

CS 472 — Provably Correct Programming

William Mansky

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

How do we prove programs correct?

- First: what's our theorem statement?

```
// x should hold a number n
r = x;
while(x > 1){
    x = x - 1;
    r = r * x;
}
// r should be n!
```

```
Lemma fact_correct:
{ x = n }
fact_program
{ r = n! }
.
```

How do we prove programs correct?

- Second: how do we prove it?

```
{ x = n }
```

Symbolic execution

```
r = x;
```

```
{ x = n /\ r = n }
```

```
while(x > 1){
```

```
    { x = a /\ r = n * n - 1 * ... * a }
```

```
    x = x - 1;
```

```
    { x = a - 1 /\ r = n * n - 1 * ... * a }
```

```
    r = r * x;
```

```
}
```

How do we prove programs correct?

- First: what's our theorem statement?
- Second: how do we prove it?
- Third: what's our *logic*?

Program logic works a bit differently than normal logic: we want to talk about owning and using *resources*

Separation Logic: Resources

HP : P

HPQ : P $\rightarrow$ Q

...

HP : P

HPQ : P $\rightarrow$ Q

HQ : Q

Separation Logic: Resources

HP : $x \mapsto 3$

HPQ : $x \mapsto 3 \text{ } -* \text{ } x \mapsto 5$

...

~~HP : $x \mapsto 3$~~

HPQ : $x \mapsto 3 \text{ } -* \text{ } x \mapsto 5$

HQ : $x \mapsto 5$

Separation Logic: Resources

- Logical facts like $x = 2$, $x < 5$, etc. stay true when they're true: we don't have to "use them up" to prove things about x
- But points-to assertions *do* get used up!
- $p \mapsto v$ is more like a *resource* than a logical statement: we can pass it around between functions, but once we use it to prove something, it's gone
- We'll need special tactics to manage resources in a program!

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.

Practical Separation Logic: Iris



- Tool #2: Iris, a separation logic framework inside Rocq
 - Default language is a simple functional language with memory
 - Proofs combine *symbolic execution* of programs with tactics for using/proving facts about resources
- Iris has lots of extensions, applications, and target languages. We'll look at a few!

Setting Up Iris (version 4.4.0)

- Install via opam:

```
opam repo add rocq-released https://rocq-prover.github.io/opam/released/
```

```
opam install coq-iris-heap-lang
```

(more instructions at <https://gitlab.mpi-sws.org/iris/iris/#working-with-iris>)

- If it's working, you should be able to run this line in your IDE:
`Require Import iris.heap_lang.proofmode.`

Setting Up Iris – Special Characters

- There's a lot of fancy notation and special characters in Iris!
- My preferred font is [DejaVu Sans Mono](#)
- You can find instructions for setting up your editor at <https://gitlab.mpi-sws.org/iris/iris/-/blob/master/docs/editor.md>
- You can also do `Require Import iris.bi.ascii.` to enable ASCII notations (full list at <https://gitlab.mpi-sws.org/iris/iris/-/blob/master/iris/bi/ascii.v>)

Iris Resources

- We will be working from my fork of the Iris Tutorial (<https://github.com/mansky1/iris-tutorial/>)
- A really good overview of the whole system from the ground up is available at <https://arxiv.org/pdf/2105.12077.pdf>
- There are links to more tutorials and lecture notes at <https://iris-project.org/#learning>
- A list of tactics is at https://gitlab.mpi-sws.org/iris/iris/-/blob/master/docs/proof_mode.md

Questions

Nobody has responded yet.

Hang tight! Responses are coming in.