Алгебра поведінок та її застосування

Олександр Летичевський
д. ф.-м. наук
відділ теоорії цифрових автоматів
Інституту кібернетики ім. В.М.Глушкова НАН України

Алгебрачна теорія взаємодії та інсерційне моделювання

Алгебраїчна теорія взаємодії сформувалася в 90-і роки як один із напрямів в загальній теорії взаємодії.

- D.R. Gilbert, A.A. Letichevsky, A universal interpreter for nondeterministic concurrent programming languages, in: M. Gabbrielli (Ed.), Fifth Compulog network area meeting on language design and semantic analysis methods, September 1996.
- A. Letichevsky and D. Gilbert. A general theory of action languages. Кибернетика и Системный Анализ, 1, 1998, 16-36, 192.
- A. Letichevsky and D. Gilbert, Interaction of agents and environments, in: *Resent trends in Algebraic Development technique, LNCS 1827* (D. Bert and C. Choppy, eds.), Springer-Verlag, 1999.

Агенти та середовища

Світ є ієрархія середовищ і агентів, занурених у ці середовища.

- 1. Агенти і середовища є сутності, що еволюціонують у часі і мають спостережувану поведінку
- 2. Занурення агента в середовище змінює поведінку цього середовища і породжує нове середовище, готове до занурення в неї нових агентів.
- 3. Середовище, що розглядається як агент, може бути занурене в середовище верхнього рівня
- 4. Нові агенти можуть занурюватися в середу, переміщаючись з середовища верхнього рівня, а також породжуватися внутрішніми агентами, вже зануреними в середовище раніше
- 5. Агенти і середовища можуть моделювати на різних рівнях абстракції зовнішнє середовище, в якій вони перебувають і інших агентів, занурених у це середовище

Алгебра поведінок

- Операції алгебри поведінок (U,A)
  - Префіксінг: B = a.B1, $a \in A$ (дії), $B \in U$, $B1 \in U$ (поведінки)
  - Недетермінований вибір: B = a1.B1 + a2.B2, a1,a2 $\in$ A, B,B1,B2$\in$U
  - Термінальні константи: $\Delta, \perp, 0$

    (Успішне закінчення, невизначений елемент та тупік)
  - Відношення апроксімації

    $B1 \sqsubseteq B2$
  - Послідовна та паралельна композиція поведінок

    B = B1;B2

    B = B1 || B2

Агенти та середовища

Ми розглядаємо агентів, як розмічену транзиційну систему

$$s \xrightarrow{a} s'$$

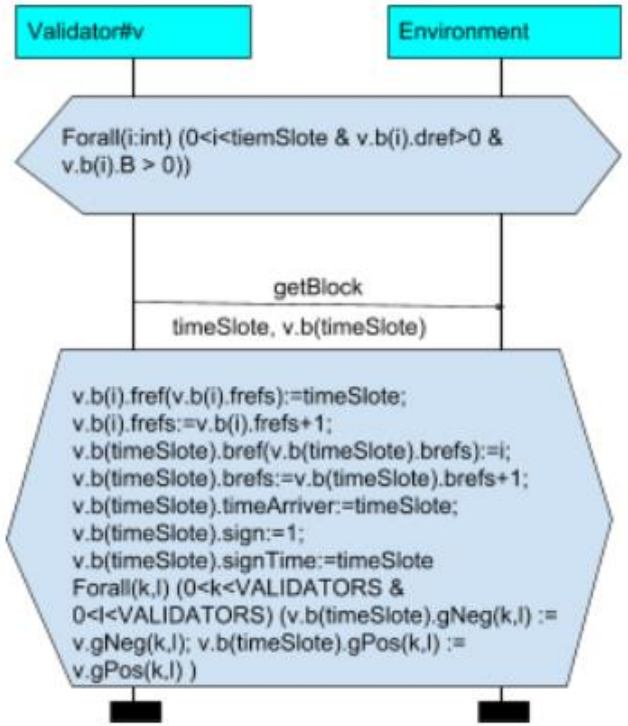
станами якої є формули, а розміткою є дії.

Ми розглядаємо формули над типизованим атрибутним середовищем. Типи атрибутів: цілочисельні, ірраціональні, символьні, булеві, перелічувані, списки, черги тощо. Розглядаються функціональні структури даних над цими типами.

Семантика дій

- Кожна дія визначається парою (P,R), де P – передумова, а R – післяумова. P – предикат в базовій мові атрибутного середовиища, R – оператори зміни середовища (присвоювання) та предикат в базовій мові.
- Іноді дія визначається трійкою (P,S,R), де S – процесна компонента, що ілюструє дану дію.

Дія може бути представлена MSC – діаграмою. В якості процесної компоненти використовують пересилання або отримання повідомлення



