REVERSE ENGINEERING, SYMBOLIC EXECUTION, AND GRÖBNER BASES

2022-04-09

Tobias Magnusson

DENNIS YURICHEV'S CHALLENGE #4

What does the following code do? Give a one to two sentence answer.

```
mov     edx,edi
shr     edx,1
and     edx,0x55555555
sub     edi,edx
mov     eax,edi
shr     edi,0x2
and     eax,0x33333333
and     edi,0x33333333
add     edi,eax
mov     eax,edi
shr     eax,0x4
add     eax,edi
and     eax,0x0f0f0f0f
imul    eax,eax,0x01010101
shr     eax,0x18
```

REGISTERS

REGISTERS

64-bit registers:

`rax, rbx, rcx, rdx, rsi, rdi, rbp, rsp, r8-r15`

REGISTERS

64-bit registers:

`rax, rbx, rcx, rdx, rsi, rdi, rbp, rsp, r8-r15`

32-bit lower half:

`eax, ebx, ecx, edx, esi, edi, ebp, esp`

REGISTERS

64-bit registers:

`rax, rbx, rcx, rdx, rsi, rdi, rbp, rsp, r8-r15`

32-bit lower half:

`eax, ebx, ecx, edx, esi, edi, ebp, esp`

16-bit lower lower half:

`ax, bx, cx, dx, si, di, bp, sp`

Back to the code.

```
mov     edx,edi
shr     edx,1
and     edx,0x55555555
sub     edi,edx
mov     eax,edi
shr     edi,0x2
and     eax,0x33333333
and     edi,0x33333333
add     edi,eax
mov     eax,edi
shr     eax,0x4
add     eax,edi
and     eax,0x0f0f0f0f
imul    eax,eax,0x01010101
shr     eax,0x18
```

INSTRUCTIONS

INSTRUCTIONS

- `mov X, Y`

INSTRUCTIONS

- `mov X, Y`
- `shr X, n`

INSTRUCTIONS

- `mov X, Y`
- `shr X, n`
- `and X, Y`

INSTRUCTIONS

- `mov X, Y`
- `shr X, n`
- `and X, Y`
- `sub X, Y`

INSTRUCTIONS

- `mov X, Y`
- `shr X, n`
- `and X, Y`
- `sub X, Y`
- `add X, Y`

INSTRUCTIONS

- `mov X, Y`
- `shr X, n`
- `and X, Y`
- `sub X, Y`
- `add X, Y`
- `imul X, Y`

INSTRUCTIONS

- `mov X, Y`
- `shr X, n`
- `and X, Y`
- `sub X, Y`
- `add X, Y`
- `imul X, Y`
- `ret`

SYMBOLIC EXECUTION

SYMBOLIC EXECUTION

What does a program do on arbitrary input?

SYMBOLIC EXECUTION

What does a program do on arbitrary input?

Goes back to mid 70s (e. g. *SELECT - a formal system for testing and debugging programs by symbolic execution*, Boyer-Elspas-Levitt, 1975).

- Initialize graph with a single node containing state and branching constraint.

- Initialize graph with a single node containing state and branching constraint.
- Go through program line-by-line.

- Initialize graph with a single node containing state and branching constraint.
- Go through program line-by-line.
  - Non-branching: create child-node with new state and old branching constraint.

- Initialize graph with a single node containing state and branching constraint.
- Go through program line-by-line.
  - Non-branching: create child-node with new state and old branching constraint.
  - Branching: for every branch, create child-node with new state **and** branching contraint.

- Initialize graph with a single node containing state and branching constraint.
- Go through program line-by-line.
  - Non-branching: create child-node with new state and old branching constraint.
  - Branching: for every branch, create child-node with new state **and** branching contraint.

Problem: Loops.

- Initialize graph with a single node containing state and branching constraint.
- Go through program line-by-line.
  - Non-branching: create child-node with new state and old branching constraint.
  - Branching: for every branch, create child-node with new state **and** branching contraint.

Problem: Loops.

Solution: Induction or heuristics (e. g. "concolic execution").

Simple example (from *A Survey of Symbolic Execution Techniques*, Baldoni et al. 2018)

Simple example (from *A Survey of Symbolic Execution Techniques*, Baldoni et al. 2018)

When does the `assert` fail?

Simple example (from *A Survey of Symbolic Execution Techniques*, Baldoni et al. 2018)

When does the `assert` fail?

```c
void foobar(int a, int b){
  int x = 1, y = 0;
  if(a != 0){
    y = 3 + x;
    if(b == 0){
      x = 2*(a + b);
    }
  }
  assert(x - y != 0);
}
```

Simple example (from *A Survey of Symbolic Execution Techniques*, Baldoni et al. 2018)

When does the `assert` fail?

```
void foobar(int a, int b){
  int x = 1, y = 0;
  if(a != 0){
    y = 3 + x;
    if(b == 0){
      x = 2*(a + b);
    }
  }
  assert(x - y != 0);
}
```

Let's go to `xournalpp`.

