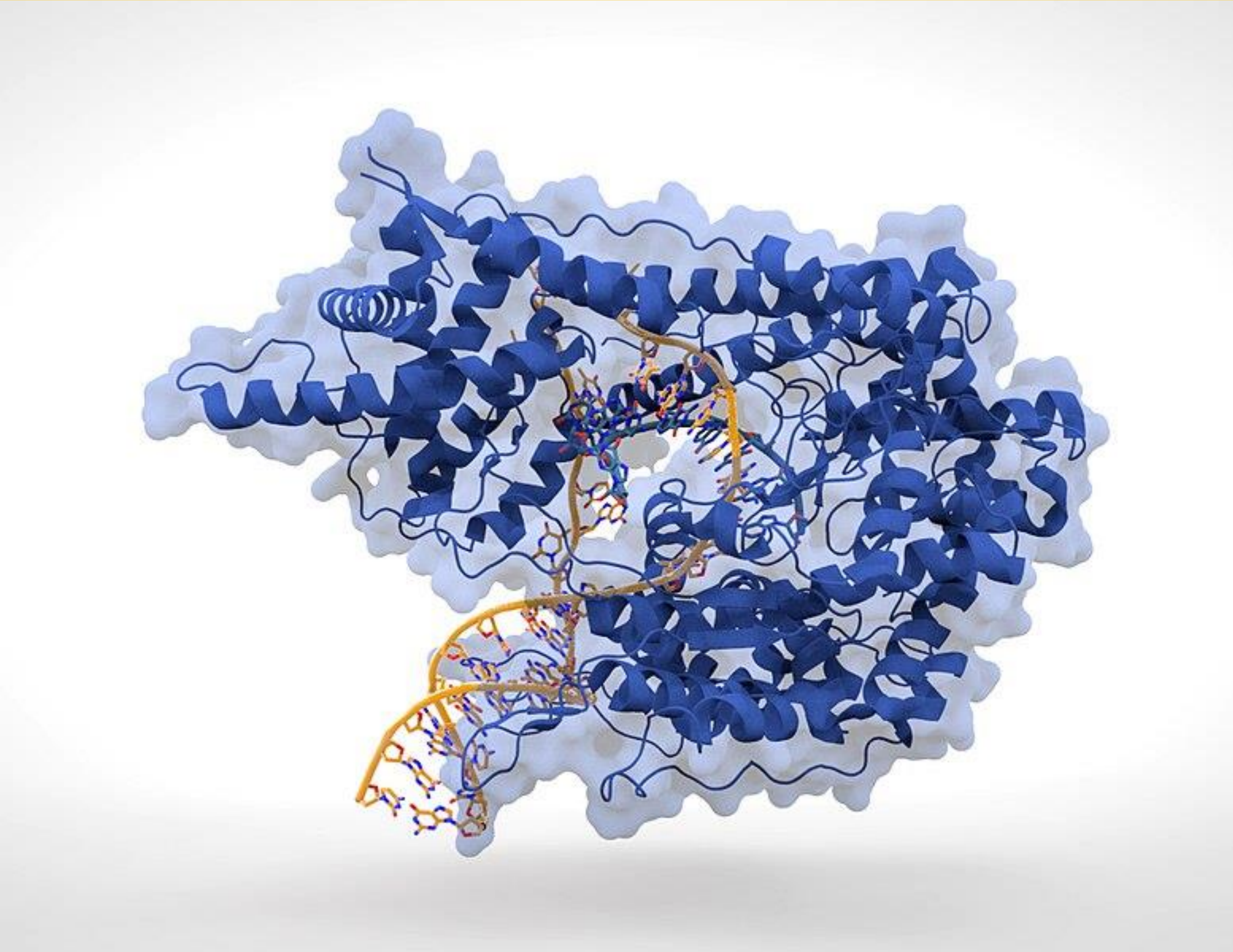


Problem Set 3 is due
This Friday, Feb 7 (10pm)

Class 8: *Circuit Size*

University of Virginia
cs3120: DMT2
Wei-Kai Lin

PS1: Recitation

Problem 5 *Claim: All cows have the same number of spots.*

From our experience (from farms or videos or other), the claim is false.[1] However, here is a proof by induction.

Proof.

1. Predicate $P(n)$: any $n+1$ cows have the same number of spots.
2. Base case, $P(0)$: any 1 cow have the same number of spots. This holds because there is only 1 cow, and its number of spots is the same.
3. Inductive case, let $n \in \mathbb{N}$.
   (a) By induction hypothesis, suppose $P(n)$ holds. That is, any $n+1$ cows have the same number of spots.
   (b) Consider any $n+2$ cows. Let $c_1, c_2, \ldots, c_{n+2}$ be (the names of) the cows.
   (c) By $P(n)$, the first $n+1$ cows, $c_1, c_2, \ldots, c_{n+1}$, have the same number of spots. Let x be the number.
   (d) By $P(n)$, the last $n+1$ cows, $c_2, \ldots, c_{n+2}$, have the same number of spots. Let y be the number.
   (e) Let z be the number of spots of the cow c_2.
   (f) We have $x = z$ because c_2 is in the first $n+1$.
   (g) We have $y = z$ because c_2 is in the last $n+1$.
   (h) By $x = y = z$, all $n+2$ cows have the same number of spots. That is, $P(n+1)$ holds.

All cows have the same number of spots (?)

Predicate, P(n): for any natural number n,
any n + 1 cows have the same number of spots.

Base case, P(0):

any 0 + 1 = 1 cows have the same number of spots.