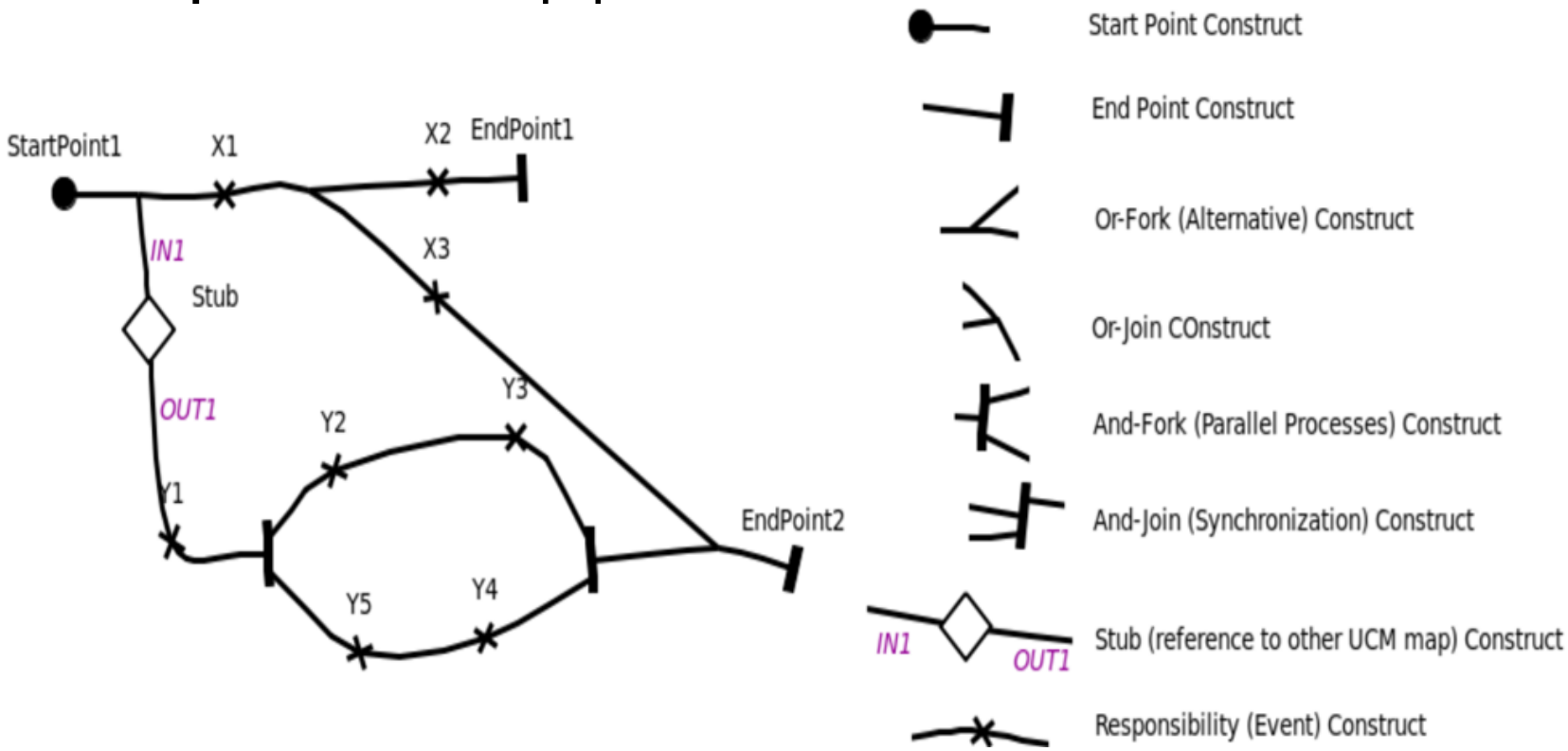
Символьне моделювання

- Початковий стан агенту визначається початковою формулою на атрибутним середовищем S0. Починаючи з цієї формули ми застосовуємо дії згідно з поведінкою, починаючи із початкової поведінки.
- Дія застосовна, якщо S0 /\ Pa1 виконувана, де Pa1 – передумова дії a1 і B0 = a1.B1.
- Наступний стан отримуємо за допомогою ПРЕДИКАТНОГО ПЕРЕТВОРЮВАЧА, що є функцією від поточного стану середовища, передумови та післяумови..

$$PT(Si, Pai, Qai) = Si+1$$

- Застосовуючи предикатний перетворювач до дій, відповідно до поведінки, ми отримуємо послідовність S0, S1,… формул, що представляють зміну станів з початкового та що відповідає послідовності дій a1, a2,… Даний процес називаємо СИМВОЛЬНИМ МОДЕЛЮВАННЯМ ПОВЕДІНКИ

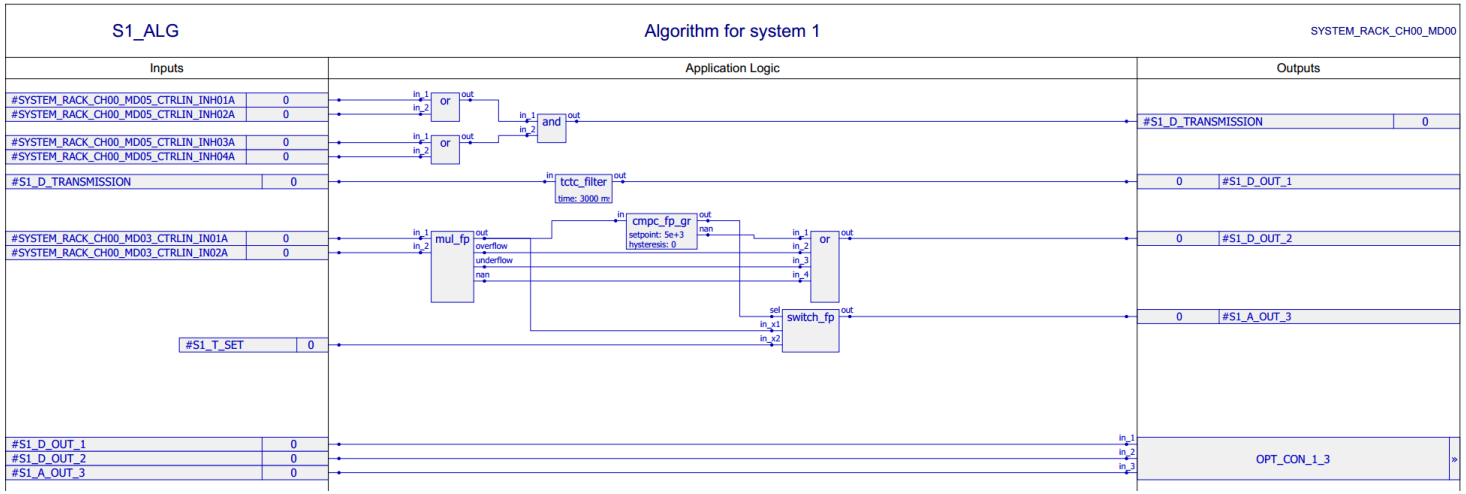
Алгебра поведінок



Алгебра поведінок має аналог в графічних специфікаціях UCM (Use Case Maps).

