



turning mathematical notation into beautiful diagrams

**Katherine Ye**

Carnegie Mellon University  
Computer Science Department



# Penrose

turning mathematical notation into beautiful diagrams

**Katherine Ye**

Carnegie Mellon University  
Computer Science Department



just a name!


# Penrose

turning mathematical notation into beautiful diagrams

**Katherine Ye**

Carnegie Mellon University  
Computer Science Department

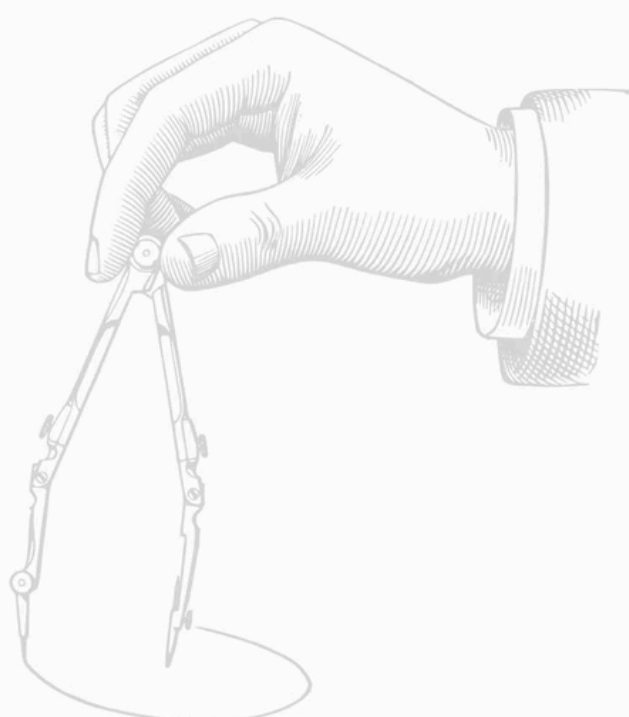
# Sneak preview:

 set theory continuous run sign in

continuous map

```
1 AutoLabel All
2
3 Set A
4 Set U
5 Set Rn
6 Label Rn  $\mathbb{R}^n$ 
7 IsSubset(U, A)
8 IsSubset(A, Rn)
9
10 Set B
11 Set V
12 Label V  $f^{-1}(U)$ 
13 Set Rm
14 Label Rm  $\mathbb{R}^m$ 
15 IsSubset(V, B)
16 IsSubset(B, Rm)
17
18 Map f
19 From(f, A, B)
20
```

editing fill




Click run to render your diagram.

resample autostep (off) download



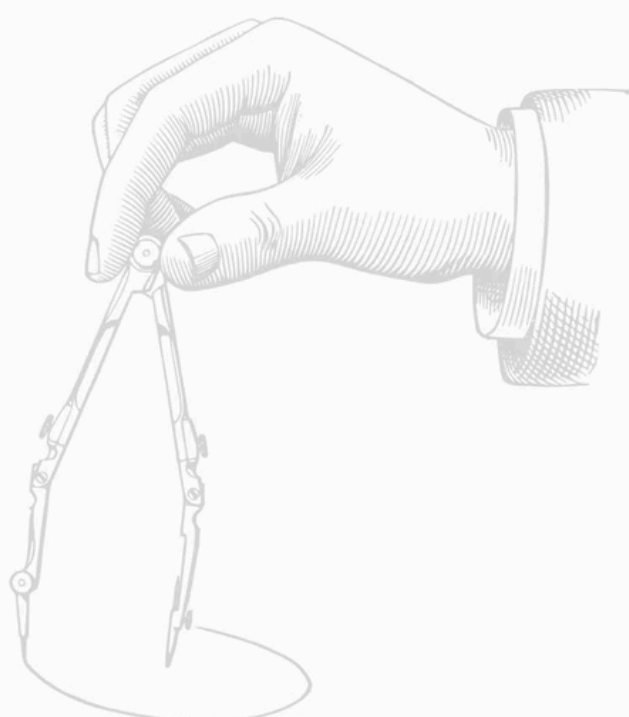
# Sneak preview:

 set theory continuous run sign in

continuous map

```
1 AutoLabel All
2
3 Set A
4 Set U
5 Set Rn
6 Label Rn  $\mathbb{R}^n$ 
7 IsSubset(U, A)
8 IsSubset(A, Rn)
9
10 Set B
11 Set V
12 Label V  $f^{-1}(U)$ 
13 Set Rm
14 Label Rm  $\mathbb{R}^m$ 
15 IsSubset(V, B)
16 IsSubset(B, Rm)
17
18 Map f
19 From(f, A, B)
20
```

editing fill



Click run to render your diagram.

resample autostep (off) download

# Who am I?



A believer in the power of **language!**

# Language plays a big role in mathematics.

## MEAN VALUE THEOREMS

**5.7 Definition** Let  $f$  be a real function defined on a metric space  $X$ . We say that  $f$  has a *local maximum* at a point  $p \in X$  if there exists  $\delta > 0$  such that  $f(q) \leq f(p)$  for all  $q \in X$  with  $d(p, q) < \delta$ .

Local minima are defined likewise.

Our next theorem is the basis of many applications of differentiation.

**5.8 Theorem** Let  $f$  be defined on  $[a, b]$ ; if  $f$  has a local maximum at a point  $x \in (a, b)$ , and if  $f'(x)$  exists, then  $f'(x) = 0$ .

The analogous statement for local minima is of course also true.

**Proof** Choose  $\delta$  in accordance with Definition 5.7, so that

$$a < x - \delta < x < x + \delta < b.$$

If  $x - \delta < t < x$ , then

$$\frac{f(t) - f(x)}{t - x} \geq 0.$$

Letting  $t \rightarrow x$ , we see that  $f'(x) \geq 0$ .

If  $x < t < x + \delta$ , then

$$\frac{f(t) - f(x)}{t - x} \leq 0,$$

which shows that  $f'(x) \leq 0$ . Hence  $f'(x) = 0$ .

**5.9 Theorem** If  $f$  and  $g$  are continuous real functions on  $[a, b]$  which are differentiable in  $(a, b)$ , then there is a point  $x \in (a, b)$  at which

$$[f(b) - f(a)]g'(x) = [g(b) - g(a)]f'(x).$$

Note that differentiability is not required at the endpoints.

**Proof** Put

$$h(t) = [f(b) - f(a)]g(t) - [g(b) - g(a)]f(t) \quad (a \leq t \leq b).$$

Then  $h$  is continuous on  $[a, b]$ ,  $h$  is differentiable in  $(a, b)$ , and

$$(12) \quad h(a) = f(b)g(a) - f(a)g(b) = h(b).$$

To prove the theorem, we have to show that  $h'(x) = 0$  for some  $x \in (a, b)$ .

If  $h$  is constant, this holds for every  $x \in (a, b)$ . If  $h(t) > h(a)$  for some  $t \in (a, b)$ , let  $x$  be a point on  $[a, b]$  at which  $h$  attains its maximum

(Theorem 4.16). By (12),  $x \in (a, b)$ , and Theorem 5.8 shows that  $h'(x) = 0$ . If  $h(t) < h(a)$  for some  $t \in (a, b)$ , the same argument applies if we choose for  $x$  a point on  $[a, b]$  where  $h$  attains its minimum.

This theorem is often called a *generalized mean value theorem*; the following special case is usually referred to as "the" mean value theorem:

**5.10 Theorem** If  $f$  is a real continuous function on  $[a, b]$  which is differentiable in  $(a, b)$ , then there is a point  $x \in (a, b)$  at which

$$f(b) - f(a) = (b - a)f'(x).$$

**Proof** Take  $g(x) = x$  in Theorem 5.9.

**5.11 Theorem** Suppose  $f$  is differentiable in  $(a, b)$ .

(a) If  $f'(x) \geq 0$  for all  $x \in (a, b)$ , then  $f$  is monotonically increasing.

(b) If  $f'(x) = 0$  for all  $x \in (a, b)$ , then  $f$  is constant.

(c) If  $f'(x) \leq 0$  for all  $x \in (a, b)$ , then  $f$  is monotonically decreasing.

**Proof** All conclusions can be read off from the equation

$$f(x_2) - f(x_1) = (x_2 - x_1)f'(x),$$

which is valid, for each pair of numbers  $x_1, x_2$  in  $(a, b)$ , for some  $x$  between  $x_1$  and  $x_2$ .

## THE CONTINUITY OF DERIVATIVES

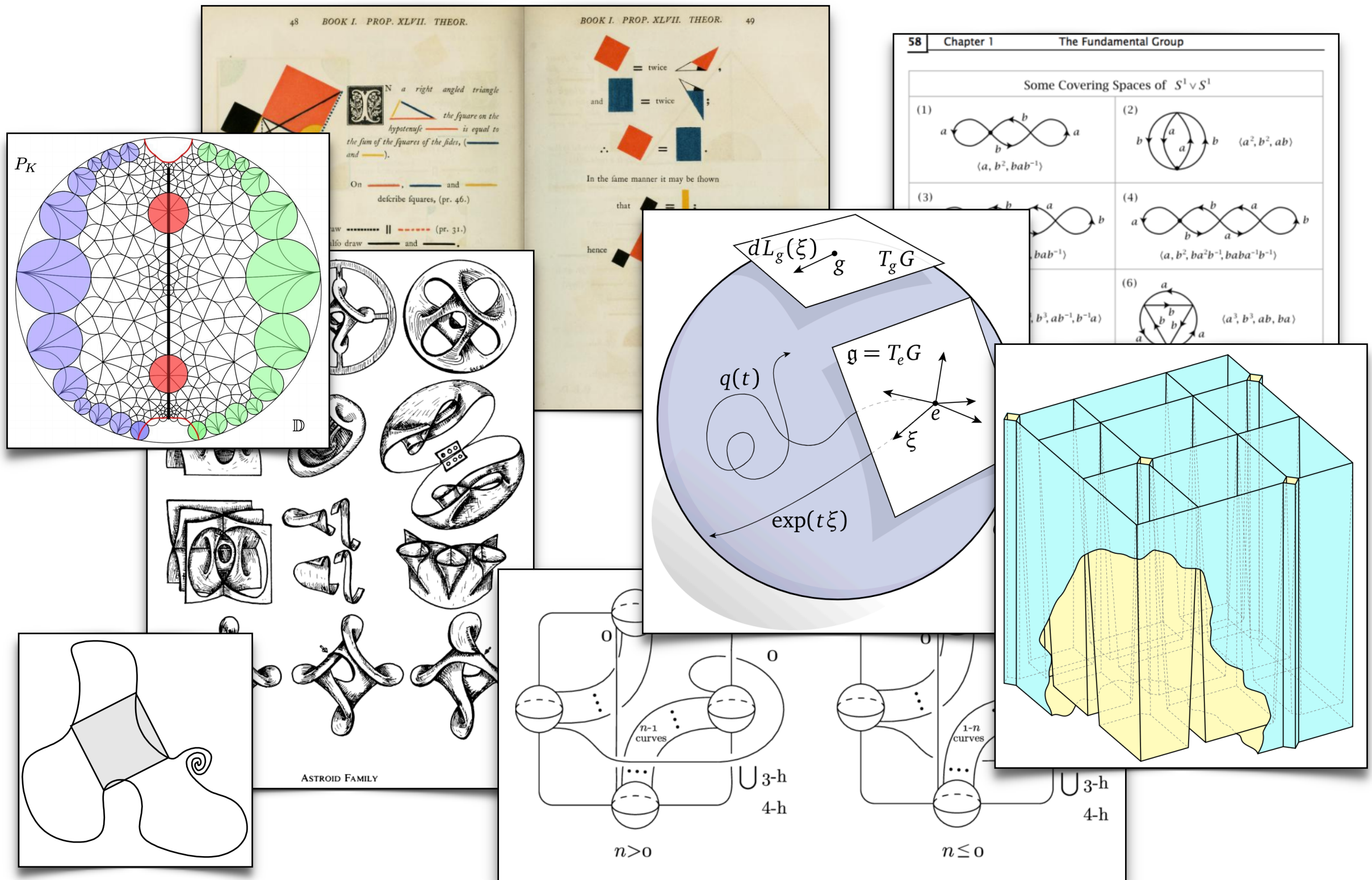
We have already seen [Example 5.6(b)] that a function  $f$  may have a derivative  $f'$  which exists at every point, but is discontinuous at some point. However, not every function is a derivative. In particular, derivatives which exist at every point of an interval have one important property in common with functions which are continuous on an interval: Intermediate values are assumed (compare Theorem 4.23). The precise statement follows.

**5.12 Theorem** Suppose  $f$  is a real differentiable function on  $[a, b]$  and suppose  $f'(a) < \lambda < f'(b)$ . Then there is a point  $x \in (a, b)$  such that  $f'(x) = \lambda$ .

A similar result holds of course if  $f'(a) > f'(b)$ .

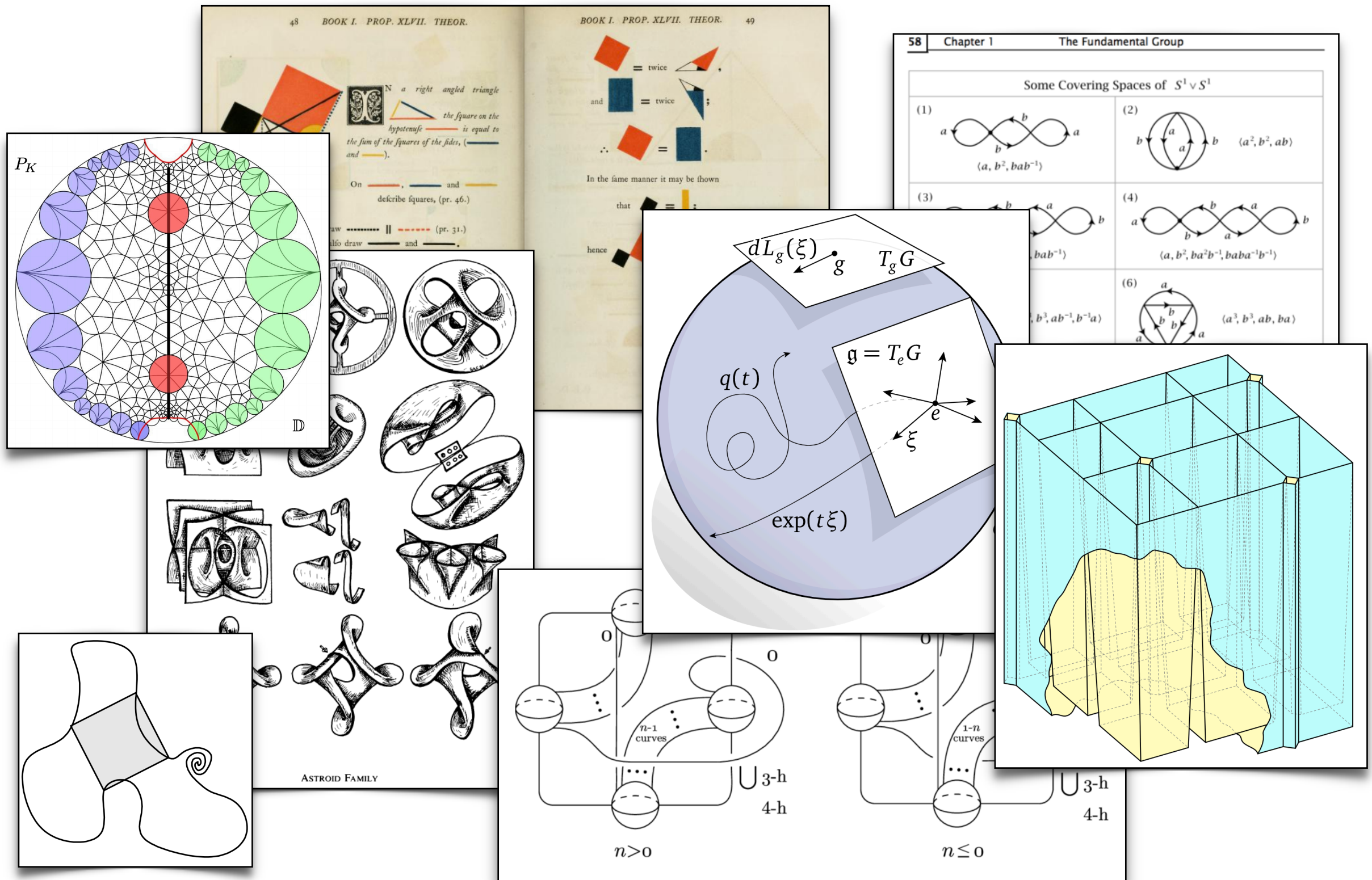
**Proof** Put  $g(t) = f(t) - \lambda t$ . Then  $g'(a) < 0$ , so that  $g(t_1) < g(a)$  for some  $t_1 \in (a, b)$ , and  $g'(b) > 0$ , so that  $g(t_2) < g(b)$  for some  $t_2 \in (a, b)$ . Hence  $g$  attains its minimum on  $[a, b]$  (Theorem 4.16) at some point  $x$  such that  $a < x < b$ . By Theorem 5.8,  $g'(x) = 0$ . Hence  $f'(x) = \lambda$ .

# But it's not the whole story





# But it's not the whole ~~story~~ picture.





“People have very powerful facilities for taking in information visually...

On the other hand, they do not have a good built-in facility for turning an internal spatial understanding back into a two-dimensional image.

So mathematicians usually have fewer and poorer figures in their papers and books than in their heads.”

**William Thurston**

(probably trying to make a diagram in Powerpoint)

**Question:** How can we do a better job of connecting language and visualization?

# Domain-Specific Language (DSL)

Idea: design a programming language to reflect the way people already naturally talk about a domain.



# Domain-Specific Language (DSL)

Idea: design a programming language to reflect the way people already naturally talk about a domain.

## MATLAB

$$\begin{pmatrix} 1 & 0 & & 0 \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 & 1 \\ 0 & & & -2 & 2 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_{J-1} \\ V_J \end{pmatrix} = \begin{pmatrix} 0 \\ (\Delta x)^2 f_2 \\ \vdots \\ (\Delta x)^2 f_{J-1} \\ (\Delta x)^2 f_J \end{pmatrix}$$

$$\mathbf{b} = 2 * \mathbf{c} + \mathbf{d}$$

$$\mathbf{x} = \mathbf{A} \backslash \mathbf{b}$$

# Domain-Specific Language (DSL)

Idea: design a programming language to reflect the way people already naturally talk about a domain.

## MATLAB

$$\begin{pmatrix} 1 & 0 & & 0 \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 & 1 \\ 0 & & & -2 & 2 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_{J-1} \\ V_J \end{pmatrix} = \begin{pmatrix} 0 \\ (\Delta x)^2 f_2 \\ \vdots \\ (\Delta x)^2 f_{J-1} \\ (\Delta x)^2 f_J \end{pmatrix}$$

```
b = 2*c + d  
x = A\b
```

## TeX

$$\int_M d\omega = \int_{\partial M} \omega$$

```
\int_M d\omega =  
\int_{\partial M}  
\omega
```

# Domain-Specific Language (DSL)

Idea: design a programming language to reflect the way people already naturally talk about a domain.

## MATLAB

$$\begin{pmatrix} 1 & 0 & & 0 \\ 1 & -2 & 1 & \\ & \ddots & \ddots & \ddots \\ & & 1 & -2 & 1 \\ 0 & & & -2 & 2 \end{pmatrix} \begin{pmatrix} V_1 \\ V_2 \\ \vdots \\ V_{J-1} \\ V_J \end{pmatrix} = \begin{pmatrix} 0 \\ (\Delta x)^2 f_2 \\ \vdots \\ (\Delta x)^2 f_{J-1} \\ (\Delta x)^2 f_J \end{pmatrix}$$

```
b = 2*c + d
x = A\b
```

## TeX

$$\int_M d\omega = \int_{\partial M} \omega$$

```
\int_M d\omega =
\int_{\partial M}
\omega
```

## CSS

<http://penrose.ink>

```
link {
  font-size: large;
  color: blue;
}
```

# Good news: we already have a nice language for mathematics

## MEAN VALUE THEOREMS

**5.7 Definition** Let  $f$  be a real function defined on a metric space  $X$ . We say that  $f$  has a *local maximum* at a point  $p \in X$  if there exists  $\delta > 0$  such that  $f(q) \leq f(p)$  for all  $q \in X$  with  $d(p, q) < \delta$ .

Local minima are defined likewise.

Our next theorem is the basis of many applications of differentiation.

**5.8 Theorem** Let  $f$  be defined on  $[a, b]$ ; if  $f$  has a local maximum at a point  $x \in (a, b)$ , and if  $f'(x)$  exists, then  $f'(x) = 0$ .

The analogous statement for local minima is of course also true.

**Proof** Choose  $\delta$  in accordance with Definition 5.7, so that

$$a < x - \delta < x < x + \delta < b.$$

If  $x - \delta < t < x$ , then

$$\frac{f(t) - f(x)}{t - x} \geq 0.$$

Letting  $t \rightarrow x$ , we see that  $f'(x) \geq 0$ .

If  $x < t < x + \delta$ , then

$$\frac{f(t) - f(x)}{t - x} \leq 0,$$

which shows that  $f'(x) \leq 0$ . Hence  $f'(x) = 0$ .

**5.9 Theorem** If  $f$  and  $g$  are continuous real functions on  $[a, b]$  which are differentiable in  $(a, b)$ , then there is a point  $x \in (a, b)$  at which

$$[f(b) - f(a)]g'(x) = [g(b) - g(a)]f'(x).$$

Note that differentiability is not required at the endpoints.

**Proof** Put

$$h(t) = [f(b) - f(a)]g(t) - [g(b) - g(a)]f(t) \quad (a \leq t \leq b).$$

Then  $h$  is continuous on  $[a, b]$ ,  $h$  is differentiable in  $(a, b)$ , and

$$(12) \quad h(a) = f(b)g(a) - f(a)g(b) = h(b).$$

To prove the theorem, we have to show that  $h'(x) = 0$  for some  $x \in (a, b)$ .

If  $h$  is constant, this holds for every  $x \in (a, b)$ . If  $h(t) > h(a)$  for some  $t \in (a, b)$ , let  $x$  be a point on  $[a, b]$  at which  $h$  attains its maximum

(Theorem 4.16). By (12),  $x \in (a, b)$ , and Theorem 5.8 shows that  $h'(x) = 0$ . If  $h(t) < h(a)$  for some  $t \in (a, b)$ , the same argument applies if we choose for  $x$  a point on  $[a, b]$  where  $h$  attains its minimum.

This theorem is often called a *generalized mean value theorem*; the following special case is usually referred to as "the" mean value theorem:

**5.10 Theorem** If  $f$  is a real continuous function on  $[a, b]$  which is differentiable in  $(a, b)$ , then there is a point  $x \in (a, b)$  at which

$$f(b) - f(a) = (b - a)f'(x).$$

**Proof** Take  $g(x) = x$  in Theorem 5.9.

**5.11 Theorem** Suppose  $f$  is differentiable in  $(a, b)$ .

(a) If  $f'(x) \geq 0$  for all  $x \in (a, b)$ , then  $f$  is monotonically increasing.

(b) If  $f'(x) = 0$  for all  $x \in (a, b)$ , then  $f$  is constant.

(c) If  $f'(x) \leq 0$  for all  $x \in (a, b)$ , then  $f$  is monotonically decreasing.

**Proof** All conclusions can be read off from the equation

$$f(x_2) - f(x_1) = (x_2 - x_1)f'(x),$$

which is valid, for each pair of numbers  $x_1, x_2$  in  $(a, b)$ , for some  $x$  between  $x_1$  and  $x_2$ .

## THE CONTINUITY OF DERIVATIVES

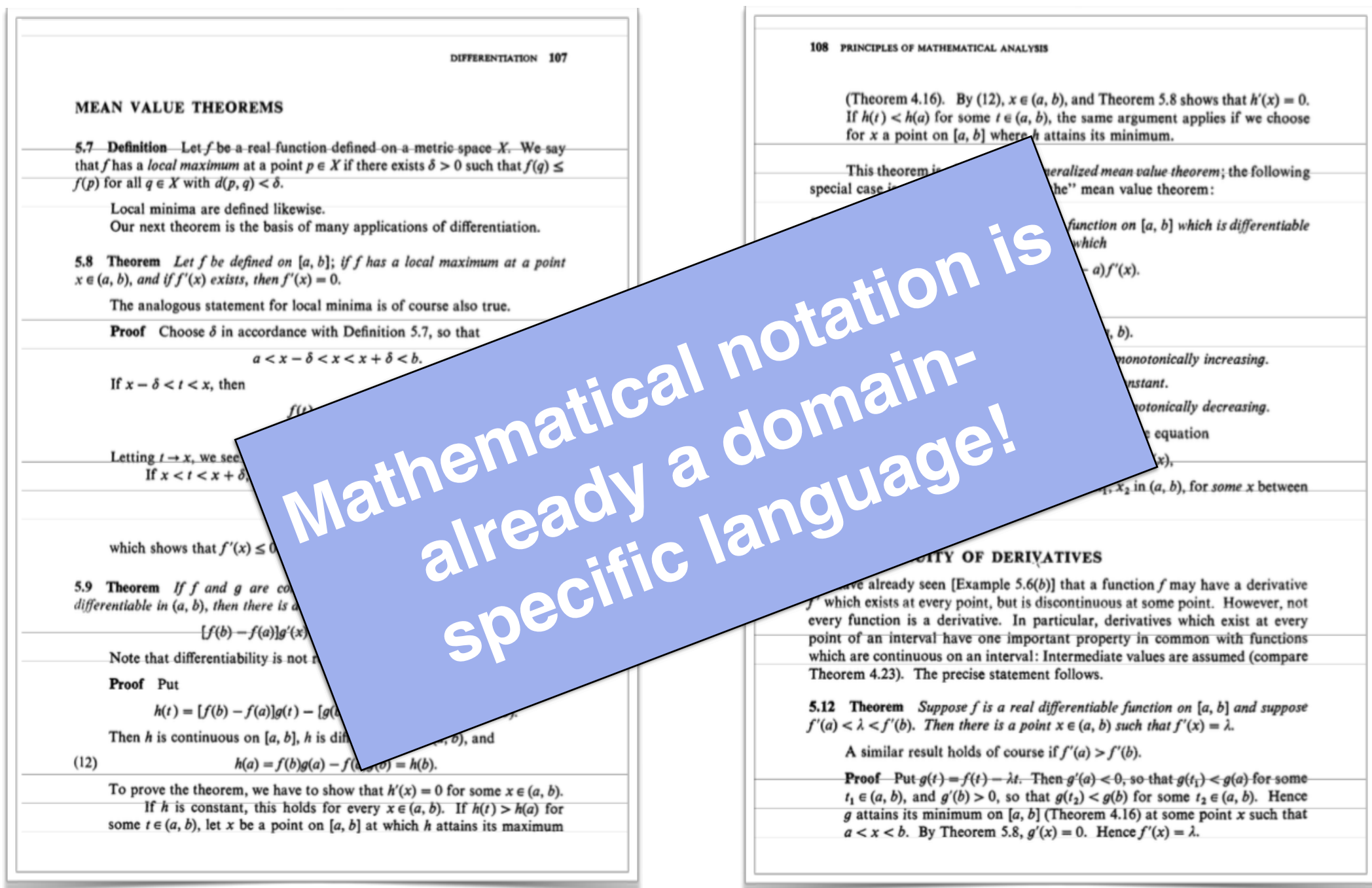
We have already seen [Example 5.6(b)] that a function  $f$  may have a derivative  $f'$  which exists at every point, but is discontinuous at some point. However, not every function is a derivative. In particular, derivatives which exist at every point of an interval have one important property in common with functions which are continuous on an interval: Intermediate values are assumed (compare Theorem 4.23). The precise statement follows.