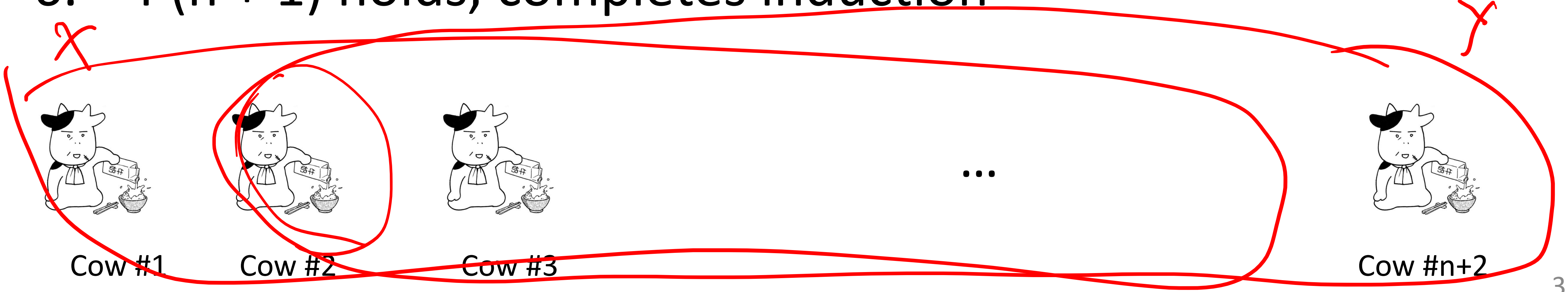
Image credit: Hyakushō Kizoku (manga, “The Noble Farmer”)

Inductive case, P(n) holds for some natural number n:

any n + 1 cows have the same number of spots.

1. Consider any n + 2 cows, arrange in a line
2. By P(n), first n + 1 cows all have x spots
3. By P(n), last n + 1 cows all have y spots
4. Cow #2 is in both the first n + 1 and last n + 1

   x = y
5. x = y because #2 have the same number of spots, P(0)
6. P(n + 1) holds, completes induction



Cow #1 Cow #2 Cow #3 Cow #n+2

Inductive case, P(n) holds for some natural number n:
any n + 1 cows have the same number of spots.

1. Consider any n + 2 cows, arrange in a line
2. By P(n), first n + 1 cows all have x spots
3. By P(n), last n + 1 cows all have y spots
4. Cow #2 is in both the first n + 1 and last n + 1
5. x = y because #2 have the same number of spots, P(0)
6. P(n + 1) holds, completes induction

Wrong! Where?

Attempt I:

P(0): any 1 cows have the same number of spots.

Holds by definition of “same number”, that is “=“

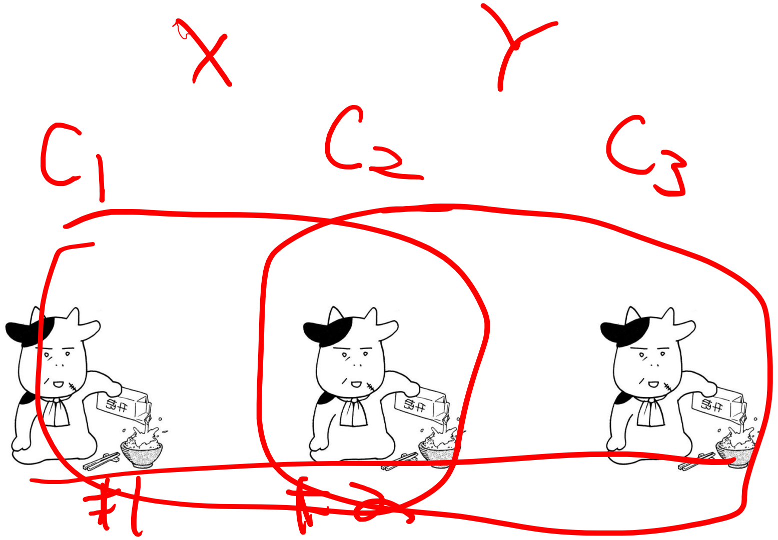
Attempt II:

Assume P(1) holds. To argue P(2):

any 3 cows have the same number of spots.

2+1

Holds:



Inductive case, P(n) holds for some natural number n:

any n + 1 cows have the same number of spots.

1. Consider any n + 2 cows, arrange in a line
2. By P(n), first n + 1 cows all have x spots
3. By P(n), last n + 1 cows all have y spots
4. Cow #2 is in both the first n + 1 and last n + 1
5. x = y because #2 have the same number of spots, P(0)
6. P(n + 1) holds, completes induction

Attempt III:

Assume P(0) holds. To argue P(1):

any 2 cows have the same number of spots.

x

y

Inductive case, P(n) holds for some natural number n:

any n + 1 cows have the same number of spots.

