CSC110 Lecture 9: Programming and Proofs

David Liu, Department of Computer Science

*Navigation tip for web slides: press **?** to see keyboard navigation controls.*

Announcements and Today's Plan

Assignment 1 due tomorrow (noon Eastern Time)

- Assignment 1 has been posted!
  - Check the FAQ (+ corrections) page
  - Additional TA office hours (schedule on Quercus)
  - **(NEW)** Academic Integrity in CSC110 advice page
  - **(NEW)** Review the Assignment Policies page, including grace credit policy
- For your submission:
  - Submit all required files (including the `a1.tex` file!)
  - Make sure your Python files all run (right-click -> “Run File in Python Console”)
  - Run `doctest` and `python_ta` on your Python files (using provided code)
  - You can submit on MarkUs multiple times!

Other announcements

- Join a Recognized Study Group
- Term Test 1 info has been posted!
  - Info on Quercus:
    - Test time and location (not MY 150!)
    - Test coverage
    - Advice for advice
  - Review provided reference sheet

Story so far

Data: data types, literals, basic operators, comprehensions

Functions: using built-in functions, methods; defining our own (top-level) functions

Logic: translating boolean expressions, filtering comprehensions, if statements

Our code is getting more complex, and will only continue to do so in the coming weeks.

How can we be confident that our code is correct?

How can we be confident that our code is correct?

1. **Testing**: in the Python console; `doctest`; `pytest`; property-based tests (`hypothesis`)
2. **Code reuse**: reuse functions that we've already defined (and are confident is correct), rather than copying the same code in multiple places.
3. **Simplification**: translate more complex code structures into simpler ones (e.g., simplifying if statements)
4. **(TODAY) Proof**: Write a formal proof that a function's implementation is correct for all valid inputs.

Today you'll learn to...

1. Understand unfamiliar mathematical definitions, and create new predicates based on these definitions.
   - Problem domain: basic number theory (divisibility and prime numbers)
2. Use definitions to simplify and expand statements in predicate logic.
3. Write proofs of simple statements using these definitions.
4. Connect mathematical statements to improve/develop algorithms.

Working with Definitions

Definitions are our way of taking a problem domain and breaking it down precisely into understandable parts.

A definition

Definition. Let $n, d \in \mathbb{Z}$. We say that d **divides** n when there exists a $k \in \mathbb{Z}$ such that $n = dk$.

Note: using this definition, 0 divides 0.

(In fact, every number divides 0!)

A definition (broken down)

Let $n, d \in \mathbb{Z}$.	The domain of the definition
We say that d **divides** n	The term being defined
when	Connection word
there exists a $k \in \mathbb{Z}$ such that $n = dk$	The body of the definition

Translating a definition into a predicate

Every definition can be turned into a predicate that asks, “Does this definition apply to these values?”

Let $n, d \in \mathbb{Z}$. We say that d **divides** n when there exists a $k \in \mathbb{Z}$ such that $n = dk$.

$$Divides(d, n) : \ \exists k \in \mathbb{Z}, \ n = dk \qquad \text{where } d, n \in \mathbb{Z}$$

$Divides(d, n)$ is often written as $d \mid n$.

Note: $d \mid n$ vs. $n \div d$

$\mid$ is a **predicate**: $d \mid n$ evaluates to a boolean.

$\div$ is an **arithmetic operator**: $n \div d$ evaluates to a number.

Expanding definitions

"For every integer x, if x divides 10, then it also divides 100."

$$\forall x \in \mathbb{Z},\ x \mid 10 \Rightarrow x \mid 100$$

$$\forall x \in \mathbb{Z},\ \underline{(\exists k_1 \in \mathbb{Z},\ 10 = k_1 x)} \Rightarrow \underline{(\exists k_2 \in \mathbb{Z},\ 100 = k_2 x)}$$

Proofs

A proof is a structured argument for convincing someone else why a statement is True.

A program is a structured text for telling a computer what to compute.

Just as we learn a programming language to write programs, we learn a proof language to write proofs.

Two things can make proofs challenging:

1. Proofs use both English and mathematical notation, and the language is less rigid than a programming language.
    - It’s up to us to write our proofs in a structured way.
2. There’s no “proof interpreter” we can run to check our proofs.
    - It’s up to us to check our proofs carefully.

Proof example 1

Prove the following statement: every integer is divisible by 1.