Конструкти UCM відповідають операціям алгебри поведінок та можуть слугувати як графічне відображення поведінки.

Вимоги та рівняння алгебри поведінок



ДІЇ:

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,
Acmpc_fp_gr(s, x1, x2, y, setpoint:real, hysteresis:real)
= (in:real, out:bool, nan:bool) 1 ->
<receive(x1 : signal(in)), send(x2 : signal (out)), signal (y : nan)>
out = (in > setpoint – hysteresis),
…

ПОВЕДІНКА:

S1_ALG = P11 || P12 || P13,
P11 = ((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 = (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)))

Вимоги та рівняння алгебри поведінок

8.9 MM Procedures for Successful Inter-CBSC CDMA to CDMA Hard Handoff Execution - Target Side

8.9.1 MM Receives A1: Handoff Request message

This following requirements define the inter-CBSC CDMA hard handoff target execution procedure. This procedure is executed as a result of receiving an A1: Handoff Request message from the MSC. MM validates the handoff request and determines whether the call should be directed towards SDU-SDF or the XC subsystem.

--this should be a parsing req.

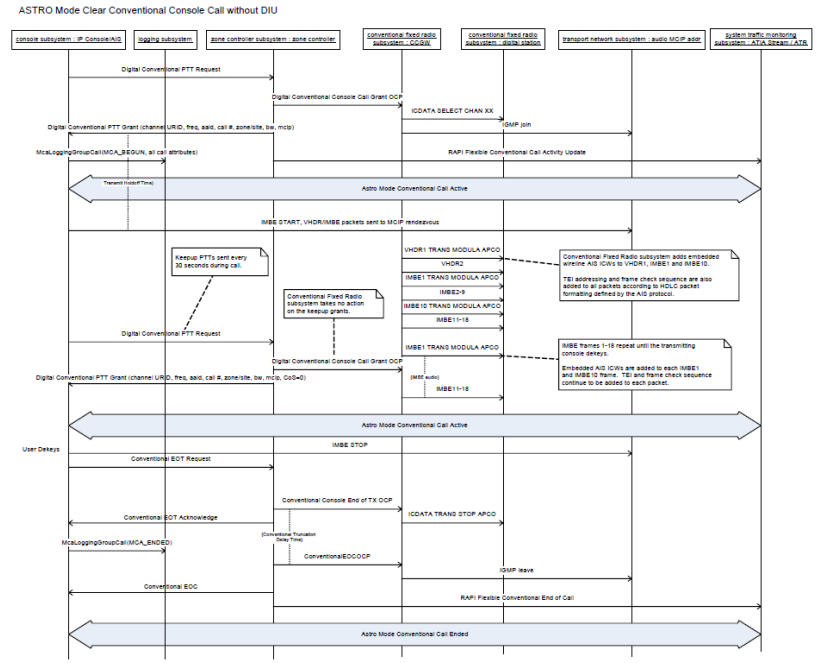
CP-HOPC.R.63286 CPSFS-MMHHOTGT-1730 | CI: DNP-MM
4813A, 6088 | Tier: 1 - Top Level
Req: [63286] Upon receipt of an A1: Handoff Request message from the MSC, th… MM shall parse the message per section 8.9.23.1, "MM Parses A1: Handoff Request Mess

…OPC.R.61934 CPSFS-MMHHOTGT-2 | CI: DNP-MM
6088, 7037 | Tier: 2
Req: [61934] BAND_CLASS shall be retained for the duration of the call. The … BAND_CLASS shall be set to the value of the environmental variable BandClassOv… it exists, otherwise BAND_CLASS shall be set to the parameter DB:BTS.BandClas… ated with the first CBSC-provisioned cell within the Cell Identifier List of the A1: … Request.
Prior Legacy No. : CPSFS-HHOTGTXC-183 SFS-HOPC-C8.1-M-D2D-34.a.4-1

…OPC.R.70416 CP-HOPC.R.70416 | CI: DNP-MM
6088, 7037, INT_NC | Tier: 2
Req [70416]: If the "See List of Entries" field within the Classmark Information … element of the A1: Handoff Request Message is "1", and none of the Band Class N… within the same element matches the target CBSC Band Class as specified in Req: … then the MM shall fail the handoff per the requirements at Req: [5209].

Table 12-3—Applicant state table

		STATE										
		VA	AA	QA	LA	VP	AP	QP	VO	AO	QO	LO
EVENT	transmitPDU!	sJ[E,I] AA	sJ[E,I] QA	-x-	sLE VO	sJ[E,I] AA	sJ[E,I] QA	-x-	-x-	-x-	-x-	sE VO
	rJoinIn	AA	QA	QA	LA	AP	QP	QP	AO	QO	QO	AO
	rJoinEmpty	VA	VA	VA	VO	VP	VP	VP	VO	VO	VO	VO
	rEmpty	VA	VA	VA	LA	VP	VP	VP	VO	VO	VO	VO
	rLeaveIn	VA	VA	VP	LA	VP	VP	VP	LO	LO	LO	VO
	rLeaveEmpty	VP	VP	VP	VO	VP	VP	VP	LO	LO	LO	VO
	LeaveAll	VP	VP	VP	VO	VP	VP	VP	LO	LO	LO	VO
	ReqJoin	-x-	-x-	-x-	VA	-x-	-x-	-x-	VP	AP	QP	VP
	ReqLeave	LA	LA	LA	-x-	VO	AO	QO	-x-	-x-	-x-	-x-
	Initialize	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO	VO



Формальна верифікація

- Досяжність властивості F(X) над атрибутним середовищем X.

Існує така поведінка, що E∧F(X) – істино, E – стан середовища .

