

Automatic Proxy App Generation through Input Capture and Generation

Ivan R. Ivanov[1,2], Aiden Grossman[3,6], Ludger Paehler[3,5], William S. Moses[4], Johannes Doerfert[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

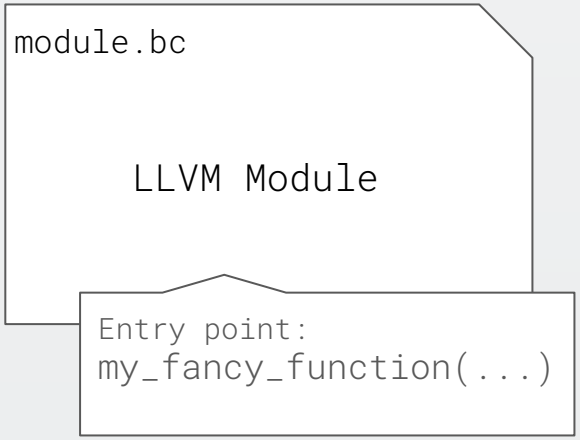
```
typedef struct LinkedList {
  int Payload;
  struct LinkedList *Next;
} LinkedList;

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);
}
```

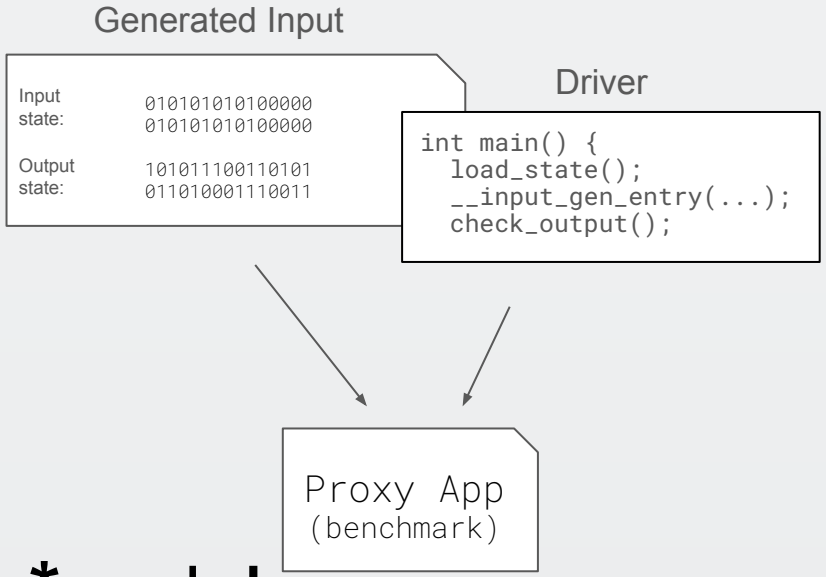
```
$ input-gen linked_list.ll --...
Length: 4, sum 1422
$ input-gen linked_list.ll --...
Length: 12, sum 5621
```

What does our tool do?

What you need



What you get



You can run any* code!

Input Generation Flow

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

1. Instrument side effects

```
define __input_gen_entry() {
entry:
  %p = __inputgen_ptr_arg()
  br %bb0
bb0:
  ...
  %a = call __inputgen_load_i64(%p, 8)
  %b = ...
  call __inputgen_store_f32(%b, %p2, 4)
  ...
```

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

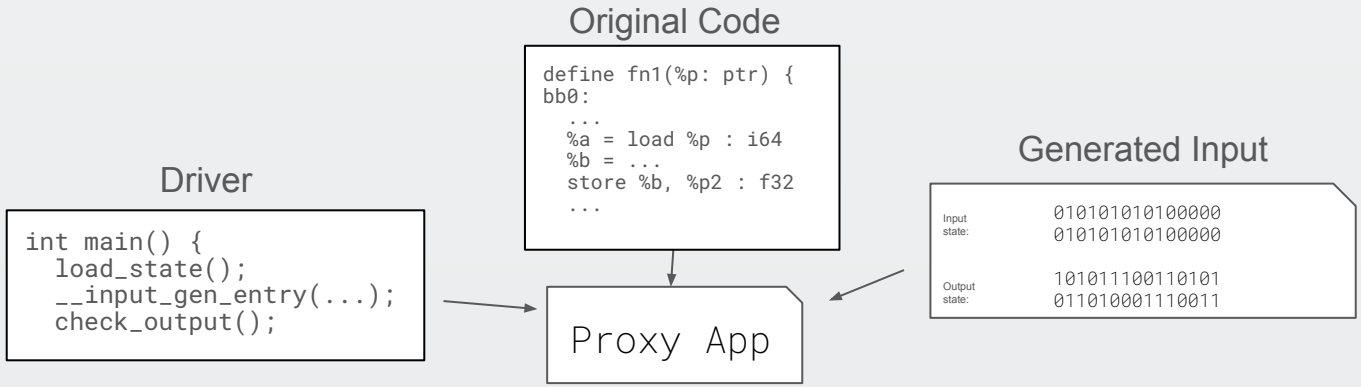
Store at the location
while not overwriting the real initial state (values we ‘pretended’ were there)

Input Generation Flow Cont.

```
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