Translation:

$$\forall n \in \mathbb{Z},\ 1 \mid n$$

$$\forall n \in \mathbb{Z},\ \underline{\exists k \in \mathbb{Z},\ n = 1 \cdot k}$$

Proof example 1, continued

$$\forall n \in \mathbb{Z},\ \underline{\exists k \in \mathbb{Z},\ n = 1 \cdot k}$$

Proof (structure only).
Let $n \in \mathbb{Z}$.
Let $k =$ _____.
We want to show that $n = 1 \cdot k$
.
...

Proof.
Let $n \in \mathbb{Z}$.
Let $k = n$.
We want to show that $n = 1 \cdot k$
.
That's just a calculation:
$1 \cdot k = 1 \cdot n = n$, since $k = n$.

Proof example 2

Prove the following statement: for all positive integers n and d, if $d \mid n$ then $d \leq n$.

Translation:

$$\forall n, d \in \mathbb{Z}^+,\ d \mid n \Rightarrow d \leq n$$

$$\forall n, d \in \mathbb{Z}^+,\ \underline{\left(\exists k \in \mathbb{Z},\ n = d \cdot k\right)} \Rightarrow d \leq n$$

Proof example 2, continued

$$\forall n, d \in \mathbb{Z}^+, \; \underline{\left(\exists k \in \mathbb{Z}, \; n = d \cdot k\right)} \Rightarrow d \leq n$$

Proof.
Let $n, d \in \mathbb{Z}^+$.
Assume there exists $k \in \mathbb{Z}$ such that $n = d \cdot k$.
We want to show that $d \leq n$.
...

Proof example 2, continued

$$\forall n, d \in \mathbb{Z}^+,\ \underline{(\exists k \in \mathbb{Z},\ n = d \cdot k)} \Rightarrow d \leq n$$

Proof. Let $n, d \in \mathbb{Z}^+$. Assume there exists $k \in \mathbb{Z}$ such that $n = d \cdot k$. We want to show that $d \leq n$.

Since $n > 0$ and $d > 0$ and $k = \frac{n}{d}$, we know $k > 0$.

Since $k \in \mathbb{Z}$ and $k > 0$, we know $k \geq 1$.

Then $d \leq d \cdot k$, since $k \geq 1$.

By our assumption, $n = d \cdot k$, and so $d \leq n$.

Proof example 2 (alternate visualization 1)

$$\forall n, d \in \mathbb{Z}^+,\ \underline{\left(\exists k \in \mathbb{Z},\ n = d \cdot k\right)} \Rightarrow d \leq n$$

Proof. Let $n, d \in \mathbb{Z}^+$. Assume there exists $k \in \mathbb{Z}$ such that $n = d \cdot k$. We want to show that $d \leq n$. Since $n > 0$ and $d > 0$ and $k = \frac{n}{d}$, we know $k > 0$. Since $k \in \mathbb{Z}$ and $k > 0$, we know $k \geq 1$. Then $d \leq d \cdot k$, since $k \geq 1$. By our assumption, $n = d \cdot k$, and so $d \leq n$.

Proof example 2 (alternate visualization 2)

$$\forall n, d \in \mathbb{Z}^+,\ \underline{(\exists k \in \mathbb{Z},\ n = d \cdot k)} \Rightarrow d \leq n$$

Proof. Let $n, d \in \mathbb{Z}^+$. Assume there exists $k \in \mathbb{Z}$ such that $n = d \cdot k$. We want to show that $d \leq n$.

Then:

Conclusion	**Justification**
$k > 0$	$n > 0$ and $d > 0$ and $k = \frac{n}{d}$ >
$k \geq 1$	$k \in \mathbb{Z}$ and $k > 0$
$d \leq d \cdot k$	$k \geq 1$
$d \leq n$	$d \leq d \cdot k$ and $n = d \cdot k$ (assumption)

Exercise 1: Practice with proofs

Divisibility and the Quotient-Remainder Theorem

```python
def divides(d: int, n: int) -> bool:
    """Return whether d divides n.
    """
    if d == 0:
        return n == 0
    else:
        return n % d == 0
```

Why does this work?

- If branch (`d == 0`):
  - in this case, $0 \mid n \Leftrightarrow n = 0$.
- Else branch (`d != 0`):
  - in this case, $d \mid n \Leftrightarrow n \,\%\, d = 0$

Quotient-Remainder Theorem

Theorem (Quotient-Remainder Theorem). For all $n, d \in \mathbb{Z}$, if $d \neq 0$ there exist unique integers $q \in \mathbb{Z}$ and $r \in \mathbb{N}$ such that $n = qd + r$ and $0 \leq r < |d|$.

We say that q is the **quotient** when n is divided by d, and that r is the **remainder** when n is divided by d.

See Section 4.5 for further details (and more example proofs).