- Статичне визначення властивості
  - Статичне доведення неповноти та несуперечливості

$$\bigcap a_i = 0$$

$$\bigcup a_i = 1$$

a_i – передумови дій при недетермінованому виборі у виразах алгебри поведінок

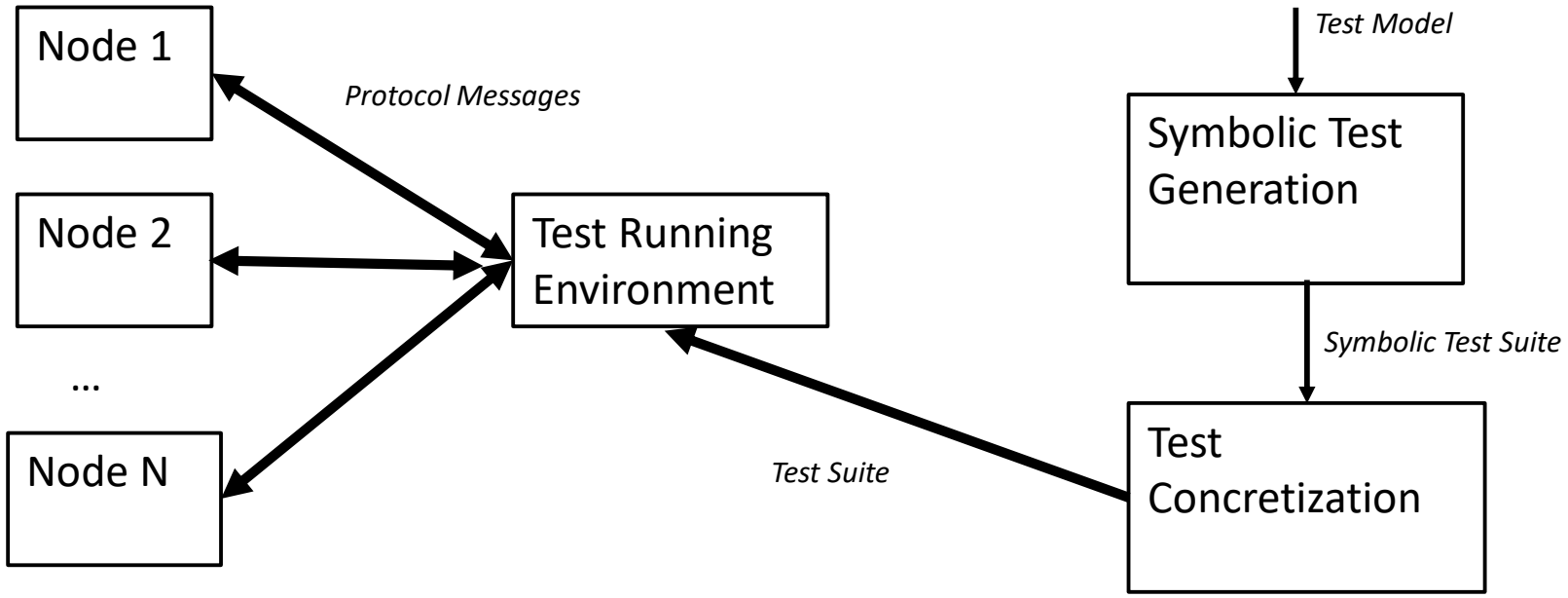
  - Визначення досяжності символьним моделюванням

- Інші способи – інваріанти

Модельне тестування

- Модель є рівняннями алгебри поведінки. Вона може бути створена із вимог до системи або отримана автоматично із специфікацій в іншій формальній мові.
- Генерація тестів із моделі методом символьного моделювання поведінок.
- Конкретизація тестів.
- Виконання тестів.

Загальна схема та проблеми модельного тестування



Проблеми модельного тестування для розподілених систем:
Експоненціальний вибух;
Для розподілених систем:

- труднощі із виявленням покриття;
- еквівалентні поведінки та надлишкове тестування

Властивості в алгебрі поведінок

- Взаємозамінність (permutability)

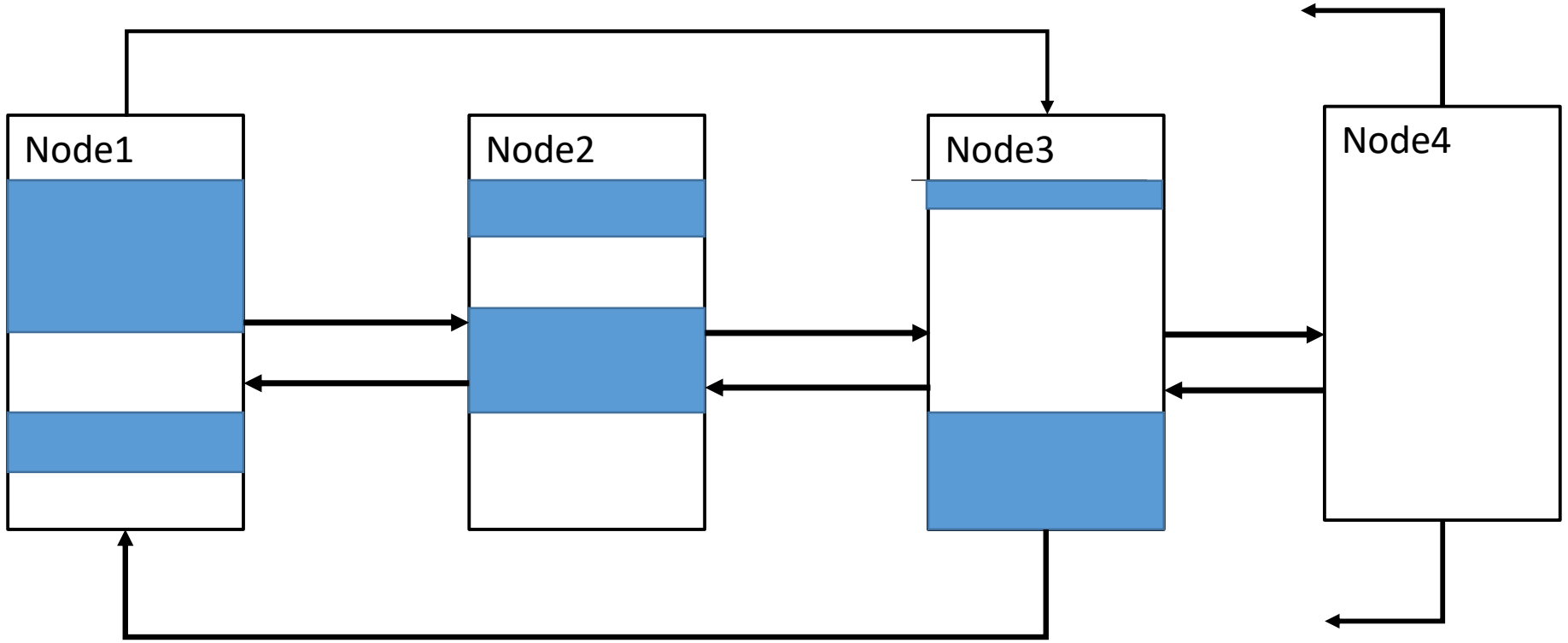
$$pt(\alpha \wedge pt(\alpha \wedge \beta, b), a) = pt(\beta \wedge pt(\alpha \wedge \beta, a), b)$$
$$pt(\alpha \wedge pt(\neg\alpha \wedge \beta, b), a) = 0$$
$$pt(\beta \wedge pt(\alpha \wedge \neg\beta, a), b) = 0$$

