Why we like programming

- **edit**
- **compile**
- **debug**
- **craving**
- **instant feedback**
- **experience coolness**

---

Why we hate proofs

- **write**
- **syntax check**
- **grade (2wks)**
- **no 2nd chance**
- **preplexion frustration**
- **repellance**

---

**If only** there were an IDE for proofs...
z' = xty
= (z:=1; z' = z×xty)

z' = z×xty
= (if y=0 then ok else y>0 ⇒ z' = z×xty)

(y>0 ⇒ z' = z×xty)

= (if y mod 2 = 0
then (x:=x×x; y:=y/2; z' = z×xty)
else (z:=z×x; y:=y-1; z' = z×xty))

unsigned fast_exp(unsigned x, unsigned y)
{
    unsigned z = 1;

    /* compute z×x×y, store in z */
    while (y != 0)
    {
        /* assume y>0, compute z×x×y, store in z */
        if (y%2 == 0) {
            x = x×x; y = y/2;
        } else {
            z = z×x; y = y-1;
        }
    }
    return z;
Useful?

Industrial perspective:

✔ unambiguous language for internal documentation
✔ critical, core fragments of code
✔ delicate hand-optimizations of originally simple algorithms
✘ not substitute for testing; not panacea
✔ good addition to toolset

❓ but how many colleagues will use it?
Useful?

Educational perspective:

✔ IDE for discrete math homework!

✔ clear rules of the game—no second-guessing the TA

⇒ graduates mind it less and use it more at work

✘ not substitute for other CS/SE homework!
Future Work

☞ if a step fails, want explanation/counter-example (see link #4)

☞ if a step works by automation, want details

☞ syntax+type check, informative messages, heuristics, help

☞ bridge to/from code files; proof templates

☞ model objects, pointers
Links


2. We model programs with the theory from:


3. We currently use this theorem prover at the back:


4. It would be nice to use a counterexample generator such as: