Specification-Driven Synthesis of Summaries for Symbolic Execution

Rafael Gonçalves , Frederico Ramos, Pedro Adão, and José Fragoso Santos
Instituto Superior Técnico, University of Lisbon

TÉCNICO LISBOA

Symbolic Summaries

- Simulate function behavior by manipulating the **symbolic state** using symbolic reflection primitives
- Main advantages:
  - Model interactions with the runtime environment (I/O, files, sockets, etc.)
  - Contain path explosion

```
size_t strlen(char* s) {
  int i = 0;
  char zero = '\0';

  while (1) {
    if (is_symbolic(&s[i], CHAR_SIZE)) {
      if (!is_sat(_solver_NEQ(&s[i], &zero, CHAR_SIZE)))
        break;
      else
        assume(_solver_NEQ(&s[i], &zero, CHAR_SIZE));
    } else if (s[i] == '\0') break;

    i++;
  }

  return i;
}
```

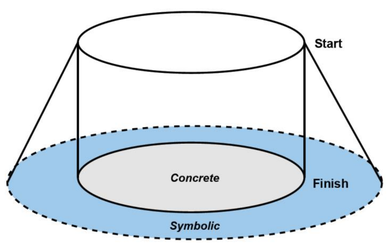
Wide adoption by state-of-the-art tools

angr

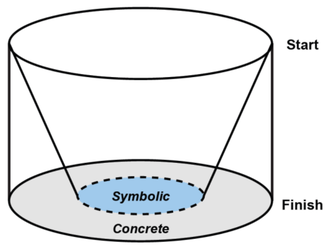
Manticore

BINSEC

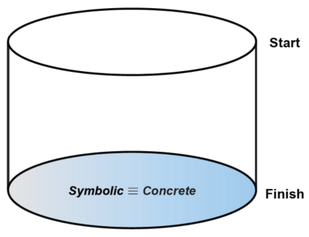
Correctness Properties



Over-approximating



Under-approximating



Exact

A summary that is neither over- nor under-approximating is **buggy** (or **unsound**)

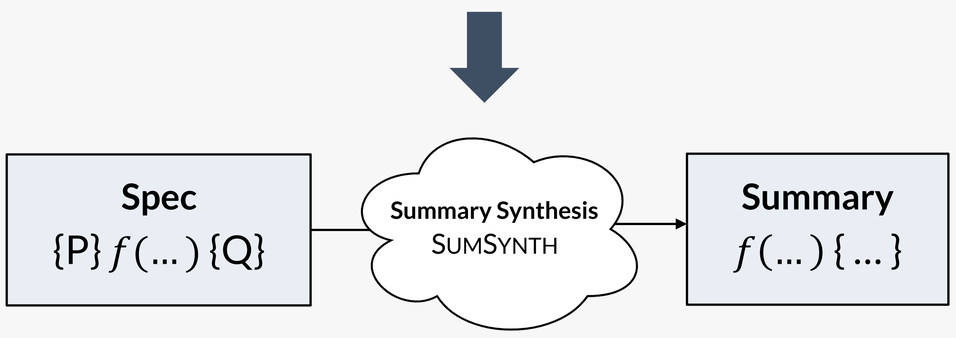
```
size_t strlen(char* s) {
  int i = 0;

  while (is_symbolic(&s[i], CHAR_SIZE) ||
    (!is_symbolic(&s[i], CHAR_SIZE) &&
    s[i] != '\0')) {
      i++;
  }

  return i;
}
```

Buggy! Why?

- Not **over-approximating**
  - **Missing Path:**
    $[str = [\hat{c}, '\backslash 0'] \wedge \hat{c} = '\backslash 0' \wedge ret = 0]$
- Not **under-approximating**
  - **Wrong Path:**
    $[str = [\hat{c}, '\backslash 0'] \wedge \hat{c} = '\backslash 0' \wedge ret = 1]$

Goal

Synthesize **summaries** from separation logic **specs**



Example

$\{\text{str}(s, \nu)\}$ `size_t strlen(char *s)` $\{\text{str}(s, \nu) * ret = \nu\}$

```
int sum_strlen(char *s) {
    if (is_certain(*s == 0))
        return 0;
    else {
        assume(*s != 0);
        int k = fold_str(s + 1);
        return k + 1;
    }
}
```

Under-approximating

```
int sum_strlen(char *s) {
    if (is_certain(*s == 0))
        return 0;
    else if (is_certain(*s != 0)) {
        int k = fold_str(s + 1);
        return k + 1;
    } else {
        int n = new_int();
        assume(n >= 0);
        return n;
    }
}
```

Over-approximating

Evaluation Plan

Synthesize **summaries** from *Verifiable C*[1] **specs**

Validate and **evaluate** with **SumBoundVerify**[2]

Target API Functions

- String manipulation (**`strlen`**, **`strcmp`**, etc.)
- Number-parsing (**`atoi`**, **`atol`**)
- Others (?)

Metrics

- Execution time
- Memory usage
- Code coverage

References

[1] Appel, A.W., Beringer, L., Cao, Q.: Verifiable C, Software Foundations, vol. 5. Electronic textbook (2022), version 1.2.1. http://softwarefoundations.cis.upenn.edu
[2] Ramos, F.: Toward Tool-Independent Summaries for Symbolic Execution. Master's thesis, Instituto Superior Técnico, Universidade de Lisboa (2021)