1. Consider any n + 2 cows, arrange in a line
2. By P(n), first n + 1 cows all have x spots
3. By P(n), last n + 1 cows all have y spots
4. Cow #2 is in both the first n + 1 and last n + 1
5. x = y because #2 have the same number of spots, P(0)
6. P(n + 1) holds, completes induction

ChatGPT: prompt

```
From our experience (from farms or videos or other), the claim is
false.\footnote{
\url{https://thedairyalliance.com/blog/udderly-unusual-cow-facts}
}
However, here is a proof by induction.
\begin{proof}
\begin{enumerate}
\item
Predicate $P(n)$: any $n+1$ cows have the same number of spots.
\item
Base case, $P(0)$: any $1$ cow have the same number of spots. This
holds because there is only 1 cow, and its number of spots is the
same.
\item
Inductive case, let $n \in \N$.
\begin{enumerate}
    \item By induction hypothesis, suppose $P(n)$ holds. That is, any
$n+1$ cows have the same number of spots.
    \item Consider any $n+2$ cows. Let $c_1, c_2, ..., c_{n+2}$ be (the
names of) the cows.
    \item By $P(n)$, the first $n+1$ cows, $c_1, c_2, ..., c_{n+1}$, have
the same number of spots. Let $x$ be the number.
    \item By $P(n)$, the last $n+1$ cows, $c_2, ..., c_{n+2}$, have the
same number of spots. Let $y$ be the number.
    \item Let $z$ be the number of spots of the cow $c_2$.
    \item We have $x = z$ because $c_2$ is in the first $n+1$.
    \item We have $y = z$ because $c_2$ is in the last $n+1$.
    \item By $x = y = z$, all $n+2$ cows have the same number of
spots. That is, $P(n+1)$ holds.
\end{enumerate}
\end{enumerate}
\end{proof}

Please briefly explain why the proof is wrong (maybe by pointing out
some incorrect steps).
```

ChatGPT response

The proof provided is attempting to use mathematical induction to show that any collection of cows, with an increasing number of cows at each step, must have the same number of spots. However, there is a critical flaw in the proof. Here's why the proof is incorrect:

Problem with the Inductive Step:

1. **Base Case**: The base case $P(0)$ states that any 1 cow has the same number of spots, which is trivially true because there is only one cow, so the number of spots is the same (just that cow's spots).
2. **Inductive Hypothesis**: The hypothesis $P(n)$ assumes that for any collection of $n+1$ cows, they have the same number of spots.
3. **Inductive Step**:
   - The proof now considers $n+2$ cows, labeled $c_1, c_2, \ldots, c_{n+2}$.
   - The proof asserts that the first $n+1$ cows have the same number of spots, and the last $n+1$ cows also have the same number of spots.

   Here's where the mistake happens:

   - The inductive argument relies on the assumption that the spots of the cows overlap. Specifically, it claims that the cow c_2 is in both the first and the last set of $n+1$ cows, and by this overlapping position, the number of spots of all the cows can be made equal.

   This reasoning is **incorrect** because:

   - **Overlapping does not guarantee equality**. The fact that c_2 is in both the first and last $n+1$ cows means that it has the same number of spots as itself, but this does not necessarily imply that the other cows in the first and last sets (i.e., $c_1, c_3, \ldots, c_{n+1}$) all have the same number of spots as each other. The assumption that the number of spots of the other cows in the two sets will match is not supported by the induction hypothesis.

   In fact, it's entirely possible for the first $n+1$ cows to have a different number of spots than the last $n+1$ cows. The overlapping cow only ensures that c_2 has the same number of spots in both groups, not that all cows in both groups must have the same number of spots.

Conclusion:

The inductive step wrongly assumes that the overlap between the two sets of cows (first $n+1$ and last $n+1$) forces the number of spots for all the cows to be the same. This assumption doesn't hold in general, so the proof fails. The proof does not properly account for the possibility that the number of spots could vary among cows that are not part of the overlap. Therefore, the claim that all cows must have the same number of spots is incorrect.

ChatGPT: base case

The proof provided is attempting to use mathematical induction to show that any collection of cows, with an increasing number of cows at each step, must have the same number of spots. However, there is a critical flaw in the proof. Here's why the proof is incorrect:

Problem with the Inductive Step:

1. **Base Case**: The base case $P(0)$ states that any 1 cow has the same number of spots, which is trivially true because there is only one cow, so the number of spots is the same (just that cow's spots).

ChatGPT: inductive step

2. **Inductive Hypothesis:** The hypothesis $P(n)$ assumes that for any collection of $n+1$ cows, they have the same number of spots.

3. **Inductive Step:**

   - The proof now considers $n+2$ cows, labeled $c_1, c_2, \ldots, c_{n+2}$.
   - The proof asserts that the first $n+1$ cows have the same number of spots, and the last $n+1$ cows also have the same number of spots.

   Here's where the mistake happens:

   - The inductive argument relies on the assumption that the spots of the cows overlap. Specifically, it claims that the cow c_2 is in both the first and the last set of $n+1$ cows, and by this overlapping position, the number of spots of all the cows can be made equal.

ChatGPT: inductive step mistake?

c2

Overlapping does not guarantee equality

…

The overlapping cow only ensures that c2 has the same number of spots in both groups, not that all cows in both groups must have the same number of spots.

Here's where the mistake happens:

- The inductive argument relies on the assumption that the spots of the cows overlap. Specifically, it claims that the cow c_2 is in both the first and the last set of $n+1$ cows, and by this overlapping position, the number of spots of all the cows can be made equal.

This reasoning is incorrect because:

- **Overlapping does not guarantee equality.** The fact that c_2 is in both the first and last $n+1$ cows means that it has the same number of spots as itself, but this does not necessarily imply that the other cows in the first and last sets (i.e., $c_1, c_3, \ldots, c_{n+1}$) all have the same number of spots as each other. The assumption that the number of spots of the other cows in the two sets will match is not supported by the induction hypothesis.

In fact, it's entirely possible for the first $n+1$ cows to have a different number of spots than the last $n+1$ cows. The overlapping cow only ensures that c_2 has the same number of spots in both groups, not that all cows in both groups must have the same number of spots.

ChatGPT: inductive step mistake?

Overlapping does not guarantee equality

...

The overla
ensures th
same num
both grou
cows in bo
have the same number of spots.

Transitivity of equality:

If x = y and y = z, then x = z.

