

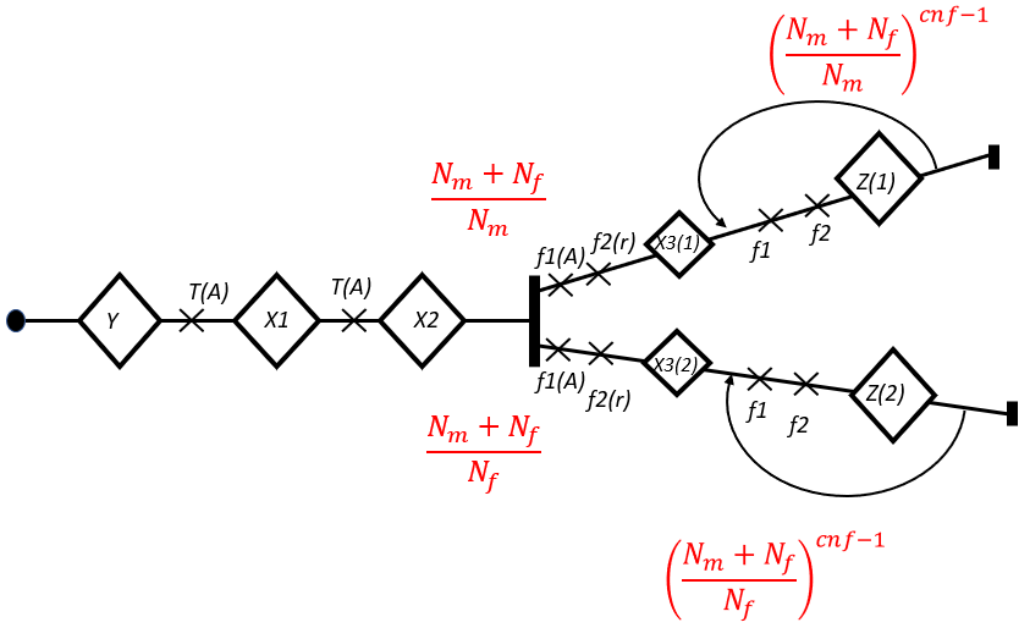
Timeslot 5: f1.f2(3).f3

Timeslot 6: f1.f2(4).f3

Timeslot 7: f1.f2(6).f3

Resolving of Behavior Equations

- Stage 1. Algebraic matching. Rewriting rules technique (apsystems.org.ua)
- Stage 2. Symbolic modelling. Satisfiability of precondition and environment. Predicate transformers theory. Works with Z3 Microsoft prover.
- Stage 3. Counterexample generation. Backward symbolic modeling.
- Stage 4. Probability evaluation.



Translation of Solidity to Behavior Algebra

```
contract SimpleDAO {
mapping (address => uint) public
credit;



function donate(address to) {
   credit[to] += msg.value;
}



function withdraw(uint amount) {
   if (credit[msg.sender]>= amount) {
      msg.sender.call.value(amount)();
      credit[msg.sender]-=amount;
   }
  }



function queryCredit(address to)
returns (uint){
   return credit[to];
  }
}
```

BDAO(x,value) = donate(x,value) || WITHDRAW(x, value),
WITHDRAW(x, value) = withdraw1(x,amount). fallback.withdraw2(x,amount)

```
donate(x,value) = SimpleDAO:1->
   <receive ether(x,value)>
   credit(x) = credit(x) + value,
```

withdraw1(x,amount) = SimpleDAO:(credit(x)>=amount)-> <call(withdraw,x,amount)),send ether(x,amount)> 1

withdraw2(x, amount) = SimpleDAO:1 -><>
credit(x) = credit(x) - amount

queryCredit(x)=SimpleDAO:1-><call(queryCredit,x), send ether(credit(x))> 1

fallback = empty + fallbackAction

Reentrancy Attack

- Pattern of Attack

DAOAttackBehavior = MaliciousCycle,

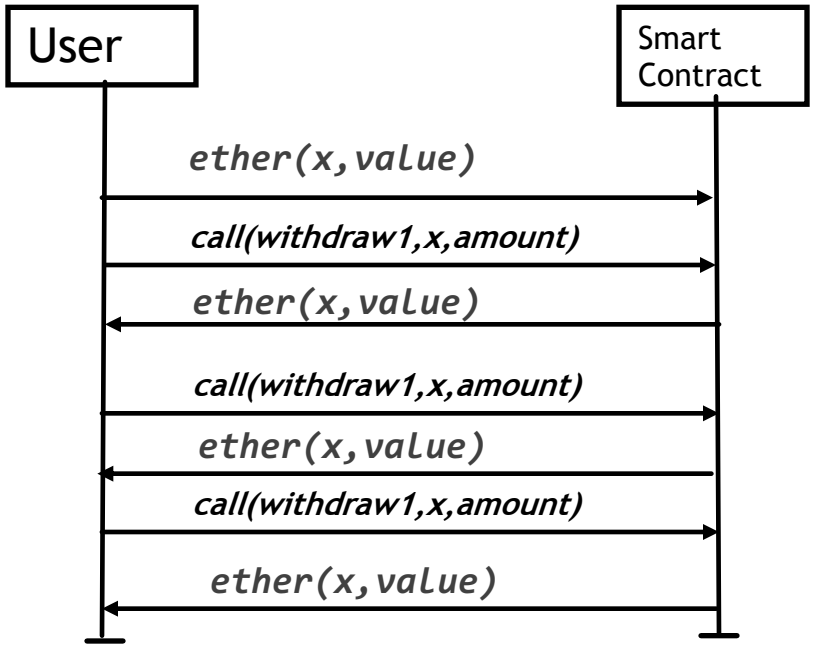
MaliciousCycle=withdraw1(x,amount).X.MaliciousCycle

withdraw1(x,amount) = (OLD_CREDIT == credit(x)) -> <> (OLD_CREDIT = credit(x))

- Resolution:

donate.withdraw1. withdraw1.withdraw1

- Trace:



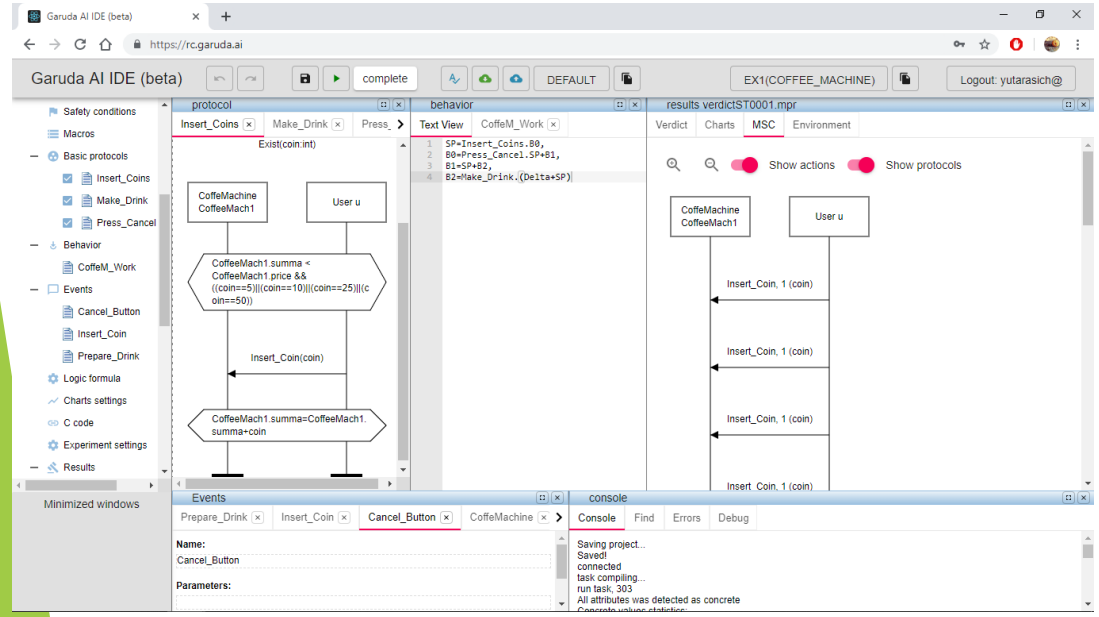
Symbolic Computation Platform for Attacks and Vulnerabilities Detection