значне скорочення тестового набору за рахунок скорочення еквівалентних сценаріїв

- Еквівалентність поведінок
  (трасова, бісімуляційна)

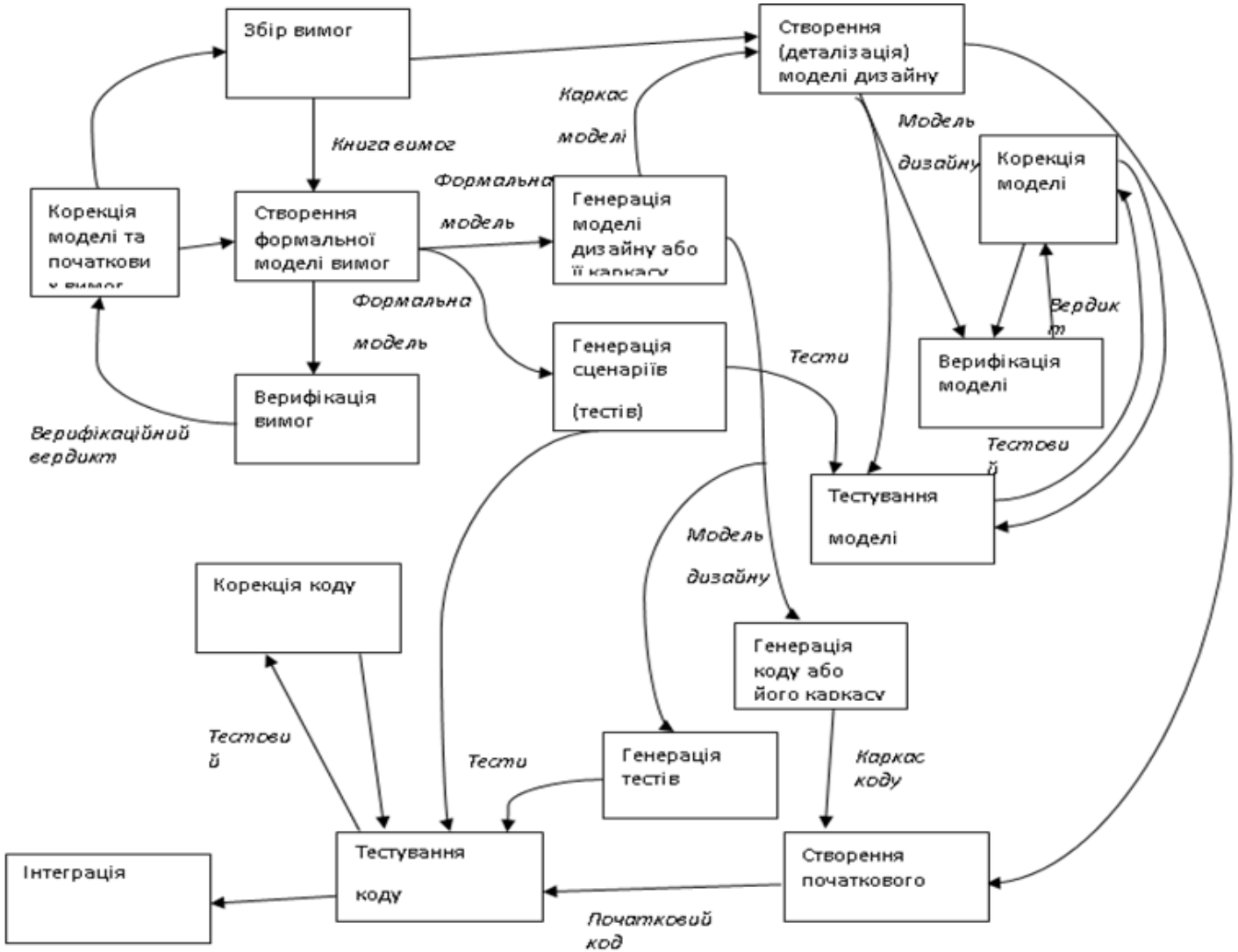
  Видалення еквівалентних сценаріїв, що критично для розподілених гомогенних мереж

Експеримент з символьним модельним тестуванням



Покриття коду з точністю до бісимуляційної еквівалентності

Модельний спосіб розробки



Кібербезпека. Пошук вразливостей в бінарному коді.