ChatGPT violates transitivity of equality!

Here's where the mistake happens:

- The inductive argument relies on the assumption that the spots of the cows overlap. Specifically, it claims that the cow c_2 is in both the first and the last set of $n + 1$ cows,
  s can be made equal.

  the first and last
  t this does not
  , $c_3, \ldots, c_{n+1}$) all have
  umber of spots of the
  on hypothesis.

In fact, it's entirely possible for the first $n + 1$ cows to have a different number of spots than the last $n + 1$ cows. The overlapping cow only ensures that c_2 has the same number of spots in both groups, not that all cows in both groups must have the same number of spots.

Problem 5, closer look

Problem 5 *Claim: All cows have the same number of spots.*

From our experience (from farms or videos or other), the claim is false.[1] However, here is a proof by induction.

Proof.

1. Predicate $P(n)$: any $n+1$ cows have the same number of spots.
2. Base case, $P(0)$: any 1 cow have the same number of spots. This holds because there is o number of spots is the same.
3. Inductive case, let $n \in \mathbb{N}$.
   (a) By induction hypothesis, suppose $P(n)$ holds. That is, any $n+1$ cows have the same
   (b) Consider any $n+2$ cows. Let $c_1, c_2, \ldots, c_{n+2}$ be (the names of) the cows.
   (c) By $P(n)$, the first $n+1$ cows, $c_1, c_2, \ldots, c_{n+1}$, have the same number of spots. Let x be the number.
   (d) By $P(n)$, the last $n+1$ cows, $c_2, \ldots, c_{n+2}$, have the same number of spots. Let y be th
   (e) Let z be the number of spots of the cow c_2.
   (f) We have $x = z$ because c_2 is in the first $n+1$.
   (g) We have $y = z$ because c_2 is in the last $n+1$.
   (h) By $x = y = z$, all $n+2$ cows have the same number of spots. That is, $P(n+1)$ hold

□

Please briefly explain why the proof is wrong (maybe by pointing out some incorrect steps). Notice there are many concepts and symbols that are not defined in class, such as $x = z$ and $y = z$ implies $x = y = z$. Please interpret them as standard math concepts you learned from other courses, such as DMT1.

explain why the proof is wrong (maybe by **pointing out some incorrect steps**).

What is said by ChatGPT?
No step number
Violation of common sense of transitivity

We see many “GPT-ish” submissions

- And many of them did not reference the resource

- “You should write up your own solutions and understand everything in them, and submit only your own work”

Recap

- Any finite function $f(x)$ is computable by LOOKUP(s_f, x)
  $C(x) = \boxed{\text{LOOKUP}(s_f}, x)$
  - Also: proving universality and non-universality

- Circuit size: the number of gates in AON circuit
  - Also: big-O notations

Circuit Complexity

How many gates?

k: 1,2,3,…

s = $b_0 b_1 \ldots b_{2^k-1}$

i represent $0, 1, \ldots, 2^k - 1$

outputs s[i] = b_i

LOOKUP$_k$(s, i):

first_half = LOOKUP$_{k-1}$(s[0:2^{k-1}], i[1:k])

second_half = LOOKUP$_{k-1}$(s[2^{k-1}:2^k], i[1:k])

return IF(i[0], second_half, first_half)

Recursion:

S(k) $\leq$ 2 · S(k-1) + c'

S(1) $\leq$ c'

Exists c, for all k, S(k) = c · 2^k

How many gates?

How many gates does this construction take?

You can computer any function
$f:\{0,1\}^n \to \{0,1\}^m$ with a NAND circuit
using no more than $c \cdot m \cdot 2^n$ gates

Note: this can be improved to $c \cdot m \cdot \frac{2^n}{n}$ (theorem 4.16 in TCS)

Where do we go from here?

Is this approach practical?

input: n-bit 2 int

How many gates need to, say, $\underbrace{\text{add}}_{2n \text{ to } n+1}$ or $\underbrace{\text{multiply}}_{2n \text{ to } 2n}$?

Are there functions that *need* $\Omega(2^n)$ gates? (stay tuned..)

$\text{LOOKUP}_{2n}(\cdots) \approx 2^{2n}$

Categorizing Functions by Circuit Size

Fact 1: No functions (n-bit to m-bit) require more than $O(m\,2^n)$ gates

Fact 2: Some functions seem to require much less

n-ADD $\Rightarrow \Theta(n) \ll 2^{2n}$

n-MULT $\Rightarrow \Theta(n^2)$

So, let's categorize functions by **the # of gates** they need

Circuit Size

SIZE

$SIZE(s)$ The set of all **functions** that can be implemented by a circuit of at most s NAND gates

$SIZE(1000 \cdot m \cdot 2^n)$ Contains all functions $f:\{0,1\}^n \to \{0,1\}^m$

size of (LOOKUPs)

TCS also uses: $SIZE_{n,m}(s)$

The set of all n-input, m-output functions that can be implemented with at most s NAND gates