Back to our case.

Back to our case.

- State?

Back to our case.

- State?
- State transitions?

Back to our case.

- State?
- State transitions?
- No branching.

Let $I = (b_0^2 - b_0, \ldots, b_{31}^2 - b_{31}) \subseteq \mathrm{F}_2[b_0, \ldots, b_{31}]$ and

$$B = \mathrm{F}_2[b_0, \ldots, b_{31}]/I$$

Let $I = (b_0^2 - b_0, \ldots, b_{31}^2 - b_{31}) \subseteq \mathrm{F}_2[b_0, \ldots, b_{31}]$ and

$$B = \mathrm{F}_2[b_0, \ldots, b_{31}]/I$$

Bitstrings of length 32 belong to B^{32}.

Let $I = (b_0^2 - b_0, \ldots, b_{31}^2 - b_{31}) \subseteq \mathrm{F}_2[b_0, \ldots, b_{31}]$ and

$$B = \mathrm{F}_2[b_0, \ldots, b_{31}]/I$$

Bitstrings of length 32 belong to B^{32}.

Least significant bit on the right.

$$\texttt{shr}((x_{31}, \ldots, x_0), 1) = (0, x_{31}, \ldots, x_1)$$

$$\texttt{and}(x, y) = (x_{31}y_{31}, \ldots, x_0y_0)$$

$$\texttt{shr}((x_{31}, \ldots, x_0), 1) = (0, x_{31}, \ldots, x_1)$$

$$\texttt{and}(x, y) = (x_{31}y_{31}, \ldots, x_0y_0)$$

How about `add`, `sub`, and `imul`?

CARRY

CARRY

Let $c, x, y \in B$. Then

$$\texttt{carry}(c, x, y) = cx + cy + xy$$

FROM CARRY TO ADD

FROM CARRY TO ADD

Let $x, y \in B^{32}$. Define $B^{32} \ni z = \mathtt{add}(x, y)$ by $c_0 = 0$

and

$$z_i = x_i + y_i + c_i$$

$$c_{i+1} = \mathtt{carry}(c_i, x_i, y_i)$$

for $0 \le i \le 31$. If $c_{32} = 1$, we have overflow.

sub AND imul

`sub` AND `imul`

Two's complement. For $x \in B^{32}$, set

$$-x = \texttt{add}((x_{31} + 1, \ldots, x_1 + 1, x_0 + 1), (0, \ldots, 0, 1))$$

`sub` AND `imul`

Two's complement. For $x \in B^{32}$, set

$$-x = \texttt{add}((x_{31} + 1, \ldots, x_1 + 1, x_0 + 1),$$
$$(0, \ldots, 0, 1))$$

`imul` is just repeated addition.

WORKING IN $\mathbb{F}_2[b]$

WORKING IN $\mathbf{F}_2[B]$

Use Singular.jl.

WORKING IN $\mathbf{F}_2[B]$

Use Singular.jl.

```
using Singular
vars = ["b$ix" for ix in 0:31]
R, b = PolynomialRing(Fp(2), vars)
(b0,b1,b2,b3,b4,b5,b6,b7,
 b8,b9,b10,b11,b12,b13,b14,b15,
 b16,b17,b18,b19,b20,b21,b22,b23,
 b24,b25,b26,b27,b28,b29,b30,b31) = b
```

Here is `add` without "reduction".

Here is add without "reduction".

```
function addeq!(l0 :: Vector{spoly{A}},
                l1 :: Vector{spoly{A}},
                l2 :: Vector{spoly{A}})
                  where A
  length(l0) == length(l1) == length(l2)
      || error("vector dimensions not equal")
  ln = length(l0)
  R = parent(l0[1])
  temp1 = zero(R)
  temp2 = zero(R)
  temp3 = zero(R)
  carry = zero(R)
  for ix in 1:ln
    addeq!(l0[ix], l1[ix])
    addeq!(l0[ix], l2[ix])
```

For others -- same idea.

GRÖBNER BASES

GRÖBNER BASES

What are they?

GRÖBNER BASES

What are they?

Gotta talk about monomial orders.

MONOMIAL ORDER

MONOMIAL ORDER

Let $M = Z_{\geq 0}$. Monomial order $\geq$:

MONOMIAL ORDER

Let $M = Z_{\geq 0}$. Monomial order $\geq$:

- Is reflexive, transitive, antisymmetric, strongly connected.

MONOMIAL ORDER

Let $M = Z_{\geq 0}$. Monomial order $\geq$:

- Is reflexive, transitive, antisymmetric, strongly connected.
- Satisfies: If $\alpha \geq \beta$ then $\alpha + \gamma \geq \beta + \gamma$.