Prime numbers

Definition. Let $p \in \mathbb{Z}$. We say p is **prime** when it is greater than 1 and the only natural numbers that divide it are 1 and itself.

“greater than 1”:

- $p > 1$

“the only natural numbers that divide it are 1 and itself”:

- $\forall d \in \mathbb{N},\ d \mid p \Rightarrow d = 1 \vee d = p$

(Note: the converse, $d = 1 \vee d = p \Rightarrow d \mid p$, holds for all integers.)

From definition to predicate, prime edition

$$\begin{aligned}&IsPrime(p):\\&\quad p > 1 \land \big(\forall d \in \mathbb{N},\ d \mid p \Rightarrow d = 1 \lor d = p\big)\\&\quad \text{where } p \in \mathbb{Z}\end{aligned}$$

$$\begin{aligned}&IsPrime(p):\\&\quad p > 1 \land \big(\forall d \in \mathbb{N},\ \underline{(\exists k \in \mathbb{Z},\ p = kd)} \Rightarrow d = 1 \lor d = p\big)\\&\quad \text{where } p \in \mathbb{Z}\end{aligned}$$

Translating into Python

$$
\begin{aligned}
&\mathit{IsPrime}(p): \\
&\quad p > 1 \land \big(\forall d \in \mathbb{N},\, d \mid p \Rightarrow d = 1 \lor d = p\big) \\
&\quad \text{where } p \in \mathbb{Z}
\end{aligned}
$$

```python
def is_prime(p: int) -> bool:
    """Return whether p is prime."""
    possible_divisors = range(1, p + 1)  # {1, 2, ..., p}
    return (
        p > 1 and
        all({d == 1 or d == p
             for d in possible_divisors if divides(d, p)})
    )
```

```python
def is_prime(p: int) -> bool:
    """Return whether p is prime."""
    possible_divisors = range(1, p + 1)  # {1, 2, ..., p}
    return (
        p > 1 and
        all({d == 1 or d == p
            for d in possible_divisors if divides(d, p)})
    )
```

This algorithm checks p possible divisors. Can we do better?

```
from math import floor, sqrt


def is_prime(p: int) -> bool:
    """Return whether p is prime."""
    possible_divisors = range(2, floor(sqrt(p)) + 1)
    return (
        p > 1 and
        all({not divides(d, p) for d in possible_divisors}
    )
```

How do we **know** this version of `is_prime` is still correct?

Prove the following statement:

$$\forall p \in \mathbb{Z},\ Prime(p) \Leftrightarrow \big(p > 1 \wedge (\forall d \in \mathbb{N},\ 2 \leq d \leq \sqrt{p} \Rightarrow d \nmid p)\big)$$

This is a bigger statement! Let’s break down the structure.

$$\forall p \in \mathbb{Z},\ Prime(p) \Leftrightarrow \big(p > 1 \wedge (\forall d \in \mathbb{N},\ 2 \le d \le \sqrt{p} \Rightarrow d \nmid p)\big)$$

Proof. Let $p \in \mathbb{Z}$.

Part 1: Prove that IF $Prime(p)$, THEN $p > 1$ and $\forall d \in \mathbb{N},\ 2 \le d \le \sqrt{p} \Rightarrow d \nmid p$.

...

Part 2: Prove that IF $p > 1$ and $\forall d \in \mathbb{N},\ 2 \le d \le \sqrt{p} \Rightarrow d \nmid p$, THEN $Prime(p)$.

...

Proof (Part 1, $Prime(p) \Rightarrow \ldots$)

Part 1: Prove that IF $Prime(p)$, THEN $p > 1$ and $\forall d \in \mathbb{N},\ 2 \leq d \leq \sqrt{p} \Rightarrow d \nmid p$.

Assume p is prime, i.e. (expanding the definition) that

- $p > 1$, and
- $\forall d_1 \in \mathbb{N},\ d_1 \mid p \Rightarrow d_1 = 1 \vee d_1 = p$

We want to show that:

- $p > 1$, and
- $\forall d \in \mathbb{N},\ 2 \leq d \leq \sqrt{p} \Rightarrow d \nmid p$

Proof (Part 1, $Prime(p) \Rightarrow \ldots$)

We want to show that:

- $p > 1$, and
- $\forall d \in \mathbb{N},\ 2 \leq d \leq \sqrt{p} \Rightarrow d \nmid p$

First, we know $p > 1$, as this is one of our assumptions.

For the second part: let $d \in \mathbb{N}$, and assume $2 \leq d \leq \sqrt{p}$. We want to prove that $d \nmid p$.

Since $d \geq 2$, we know $d \neq 1$.

Since $p > 1$, we know $\sqrt{p} < p$, and so $d \leq \sqrt{p} < p$.