X86 асемблер:

```
8049865:  2d e0 01 1d 08      sub eax,0x81d01e0
804986a:  c1 f8 02            sar eax,0x2
804986d:  89 c2               mov edx,eax
8049876:  75 01               jne 8049879
```

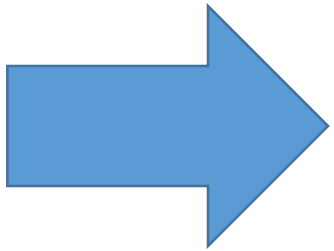
Модель коду я вирази алгебри поведінок:

```
B8049865 = sub(1,eax,0x81d01e0).B804986a,
B804986a = sar(1,eax,0x2).B804986d,
B804986d = mov(1,edx,eax,mov).B8049876,
B8049876 = jmp(1,jne).B8049879
```

Трансляція x86 в алгебру поведінок

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

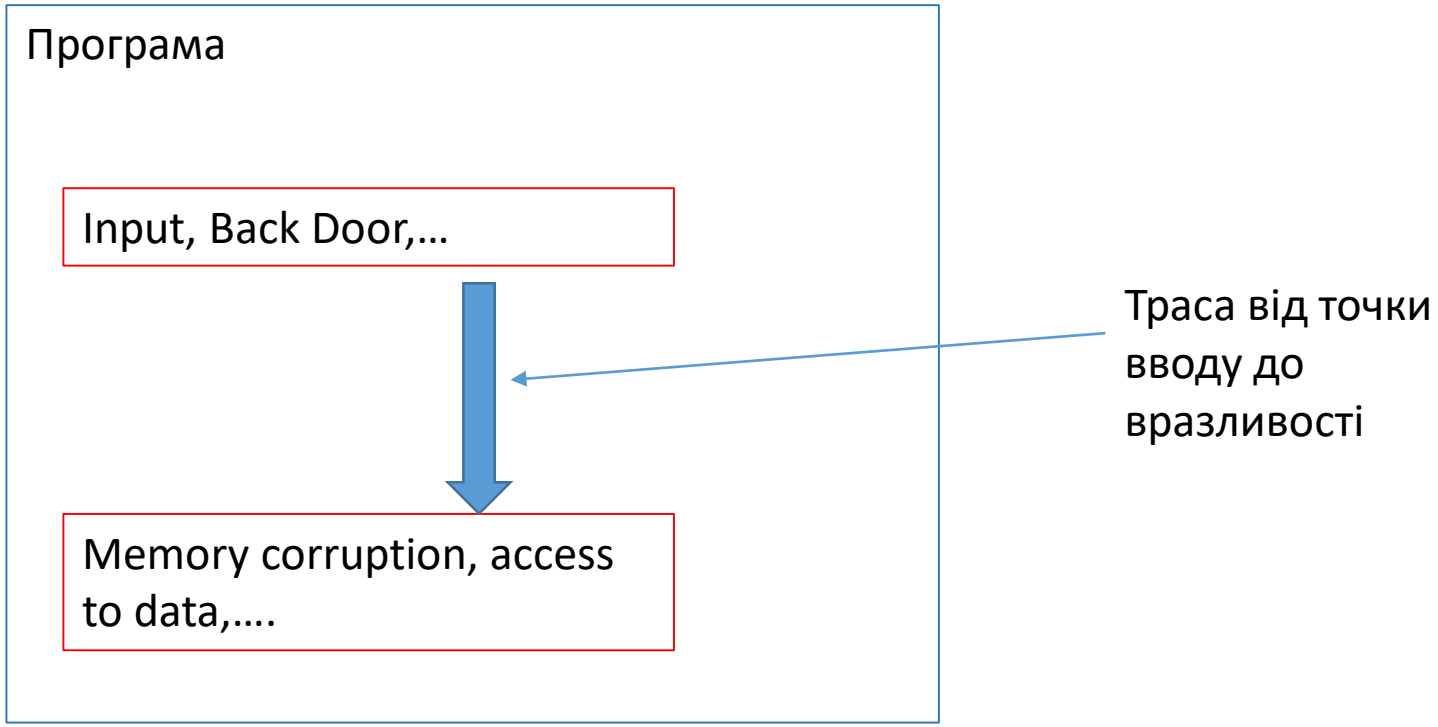
```
B425060 = a_push_33766.B425062,
B425062 = a_push_33767.B425064,
B425064 = a_mov_33768.B425067,
B425067 = a_push_33769.B425068,
B425068 = a_push_33770.B425069,
B425069 = a_mov_33771.B42506c,
B42506c = a_mov_33772.B42506e,
B42506e = a_sub_33773.B425072,
B425072 = a_call_33774.call B407550.B425077,
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)),
```

Алгебраїчні шаблони вразливостей

VulnerabilityPattern = IntruderInput; ProgramBehavior; VulnerabilityPoint



Алгебраїчні шаблони вразливостей

BUFFER OVERFLOW вразливість. Поведінка:

vulnerabilityBufferOverflow = input; X1; allocateStack; X2; writeStack,

input = mov(9, eax, 0x66). mov(10, ebx, 0x11). lea(11, ecx, MemoryOperand). call(12, MemoryOperand) + …,

allocateStack = push(1,ebp).mov(2,ebp,esp).sub(3,esp,N),

writeStack = movs(8, MemoryOperand, MemoryOperand) + mov(5, MemoryOperand, regGen)

Алгебраїчні шаблони вразливостей

Дії:

call(12,MemoryOperand) = Forall(i:int, 0<=i<LengthSocket) -> Input(ecx + i) = true,
mov(2,ebp,esp)=1->StackAddr=ebp,

movs(8,MemoryOperand,MemoryOperand) = Input(RefMemorySrc) &&(StackAddr==RefMemoryDest) -> 1,
mov(5, MemoryOperand, regGen) = Input(RefMemorySrc) && (StackAddr == RefMemoryDest) -> 1,

mov(x, GenReg, MemoryOperand) = Input(RefMemorySrc) -> Input(GenReg),
mov(x, MemoryOperand, MemoryOperand) = Input(RefMemorySrc) -> Input(RefMemoryDest)

Алгебраїчне зіставлення поведінок

Знайти поведінку, що веде до вразливості, представлену алгебраїчним шаблоном.
Задача вирішується в два етапи.

