Automatic Proxy App Generation through Input Capture and Generation

Ivan R. Ivanov\textsuperscript{1,2}, Aiden Grossman\textsuperscript{3,6}, Ludger Paehler\textsuperscript{3,5}, William S. Moses\textsuperscript{4}, Johannes Doerfert\textsuperscript{3}

1: Tokyo Tech, 2: RIKEN, 3: LLNL, 4: UIUC, 5: TUM, 6: UC Davis

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-PRES-862736.
Input Generation Example

typedef struct LinkedList {
    int Payload;
    struct LinkedList *Next;
} LinkedList;

void sum(LinkedList *LL) {
    int S = 0, L = 0;
    while (LL != 0) {
        S += LL->Payload;
        L += 1;
        LL = LL->Next;
    }
    printf("Length: %i, sum %i\n", L, S);
}
What does our tool do?

What you need

- **module.bc**
- **LLVM Module**
- **Entry point:** my_fancy_function(...)

What you get

**Generated Input**

Input state:
010101010100000
010101010100000

Output state:
101011100110101
011010001110011

**Driver**

```
int main() {
load_state();
__input_gen_entry(...);
check_output();
}
```

**Proxy App**

(benchmark)

You can run any* code!
Input Generation Flow

1. Instrument side effects

Allocate random amount of mem and return it

Have we previously stored here? Yes? -> Return that
No? -> Pull out a value out of thin air and pretend it was stored there

Store at the location while not overwriting the real initial state (values we ‘pretended’ were there)
void generate_inputs() {
    while (true) {
        RandomSeed = ...
        InputGeneratorRuntime = new(RandomSeed);
        __input_gen_entry();
        if (finished_successfully)
            dump_generated_input_state();
    }
}

We explore various random seeds and store the generated inputs.
Evaluation, Future Work

ComPile: Dataset of ~750,000 LLVM IR modules
We ran input gen on 50 of them and got the following results:

- Number of functions: 853
- Number of functions instrumented: 373 (-480)
- Number of functions input gen succeeded: 192 (-181)
- Number of functions for which generated input ran: 181 (-11)

Future plans:

- ‘Hints’ to the inputgen runtime from static analysis
- Focusing on (evaluating) branch coverage
- Matching profile information (branch probabilities) -> Scaling out programs
Thank you!
Input Recording Flow

1. Instrument side effects

```assembly
define fn1(i32 %arg) {
  bb0:
  ...
  %a = load %p : i64
  %b = ...
  store %b, %p2 : f32
  ...
}
```

2. Record state

```assembly
define fn1(i32 %arg) {
  entry:
  __record_i32_arg(%arg)
  br %bb0
  bb0:
  ...
  %a = call __record_load(%p, 8)
  %b = ...
  call __record_store(%b, %p2, 4)
  ...
}
```

3. Generate Driver

```c
int main() {
  load_state();
  fn1(...);
  check_output();
}
```

4. Bundle everything

Input state: 01010101010100000
Output state: 011010001110011

Proxy App