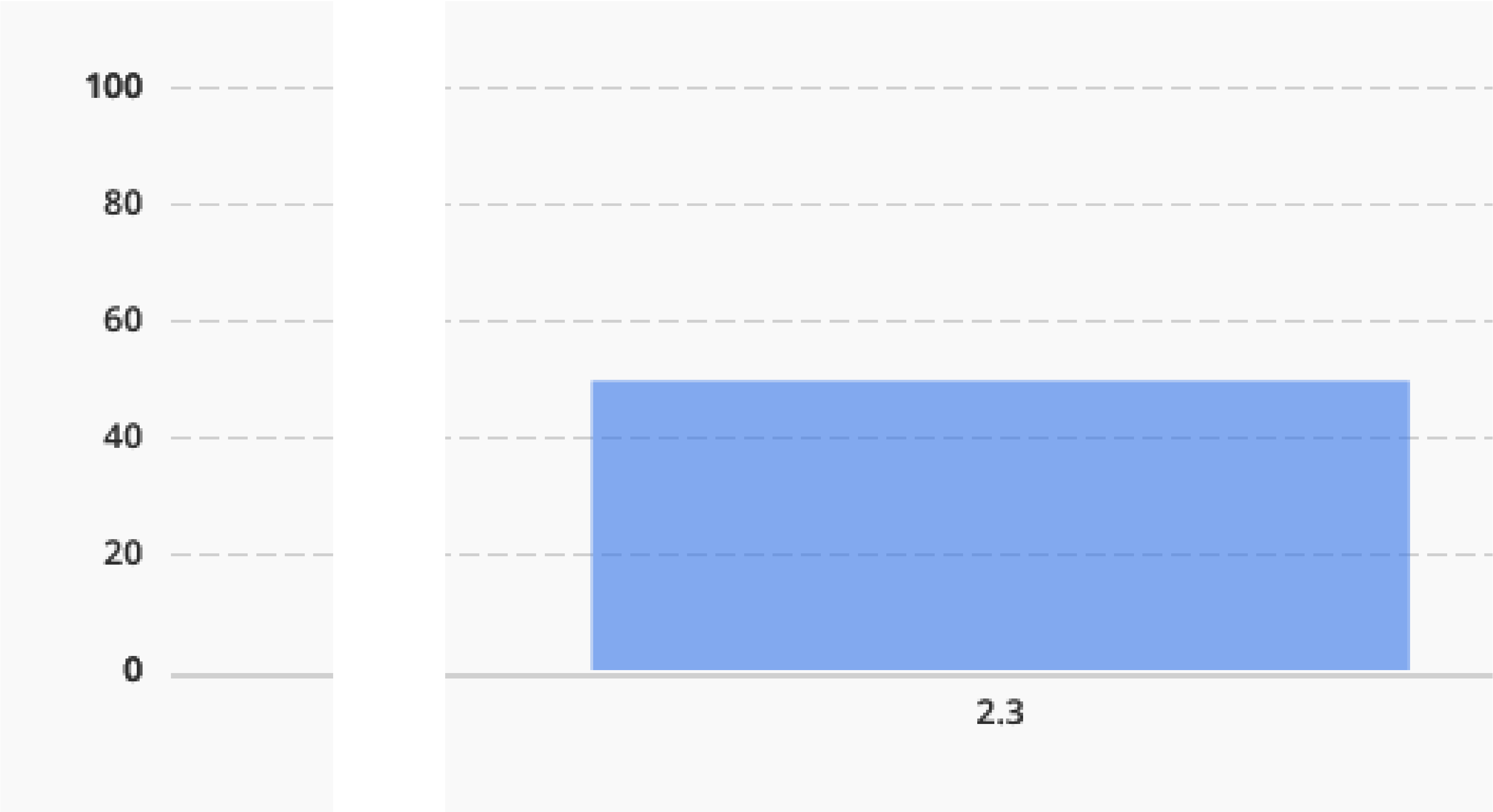
$n=10$

$m=10$

$f \in SIZE(1000 \cdot 10 \cdot 2^{10})$

$: \{\ \}^{10} \to \{\ \}^{10}$

PRR3



Q2.3

3 Points

Suppose that C is a NAND circuit that consists of s gates. Also suppose that C computes the function $f : \{0,1\}^n \to \{0,1\}$. Which of following must hold? Select all that hold.

- [x] $C \in SIZE_{n,1}(s)$
- [x] $C \in SIZE_{n,1}(10s)$
- [x] $f \in SIZE_{n,1}(s)$
- [x] $f \in SIZE_{n,1}(10s)$
- [ ] $f \in SIZE_{n,1}(0.1s)$