1. Вирішенням рівняння алгебри поведінок.
Нехай *B0* система рівнянь алгебри поведінок:
B0 = R(a1, a2, … , B1, B2, …,),
B1 = R(a11, a12, …, B11, B12, …), …
де *B1, B2, … B11, B12, ...* поведінки, *a1, a2, …* є дії. *B0* отримано трансляцією бінарного коду в алгебру поведінок.

Трбеа знайти таку поведінку X, що B0 = X;Y і
X = *vulnerabilityBufferOverflow*

2. Доведенням досяжності поведінки методом символьного моделювання.

Алгебраїчна модель поведінки зловмисника

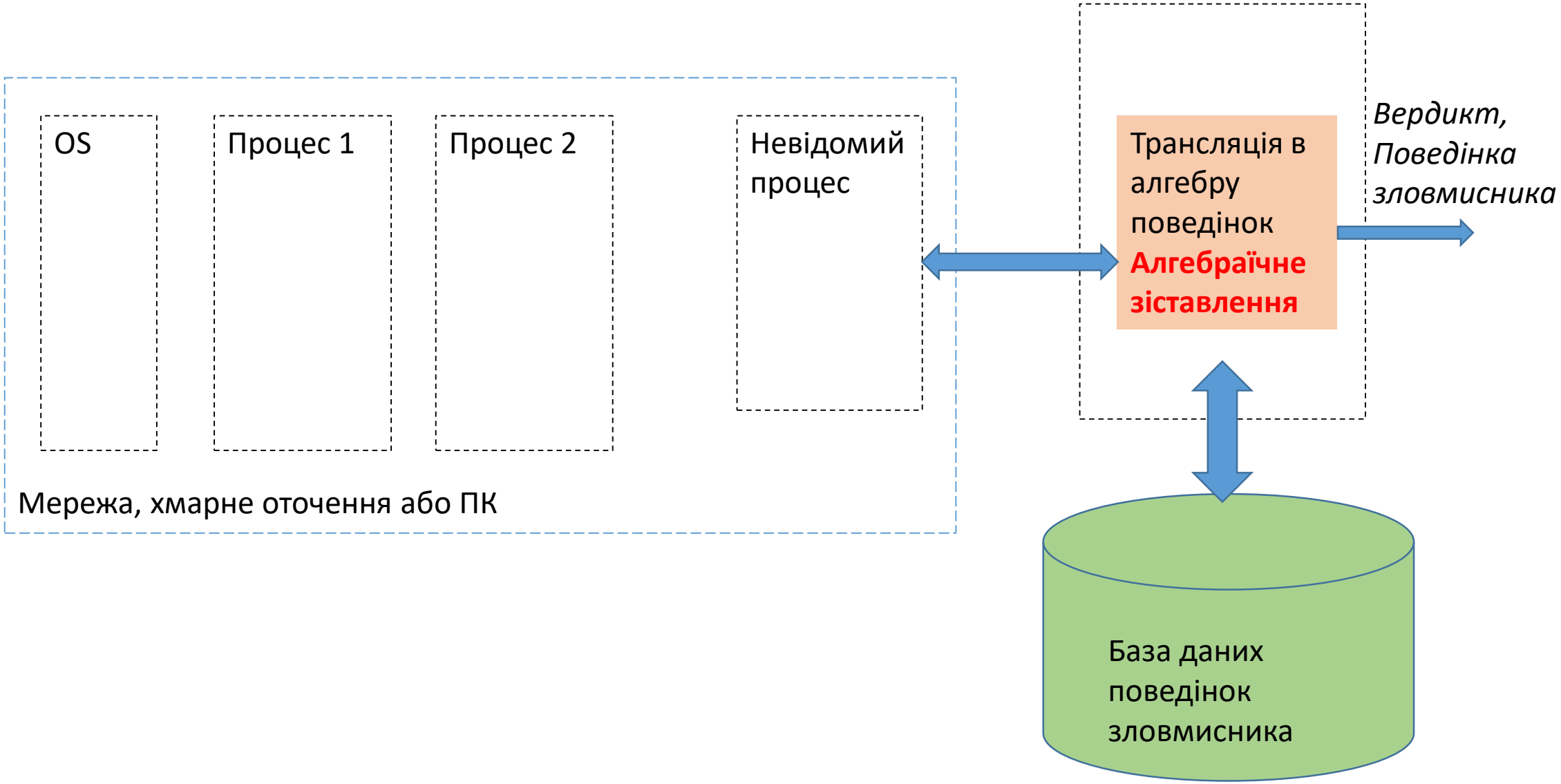
- Приклад атаки **Spectre**:

```
    /* Bit twiddling to set x=training_x if j%6!=0 or malicious_x if j%6==0 */
    /* Avoid jumps in case those tip off the branch predictor */
    x = ((j % 6) - 1) & ~0xFFFF; /* Set x=FFF.FF0000 if j%6==0, else x=0 */
    x = (x | (x >> 16)); /* Set x=-1 if j&6=0, else x=0 */
    x = training_x ^ (x & (malicious_x ^ training_x));

    /* Call the victim! */
    victim_function(x);
}

/* Time reads. Order is lightly mixed up to prevent stride prediction */
for (i = 0; i < 256; i++) {
    mix_i = ((i * 167) + 13) & 255;
    addr = &array2[mix_i * 512];
    time1 = __rdtscp(&junk); /* READ TIMER */
    junk = *addr; /* MEMORY ACCESS TO TIME */
    time2 = __rdtscp(&junk) - time1; /* READ TIMER & COMPUTE ELAPSED TIME */
    if (time2 <= CACHE_HIT_THRESHOLD && mix_i != array1[tries % array1_size])
        results[mix_i]++; /* cache hit - add +1 to score for this value */
}
```

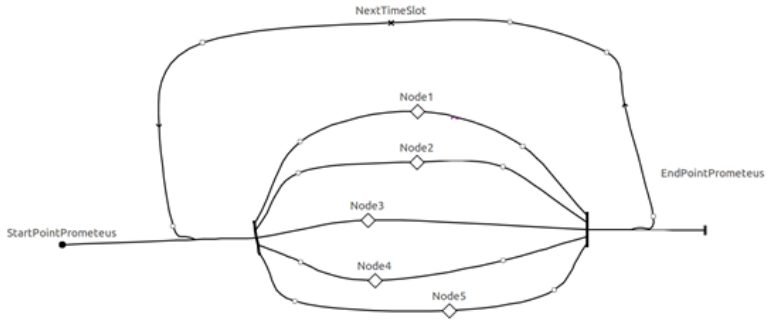
```
400801:   48 8d 45 94          lea    rax,[rbp-0x6c]
400805:   48 89 45 d8          mov    QWORD PTR [rbp-0x28],rax
400809:   0f 01 f9             rdtscp
40080c:   89 ce                mov    esi,ecx
40080e:   48 8b 4d d8          mov    rcx,QWORD PTR [rbp-0x28]
400812:   89 31                mov    DWORD PTR [rcx],esi
400814:   48 c1 e2 20          shl    rdx,0x20
400818:   48 09 d0             or     rax,rdx
40081b:   48 89 c3             mov    rbx,rax
40081e:   48 8b 45 d0          mov    rax,QWORD PTR [rbp-0x30]
400822:   0f b6 00             movzx  eax,BYTE PTR [rax]
400825:   0f b6 c0             movzx  eax,al
400828:   89 45 94             mov    DWORD PTR [rbp-0x6c],eax
40082b:   48 8d 45 94          lea    rax,[rbp-0x6c]
40082f:   48 89 45 c0          mov    QWORD PTR [rbp-0x40],rax
400833:   0f 01 f9             rdtscp
400836:   89 ce                mov    esi,ecx
400838:   48 8b 4d c0          mov    rcx,QWORD PTR [rbp-0x40]
40083c:   89 31                mov    DWORD PTR [rcx],esi
40083e:   48 c1 e2 20          shl    rdx,0x20
400842:   48 09 d0             or     rax,rdx
400845:   48 29 d8             sub    rax,rbx
400848:   48 89 c3             mov    rbx,rax
40084b:   48 83 fb 50          cmp    rbx,0x50
40084f:   77 3e                ja     40088f <readMemoryByte+0x205>
400851:   8b 45 9c             mov    eax,DWORD PTR [rbp-0x64]
400854:   8b 0d 06 18 20 00    mov    ecx,DWORD PTR [rip+0x201806]
```

B = X1;a1(rdtscp).a2(mov,t1 = esi,ecx).X2;B1,
B1 = a3(mov,i = rax,A).X3;B2,
B2 = X4;a4(rdtscp).a5(mov,t2 = esi,ecx).X5;B3,
B3 = a6(t2 - t1 < CASH_THRESHOLD).B + a7(t2 - t1 >=CASH_THRESHOLD).B4,
B4 = ...

Виявлення атаки



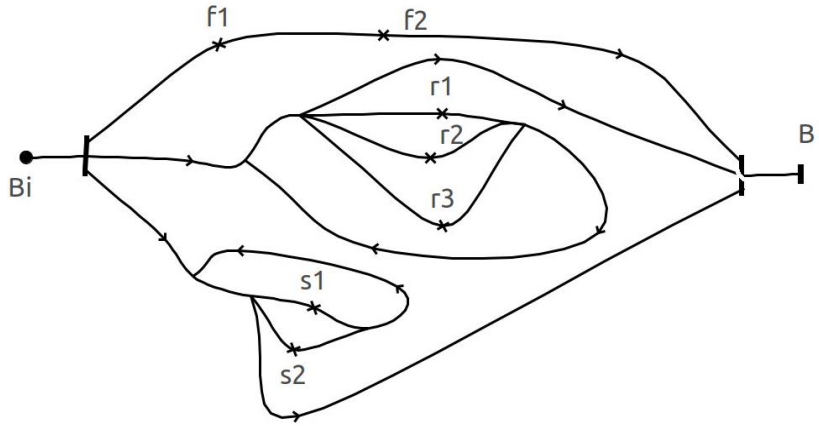
Аналіз децентралізованих систем



Маємо систему в якій вузли працюють паралельно. Поведінка представляє паралельну композицію.

$B = (B1 \,||\, B2 \,||\, \ldots \,||\, Bn);B$

$B = (||\, Bi);B$



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

$F = f1.f2.B,$

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

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

Верифікація властивостей (Відновлення мережі)

Формула представляє властивість, яка визначає властивість відновлення мережі. Тобто при тимчасовому роз'єднанні мережі, після відновлення всі вузли отримають пропущені блоки.

Forall (i,k) (1<=i<=VALIDATORS, 1<= k<= timeSlot)
BlockNumber(k,i) = k

VALIDATORS є кількість вузлів, *BlockNumber(k,i)* кількість отриманих блоків вузлом *i* в часовий слот *k*.

Це властивість життєстійкості, що має виконуватись в час, коли мережа буда відновлена. Властивість доводиться оберненим символьним моделюванням із точки відновлення.

Висновки

- Алгебра поведінок є потужним апаратом для формалізації агентів в найрізноманітніших областях індустрії. Теорія агентів та середовищ є природньою та зручною для формалізації
- Алгебра поведінок вирішує проблеми верифікації, тестування, кібербезпеки, перетворення об’єктів та є тим інструментом, що долає розрив між інженерним та формальним підходом.
- Практична здатність алгебри поведінок підтверджена співпрацею із багатьма іноземними та українськими компаніями та університетами.

Партнери та співвиконавці

- ІПС, Motorola, Intel (2000 – 2009)
- University of Missouri, Rolla(2009-2010)
- Uniquesoft (2010-2016)
- Херсонський університет
- Вінницький університет
- Харківський Авіаційний Інститут
- Skillonomy, Pandora
- London University College
- GARUDA.AI

Та інші…

Дякую за увагу!

oleksandr.letychevskyi@litsoft.com.ua

Інститут кібернетики ім.В.М.Глушкова НАН України