**5.12 Theorem** Suppose  $f$  is a real differentiable function on  $[a, b]$  and suppose  $f'(a) < \lambda < f'(b)$ . Then there is a point  $x \in (a, b)$  such that  $f'(x) = \lambda$ .

A similar result holds of course if  $f'(a) > f'(b)$ .

**Proof** Put  $g(t) = f(t) - \lambda t$ . Then  $g'(a) < 0$ , so that  $g(t_1) < g(a)$  for some  $t_1 \in (a, b)$ , and  $g'(b) > 0$ , so that  $g(t_2) < g(b)$  for some  $t_2 \in (a, b)$ . Hence  $g$  attains its minimum on  $[a, b]$  (Theorem 4.16) at some point  $x$  such that  $a < x < b$ . By Theorem 5.8,  $g'(x) = 0$ . Hence  $f'(x) = \lambda$ .

# Good news: we already have a nice language for mathematics





48 BOOK I. PROP. XLVII. THEOR.

**Euclid's Elements, Book I, Proposition 47:**

**THEOREM.** In a right angled triangle the square on the hypotenuse is equal to the sum of the squares of the sides, (— and —).

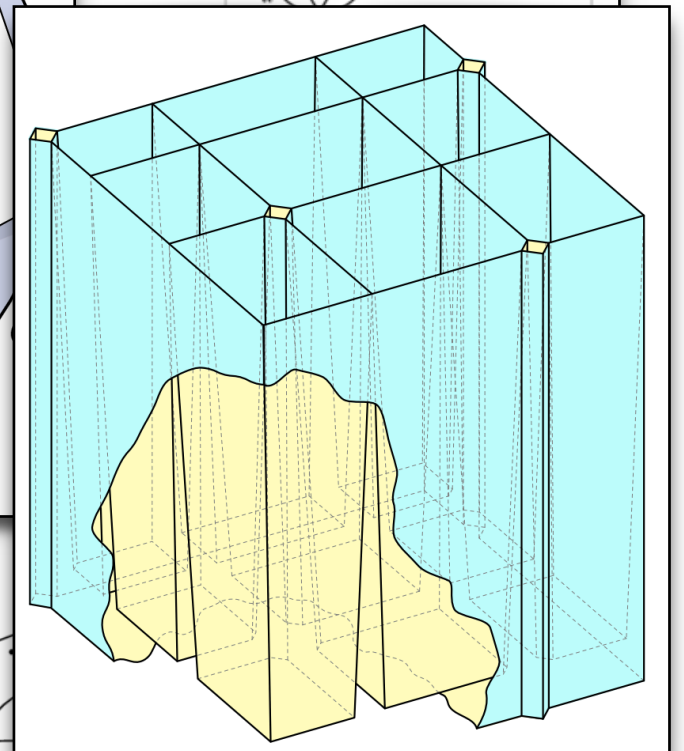
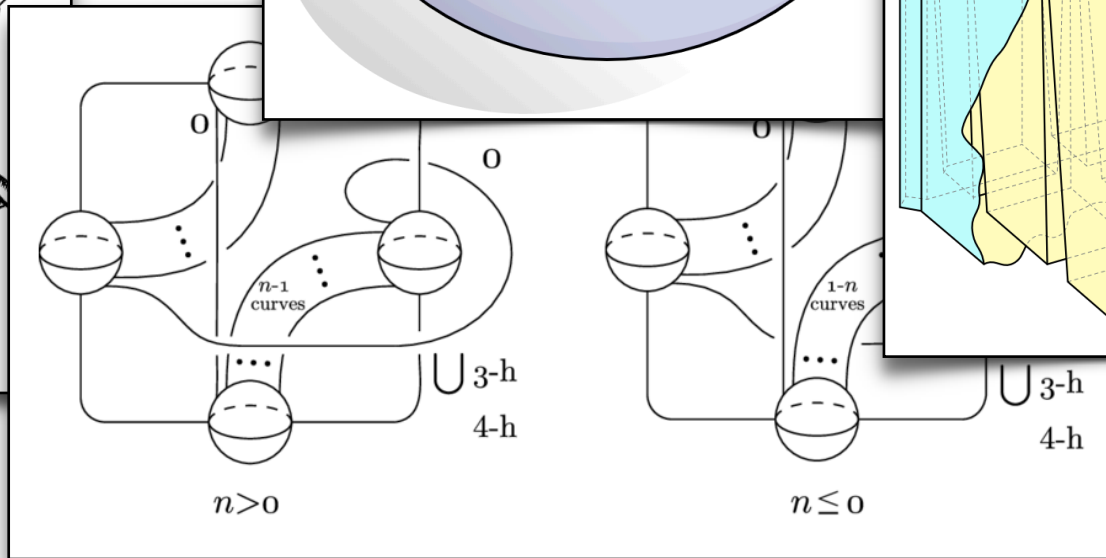
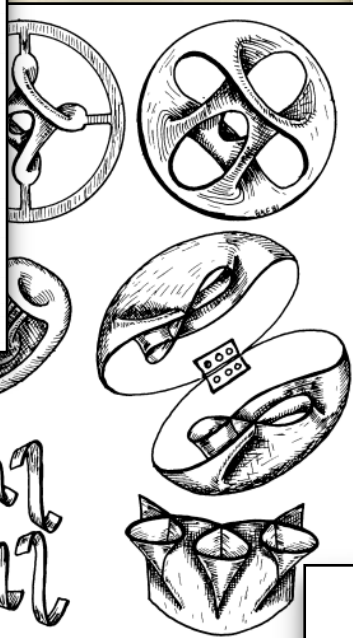
On —, — and — describe squares, (pr. 46.)

Now ———— || ———— (pr. 31.)

Also draw — and —.

**Modern Diagrams:**

- Top Left:** A circle with a network of inscribed circles, some shaded green.
- Bottom Left:** Two circular diagrams showing complex geometric constructions, possibly related to the proof of Proposition 47.
- Right:** A diagram illustrating the relationship between a curve and a tangent line. The curve is labeled  $q(t)$  and the tangent line is labeled  $dL_g(\xi)$ .



# How do we get there?

## **Outline of this talk:**

- I. What do we want from a diagramming tool?
- II. What do tools look like now?
- III. A new language-based tool
- IV. What does a language-based approach enable?

# **Part I:** What makes a good tool?



Picture an ideal tool for making mathematical diagrams...



...what features might it have?



**Accessibility:** it should be possible to use without steep learning curve/deep expertise



**Accessibility:** it should be possible to use without steep learning curve/deep expertise



**Universality:** it should be extensible, i.e., able to generate diagrams from any area of math, using any visual representation



**Accessibility:** it should be possible to use without steep learning curve/deep expertise



**Universality:** it should be extensible, i.e., able to generate diagrams from any area of math, using any visual representation



**Beauty:** it should be possible to make diagrams that *approach* the quality of professional illustrations



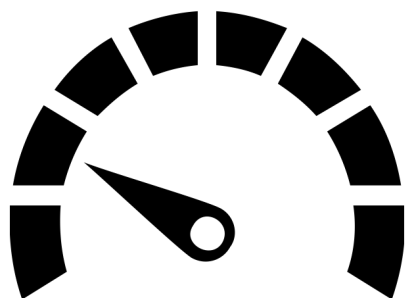
**Accessibility:** it should be possible to use without steep learning curve/deep expertise



**Universality:** it should be extensible, i.e., able to generate diagrams from any area of math, using any visual representation



**Beauty:** it should be possible to make diagrams that *approach* the quality of professional illustrations



**Productivity:** it should be reasonably fast to make or change diagrams

Probably too much to ask for...

# Probably too much to ask for...

- Solve hard problems  
(e.g., prove Fermat's last theorem)

# Probably too much to ask for...

- Solve hard problems  
(e.g., prove Fermat's last theorem)
- Invent novel visualization methods  
(e.g., sphere eversion)



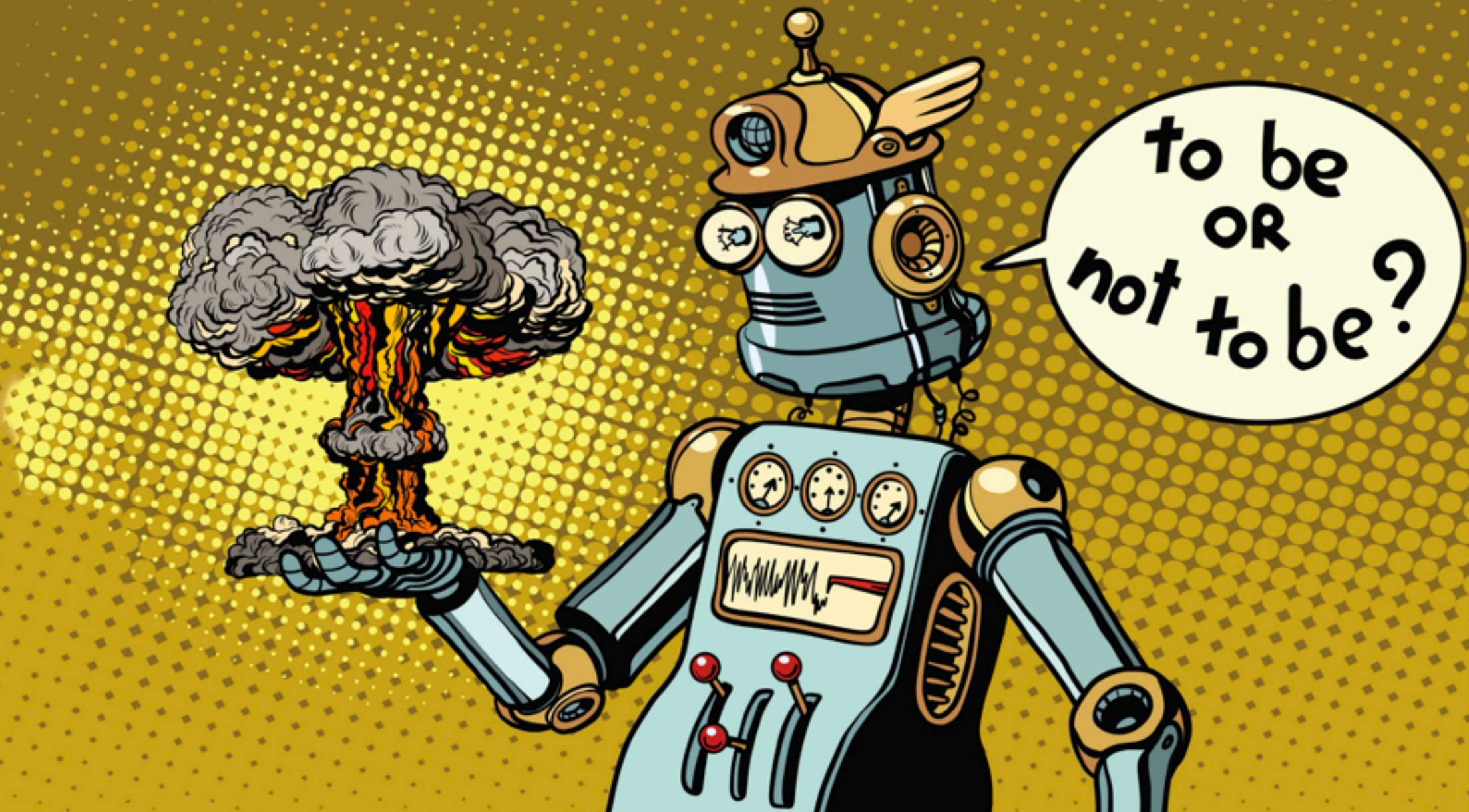
# Probably too much to ask for...

- Solve hard problems  
(e.g., prove Fermat's last theorem)
- Invent novel visualization methods  
(e.g., sphere eversion)
- Guarantee that diagrams formally encode mathematics

# Probably too much to ask for...

- Solve hard problems  
(e.g., prove Fermat's last theorem)
- Invent novel visualization methods  
(e.g., sphere eversion)
- Guarantee that diagrams formally encode mathematics
- Provide unified notation for all of mathematics

Probably too much to ask for...



**In general:** shouldn't expect your diagramming tool to do things that even expert mathematicians can't do!

Does such a holy grail exist?

Let's take a look at the state of the art...