Therefore $d \neq 1$ and $d \neq p$, and so $d \nmid p$ (by the definition of prime).

Proof (Part 2, $\cdots \Rightarrow Prime(p)$)

Part 2: Prove that IF $p > 1$ and $\forall d \in \mathbb{N},\ 2 \leq d \leq \sqrt{p} \Rightarrow d \nmid p$, THEN $Prime(p)$.

Assume that:

- $p > 1$, and
- $\forall d \in \mathbb{N},\ 2 \leq d \leq \sqrt{p} \Rightarrow d \nmid p$

We want to show that p is prime, i.e., that

- $p > 1$, and
- $\forall d_1 \in \mathbb{N},\ d_1 \mid p \Rightarrow d_1 = 1 \vee d_1 = p$

Proof (Part 2, $\cdots \Rightarrow Prime(p)$)

We want to show that p is prime, i.e., that

- $p > 1$, and
- $\forall d_1 \in \mathbb{N},\ d_1 \mid p \Rightarrow d_1 = 1 \vee d_1 = p$

First, we know $p > 1$, as this is one of our assumptions.

For the second part: let $d_1 \in \mathbb{N}$, and assume $d_1 \mid p$, i.e., that $\exists k \in \mathbb{Z}$ such that $p = d_1 k$. We want to show that $d_1 = 1 \vee d_1 = p$.

Since $d_1 \mid p$, by our earlier assumption we know $d_1 < 2$ or $d_1 > \sqrt{p}$.

This "or" gives us two cases in our proof.

Proof (Part 2, $\cdots \Rightarrow Prime(p)$)

We want to show that $d_1 = 1 \vee d_1 = p$.

Since $d_1 \mid p$, by our earlier assumption we know $d_1 < 2$ or $d_1 > \sqrt{p}$.

Case 1: assume $d_1 < 2$.
Then since $d_1 \in \mathbb{N}$, we know $d_1 = 1$ or $d_1 = 0$.
Then since $d_1 \mid p$, we know $d_1 \neq 0$, and so $d_1 = 1$.

Proof (Part 2, $\cdots \Rightarrow Prime(p)$)

We want to show that $d_1 = 1 \vee d_1 = p$.

Since $d_1 \mid p$, by our earlier assumption we know $d_1 < 2$ or $d_1 > \sqrt{p}$.

Case 2: assume $d_1 > \sqrt{p}$. Now we calculate:

$$
\begin{aligned}
d_1 k &= p && \text{(by definition of divisibility)} \\
k &= \frac{p}{d_1} \\
k &< \frac{p}{\sqrt{p}} && \text{(since } d_1 > \sqrt{p}\text{)} \\
k &< \sqrt{p}
\end{aligned}
$$

We want to show that $d_1 = 1 \vee d_1 = p$.

Case 2: assume $d_1 > \sqrt{p}$.

...

So $k < \sqrt{p}$.

By our earlier assumption again, since $k \mid p$, we know $k < 2$ or $k > \sqrt{p}$.

Since $k < \sqrt{p}$, this means $k < 2$.

By the same reasoning as earlier, $k = 1$.

Since $p = d_1 k$, we conclude that $d_1 = p$.

Putting it all together

$$\forall p \in \mathbb{Z},\ Prime(p) \Leftrightarrow \big(p > 1 \land (\forall d \in \mathbb{N},\ 2 \leq d \leq \sqrt{p} \Rightarrow d \nmid p)\big)$$

$$\forall p \in \mathbb{Z},\ Prime(p) \Leftrightarrow \underline{\text{is\_prime(p) returns True}}$$

```python
def is_prime(p: int) -> bool:
    """Return whether p is prime."""
    possible_divisors = range(2, floor(sqrt(p)) + 1)
    return (
        p > 1 and
        all({not divides(d, p) for d in possible_divisors}
    )
```

```python
def is_prime(p: int) -> bool:
    """Return whether p is prime."""
    possible_divisors = range(2, floor(sqrt(p)))  # CHANGE
    return (
        p > 1 and
        all({not divides(d, p) for d in possible_divisors}
    )
```

$$\forall p \in \mathbb{Z},\ Prime(p) \Leftrightarrow \big(p > 1 \wedge (\forall d \in \mathbb{N},\ 2 \leq d \leq \sqrt{p} \Rightarrow d \nmid p)\big)$$

$$\forall p \in \mathbb{Z},\ Prime(p) \Leftrightarrow \big(p > 1 \wedge (\forall d \in \mathbb{N},\ 2 \leq d < \sqrt{p} \Rightarrow d \nmid p)\big)$$

