What Are We Going To Do About Libraries? A Work in Non-Progress Talk

Martin Nyx Brain

University of Oxford

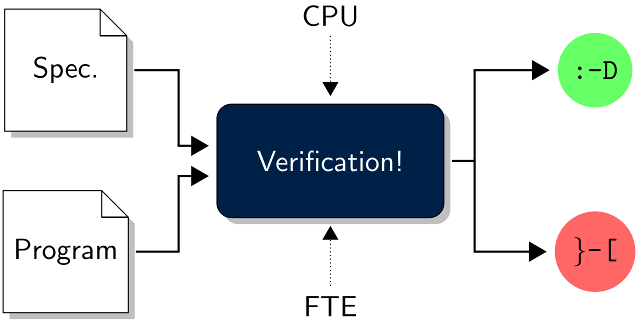
September 18, 2018

A Work in Non-Progress Talk

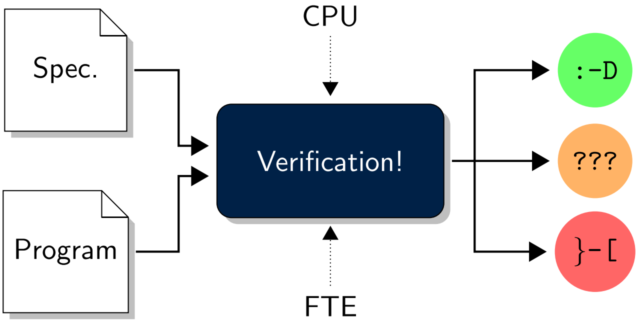
Don't say I didn't warn you...

- No answers; only problems.
- No results; only opinions.

Verification

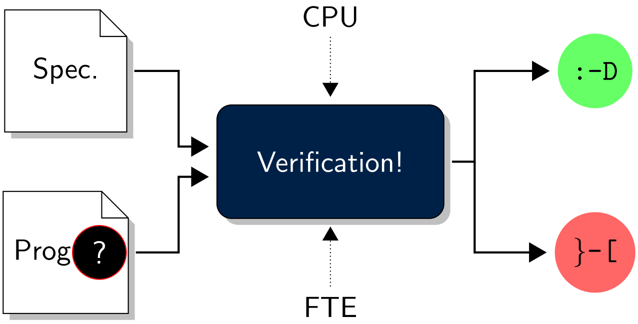


Verification



"The Library Problem"

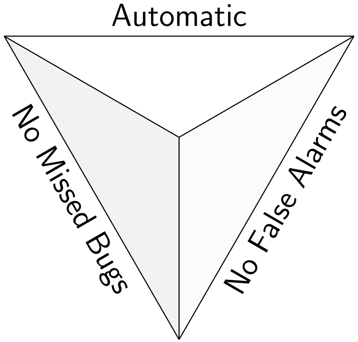
Parts of the program are not available or desireable to analyse



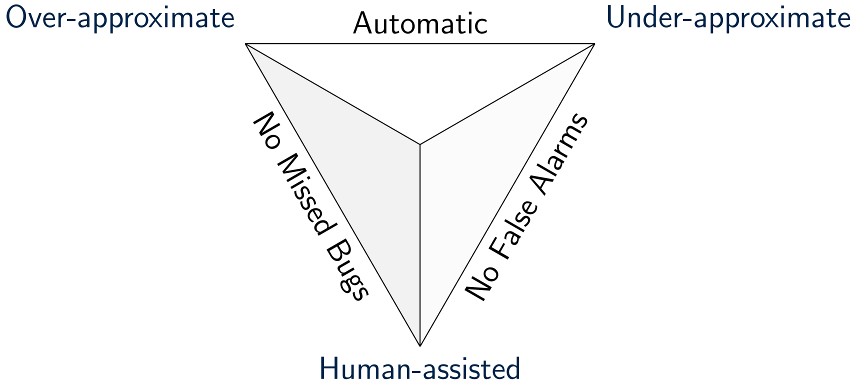
Because. . .

- Source unavailable
- External functionality
- Out of scope
- Platform independence
- Unspecified / imp. def.
- Too complex
- Program not finished
- This *is* the library