## **Part II:** What do tools look like now?



**Providence 2019**



# **DIAGRAM TOOL OLYMPICS**

**Providence 2019**



# **DIAGRAM TOOL OLYMPICS**



**Providence 2019**



# **DIAGRAM TOOL OLYMPICS**

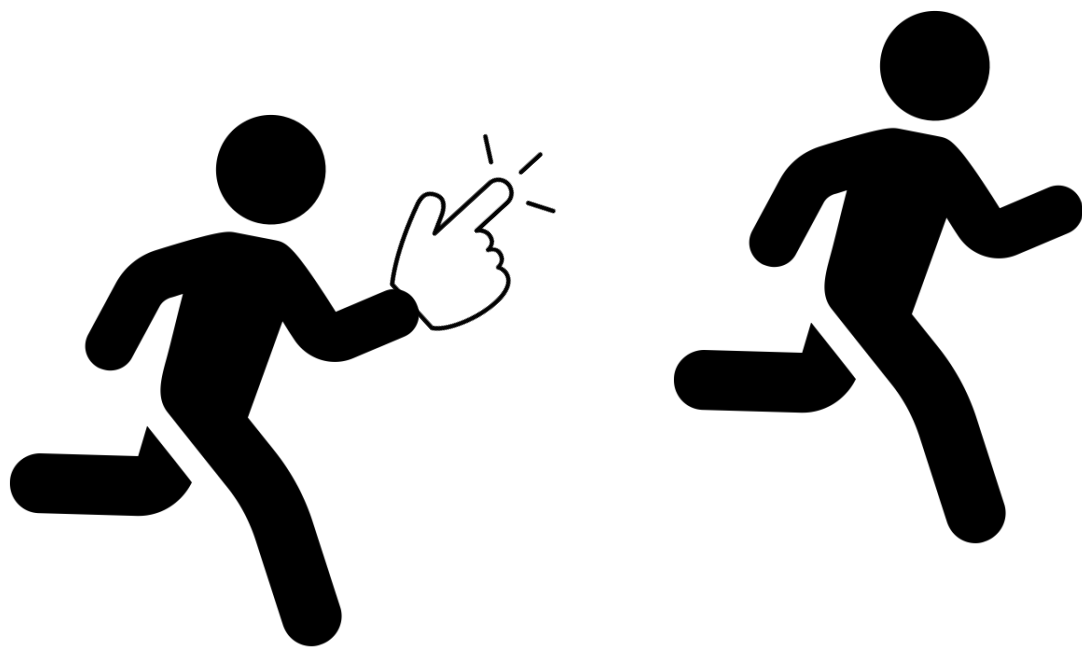




**Providence 2019**



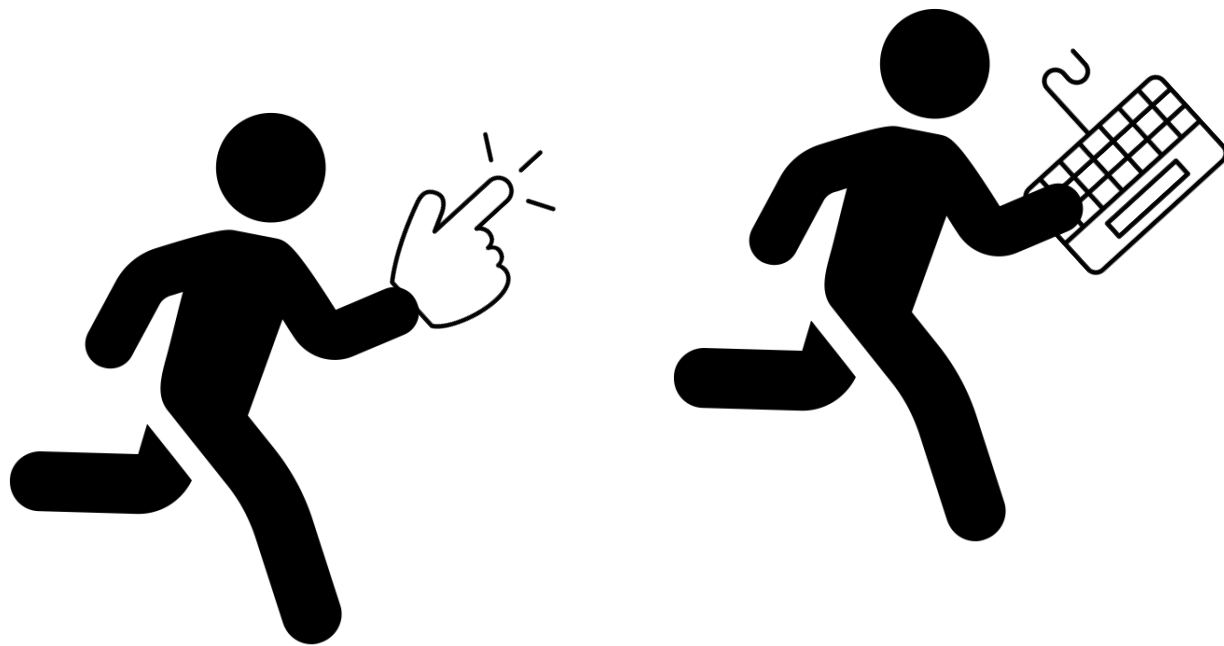
# **DIAGRAM TOOL OLYMPICS**



**Providence 2019**



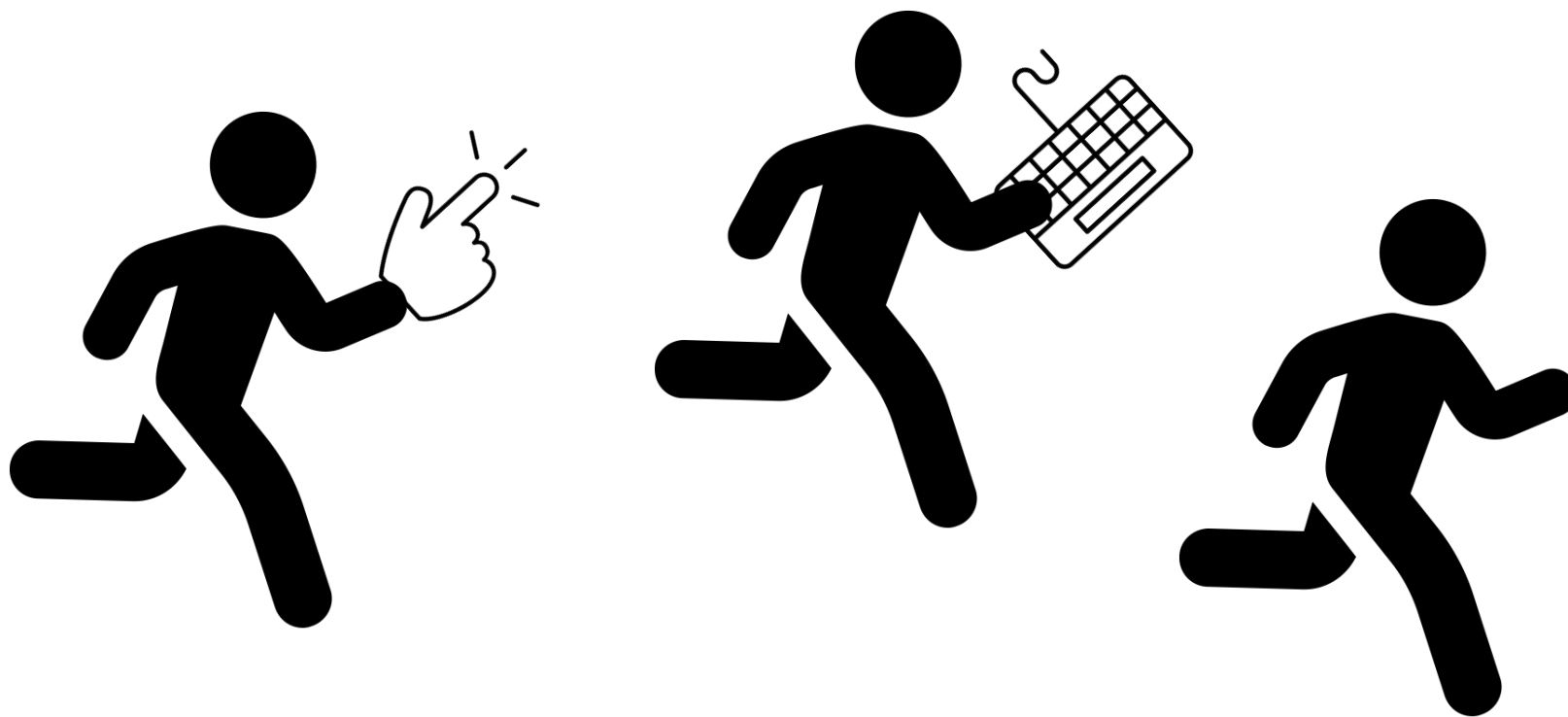
# **DIAGRAM TOOL OLYMPICS**



**Providence 2019**



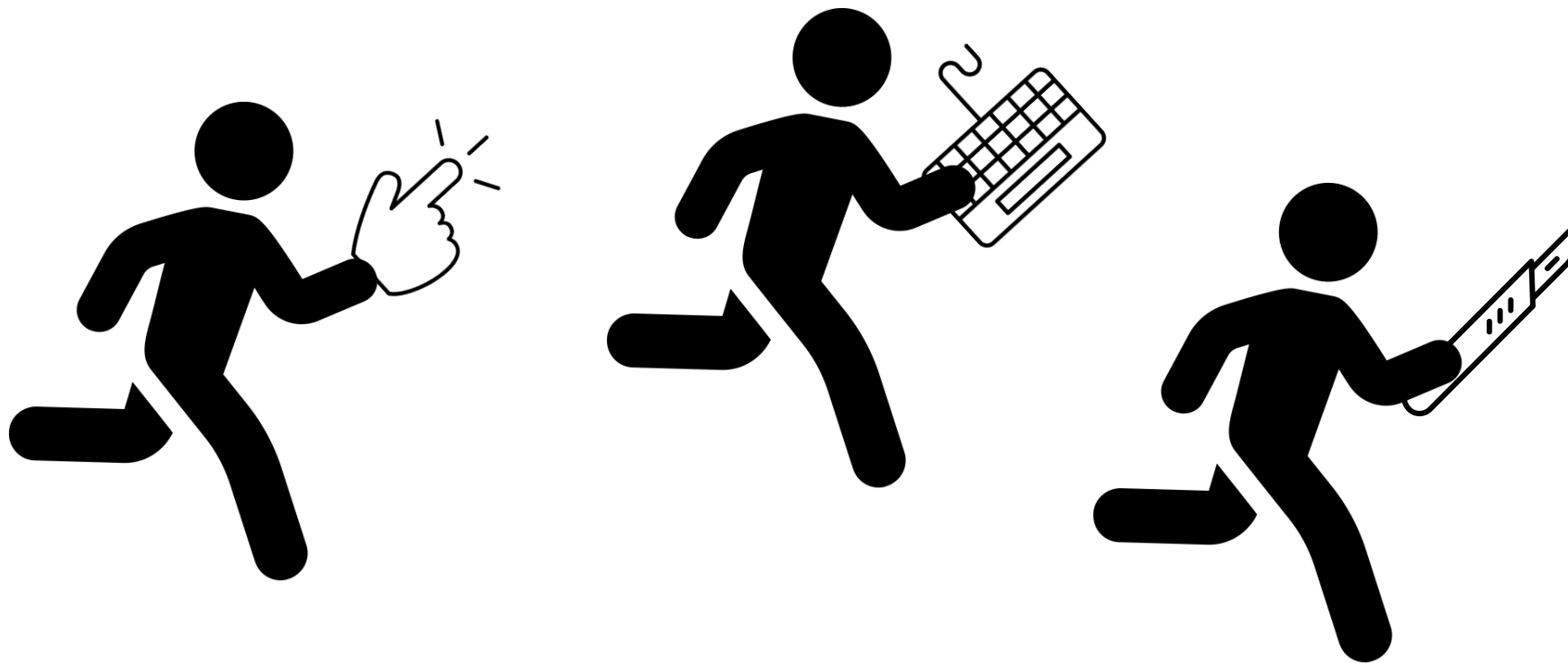
# **DIAGRAM TOOL OLYMPICS**



**Providence 2019**



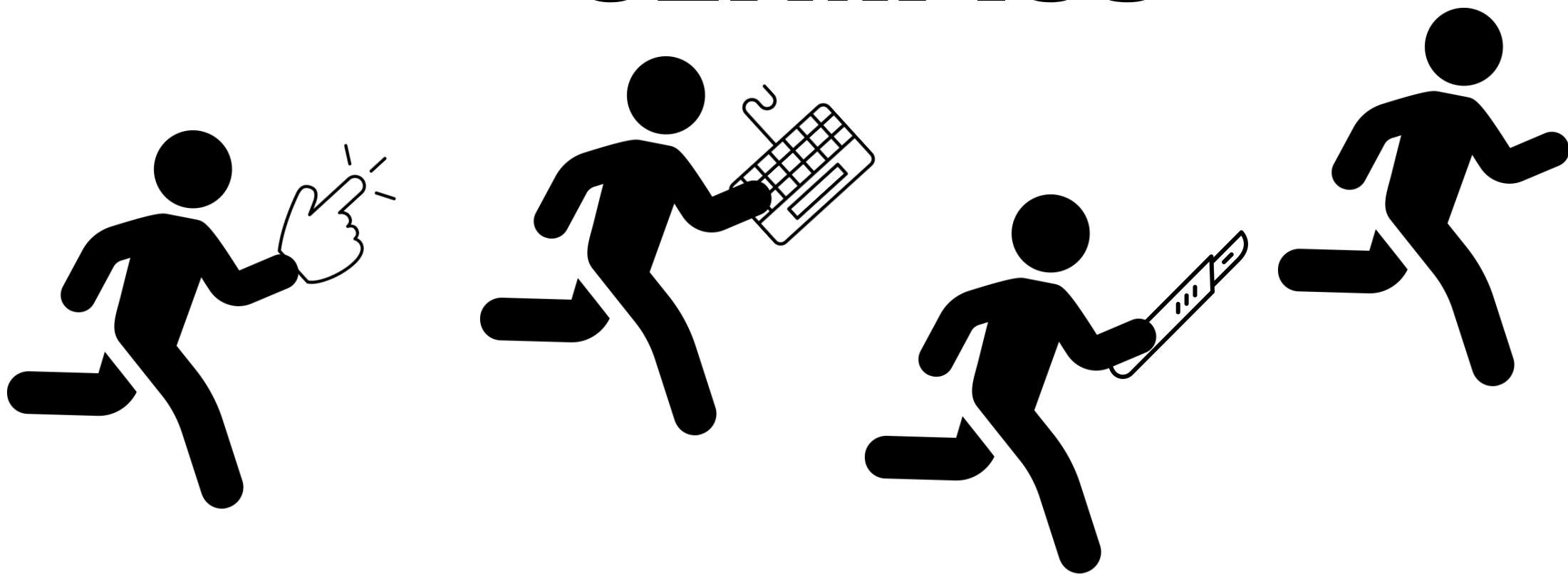
# **DIAGRAM TOOL OLYMPICS**



**Providence 2019**



# **DIAGRAM TOOL OLYMPICS**

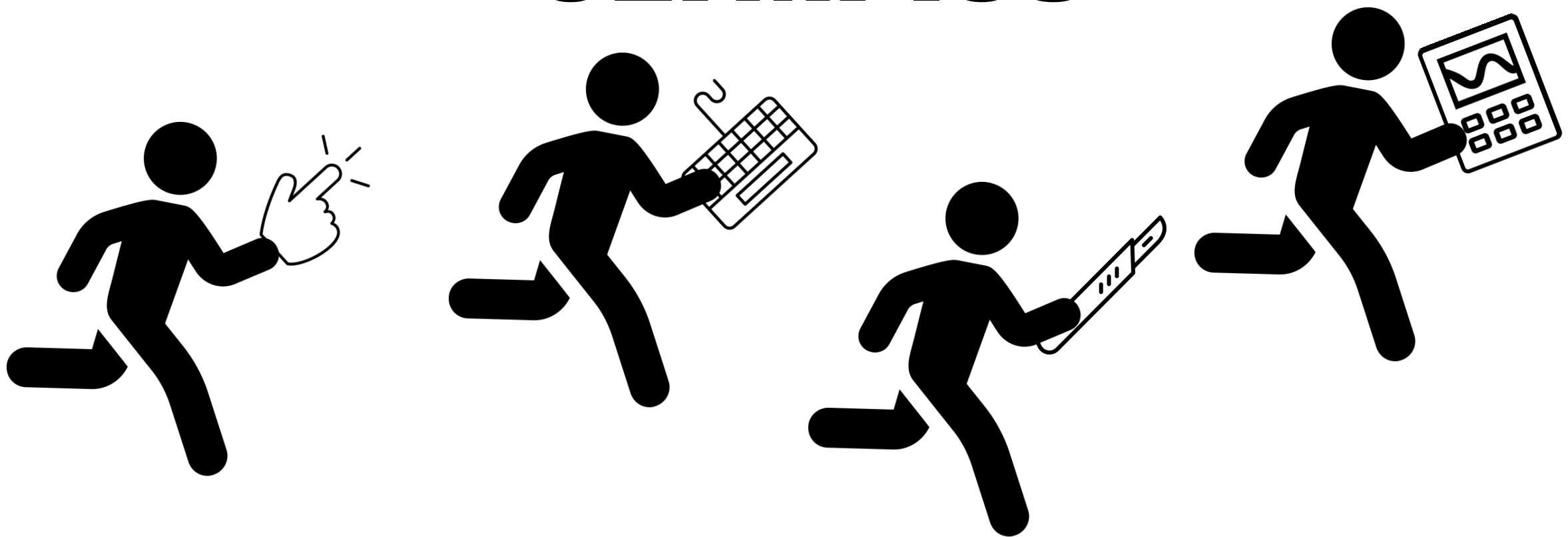




**Providence 2019**



# **DIAGRAM TOOL OLYMPICS**

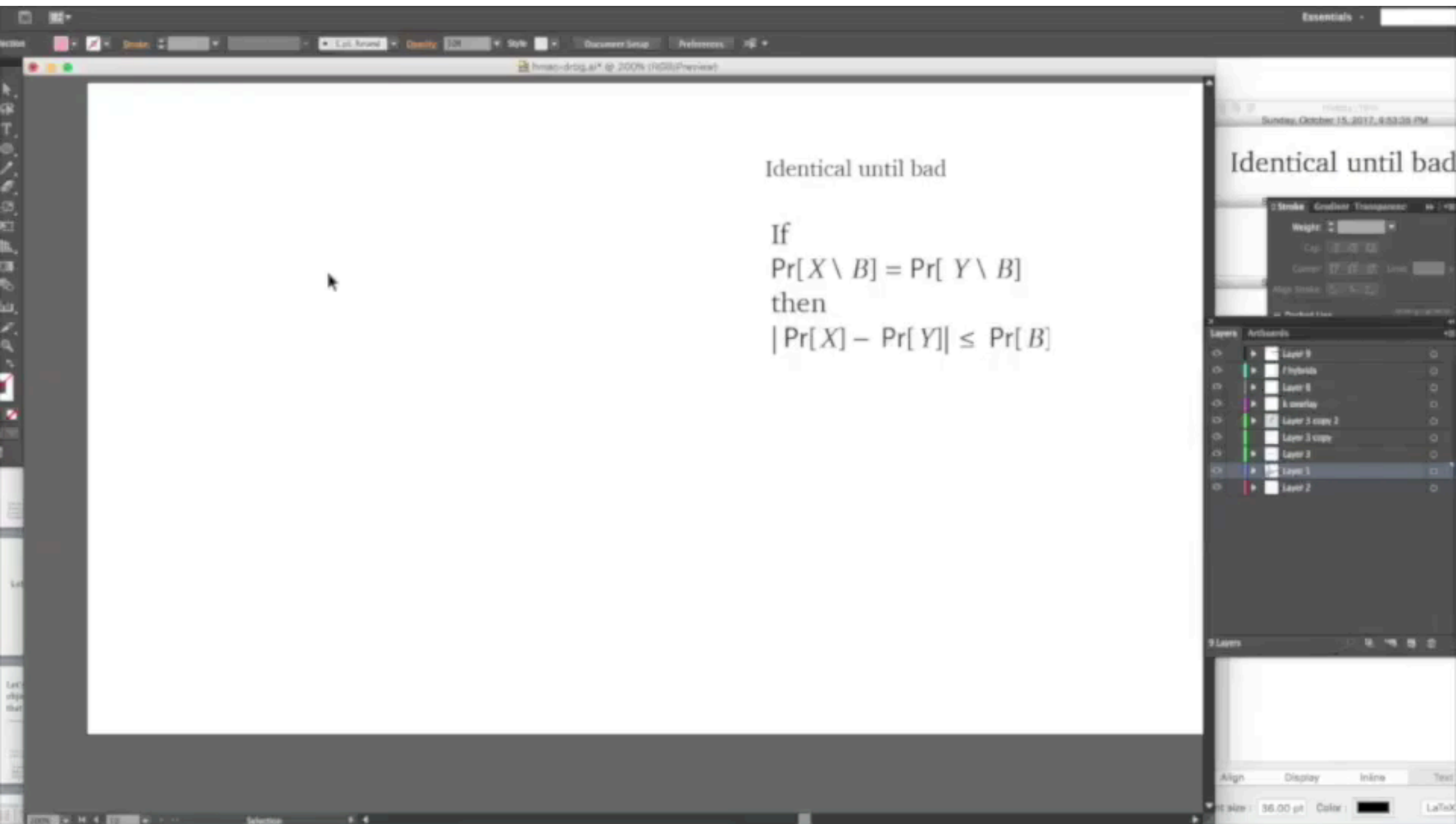




# **Graphical User Interface (GUI)**

Examples: Adobe Illustrator, Inkscape

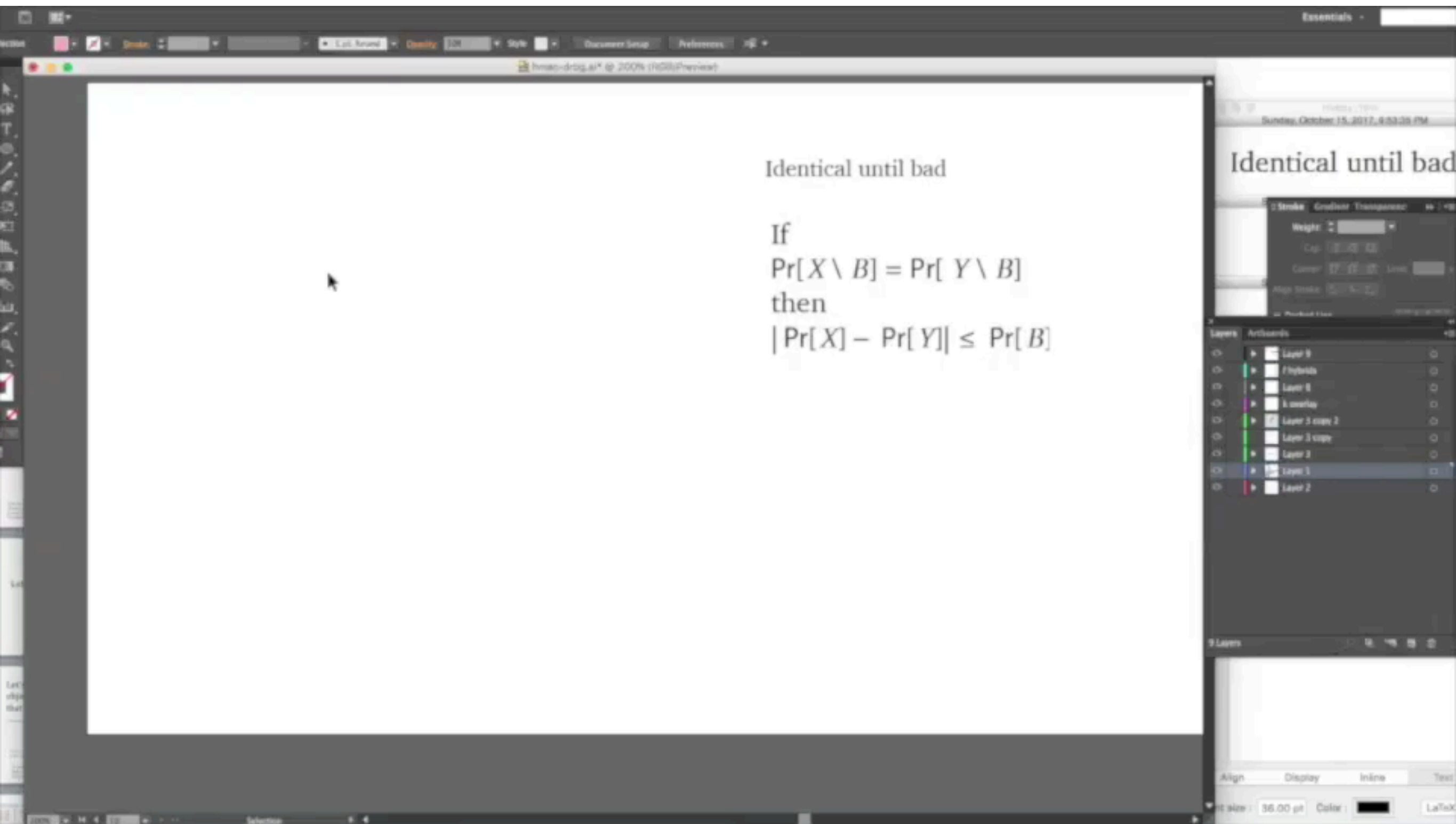
Lots of clicking and dragging... (sped up 40x)



Very hard to change mathematical content later!

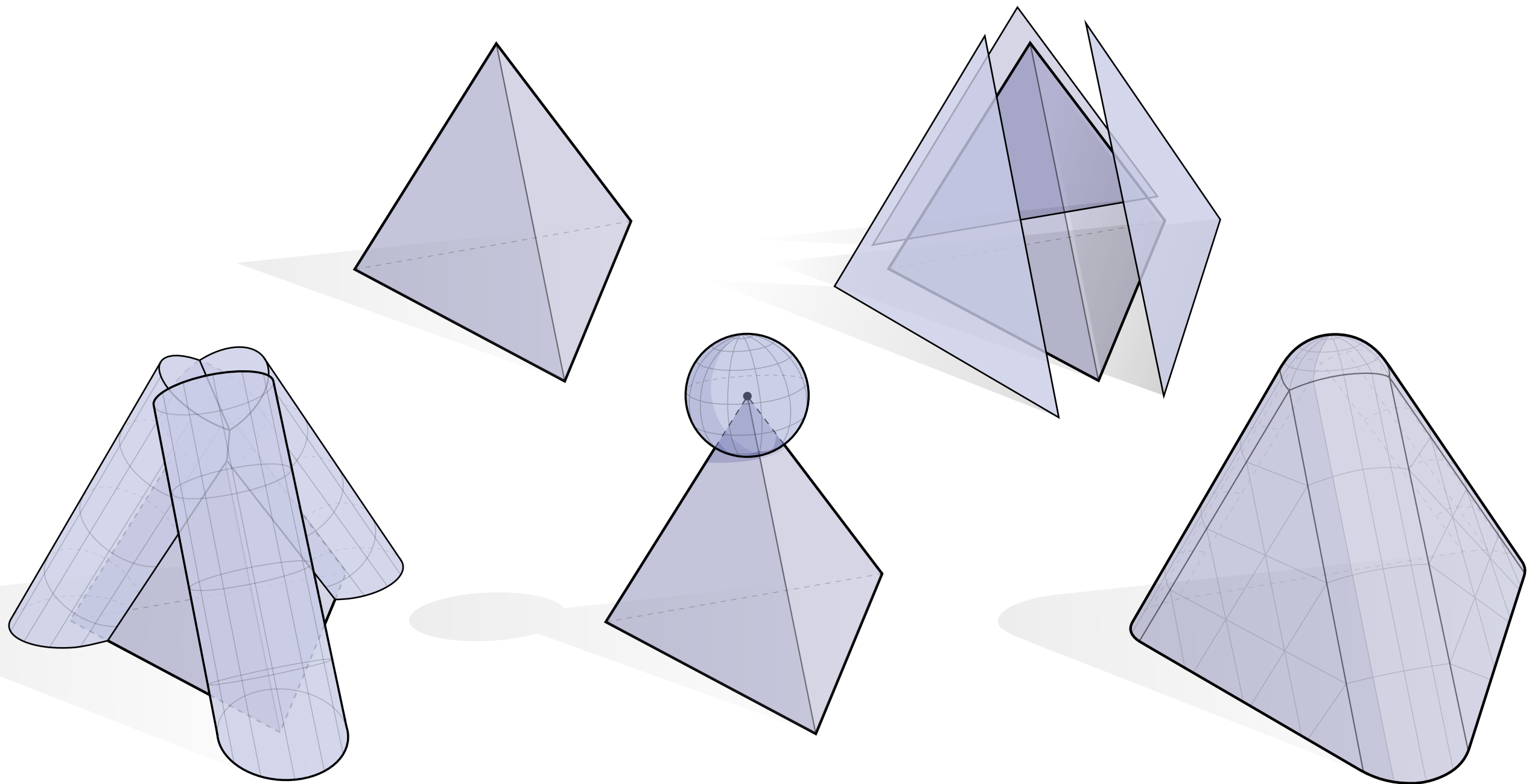


Lots of clicking and dragging... (sped up 40x)



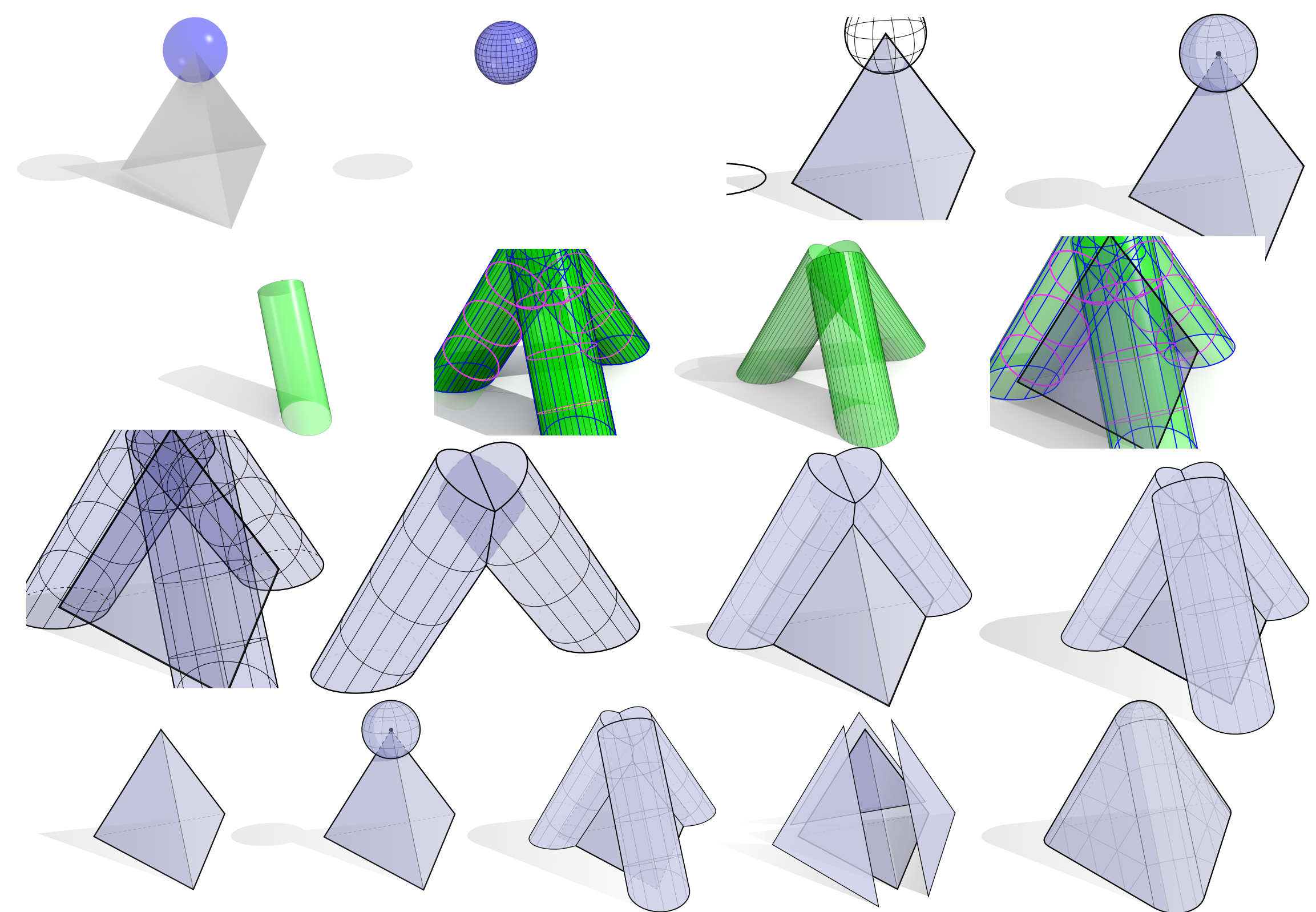
Very hard to change mathematical content later!

# Example: Illustrating Steiner's polyhedral formula



Looks easy, right?

# Reality: **8 hours** of clicking & dragging



Source: Keenan Crane



# Graphical User Interface

**Accessibility**



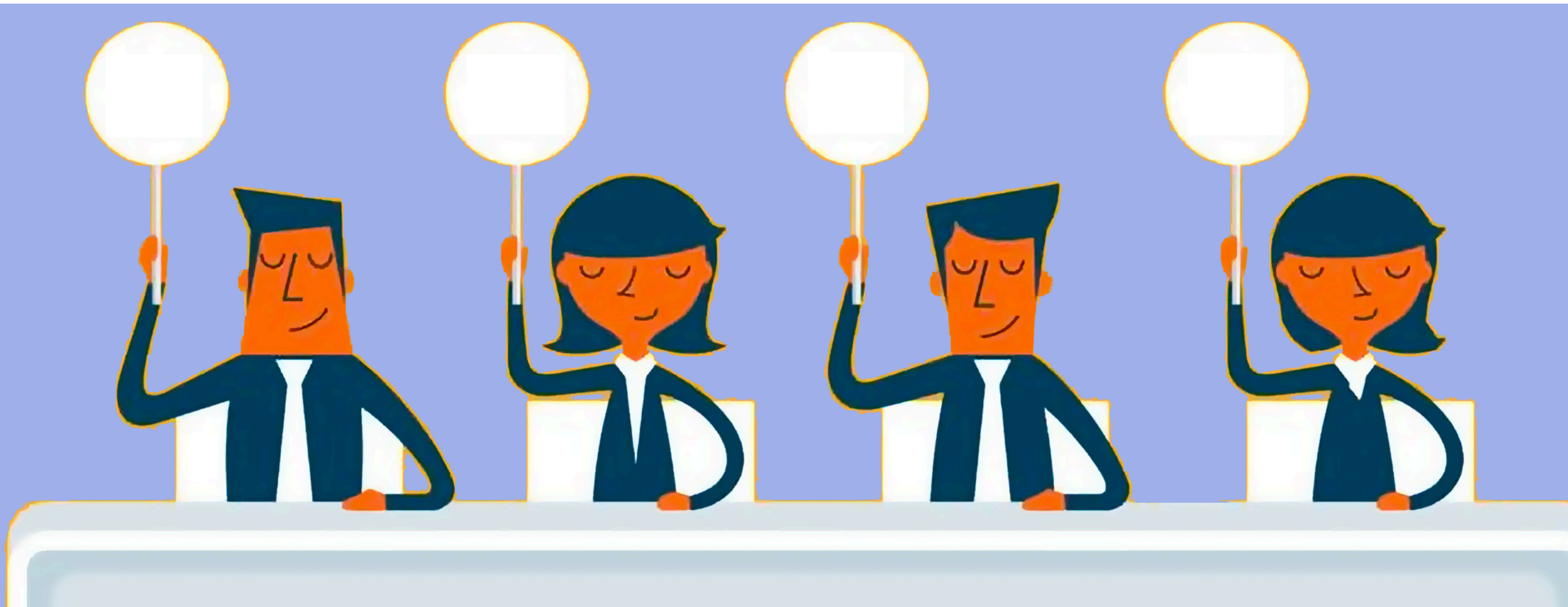
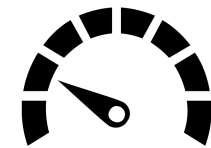
**Universality**