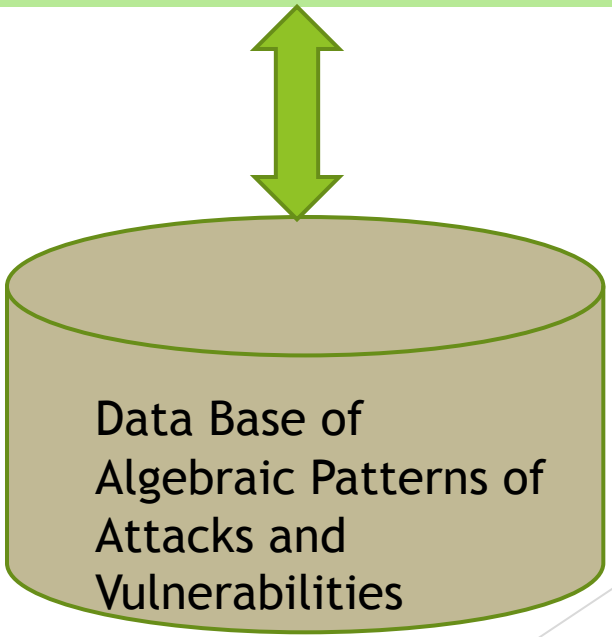
Source Code

Creation of model, conversion to algebraic behavior

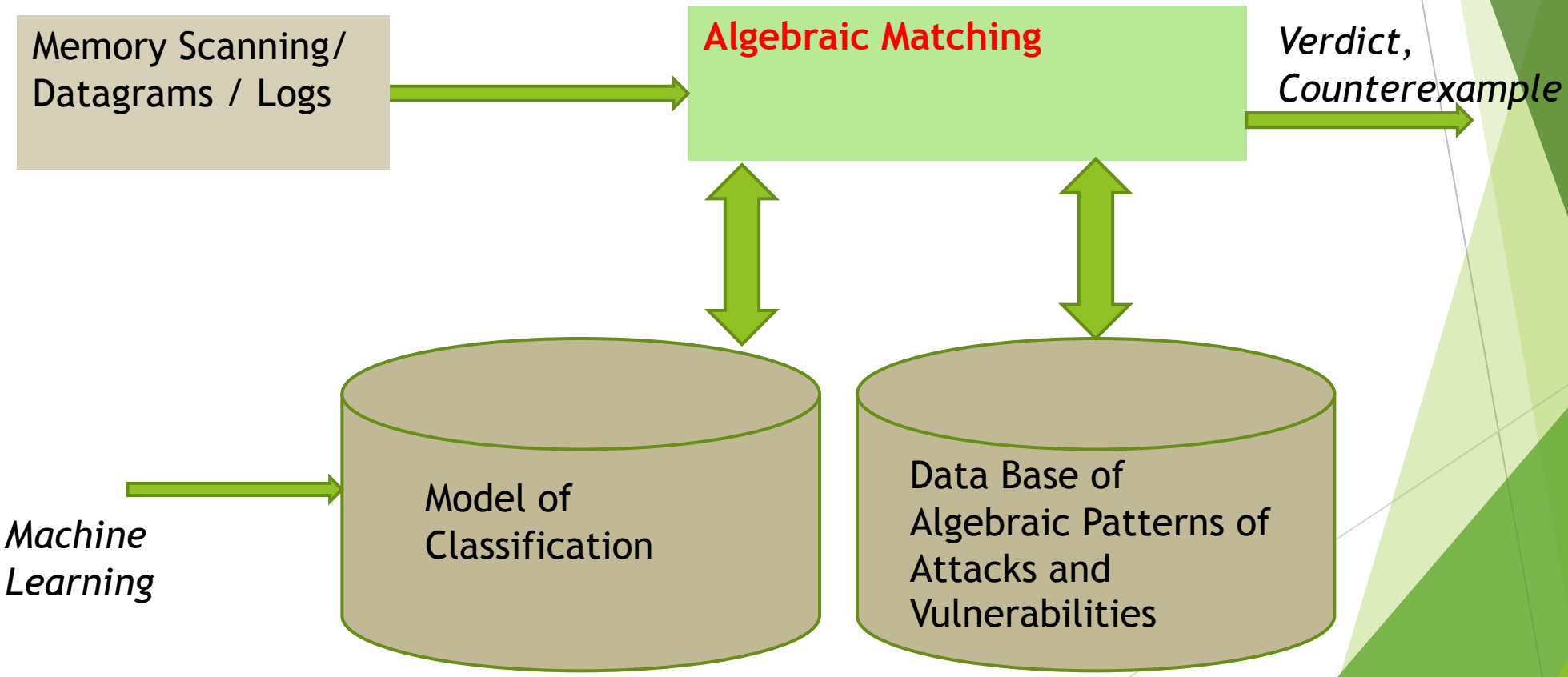
Algebraic Matching:
- Behavior matching;
- Symbolic modeling;
- Exploit generation

Verdict, Counterexample





Symbolic Computation Platform for Attacks and Vulnerabilities Detection



Thanks for attention

Contact me:

oleksandr.letychevskyi@litsoft.com.ua

www.litsoft.com.ua

LitSoft