X

$SIZE(s) \subseteq SIZE(s+1)$

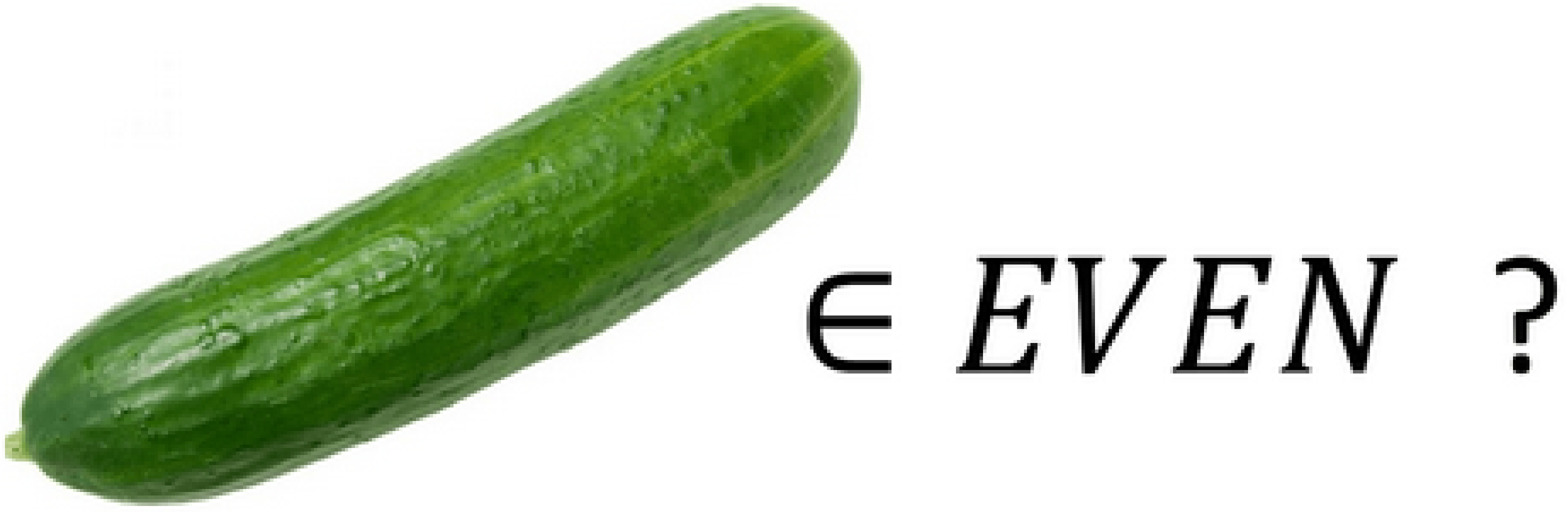
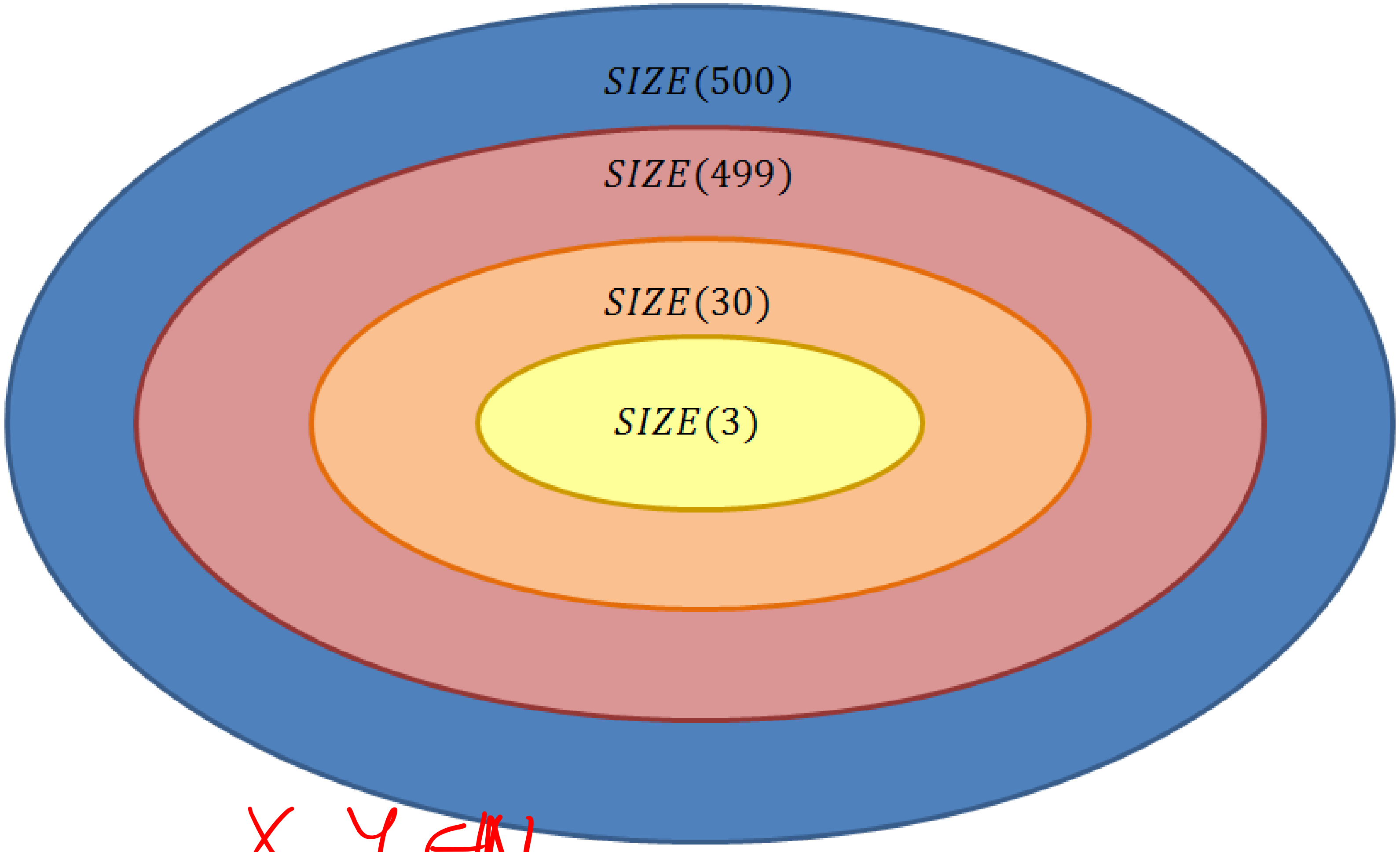


Figure 4.13: A "category error" is a question such as "is a cucumber even or odd?" which does not even make sense. In this book one type of category error you should watch out for is confusing functions and programs (i.e., confusing specifications and implementations). If C is a circuit or program, then asking if $C \in SIZE_{n,1}(s)$ is a category error, since $SIZE_{n,1}(s)$ is a set of functions and not programs or circuits.