**Beauty**



**Productivity**





# Graphical User Interface

**Accessibility**



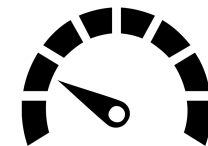
**Universality**



**Beauty**



**Productivity**



4







# Graphical User Interface

**Accessibility**



**Universality**



**Beauty**



**Productivity**



4

10





# Graphical User Interface

**Accessibility**



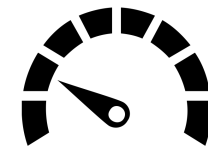
**Universality**



**Beauty**



**Productivity**



4

10

10





# Graphical User Interface

**Accessibility**



**Universality**



**Beauty**



**Productivity**



4



10



10



1







# **Low-Level Languages**

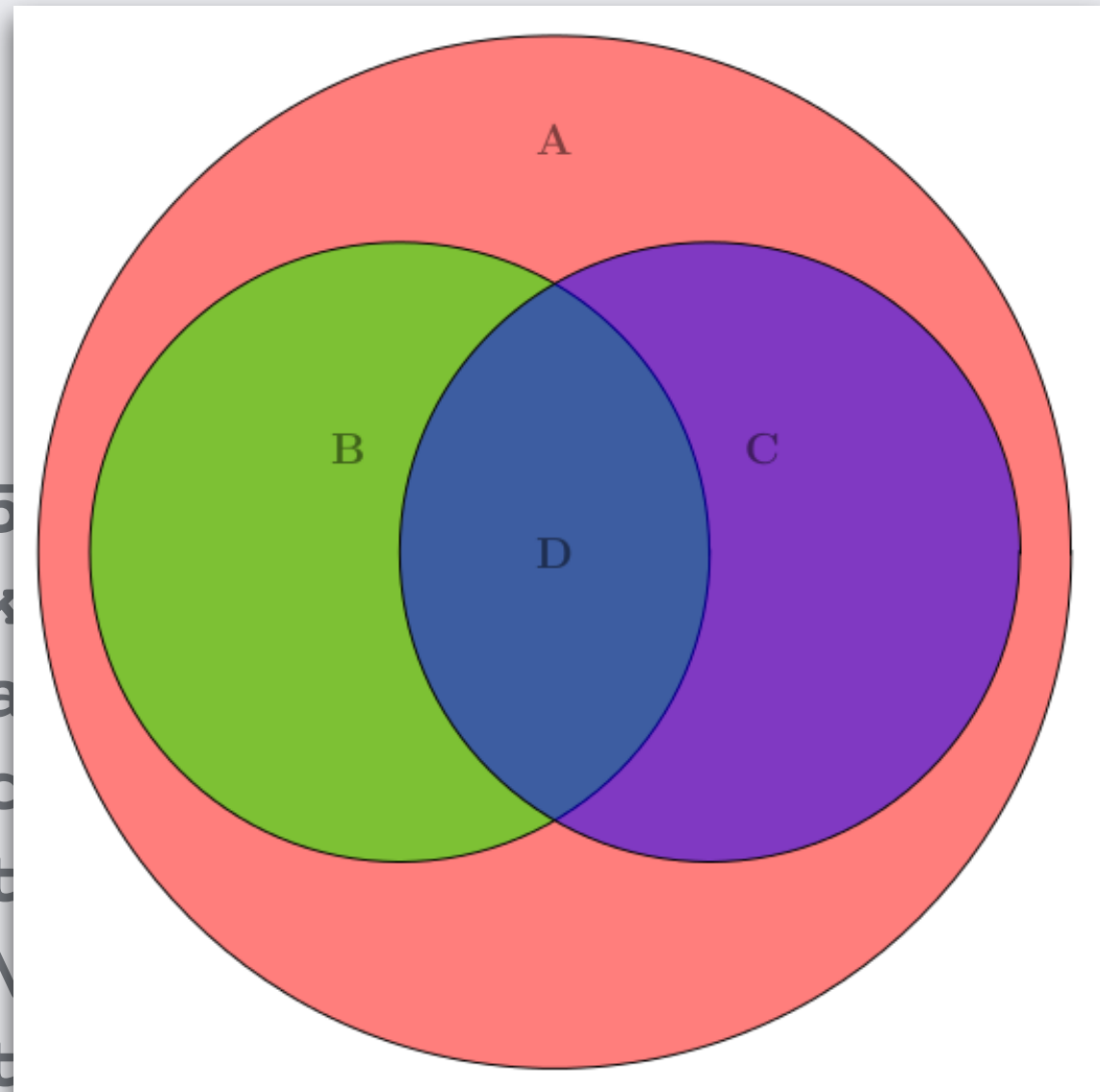
Examples: PostScript, TikZ

# Illustrating basic set relationships in TikZ

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
\pagestyle{empty}
\begin{tikzpicture}
  \begin{scope}[shift={(3cm,-5cm)}, fill opacity=0.5]
    \draw[fill=red, draw = black] (0,0) circle (5);
    \draw[fill=green, draw = black] (-1.5,0) circle (3);
    \draw[fill=blue, draw = black] (1.5,0) circle (3);
    \node at (0,4) (A) {\large\textbf{A}};
    \node at (-2,1) (B) {\large\textbf{B}};
    \node at (2,1) (C) {\large\textbf{C}};
    \node at (0,0) (D) {\large\textbf{D}};
  \end{scope}
\end{tikzpicture}
\end{document}
```

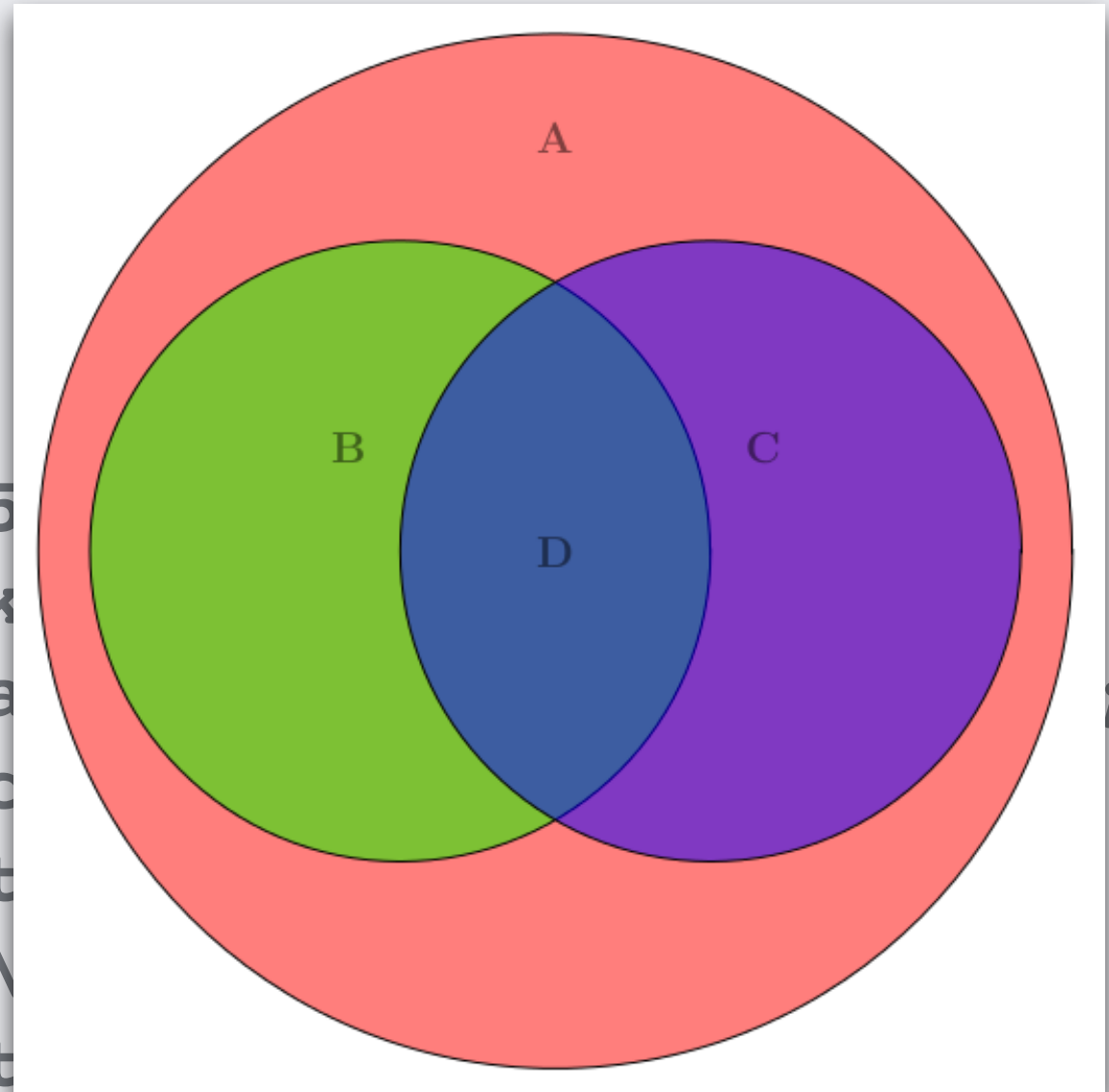
# Illustrating basic set relationships in TikZ

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
\pagestyle{empty}
\begin{tikzpicture}
  \begin{scope}[shift={(3cm,-5cm)}]
    \draw[fill=red, draw=black] circle (3cm);
    \draw[fill=green, draw=black] circle (2cm);
    \draw[fill=blue, draw=black] circle (2cm);
    \node at (0,4) (A) {\large\textcolor{red}{A}};
    \node at (-2,1) (B) {\large\textcolor{green}{B}};
    \node at (2,1) (C) {\large\textcolor{blue}{C}};
    \node at (0,0) (D) {\large\textbf{D}};
  \end{scope}
\end{tikzpicture}
\end{document}
```



# Illustrating basic set relationships in TikZ

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
\pagestyle{empty}
\begin{tikzpicture}
  \begin{scope}[shift={(3cm,-5cm)}]
    \draw[fill=red, draw=black] circle (3cm);
    \draw[fill=green, draw=black] circle (2cm);
    \draw[fill=blue, draw=black] circle (2cm);
    \node at (0,4) (A) {\large A};
    \node at (-2,1) (B) {\large B};
    \node at (2,1) (C) {\large C};
    \node at (0,0) (D) {\large D};
  \end{scope}
\end{tikzpicture}
\end{document}
```

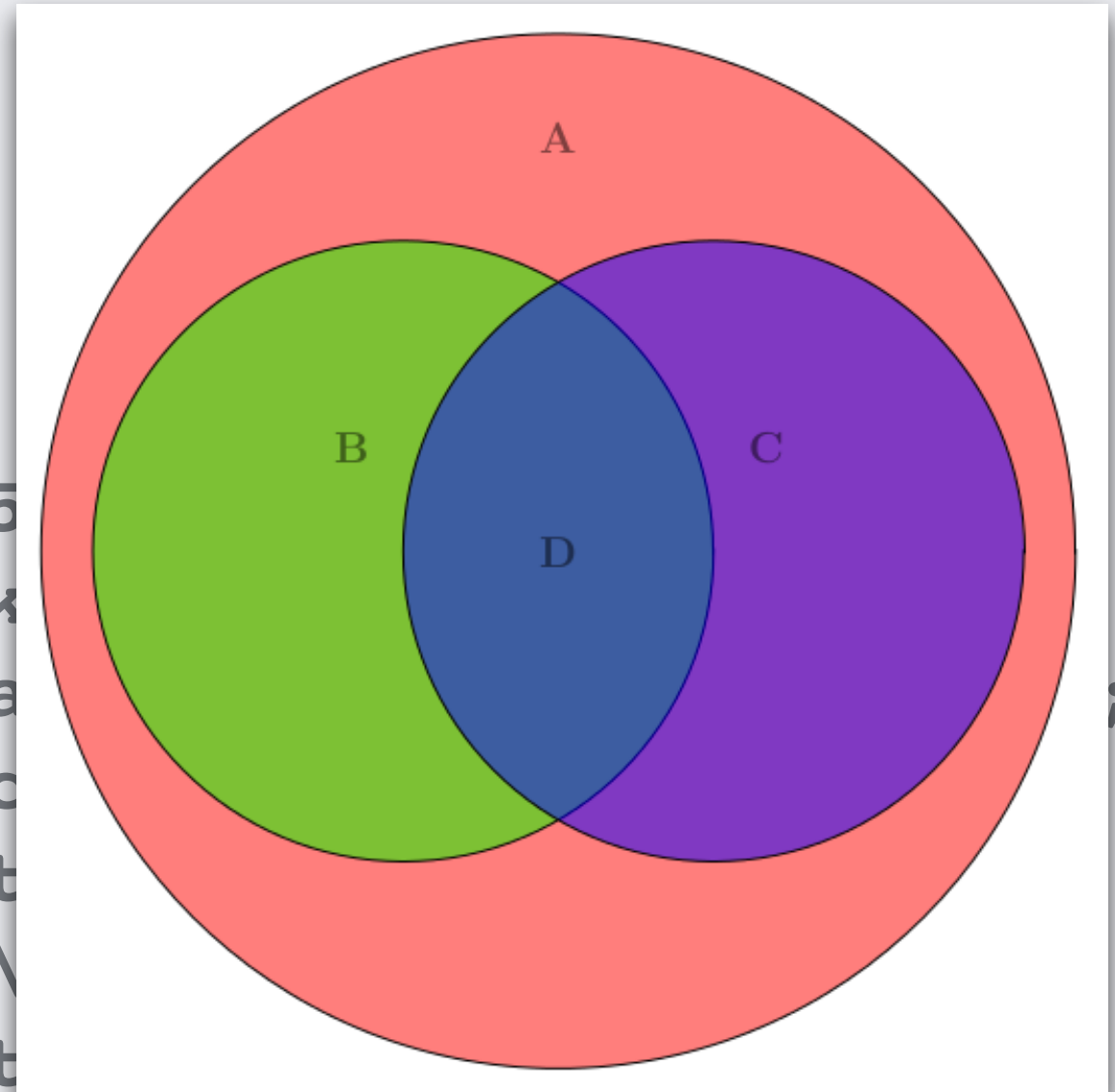


Low-level specification of coordinates

**Meaning** of the diagram is **lost**

# Illustrating basic set relationships in TikZ

```
\documentclass{article}
\usepackage{tikz}
\begin{document}
\pagestyle{empty}
\begin{tikzpicture}
  \begin{scope}[shift={(3cm,-5cm)}]
    \draw[fill=red, draw=black] circle[radius=4cm];
    \draw[fill=green, draw=black] circle[radius=2.5cm];
    \draw[fill=blue, draw=black] circle[radius=2.5cm];
    \node at (0,4) (A) {\large A};
    \node at (-2,1) (B) {\large B};
    \node at (2,1) (C) {\large C};
    \node at (0,0) (D) {\large D};
  \end{scope}
\end{tikzpicture}
\end{document}
```



Low-level specification of coordinates

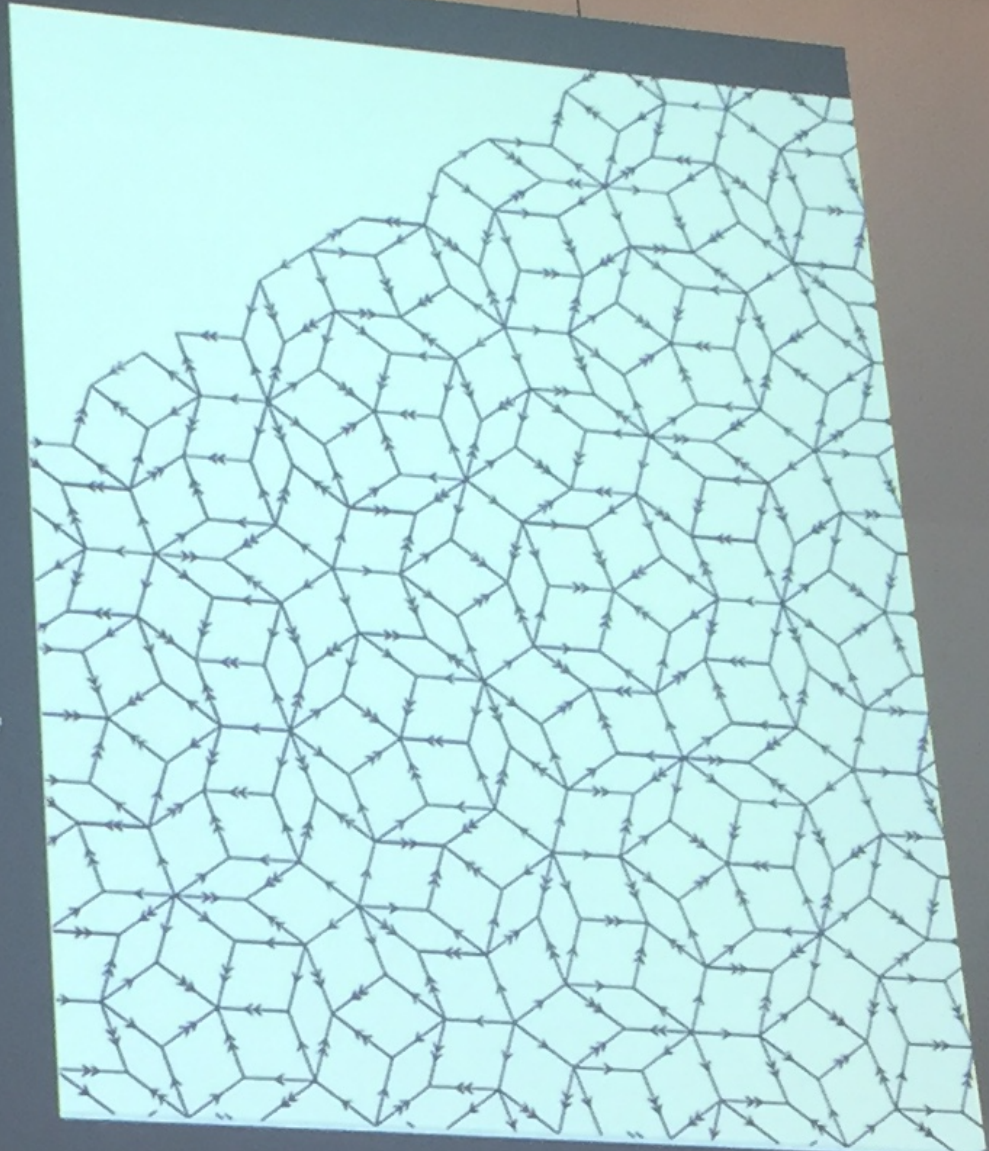
**Meaning** of the diagram is **lost**

Also hard to modify mathematical content



# Can however handle significant complexity...