MONOMIAL ORDER

Let $M = Z_{\geq 0}$. Monomial order $\geq$:

- Is reflexive, transitive, antisymmetric, strongly connected.
- Satisfies: If $\alpha \geq \beta$ then $\alpha + \gamma \geq \beta + \gamma$.
- Satisfies: Every $\varnothing \neq A \subseteq M$ has smallest element w. r. t. $>$.

MONOMIAL ORDER

Let $M = Z_{\geq 0}$. Monomial order $\geq$:

- Is reflexive, transitive, antisymmetric, strongly connected.
- Satisfies: If $\alpha \geq \beta$ then $\alpha + \gamma \geq \beta + \gamma$.
- Satisfies: Every $\varnothing \neq A \subseteq M$ has smallest element w. r. t. $>$.

For $x^{\alpha}, x^{\beta} \in k[x_1, \ldots, x_n]$ define:

$$x^{\alpha} \geq x^{\beta} \text{ iff } \alpha \geq \beta$$

MONOMIAL ORDER

Let $M = Z_{\geq 0}$. Monomial order $\geq$:

- Is reflexive, transitive, antisymmetric, strongly connected.
- Satisfies: If $\alpha \geq \beta$ then $\alpha + \gamma \geq \beta + \gamma$.
- Satisfies: Every $\varnothing \neq A \subseteq M$ has smallest element w. r. t. $>$.

For $x^\alpha, x^\beta \in k[x_1, \ldots, x_n]$ define:

$$x^\alpha \geq x^\beta \text{ iff } \alpha \geq \beta$$

Some examples

Lexicographical:

$\alpha >_{\text{lex}} \beta$ iff leftmost nonzero of $\alpha - \beta$ is positive

Lexicographical:

$\alpha >_{lex} \beta$ iff leftmost nonzero of $\alpha - \beta$ is positive

Graded lex.:

$\alpha >_{grlex} \beta$ iff $|\alpha| > |\beta|$, or $|\alpha| = |\beta|$ and $\alpha >_{lex} \beta$

Lexicographical:

$\alpha >_{\text{lex}} \beta$ iff leftmost nonzero of $\alpha - \beta$ is positive

Graded lex.:

$\alpha >_{\text{grlex}} \beta$ iff $|\alpha| > |\beta|$, or $|\alpha| = |\beta|$ and $\alpha >_{\text{lex}} \beta$

Graded rev. lex.:

$\alpha >_{\text{grevlex}} \beta$ iff $|\alpha| > |\beta|$,
or $|\alpha| = |\beta|$ and r.-most nonzero of $\alpha - \beta$ is negative

Lexicographical:

$\alpha >_{\text{lex}} \beta$ iff leftmost nonzero of $\alpha - \beta$ is positive

Graded lex.:

$\alpha >_{\text{grlex}} \beta$ iff $|\alpha| > |\beta|$, or $|\alpha| = |\beta|$ and $\alpha >_{\text{lex}} \beta$

Graded rev. lex.:

$\alpha >_{\text{grevlex}} \beta$ iff $|\alpha| > |\beta|$,
or $|\alpha| = |\beta|$ and r.-most nonzero of $\alpha - \beta$ is negative

And others.

TERMINOLOGY

TERMINOLOGY

Let $f = \sum_{\alpha} c_{\alpha} x^{\alpha} \in k[x_1, \ldots, x_n]$ and $>$ monomial order.

TERMINOLOGY

Let $f = \sum_{\alpha} c_{\alpha} x^{\alpha} \in k[x_1, \ldots, x_n]$ and $>$ monomial order.

- $\text{multideg}(f) = \max$

TERMINOLOGY

Let $f = \sum_{\alpha} c_{\alpha} x^{\alpha} \in k[x_1, \ldots, x_n]$ and $>$ monomial order.

- $\text{multideg}(f) = \max$
- \text{LC}(f)=c_{\text{multideg}(f)}

TERMINOLOGY

Let $f = \sum_{\alpha} c_{\alpha} x^{\alpha} \in k[x_1, \ldots, x_n]$ and $>$ monomial order.

- $\text{multideg}(f) = \max$
- \text{LC}(f)=c_{\text{multideg}(f)}
- \text{LM}(f)=x^{\text{multideg}(f)}

TERMINOLOGY

Let $f = \sum_{\alpha} c_{\alpha} x^{\alpha} \in k[x_1, \ldots, x_n]$ and $>$ monomial order.

- $\text{multideg}(f) = \max$
- \text{LC}(f)=c_{\text{multideg}(f)}
- \text{LM}(f)=x^{\text{multideg}(f)}
- \text{LT}(f)=\text{LC}(f)\cdot\text{LM}(f)

TERMINOLOGY

Let $f = \sum_{\alpha} c_{\alpha} x^{\alpha} \in k[x_1, \dots, x_n]$ and $>$ monomial order.