Two Different "Roles"

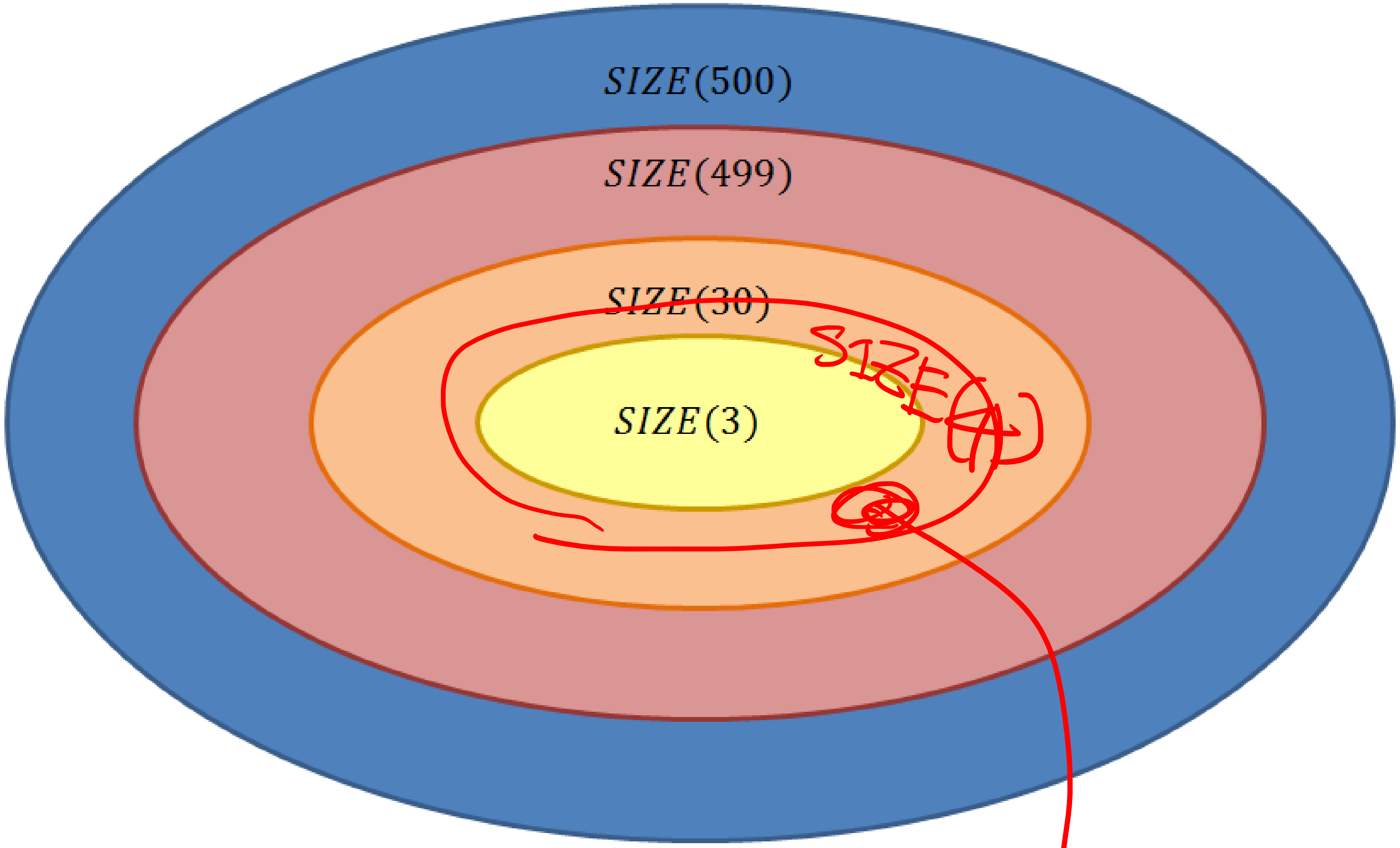
Function: What we want. Examples:
- Problem (we want to solve)
- Specification (of any module, or what customers want)

Circuit: How do we achieve it. Similar concepts:
- Program (implemented in our favorite PL)
- Implementation (to satisfy the need of customers)



Theorem: $x, y \in \mathbb{N}$

If $x \le y$, then $SIZE(x) \subseteq SIZE(y)$



If $x \leq y$, then $SIZE(x) \subseteq SIZE(y)$

But is the inclusion **strict**? Again, stay tuned for that..