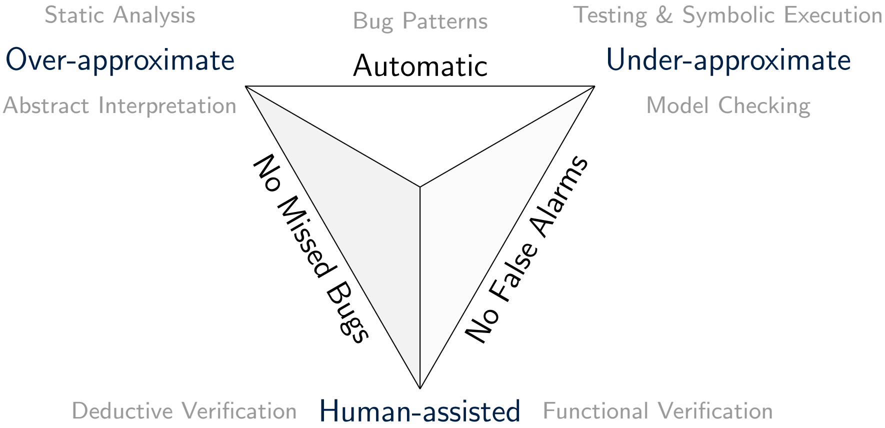
The Pyramid Model of Verification



The Pyramid Model of Verification

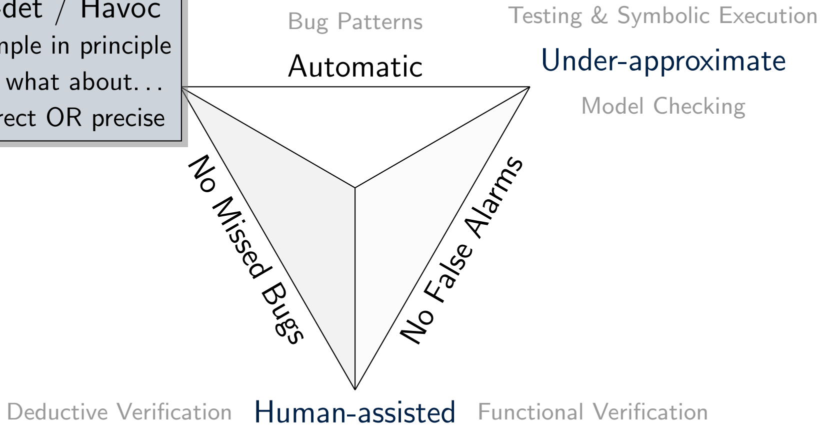


The Pyramid Model of Verification



The Over-approximate Solution : Just Over-approximate

Non-det / Havoc
+ Simple in principle
- But what about…
- Correct OR precise



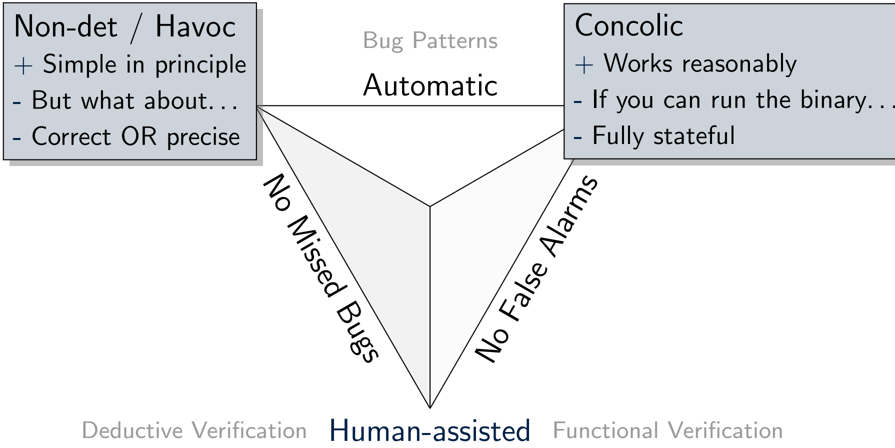
The Over-approximate Solution : Just Over-approximate

```
size_t f00(void*, size_t, size_t, struct s *)
```

The Over-approximate Solution : Just Over-approximate

```
size_t fread(void*, size_t, size_t, FILE *)
```

The Under-approximate Solution : "Concolic"



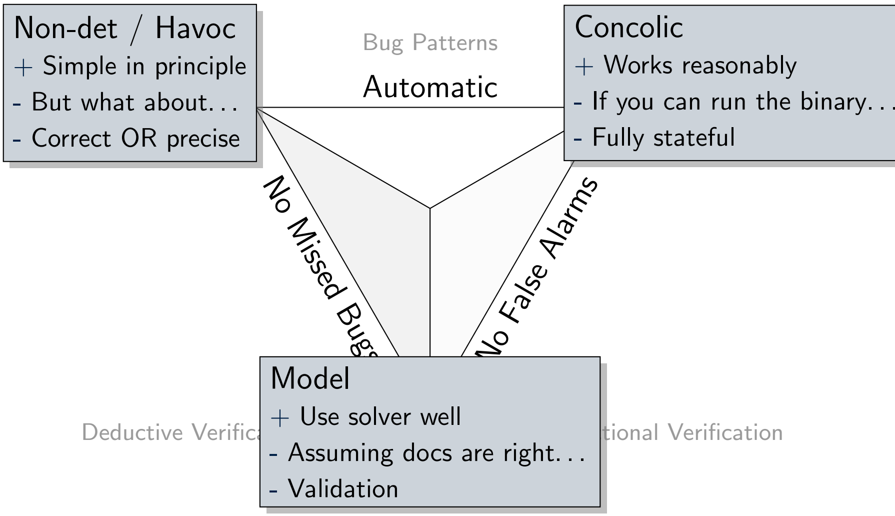
The Under-approximate Solution : “Concolic”

```
ssize_t f01(int, const void*, size_t, int,
        const struct t*, size_t)
```

The Under-approximate Solution : “Concolic”

```
ssize_t sendto(int, const void*, size_t, int,
        const struct sockaddr*, socklen_t)
```

The Human-assisted Solution : Write Models



The Human-assisted Solution : Write Models

```
void * realloc(void *ptr, size_t size)
```

Should we model...