```
474 /DIR exch def
475 /TYPE exch def
476 /y3 exch def
477 /x3 exch def
478 /y2 exch def
479 /x2 exch def
480 /y1 exch def
481 /x1 exch def
482
483 % 0: tL → tL
484 MAP 0 eq {
485
486 [x3 y3 x1 y1 0 1] Third
487
488 }{
489
490 % 1: tR → tR
491 MAP 1 eq {
492
493 [x2 y2 x3 y3 0 0] Third
494
495 }{
496
497 % 2: TL → TL
498 MAP 2 eq {
499
500 /m { y3 y1 sub x3 x1 sub div } def
501 /n { x3 x1 sub dup mul y3 y1 sub dup mul add sqrt } def
502 /l { x2 x1 sub dup mul y2 y1 sub dup mul add sqrt } def
503
504 x1 x3 sub 0 lt {
505   /x4 { n tau mul dup mul m dup mul 1 add div sqrt x3 add } def
506 }{
507   /x4 { n tau mul dup mul m dup mul 1 add div sqrt neg x3 add } def
508 }ifelse
509
510 /y4 { x4 x3 sub m mul y3 add } def
511 [x4 y4 x1 y1 x2 y2 1 1]
512
513 }{
514
515 % 3: TR → TR
516 MAP 3 eq {
517
518 /m { y3 y2 sub x3 x2 sub div } def
519 /n { x3 x2 sub dup mul y3 y2 sub dup mul add sqrt } def
520 /l { x2 x1 sub dup mul y2 y1 sub dup mul add sqrt } def
```





# Low-Level Languages

**Accessibility**



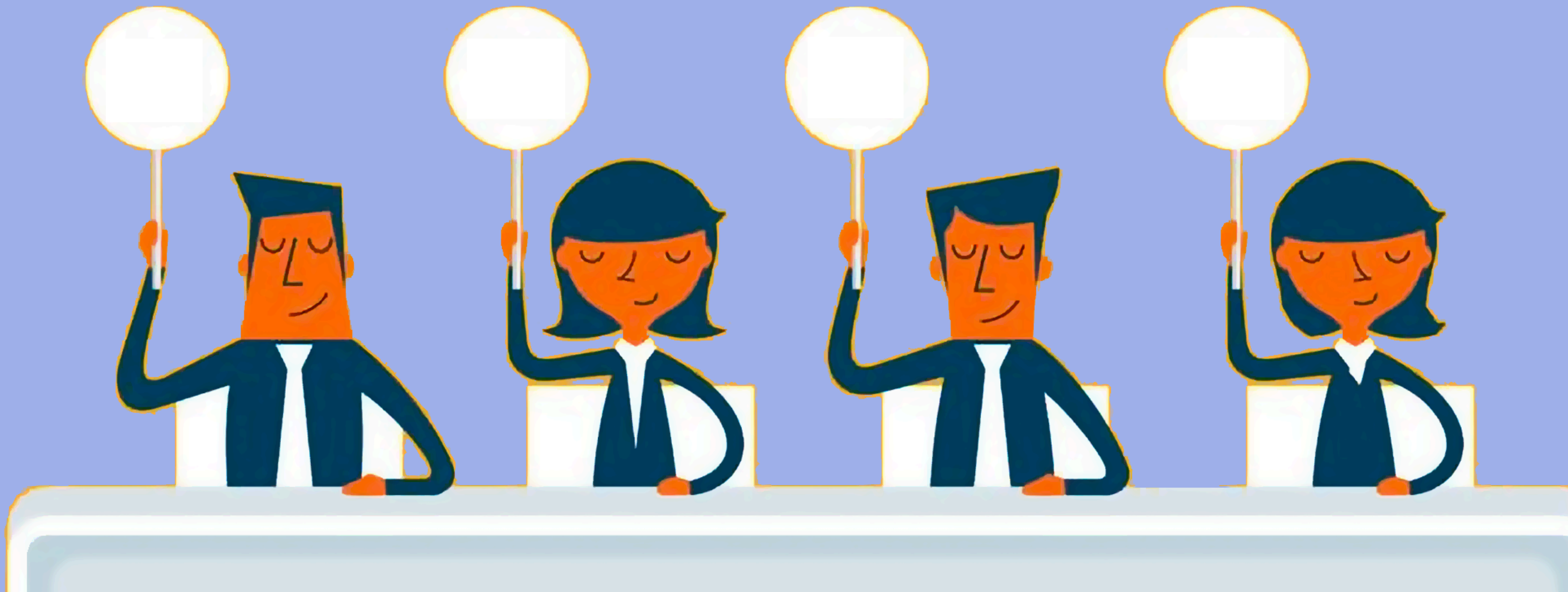
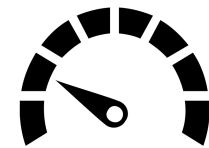
**Universality**



**Beauty**



**Productivity**







# Low-Level Languages

Accessibility



Universality



Beauty



Productivity



2







# Low-Level Languages

Accessibility



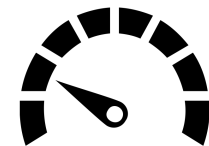
Universality



Beauty



Productivity



2

8



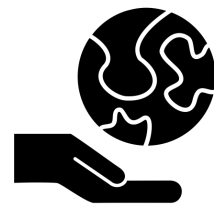


# Low-Level Languages

Accessibility



Universality



Beauty



Productivity



2



8



4





# Low-Level Languages

Accessibility



Universality



Beauty



Productivity



2



8



4



5

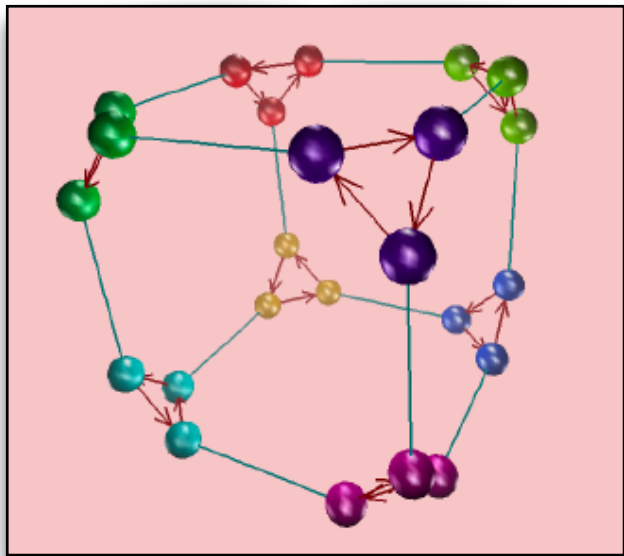




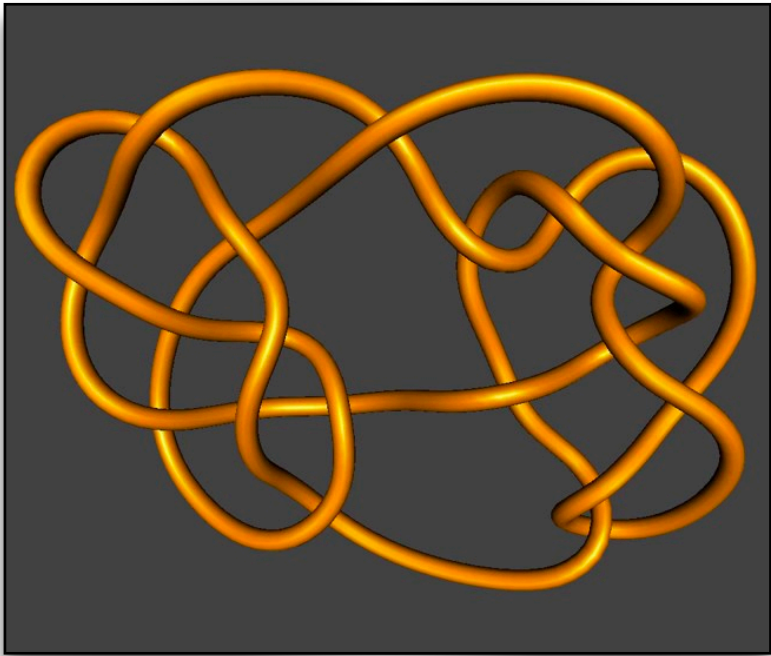
# **Specialized Packages**

Examples: KnotPlot, Group Explorer

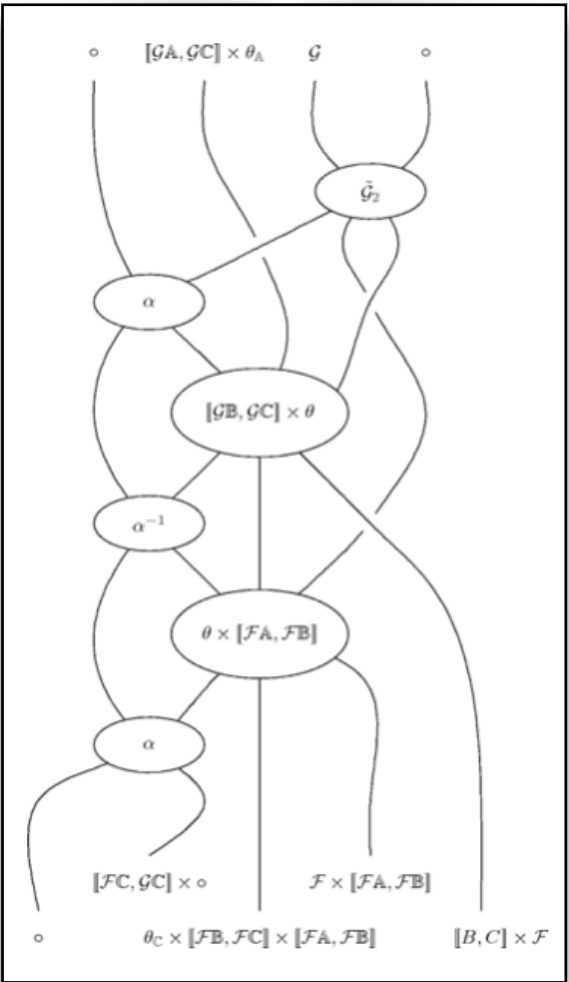
# More examples



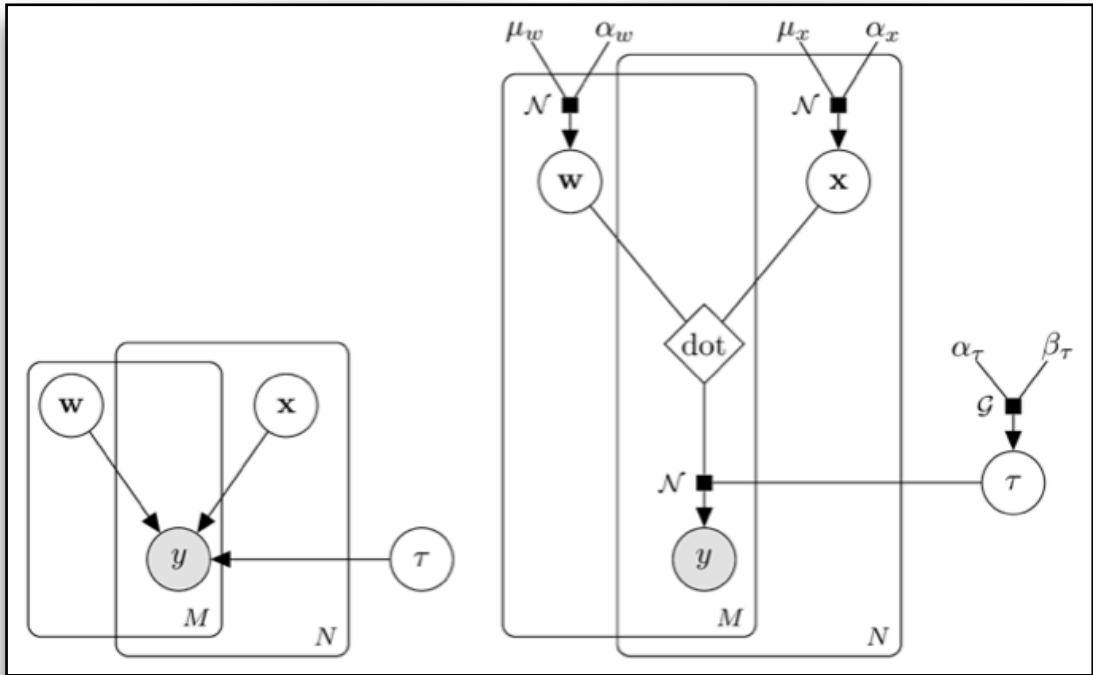
Group Explorer



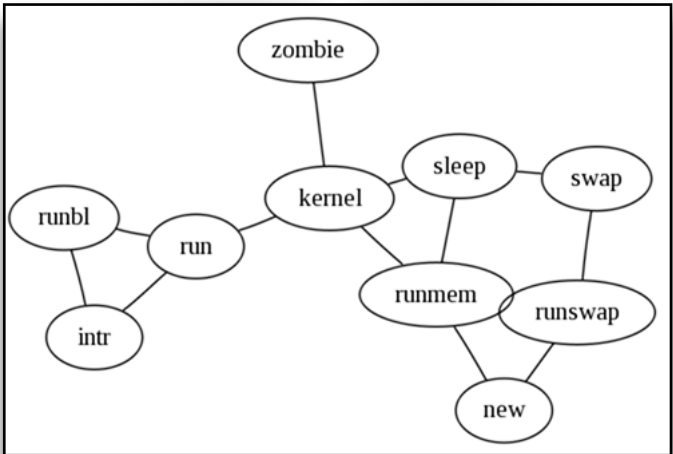
KnotPlot



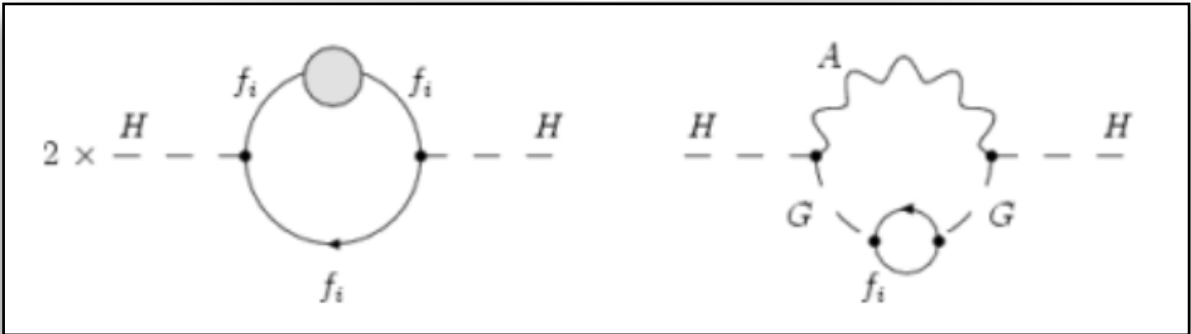
strid



BayesNet



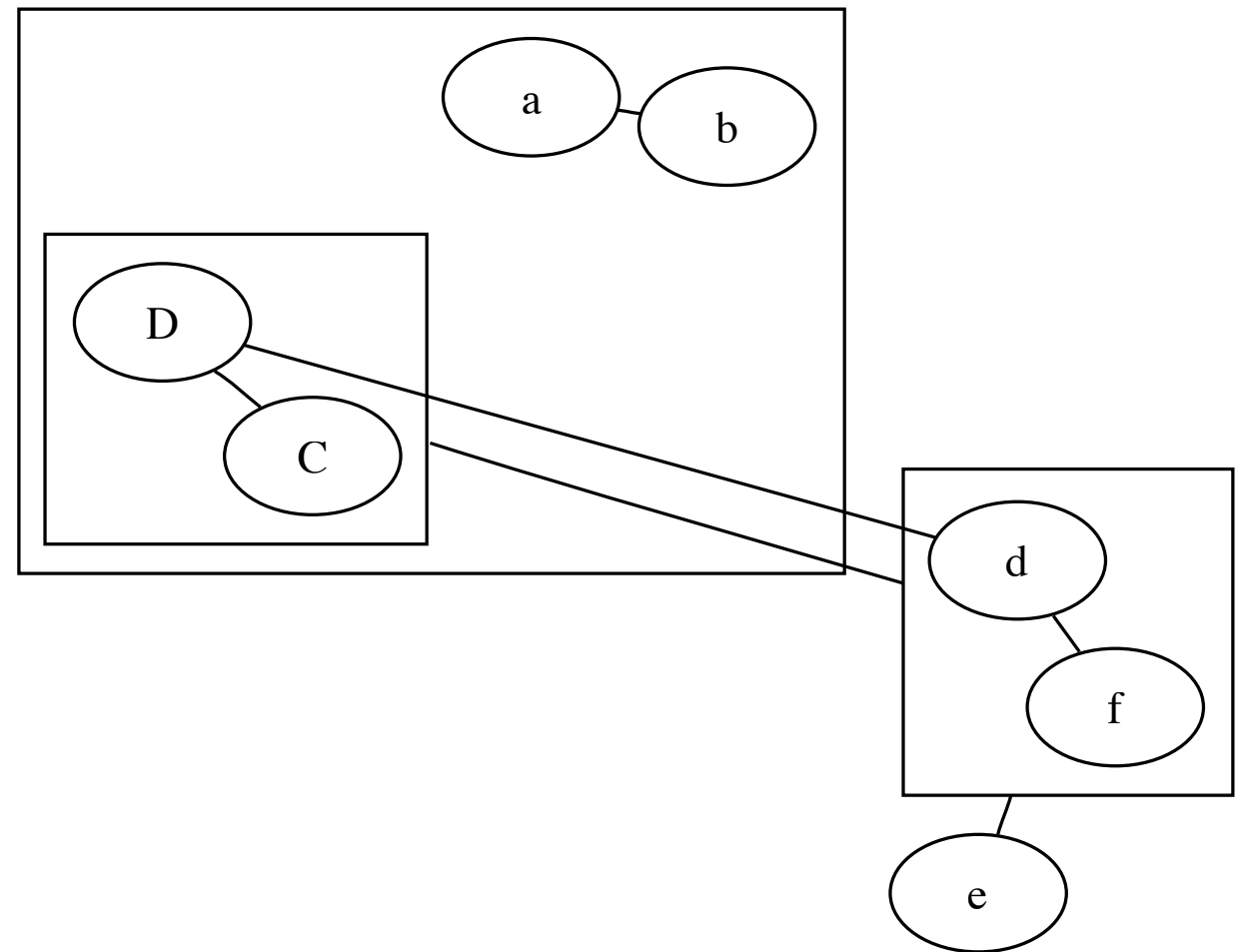
GraphViz



JaxoDraw

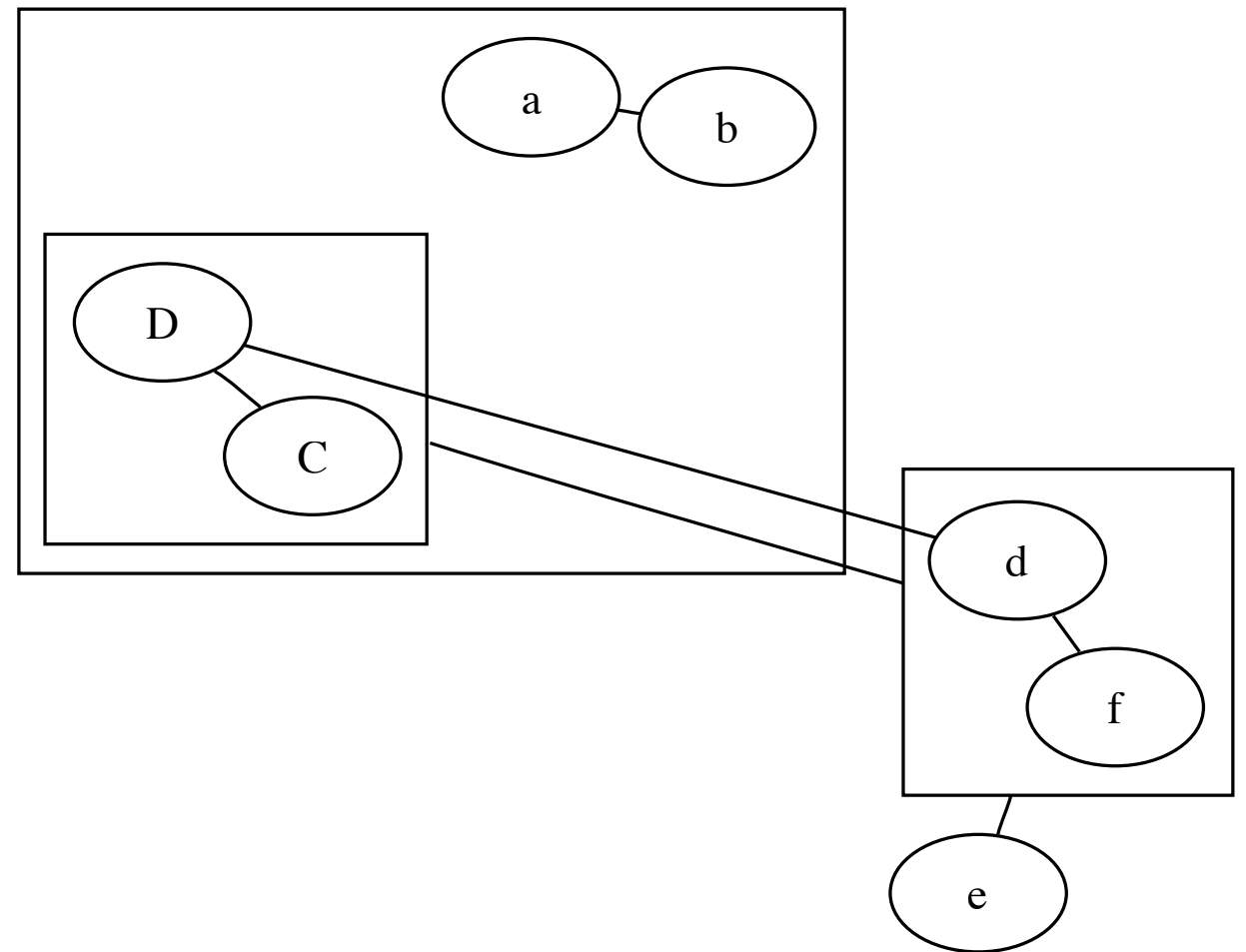
# Example: graph visualization using Graphviz

```
graph G {  
    e  
    subgraph clusterA {  
        a -- b;  
        subgraph clusterC {  
            C -- D;  
        }  
    }  
    subgraph clusterB {  
        d -- f  
    }  
    d -- D  
    e -- clusterB  
    clusterC -- clusterB  
}
```



# Example: graph visualization using Graphviz

```
graph G {  
  e  
  subgraph clusterA {  
    a -- b;  
    subgraph clusterC {  
      C -- D;  
    }  
  }  
  subgraph clusterB {  
    d -- f  
  }  
  d -- D  
  e -- clusterB  
  clusterC -- clusterB  
}
```



High-level & clean—but only works for graphs!





# Specialized Packages

**Accessibility**



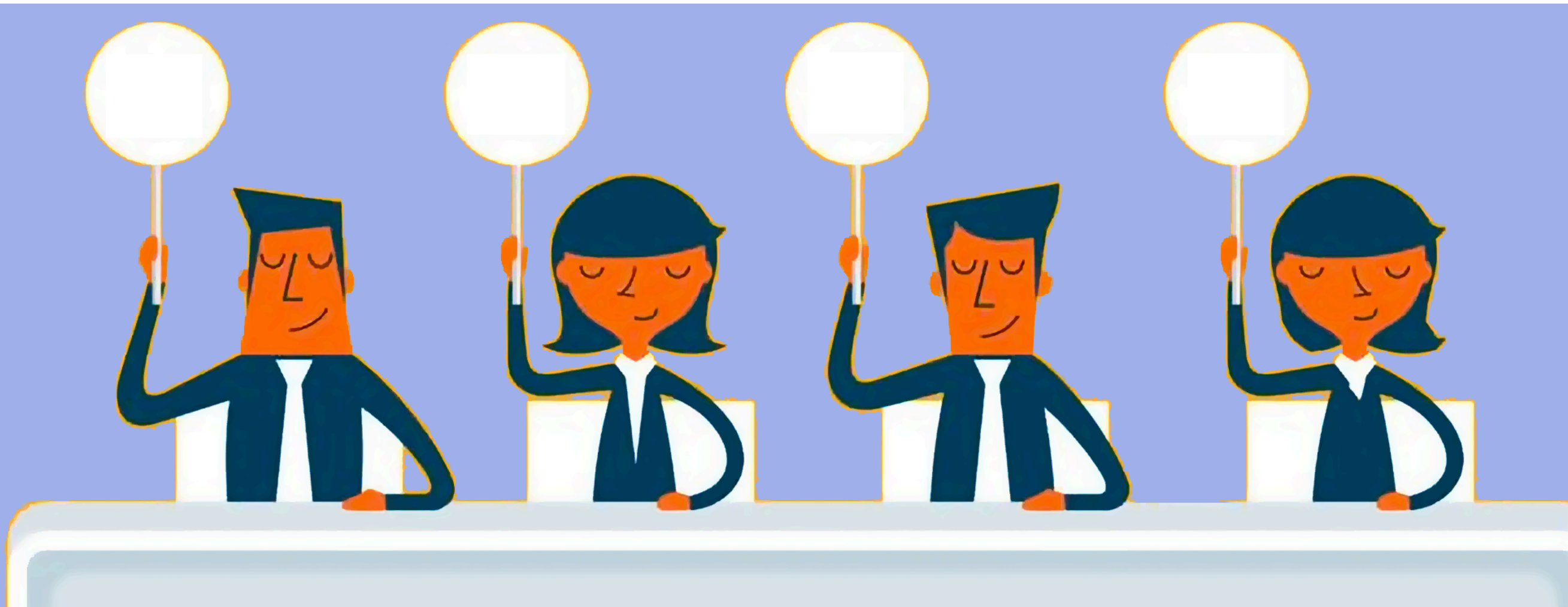
**Universality**



**Beauty**



**Productivity**





# Specialized Packages

**Accessibility**



**Universality**



**Beauty**



**Productivity**



8





# Specialized Packages

**Accessibility**



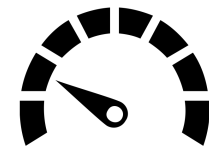
**Universality**



**Beauty**



**Productivity**



8

1





# Specialized Packages

**Accessibility**



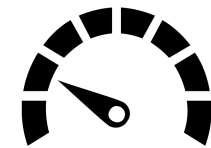
**Universality**



**Beauty**



**Productivity**



8



1



7



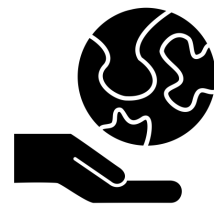


# Specialized Packages

**Accessibility**



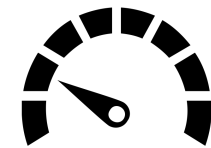
**Universality**



**Beauty**



**Productivity**



8



1



7



9





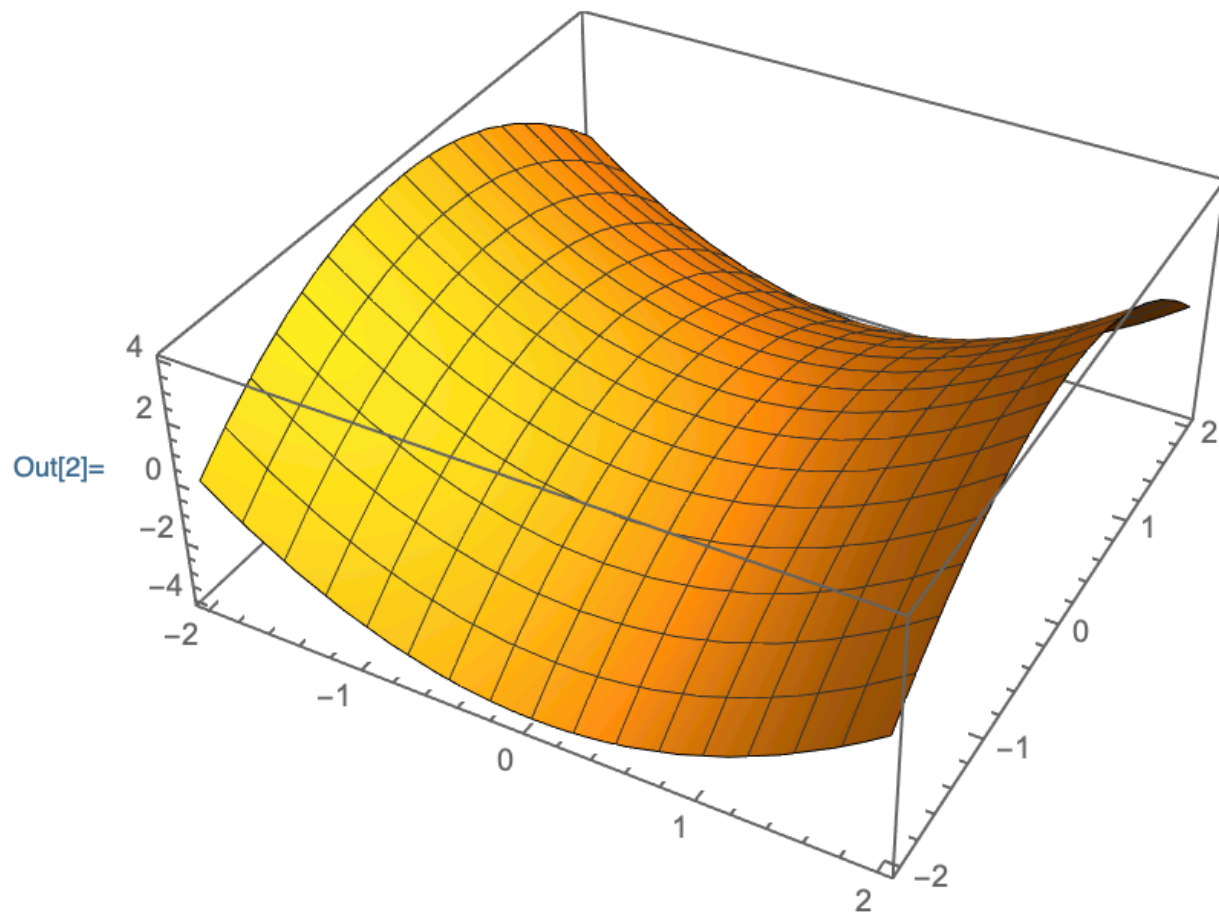
# Plotting Software

Examples: Matplotlib, MATLAB



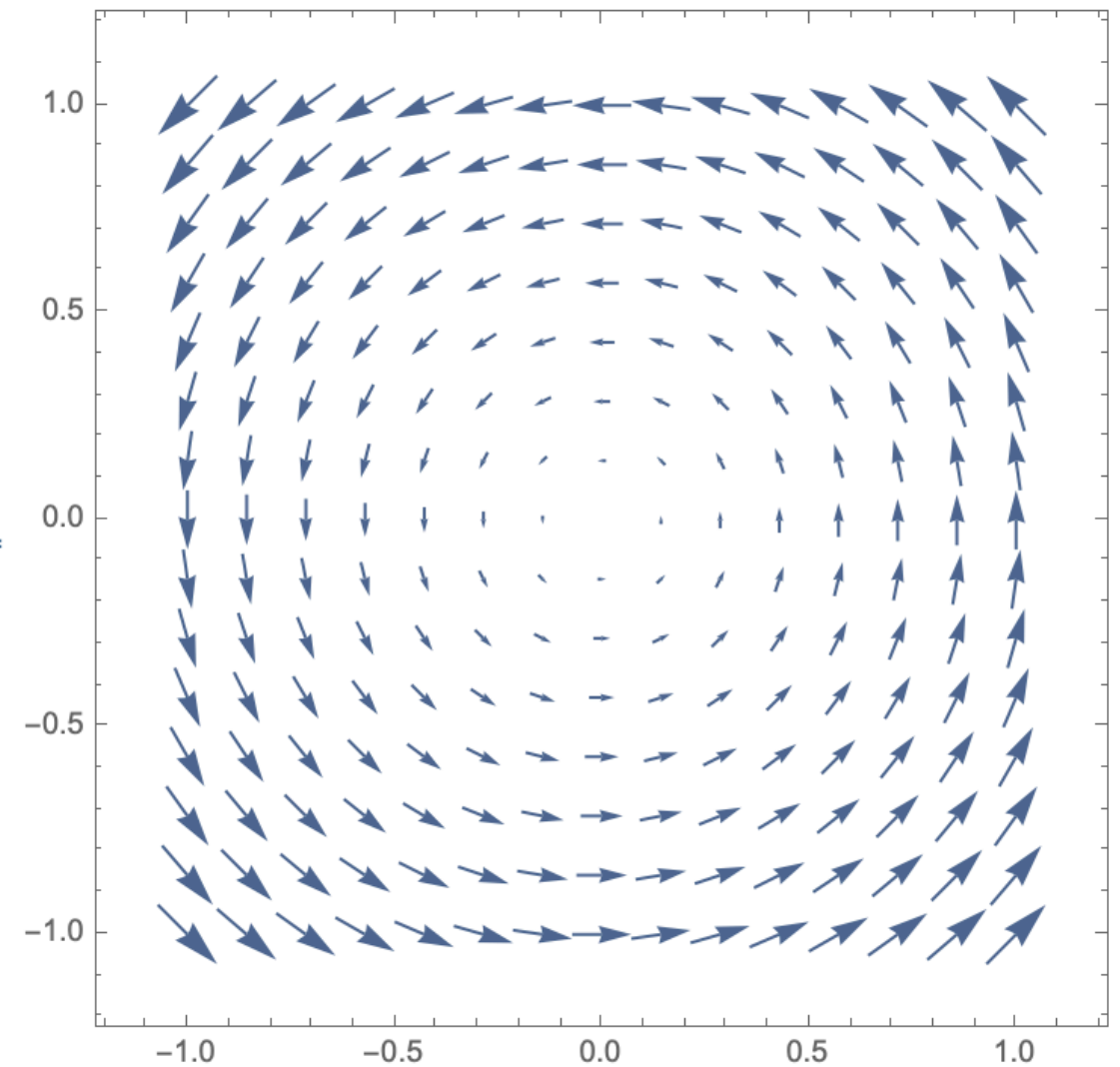
# Example: Mathematica