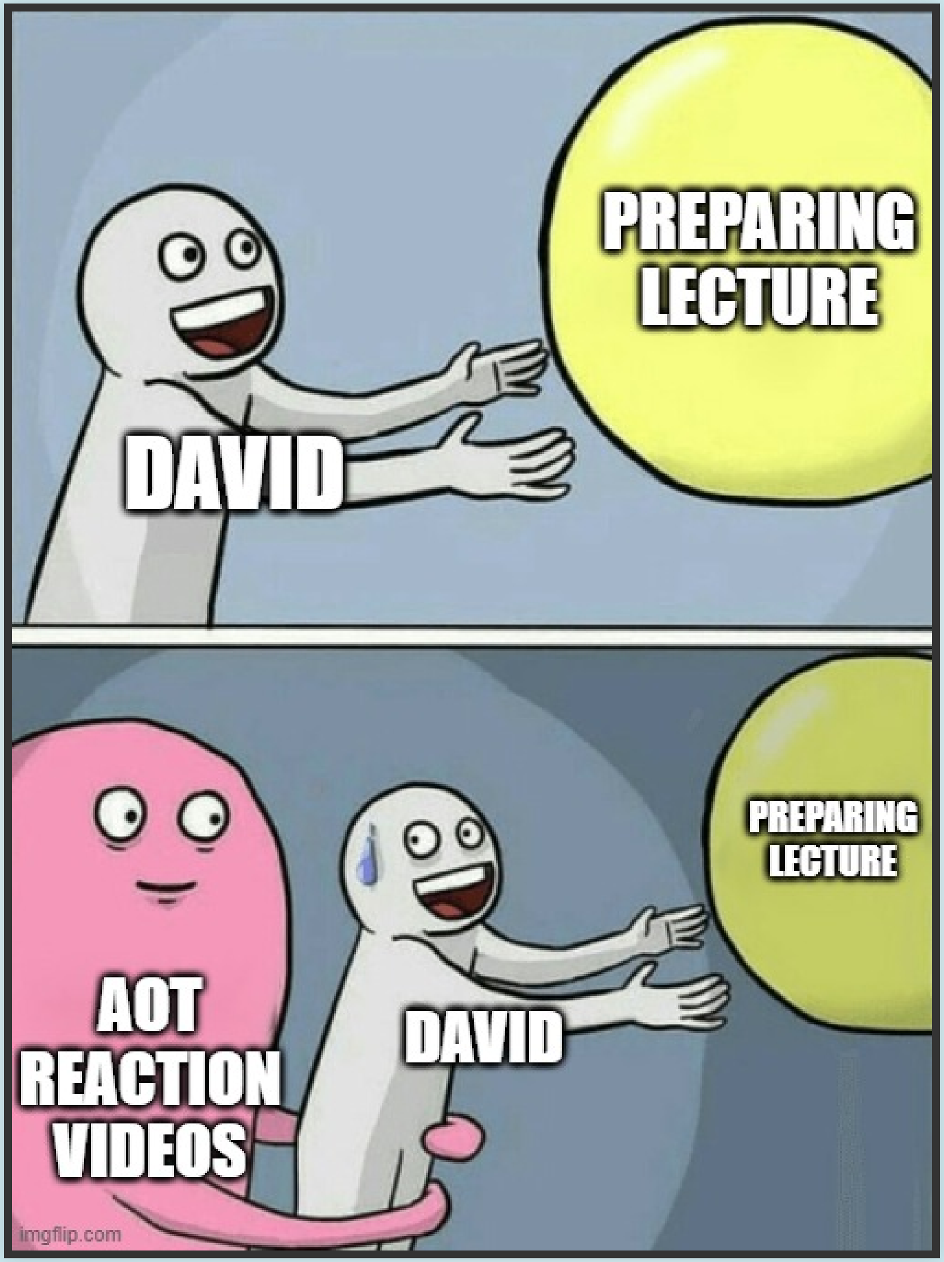
Summary

Today you learned to...

1. Understand unfamiliar mathematical definitions, and create new predicates based on these definitions.
   - Problem domain: basic number theory (divisibility and prime numbers)
2. Use definitions to simplify and expand statements in predicate logic.
3. Write proofs of simple statements using these definitions.
4. Connect mathematical statements to improve/develop algorithms.

Homework

- Readings:
  - Today: 4.6, 4.7
  - Next class: 5.1 (to be posted soon)
- Finish up Assignment 1
- Review for Term Test 1


DAVID
PREPARING LECTURE
AOT REACTION VIDEOS
DAVID
PREPARING LECTURE
imgflip.com