- $\text{multideg}(f) = \max$
- \text{LC}(f)=c_{\text{multideg}(f)}
- \text{LM}(f)=x^{\text{multideg}(f)}
- \text{LT}(f)=\text{LC}(f)\cdot\text{LM}(f)

Let \{0\}\neq I\subseteq k[x_1,\dots,x_n] ideal. Then \text{LT}(I)=\{z:\exists f\in I.\,\text{LT}(f)=z\}

REDUCTION

REDUCTION

Fix order. Let $F=(f_1,\dots,f_s)$ be polynomials in $k[x_1,\dots,x_n]$. Let $f\in k[x_1,\dots,x_n]$.

REDUCTION

Fix order. Let $F=(f_1,\dots,f_s)$ be polynomials in $k[x_1,\dots,x_n]$. Let $f\in k[x_1,\dots,x_n]$.

Then exists $a_i,r\in k[x_1,\dots,x_n]$ such that $f=a_1f_1+\dots+a_sf_s+r$, where $r=0$ or is lin. comb. of monomials neither of which is div. by $\text{LT}(f_i)$.

REDUCTION

Fix order. Let $F=(f_1,\dots,f_s)$ be polynomials in $k[x_1,\dots,x_n]$. Let $f\in k[x_1,\dots,x_n]$.

Then exists $a_i,r\in k[x_1,\dots,x_n]$ such that $f=a_1f_1+\dots+a_sf_s+r$, where $r=0$ or is lin. comb. of monomials neither of which is div. by $\text{LT}(f_i)$.

Instead of algorithm -- let's look at a generic example.

Back to `xournalpp`.

GRÖBNER BASES, DEFINITION

GRÖBNER BASES, DEFINITION

Let $A=k[x_1,\dots,x_n]$ with fixed monomial order. Let $I\subseteq A$ ideal.

GRÖBNER BASES, DEFINITION

Let $A=k[x_1,\dots,x_n]$ with fixed monomial order. Let $I\subseteq A$ ideal.

Say $g_1,\dots,g_t\in I$ Gröbner basis if $\langle\text{LT}(g_1),\dots\text{LT}(g_t)\rangle=\langle\text{LT}(I)\rangle$

GRÖBNER BASES, DEFINITION

Let $A=k[x_1,\dots,x_n]$ with fixed monomial order. Let $I\subseteq A$ ideal.

Say $g_1,\dots,g_t\in I$ Gröbner basis if $\langle\text{LT}(g_1),\dots\text{LT}(g_t)\rangle=\langle\text{LT}(I)\rangle$

Non-example: $g_1=xy-1$ and $g_2=y^2-1$ in $k[x,y]$ with grevlex. Have $x\in\text{LT}(\langle g_1,g_2\rangle)$ but $x\notin\langle xy,y^2\rangle$.

GRÖBNER BASES, PROPERTIES

GRÖBNER BASES, PROPERTIES

- Reduction is unique! (Proof omitted.)

GRÖBNER BASES, PROPERTIES

- Reduction is unique! (Proof omitted.)
- Equivalent to defining property.

Back to example.

Back to example.

Have that $\{b_0^2-b_0,\dots,b_{31}^2-b_{31}\}$, is Gröbner basis!

Back to example.

Have that $\{b_0^2-b_0,\dots,b_{31}^2-b_{31}\}$, is Gröbner basis!

So, pick order and reduce.

Inserting reduction in Singular.

Inserting reduction in Singular.

```
function _addeq!(l0 :: Vector{spoly{A}},
                 l1 :: Vector{spoly{A}},
                 l2 :: Vector{spoly{A}},
                 I  :: sideal{spoly{A}}) where A
  ...
  for ix in 1:ln
    zero!(l0_ix_unred)
    addeq!(l0_ix_unred, l1[ix])
    addeq!(l0_ix_unred, l2[ix])
    addeq!(l0_ix_unred, carry)
    l0[ix] = reduce(l0_ix_unred, I)
    mul!(temp1, l1[ix], l2[ix])
    mul!(temp2, carry, l2[ix])
    mul!(temp3, carry, l1[ix])
    zero!(carry_unred)
```

Time for a demo.

We obtain the bit-vector e with $e_i=\sum_{0\leq j_1\le \dots\le j_{2^i}\leq 31}b_{j_1}\cdots b_{j_{2^i}}$ for $0\leq i\leq 5$, and the rest zero.

We obtain the bit-vector e with $e_i=\sum_{0\leq j_1\le \dots\le j_{2^i}\leq 31}b_{j_1}\cdots b_{j_{2^i}}$ for $0\leq i\leq 5$, and the rest zero.

That is: say `edi` has n ones, then $e_i=\binom{n}{2^i}\text{ mod }2$.

Questions?