```
In[2]:= Plot3D[x2 - y2, {x, -2, 2}, {y, -2, 2}]
```



```
In[4]:= VectorPlot[{-y, x}, {x, -1, 1}, {y, -1, 1}]
```

Out[4]=



Still need to give explicit coordinates

Meaning of expressions easily lost

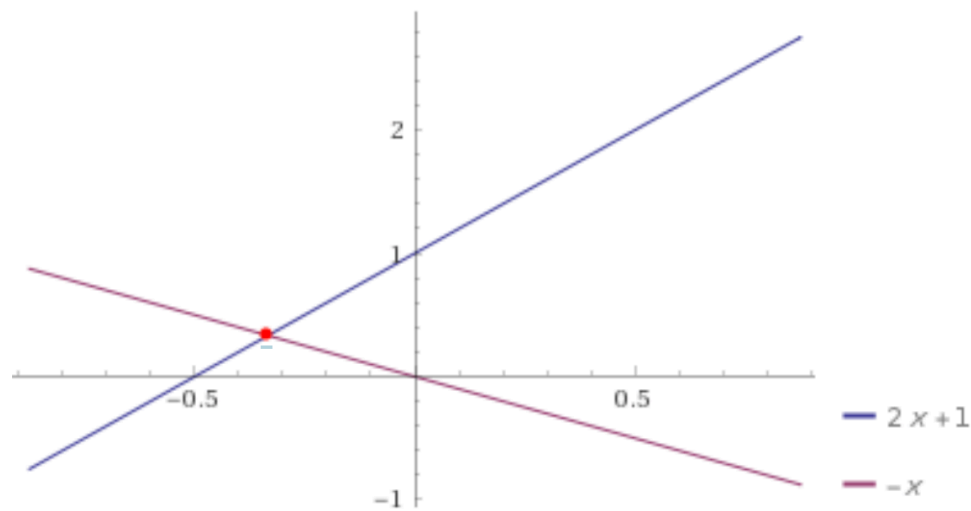


# Example: Wolfram Alpha

intersection of two lines



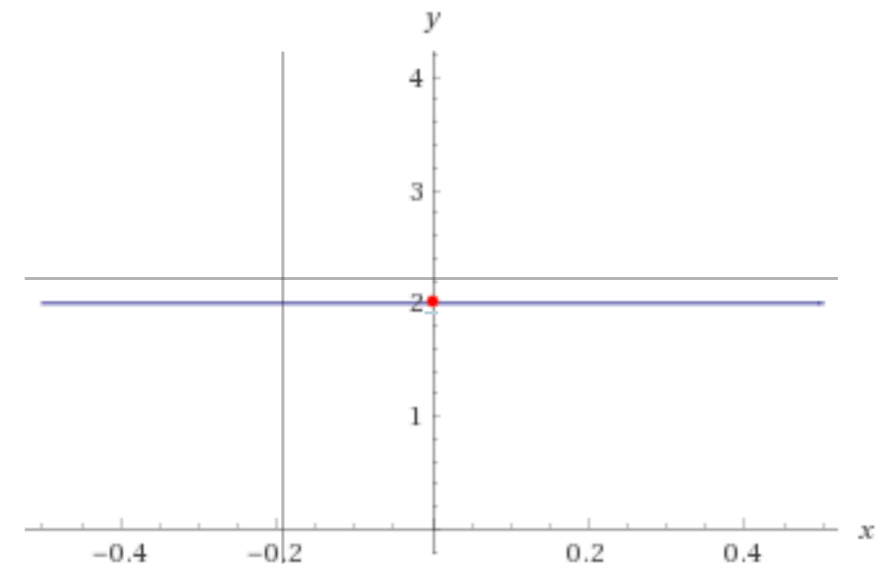
Plot:



intersection of two spheres



Plot:



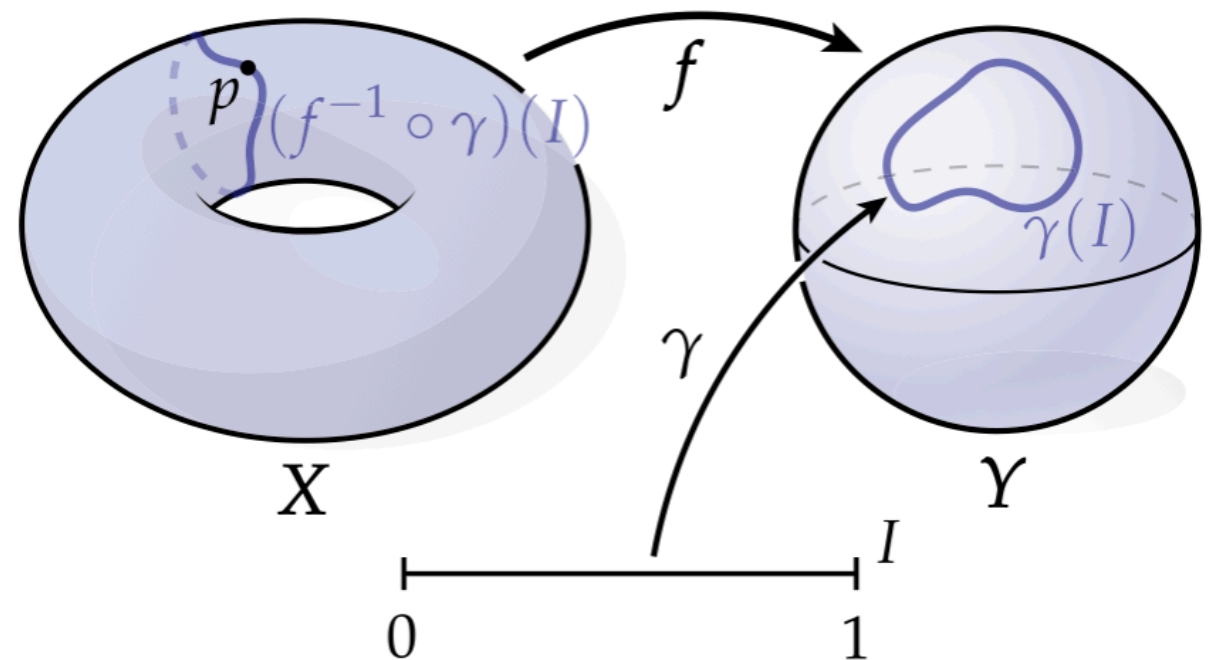
Using closest Wolfram|Alpha interpretation: **intersection of two**



Handles more conceptual statements  
Cases it can handle are fairly “canned”...

Imagine attempting something like this\*...