Programs as Data

A Big Idea in Theory of Computing

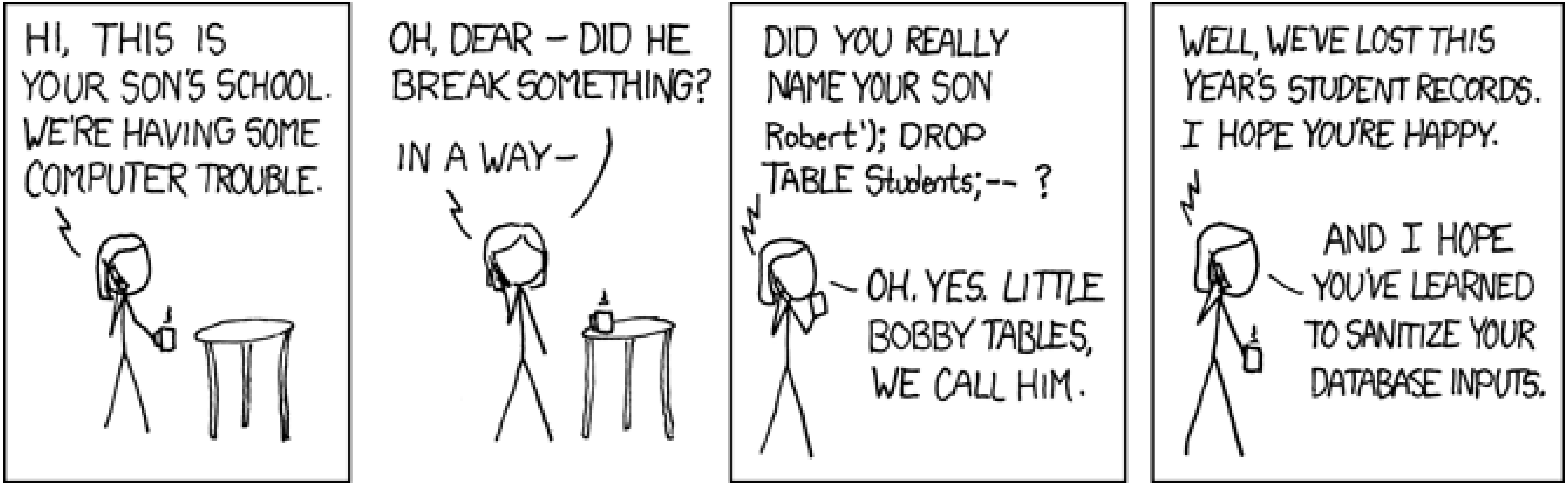
Big Idea 6

A program is a piece of text, and so it can be fed as input to other programs.

Program: an instance (in a computation model) that performs computation (on some data)

Data: a sequence of symbols, such as bits (which can be computed, such as copy, truncate, concatenate….)

Could be Bad

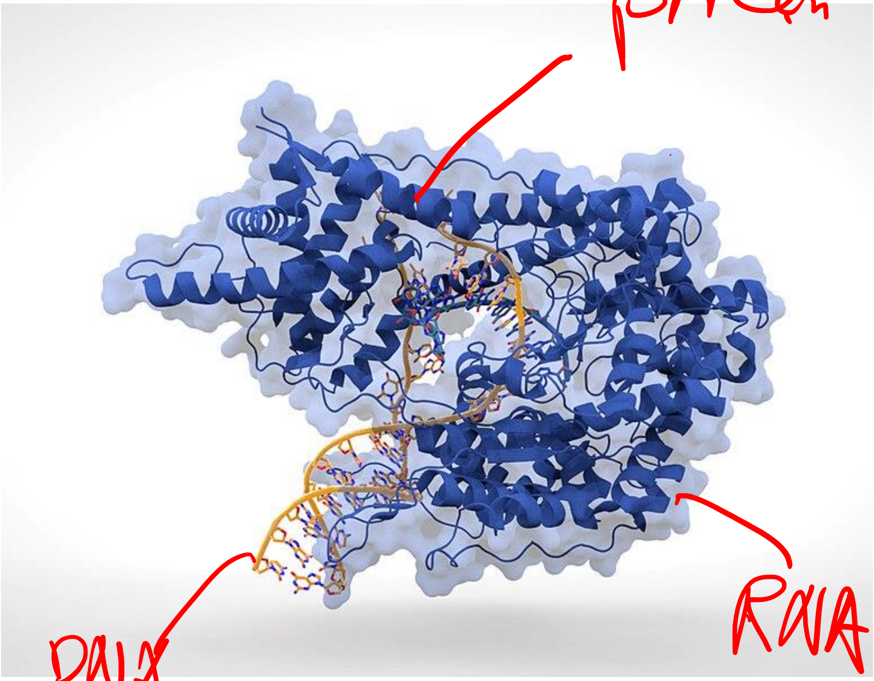


As illustrated in this xkcd cartoon, many exploits, including buffer overflow, SQL injections, and more, utilize the blurry line between “active programs” and “static strings”.

Can be GOOD

DNA (produce)→ Creatures

Creatures (copy, modify) → ~~NDA~~
DNA



> *"The term code script is, of course, too narrow. The chromosomal structures are at the same time instrumental in bringing about the development they foreshadow. They are law-code and executive power - or, to use another simile, they are architect's plan and builder's craft - in one."* , Erwin Schrödinger, 1944.

Image credit: https://en.wikipedia.org/wiki/T7_RNA_polymerase
T7 RNA polymerase (blue) producing m-RNA (light blue) from DNA (orange)

How can we represent a straightline NAND program?

t0

~~temp0~~ = NAND(X[0],X[1])

t1 t0

~~temp1~~ = NAND(X[0],~~temp0~~)

t0

temp2 = NAND(X[1],~~temp0~~)

t1

Y[0] = NAND(~~temp1~~,temp2)

DAG

Graph

edges

var → gate.

var ↗

99 characters. Can it be shorter?

Unicode ASCII char → 8 bits

How can we represent a straightline NAND program?

Recall: we have n input variables (gates), ℓ lines of code that each is also a "gate", and m outputs. Total number of gates $s = n + \ell + m$

To encode: write (n, ℓ, m) followed by a list of ℓ "internal-gate descriptions" followed by m "output-gate descriptions".

log s bit

Each of the NAND gates, NAND(input1, input2), is described by:

$$(input1, input2)$$

(line #) (input 1, input 2)

So, we have a list of $\ell + m \leq s$ triples, each of size $O(1) + 2\log s$ bits

ints

Example

input 0, 1

2 temp0 = NAND(X[0],X[1])

3 temp1 = NAND(X[0],temp0)

4 temp2 = NAND(X[1],temp0)

Y[0] = NAND(temp1,temp2)

2

(0, 1)

(0, 2)

(1, 2)

(3, 4)

$O(S \cdot \log S)$