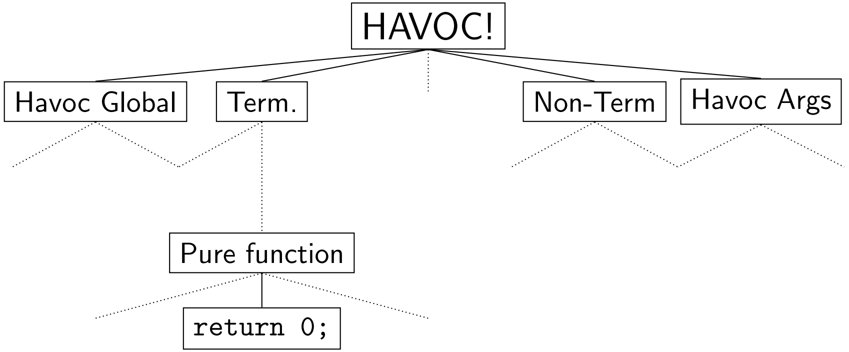
- When is `size` too much?
- Return `NULL`?
- Return `NULL` is sticky?
- Alignment of result?
- When does it return `ptr`?
- `errno` set?

Possible Approaches

1. Isn’t this what game semantics is supposed to fix?

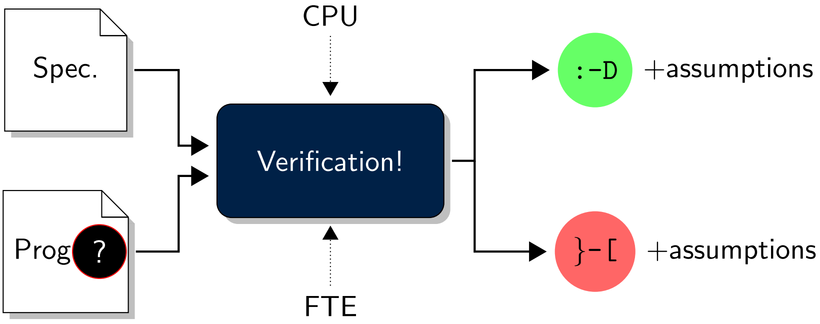
Possible Approaches

1. Isn't this what game semantics is supposed to fix?
2. Lattice-based (formula) abstraction refinement



Possible Approaches

1. Isn't this what game semantics is supposed to fix?
2. Lattice-based (formula) abstraction refinement
3. What is "the answer" anyway?



Possible Approaches

1. Isn't this what game semantics is supposed to fix?
2. Lattice-based (formula) abstraction refinement
3. What is "the answer" anyway?
4. Opaque handles $\longrightarrow$ automata?

```
FILE *fopen(const char *pathname, const char *mode);
size_t fread(void *ptr, size_t size, size_t nmemb, FILE *stream)
size_t fwrite(const void *ptr, size_t size, size_t nmemb, FILE *
int feof(FILE *stream);
int ferror(FILE *stream);
int fclose(FILE *stream);
```

Possible Approaches

1. Isn't this what game semantics is supposed to fix?
2. Lattice-based (formula) abstraction refinement
3. What is "the answer" anyway?
4. Opaque handles $\longrightarrow$ automata?
5. The spec is in the caller!

```
struct dirent *d =
  readdir(root);

if (d == NULL) {
  perror("Directory empty");
  return errno;
} else {
  ...
```

```
do {
  struct dirent *d =
    readdir(tmp);

  if (strcmp(d->d_name, "vmlinuz")
    ...
```

Possible Approaches

1. Isn't this what game semantics is supposed to fix?
2. Lattice-based (formula) abstraction refinement
3. What is "the answer" anyway?
4. Opaque handles $\longrightarrow$ automata?
5. The spec is in the caller!
6. Is modular symbolic execution impossible? Prove it!

Assuming independence is an (the only?) over-approximation. . .

Conclusions

1. The library problem is the pressing problem for practical application of verification tools
(that can be solved by theoretical advances).
2. Current approaches are not practical / cost-effective.
3. Your solution here?

Conclusions

1. The library problem is the pressing problem for practical application of verification tools
(that can be solved by theoretical advances).
2. Current approaches are not practical / cost-effective.
3. Your solution here?

Thank you for your time and attention.

Made using only Free Software