```
TopologicalSpace X,Y
 $\pi_1(X)$  = DirectProduct(Ints,Ints)
 $\pi_1(Y)$  = TrivialGroup
I := [0,1] Subset Reals
ContinuousMap f : X -> Y
ContinuousMap gamma : I -> Y
gamma(0) = gamma(1)
eta1 = Image(gamma)
eta2 = PreImage(eta1,f)
p In eta2
```



\*Note: just a mockup!



# Plotting Software

**Accessibility**



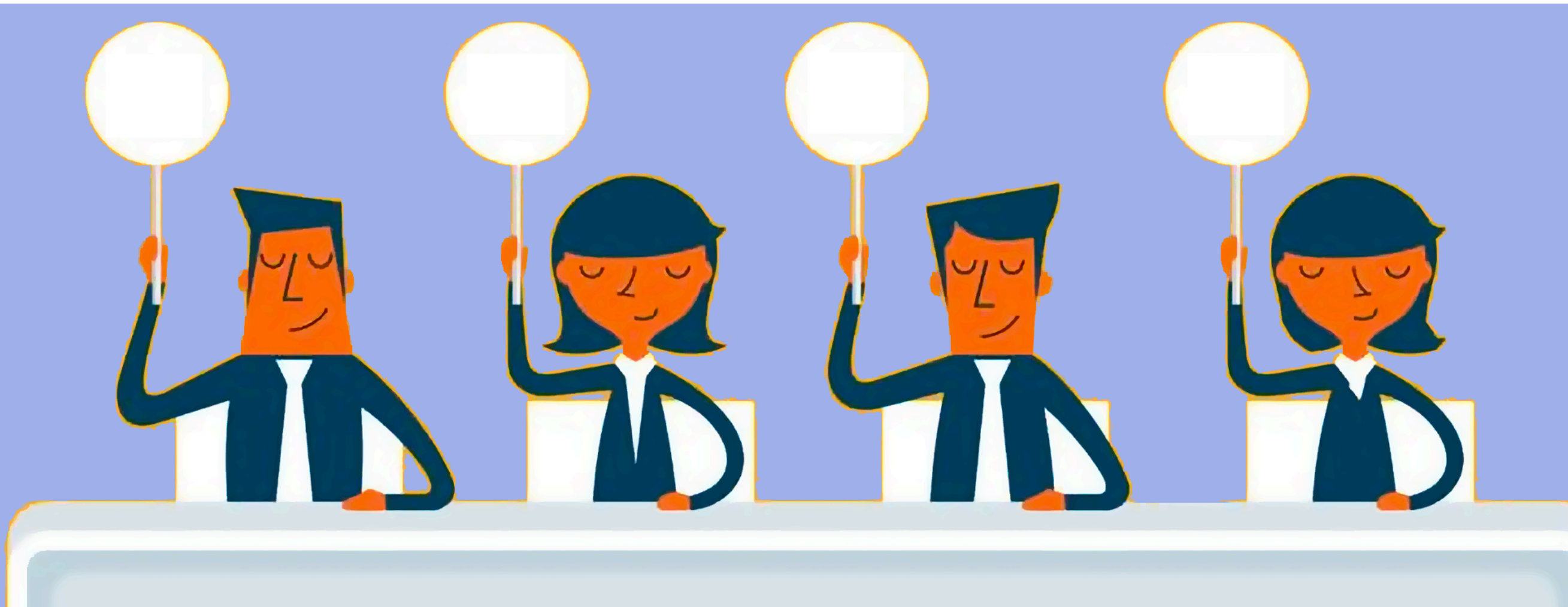
**Universality**



**Beauty**



**Productivity**



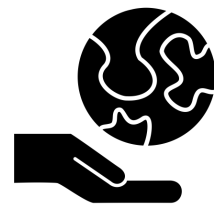


# Plotting Software

**Accessibility**



**Universality**



**Beauty**



**Productivity**



8



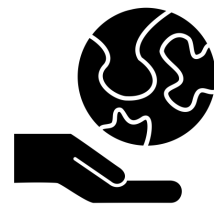


# Plotting Software

**Accessibility**



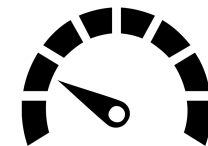
**Universality**



**Beauty**



**Productivity**



8

2





# Plotting Software

**Accessibility**



**Universality**



**Beauty**



**Productivity**



8



2



6





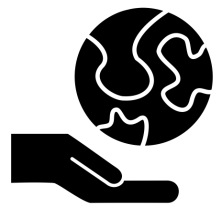


# Plotting Software

**Accessibility**



**Universality**



**Beauty**



**Productivity**



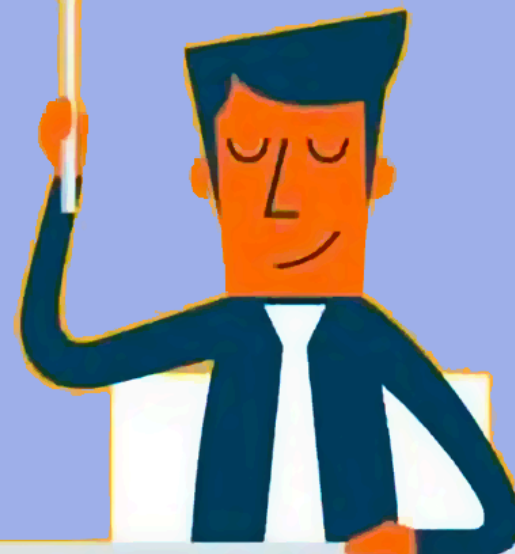
8



2



6



6



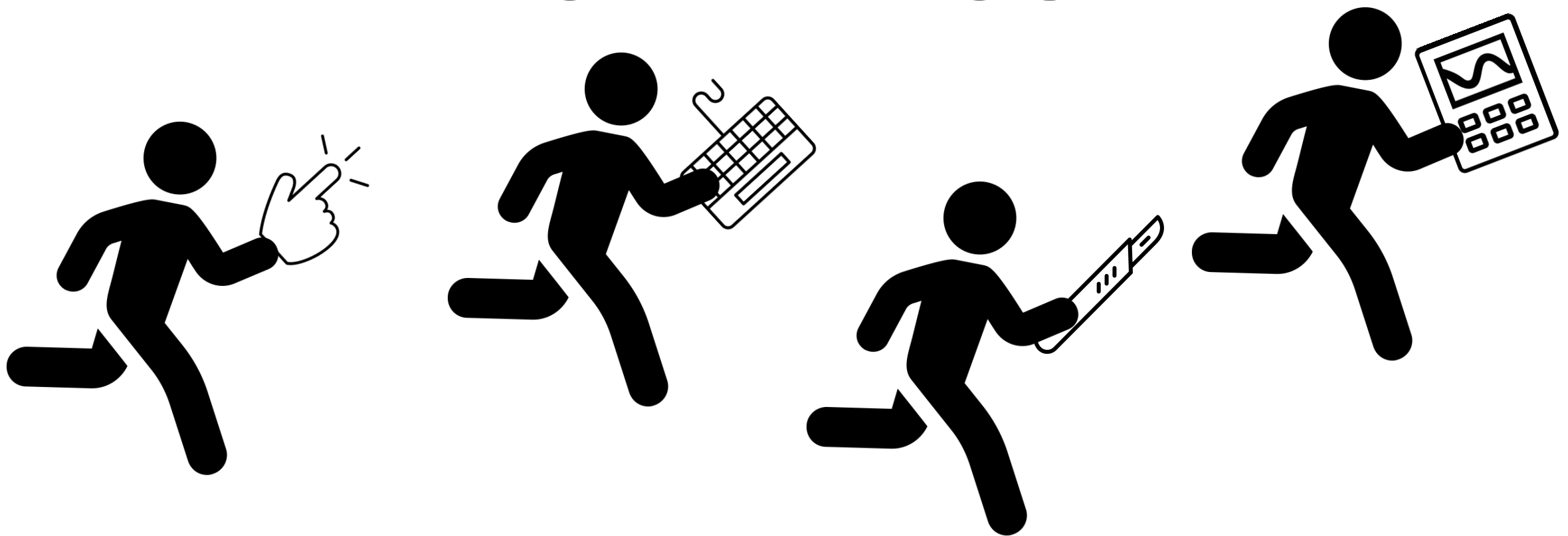


Diagramming is still hard!

**Providence 2019**



# DIAGRAM TOOL OLYMPICS



**Providence 2019**



# DIAGRAM TOOL OLYMPICS



**Providence 2019**



# DIAGRAM TOOL OLYMPICS



**Providence 2019**



# DIAGRAM TOOL OLYMPICS





**Providence 2019**



# DIAGRAM TOOL OLYMPICS



## **Part III:** A new language-based tool



A yellow triangle pointing downwards, tilted to the right, containing the text "Work in progress!".

**Work  
in  
progress!**

**Part III:** A new language-based tool

*How would you visualize this statement?*

*How would you visualize this statement?*

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

*How would you visualize this statement?*

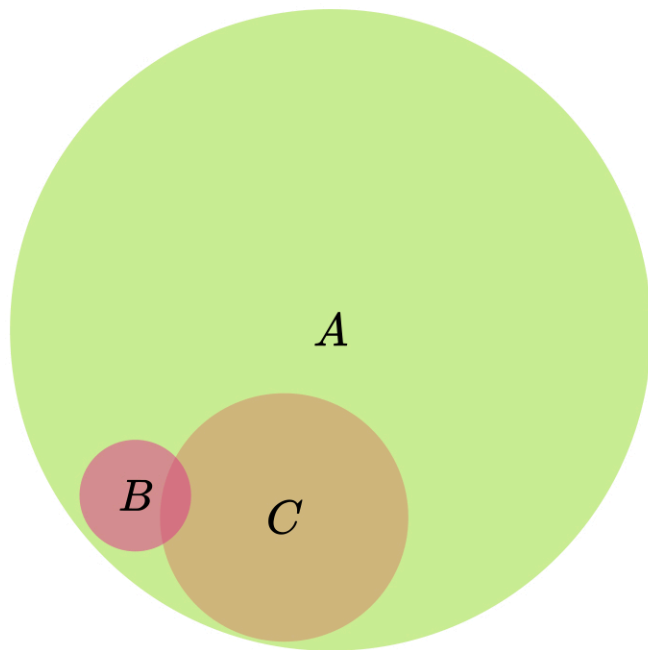
Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

one style

*How would you visualize this statement?*

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

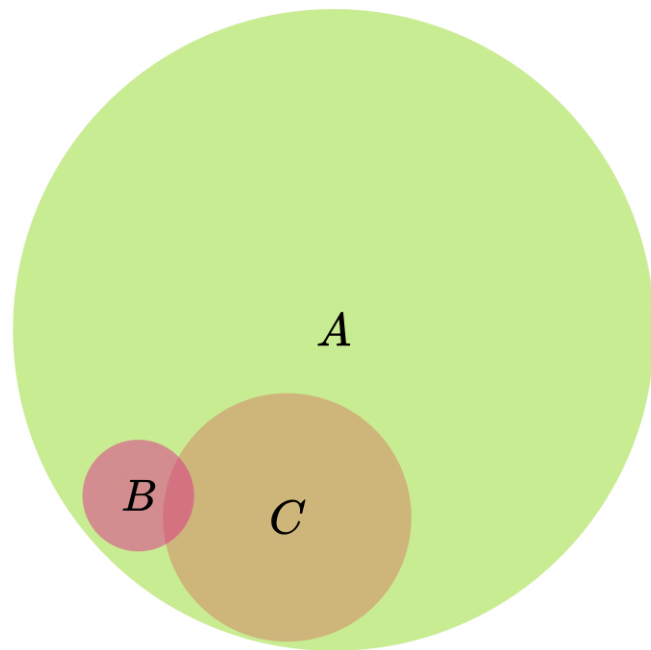
one style



*How would you visualize this statement?*

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

one style



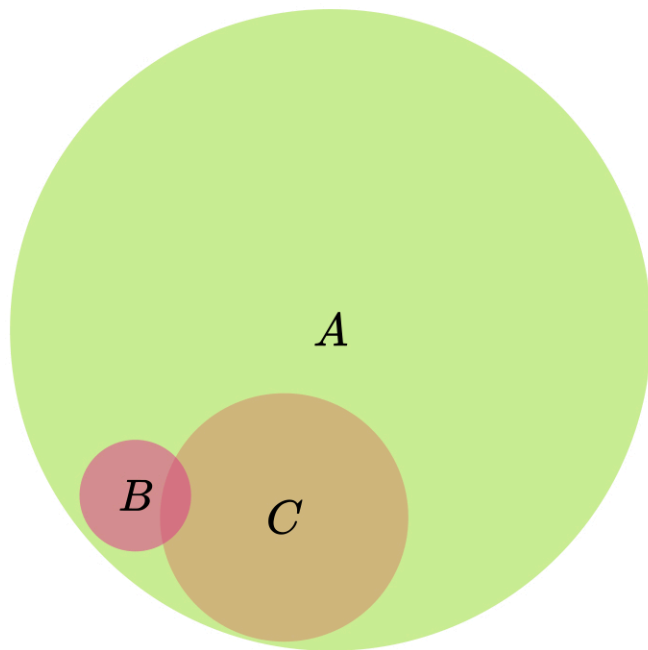
another style



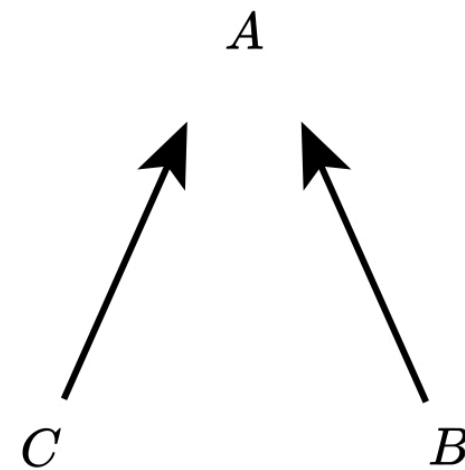
*How would you visualize this statement?*

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

one style



another style

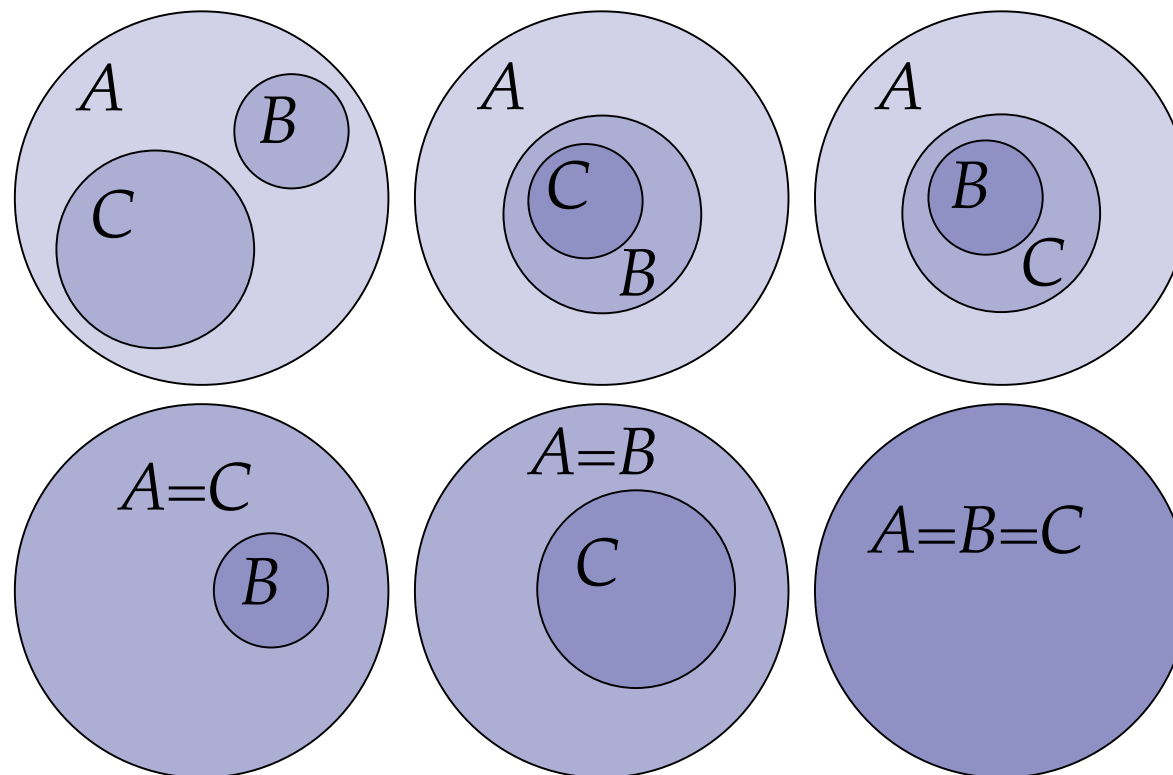


Also: miss a lot of possibilities if  
you draw just one diagram...

$$B \subseteq A \text{ and } C \subseteq A$$

Also: miss a lot of possibilities if  
you draw just one diagram...

$$B \subseteq A \text{ and } C \subseteq A$$



Let's try this in Penrose  
(demo)

**Question:** How can we do a better job of connecting language and visualization?

## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

Set



## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

logical object

# Set

## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

logical object

# Set

 $\subseteq$

## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

logical object

# Set

logical relationship

# $\subseteq$

## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

logical object

Set



logical relationship

$\subseteq$

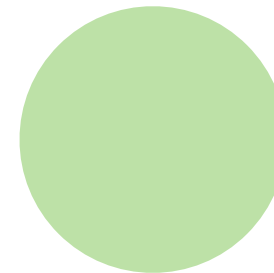


# Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

logical object

Set



logical relationship

$\subseteq$



# Key idea

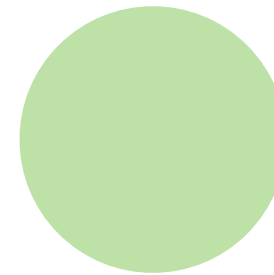
Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

logical object

Set



visual object



logical relationship

$\subseteq$



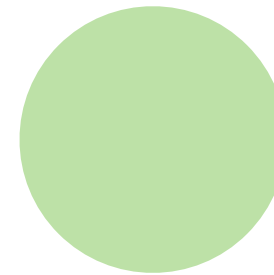


# Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

logical object

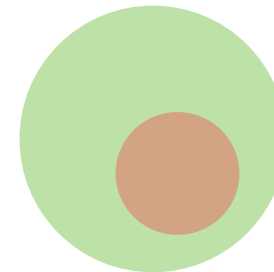
Set



visual object

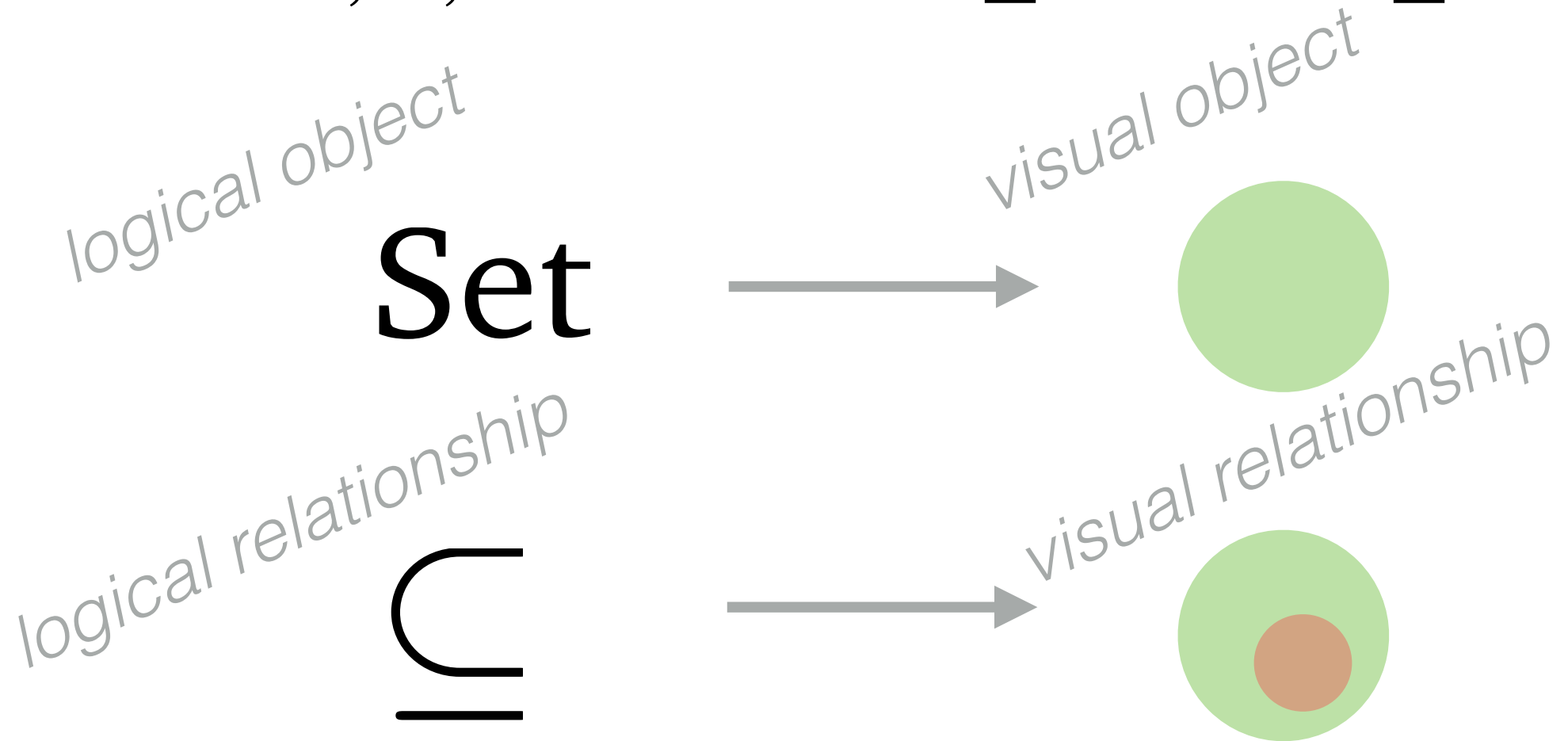
logical relationship

$\subseteq$



# Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .



## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

Set 

$\subseteq$  

## Key idea

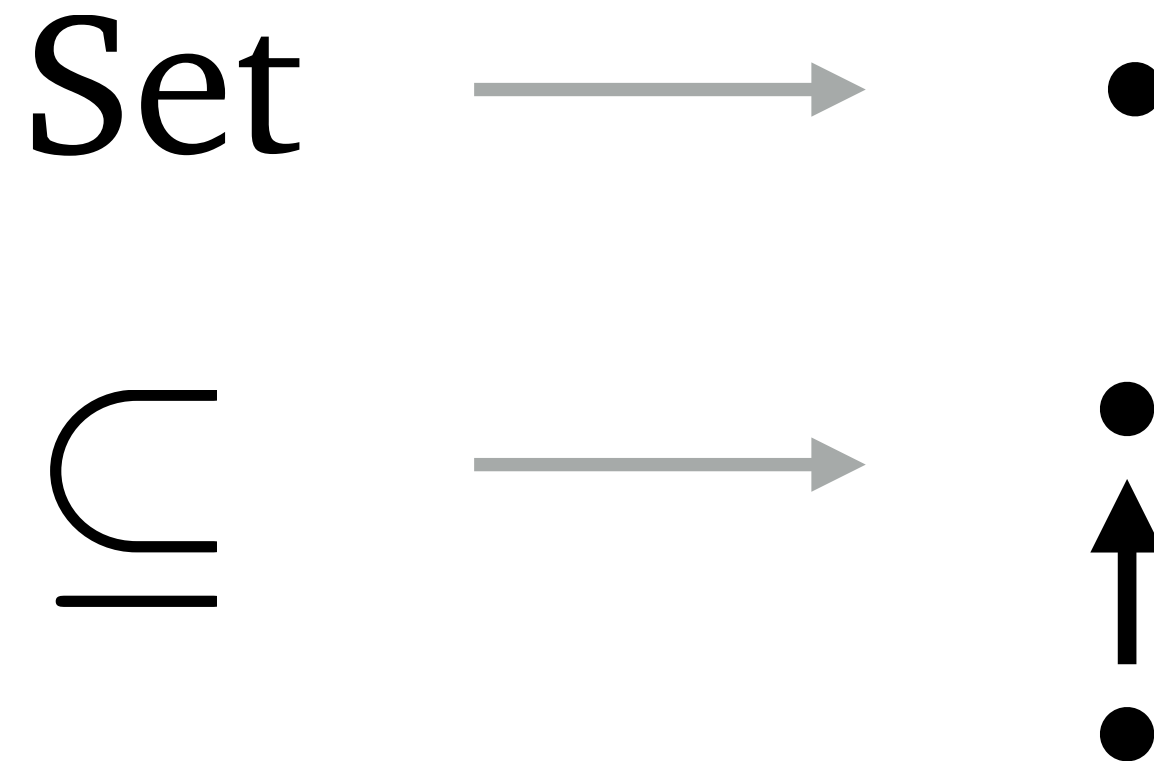
Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

Set  $\longrightarrow$   $\bullet$

$\subseteq$   $\longrightarrow$

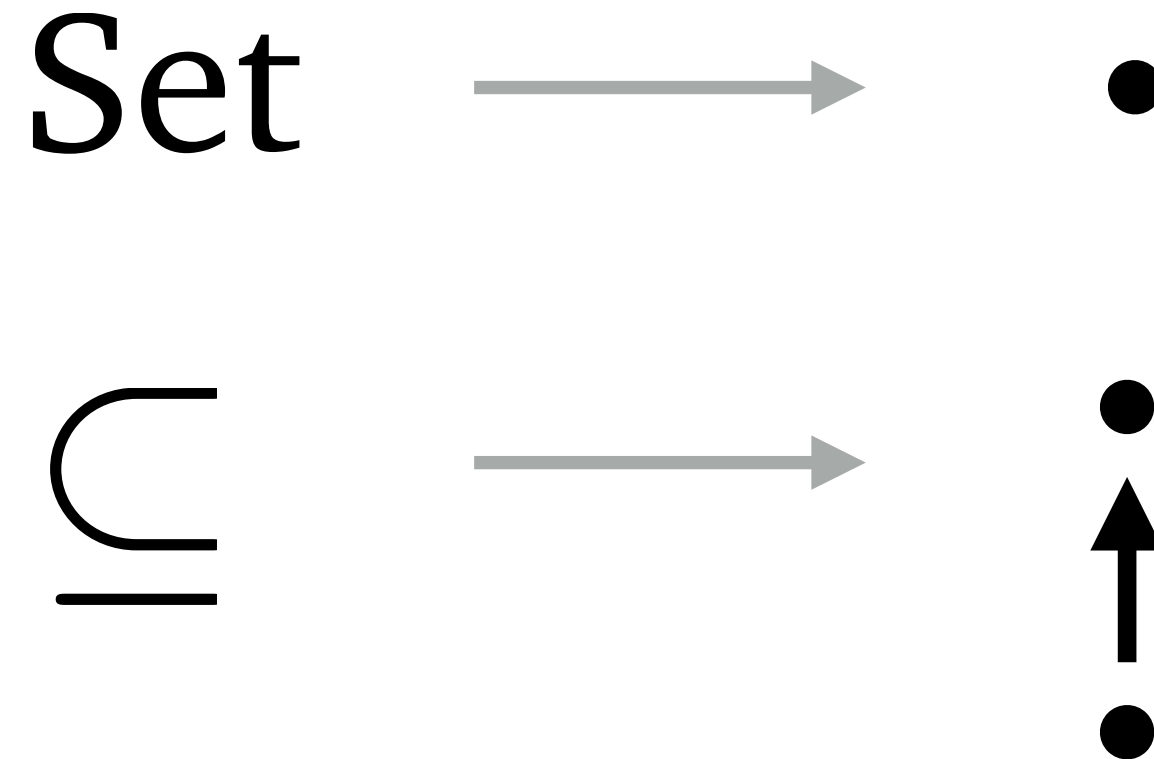
## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .



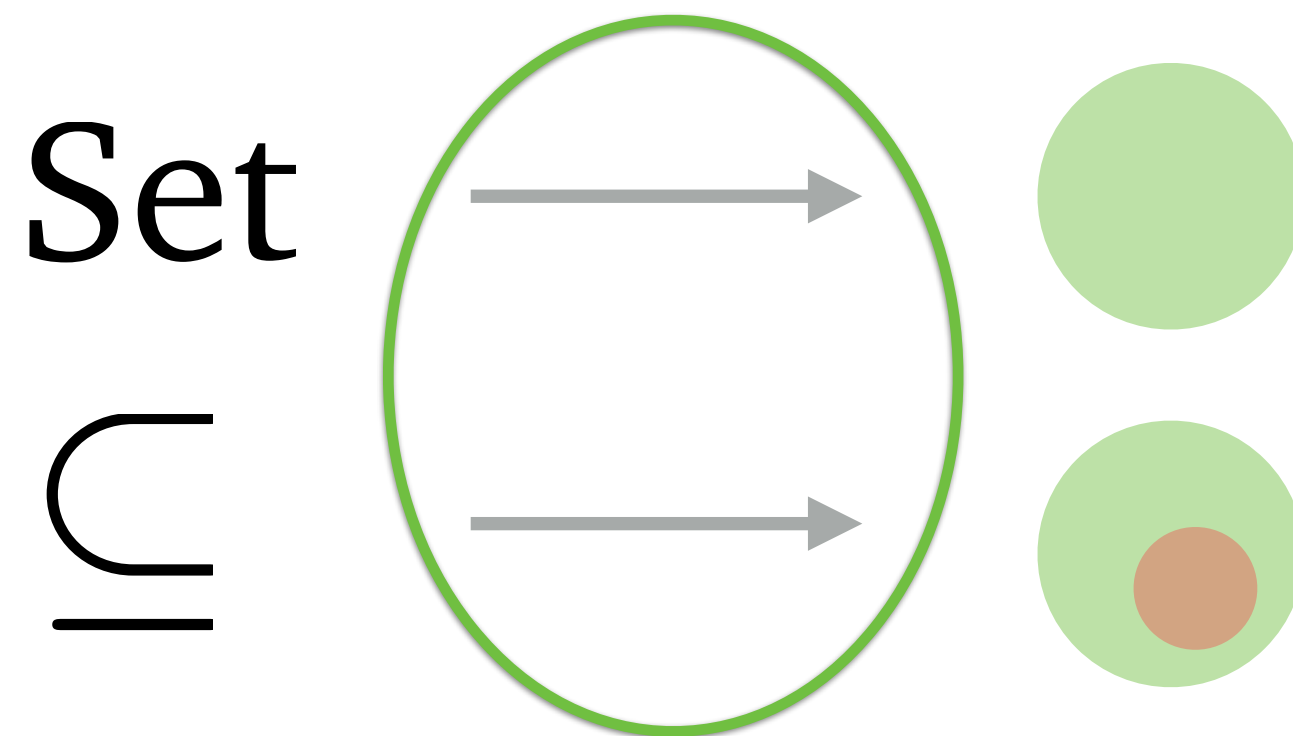
## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .



It is powerful to formally encode the **mapping** from abstract objects and relationships to their visual representations.

How did we encode this  
diagramming knowledge in a *tool*?





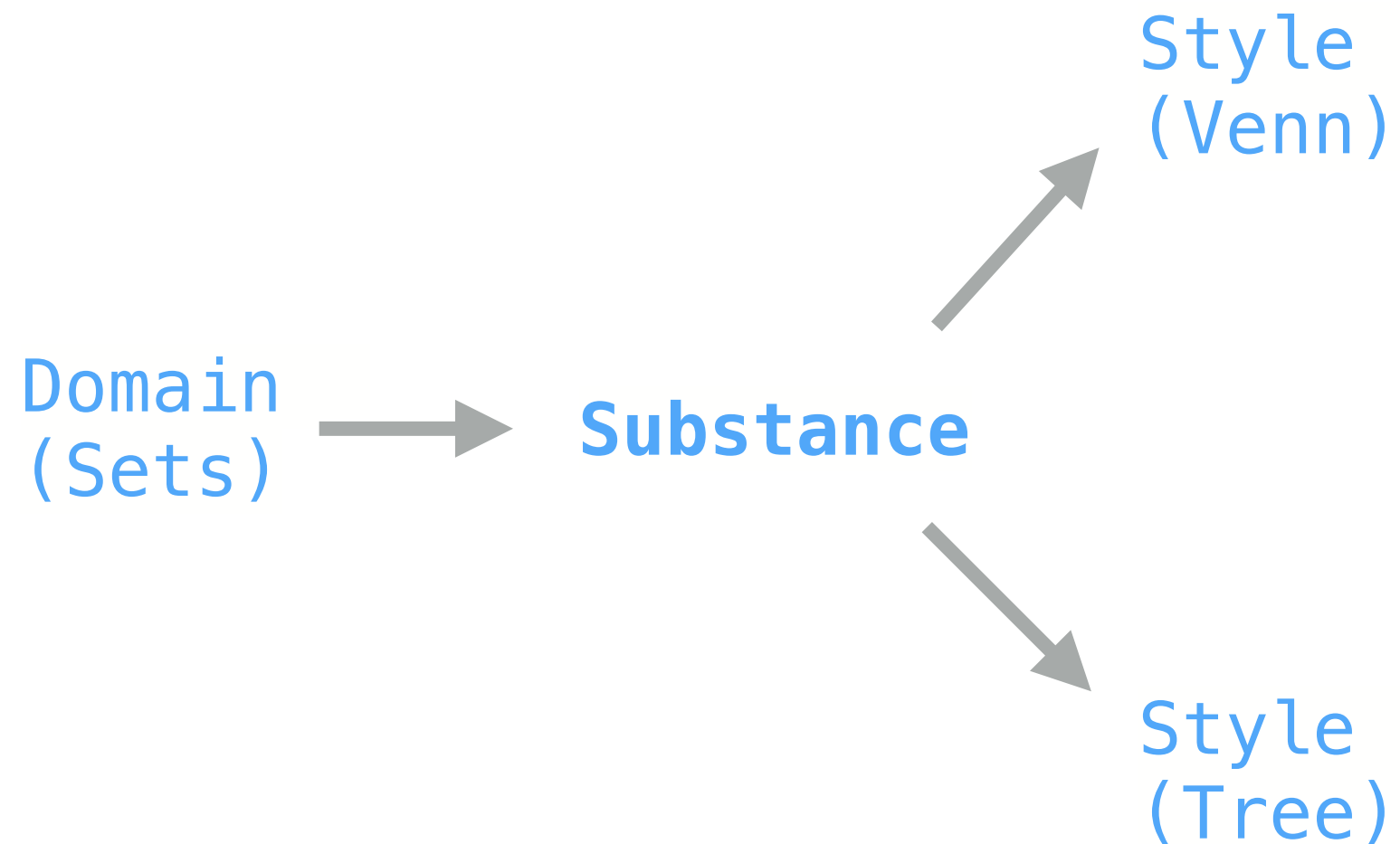
How did we encode this diagramming knowledge in a *tool*?

Domain  
(Sets)

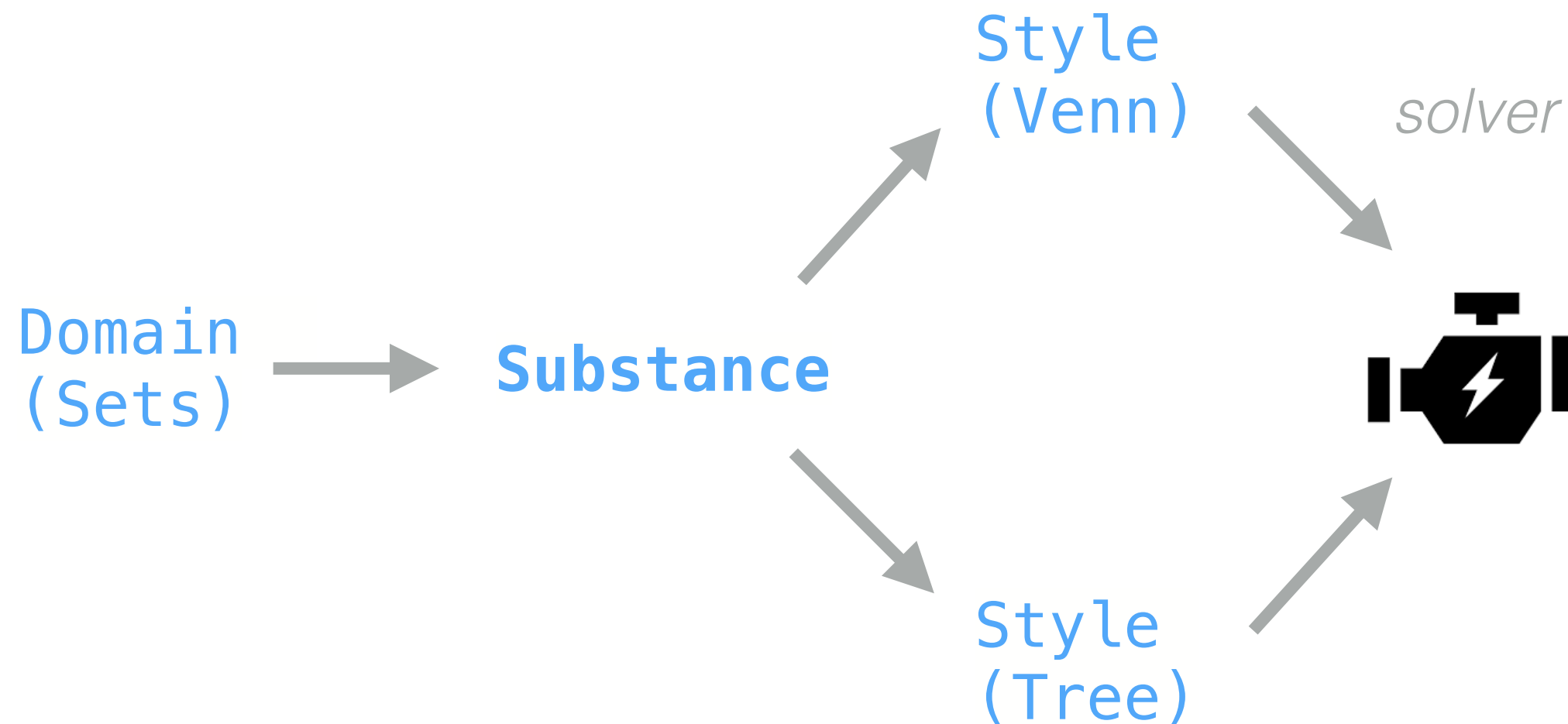
How did we encode this diagramming knowledge in a *tool*?

Domain  
(Sets) → Substance

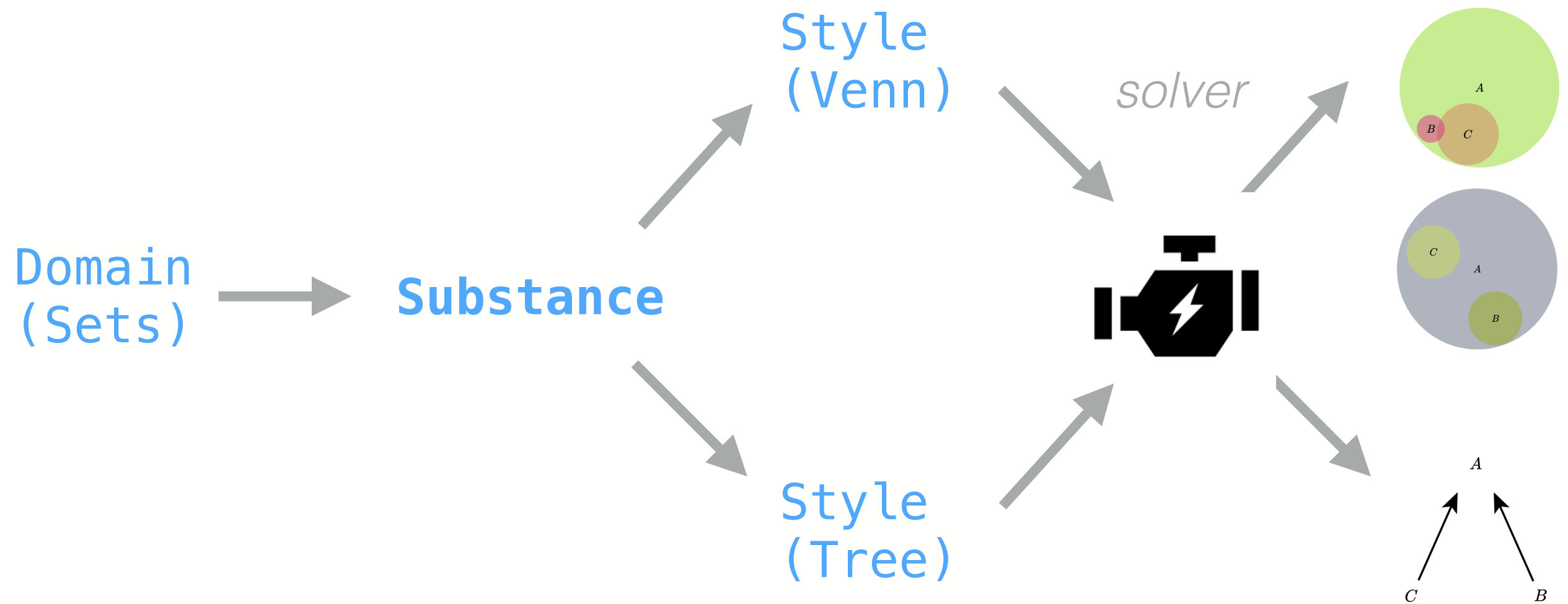
How did we encode this diagramming knowledge in a *tool*?



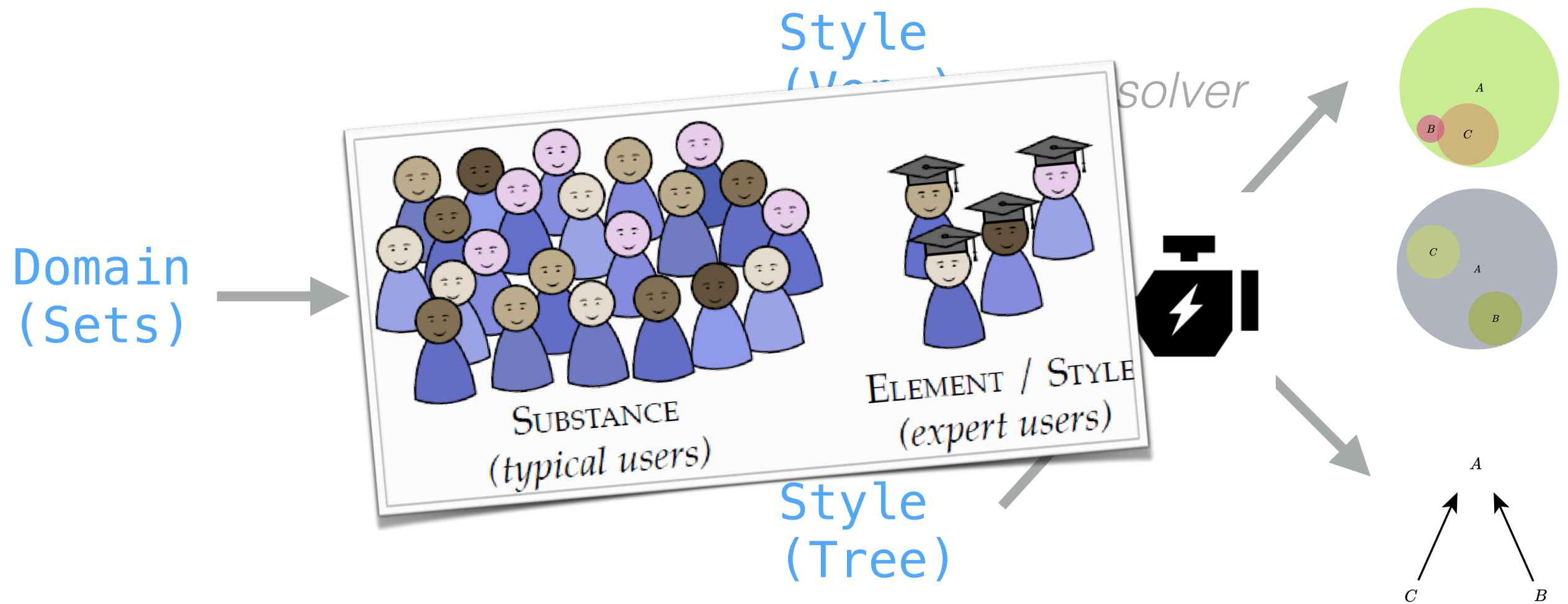
How did we encode this diagramming knowledge in a *tool*?

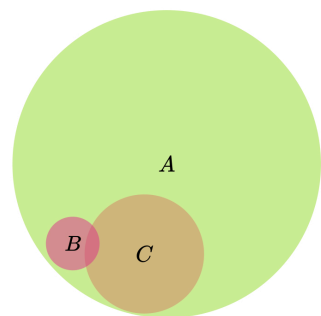


How did we encode this diagramming knowledge in a *tool*?



How did we encode this diagramming knowledge in a *tool*?

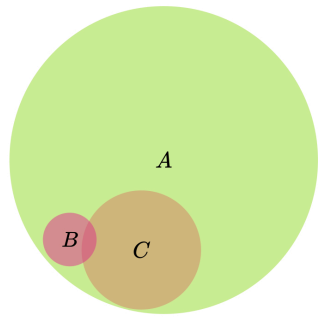




**Penrose**



*L***A***T*<sub>E</sub>*X*



**Penrose**



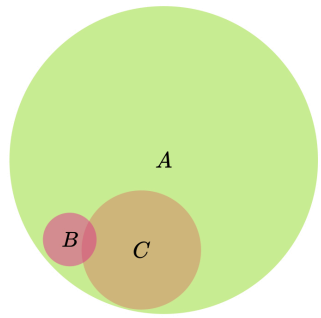
*L<sup>A</sup>T<sub>E</sub>X*

Domain



(no analogy)





**Penrose**



*LaTeX*

Domain

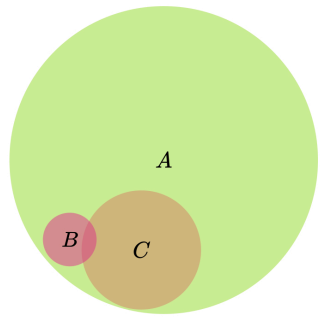


(no analogy)

Substance



TeX file



**Penrose**



*L***A***T***E***X*

Domain



(no analogy)

Substance

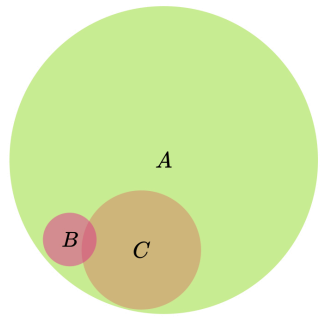


TeX file

Style



TeX style



**Penrose**



*LaTeX*

Domain



(no analogy)

Substance



TeX file

Style

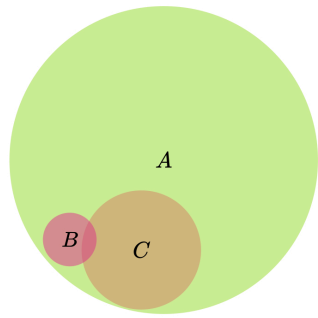


TeX style

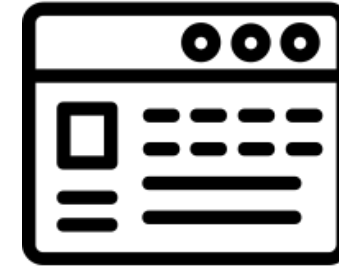
solver



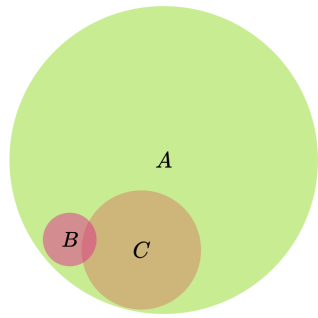
TeX layout engine



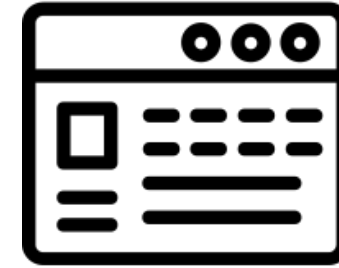
**Penrose**



**Web  
design**



**Penrose**

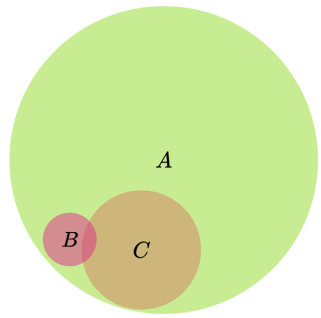


**Web  
design**

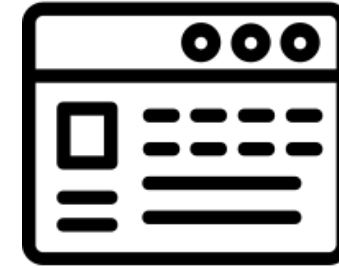
Domain



(no analogy)



**Penrose**



**Web  
design**

Domain

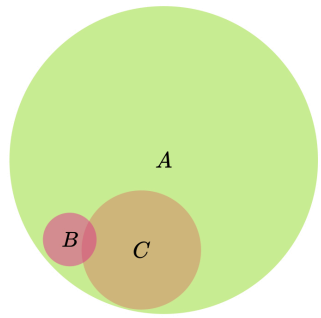


(no analogy)

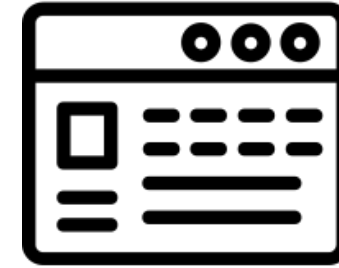
Substance



HTML



**Penrose**



**Web  
design**

Domain



(no analogy)

Substance

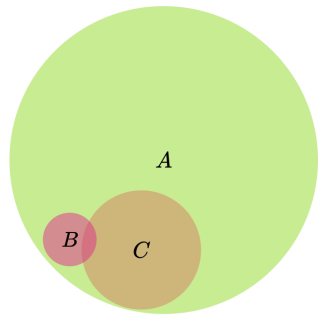


HTML

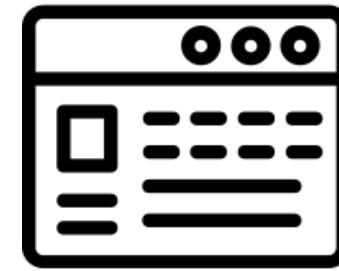
Style



CSS



**Penrose**



**Web  
design**

Domain



(no analogy)

Substance



HTML

Style



CSS

solver



browser layout  
engine



A deeper dive  
with another example

# The Domain language

# The Domain language

**type** VectorSpace

# The Domain language

```
type VectorSpace
```

```
type Vector
```

# The Domain language

```
type VectorSpace
```

```
type Vector
```

# The Domain language

**type** VectorSpace

**type** Vector

**predicate** In : Vector v \* VectorSpace V

# The Domain language

**type** VectorSpace

**type** Vector

**predicate** In : Vector v \* VectorSpace V

**function** addV : Vector \* Vector -> Vector

# The Domain language

**type** VectorSpace

**type** Vector

**predicate** In : Vector v \* VectorSpace V

**function** addV : Vector \* Vector -> Vector



# The Domain language

**type** VectorSpace

**type** Vector

**predicate** In : Vector v \* VectorSpace V

**function** addV : Vector \* Vector -> Vector

**notation** "Vector a  $\in$  U" ~ "Vector a; In(a,U)"

# The Domain language

**type** VectorSpace

**type** Vector

**predicate** In : Vector v \* VectorSpace V

**function** addV : Vector \* Vector -> Vector

**notation** "Vector a  $\in$  U" ~ "Vector a; In(a,U)"

**notation** "v1 + v2" ~ "addV(v1,v2)"

Expressing relationships in a domain  
(Adding three vectors)

Expressing relationships in a domain  
(Adding three vectors)

**VectorSpace U**

Expressing relationships in a domain  
(Adding three vectors)

**VectorSpace**  $U$

**Vector**  $u_1, u_2, u_3, u_4, u_5 \in U$

Expressing relationships in a domain  
(Adding three vectors)

**VectorSpace** U

**Vector** u1, u2, u3, u4, u5 ∈ U

u3 := u1 + u2

Expressing relationships in a domain  
(Adding three vectors)

**VectorSpace** U

**Vector** u1, u2, u3, u4, u5  $\in$  U

u3 := u1 + u2

u5 := u3 + u4

Expressing relationships in a domain  
(Adding three vectors)

*notation for In*

**VectorSpace** U

**Vector** u1, u2, u3, u4, u5  **∈** U

u3 := u1 + u2

u5 := u3 + u4



Expressing relationships in a domain  
(Adding three vectors)

*notation for In*

**VectorSpace** U

**Vector** u1, u2, u3, u4, u5  $\in$  U

u3 := u1 + u2

u5 := u3 + u4

*notation for AddV*

Expressing relationships in a domain  
(Adding three vectors)

**VectorSpace**

**Vector**

$u_3$  :

$u_5$  :=

*notation for  $\in$*

$u_5 \in U$

**No coordinates  
needed!**

*nota*

# Encoding a visual representation of vector addition

Encoding a visual representation  
of vector addition

**VectorSpace U**

Encoding a visual representation  
of vector addition

**VectorSpace**  $U$

**Vector**  $u_1, u_2, u_3, u_4, u_5 \in U$

# Encoding a visual representation of vector addition

**VectorSpace** U

**Vector**  $u_1, u_2, u_3, u_4, u_5 \in U$

$u_3 := u_1 + u_2$

# Encoding a visual representation of vector addition

**VectorSpace** U

**Vector**  $u_1, u_2, u_3, u_4, u_5 \in U$

$u_3 := u_1 + u_2$

$u_5 := u_3 + u_4$

# Encoding a mapping in the Style language



# Encoding a mapping in the Style language

For every vector  
in a vector space,

# Encoding a mapping in the Style language

For every vector  
in a vector space,

**Vector V**  
**with VectorSpace U**  
**where  $v \in U$  {**

**}**

# Encoding a mapping in the Style language

For every vector  
in a vector space,

**Vector  $V$**   
**with `VectorSpace`  $U$**   
**where  $v \in U$  {**

draw it as a little arrow  
rooted at the origin

**}**

# Encoding a mapping in the Style language

For every vector  
in a vector space,

draw it as a little arrow  
rooted at the origin

```
Vector V  
with VectorSpace U  
where  $v \in U$  {  
     $v.shape = \mathbf{Arrow}$  {  
         $start = U.shape.center$   
    }  
}
```

# Encoding a mapping in the Style language

For every vector  
in a vector space,

draw it as a little arrow  
rooted at the origin

place its label  
near the arrowhead

```
Vector V  
with VectorSpace U  
where  $v \in U$  {  
     $v.shape = \mathbf{Arrow}$  {  
         $start = U.shape.center$   
    }  
  
}
```

# Encoding a mapping in the Style language

For every vector  
in a vector space,

draw it as a little arrow  
rooted at the origin

place its label  
near the arrowhead

```

Vector V
with VectorSpace U
where  $v \in U$  {

    v.shape = Arrow {
        start = U.shape.center
    }

    encourage nearHead(v.shape,
                        v.text)

}

```

# Encoding a mapping in the Style language

For every vector  
in a vector space,

draw it as a little arrow  
rooted at the origin

place its label  
near the arrowhead

make sure the vector's shape  
is in the vector space's shape

```

Vector V
with VectorSpace U
where  $v \in U$  {

    v.shape = Arrow {
        start = U.shape.center
    }

    encourage nearHead(v.shape,
                        v.text)

}

```

# Encoding a mapping in the Style language

For every vector  
in a vector space,

draw it as a little arrow  
rooted at the origin

place its label  
near the arrowhead

make sure the vector's shape  
is in the vector space's shape

```

Vector V
with VectorSpace U
where  $v \in U$  {
    v.shape = Arrow {
        start = U.shape.center
    }

    encourage nearHead(v.shape,
                        v.text)

    ensure contains(U.shape,
                    v.shape)

}

```



# Encoding a mapping in the Style language

# Encoding a mapping in the Style language

For every vector  
that's the sum of vectors,

# Encoding a mapping in the Style language

For every vector  
that's the sum of vectors,

```

Vector u
with Vector v, w; VectorSpace U
where u := v + w; u, v, w ∈ U {

```

# Encoding a mapping in the Style language

For every vector  
that's the sum of vectors,

**Vector** u  
**with** **Vector** v, w; **VectorSpace** U  
**where**  $u := v + w$ ;  $u, v, w \in U$  {

draw the end of the arrowhead  
as the vector sum

}

# Encoding a mapping in the Style language

For every vector  
that's the sum of vectors,

draw the end of the arrowhead  
as the vector sum

```
Vector u  
with Vector v, w; VectorSpace U  
where  $u := v + w$ ;  $u, v, w \in U$  {
```

```
u.shape.end = v.shape.end +  
              w.shape.end
```

```
}
```

# Encoding a mapping in the Style language

For every vector  
that's the sum of vectors,

draw the end of the arrowhead  
as the vector sum

using the “tip-to-tail”  
mnemonic for the vectors  
being summed

```
Vector u  
with Vector v, w; VectorSpace U  
where u := v + w; u, v, w ∈ U {
```

```
u.shape.end = v.shape.end +  
w.shape.end
```

```
}
```

# Encoding a mapping in the Style language

For every vector  
that's the sum of vectors,

draw the end of the arrowhead  
as the vector sum

using the “tip-to-tail”  
mnemonic for the vectors  
being summed

```
Vector u  
with Vector v, w; VectorSpace U  
where u := v + w; u, v, w ∈ U {
```

```
u.shape.end = v.shape.end +  
w.shape.end
```

```
u.v_shadow = Arrow {  
  start = w.shape.end  
  end   = u.shape.end  
  style = “dashed”  
}
```

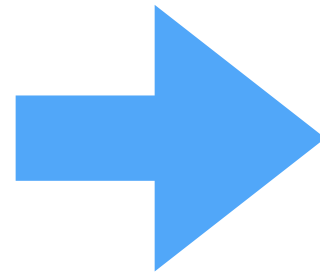
```
u.w_shadow = Arrow { ... }
```

```
}
```

# Visualizing the notation

Domain program

Substance  
program



**VectorSpace** U

**Vector** u1, u2, u3, u4, u5 ∈ U

u3 := u1 + u2

u5 := u3 + u4

Style program



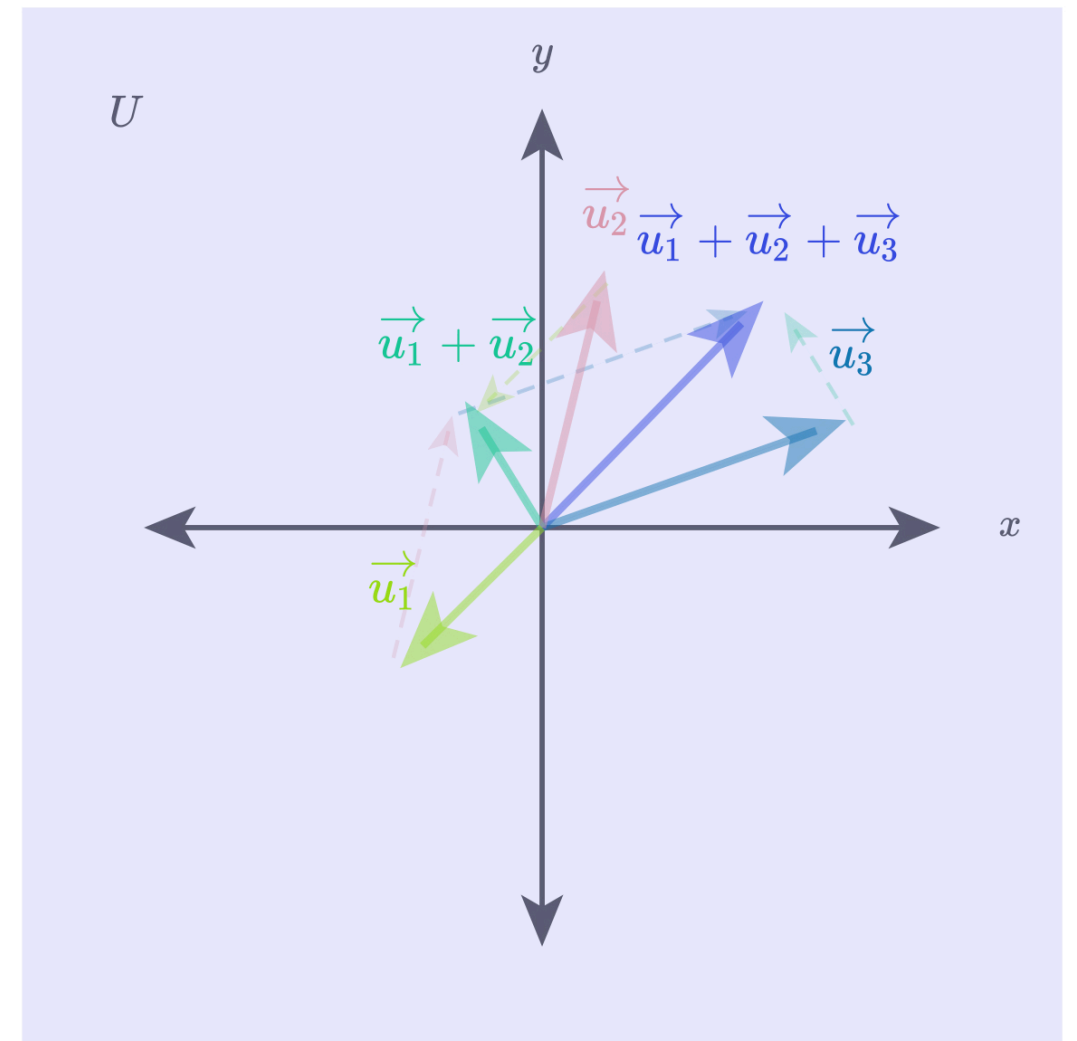
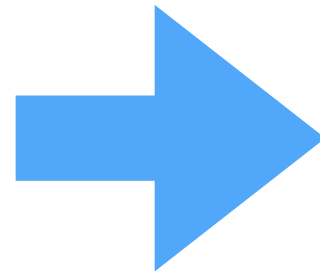
# Visualizing the notation

Domain program

Substance  
program

**VectorSpace**  $U$   
**Vector**  $u_1, u_2, u_3, u_4, u_5 \in U$   
 $u_3 := u_1 + u_2$   
 $u_5 := u_3 + u_4$

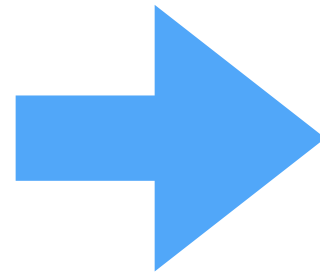
Style program



# Visualizing the notation

Domain program

Substance  
program



**VectorSpace** U

**Vector** u1, u2, u3, u4, u5 ∈ U

u3 := u1 + u2

u5 := u3 + u4

Style program

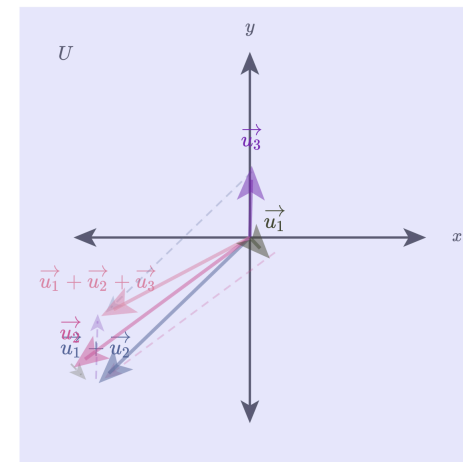
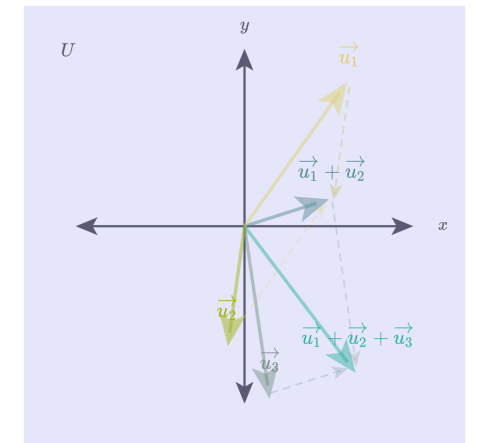
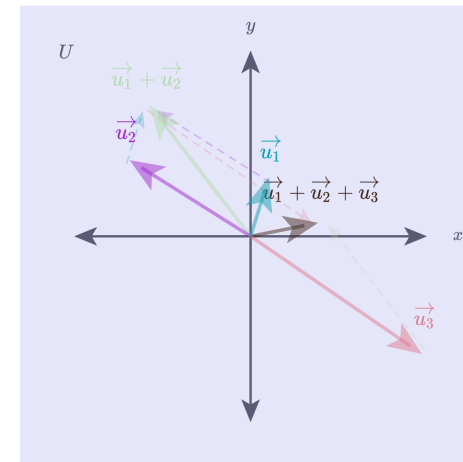
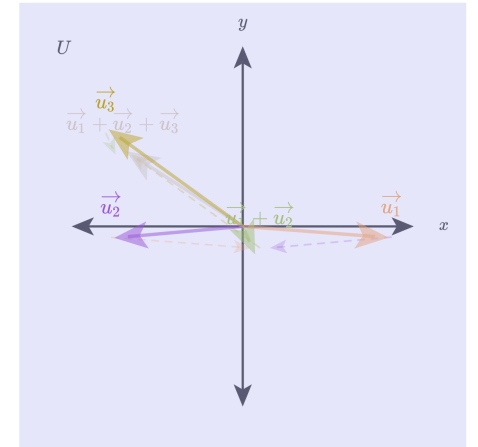
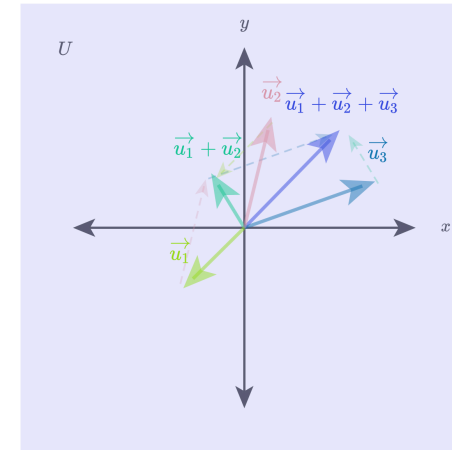
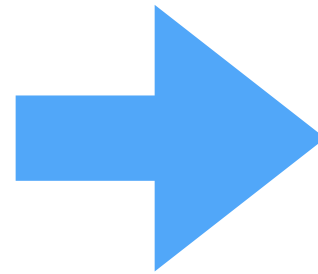
# Visualizing the notation

Domain program

Substance  
program

**VectorSpace** U  
**Vector** u1, u2, u3, u4, u5 ∈ U  
u3 := u1 + u2  
u5 := u3 + u4

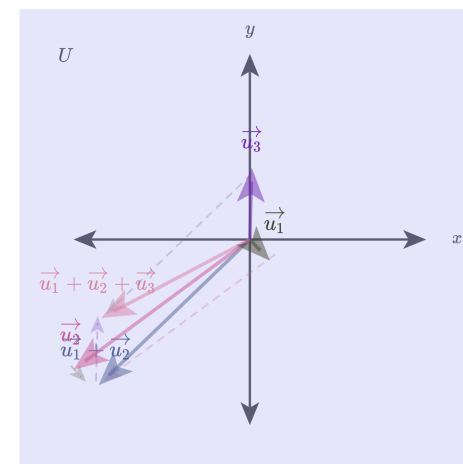
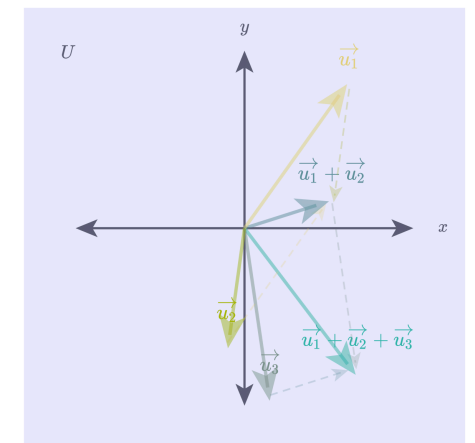
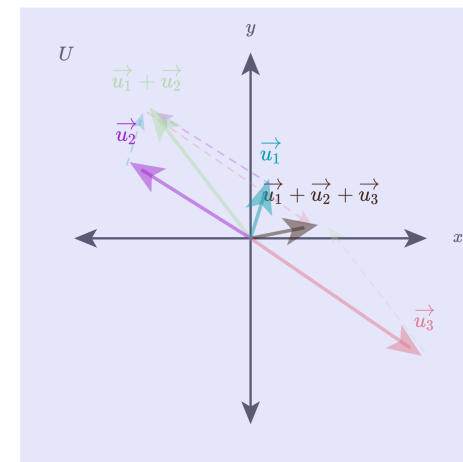
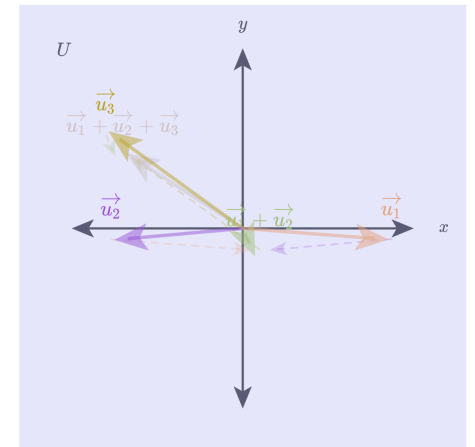
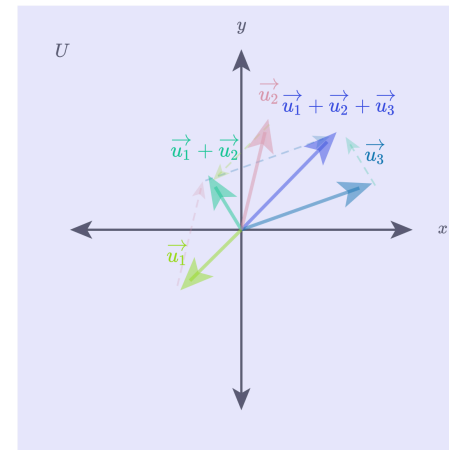
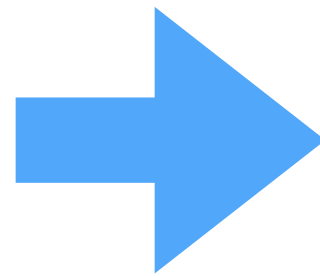
Style program



# Visualizing the notation

## Key idea:

Every diagram is just one of many solutions to an underlying constrained optimization problem!



# Example: Containment

The Style directive

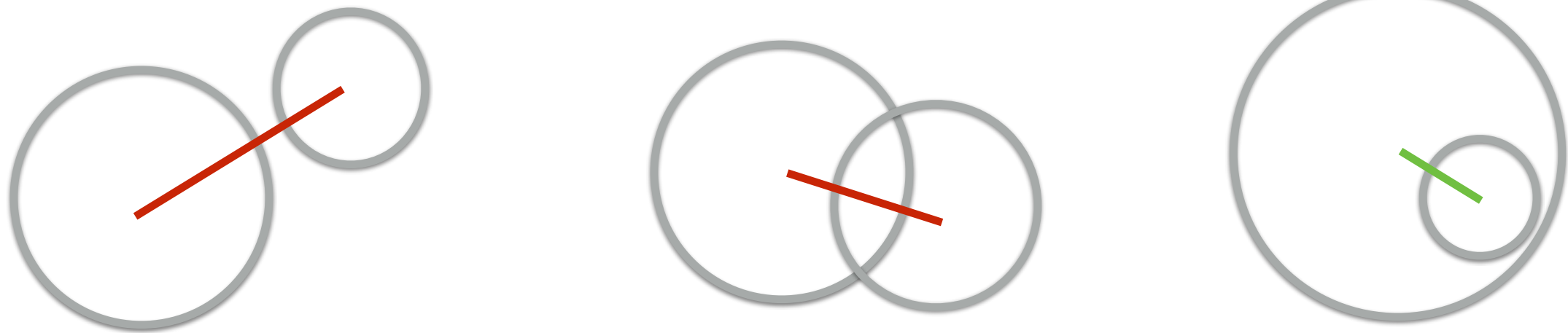
```
ensure Y.shape contains X.shape
```

# Example: Containment

The Style directive

**ensure** Y.shape **contains** X.shape

gets automatically translated into the constraint (for circles)



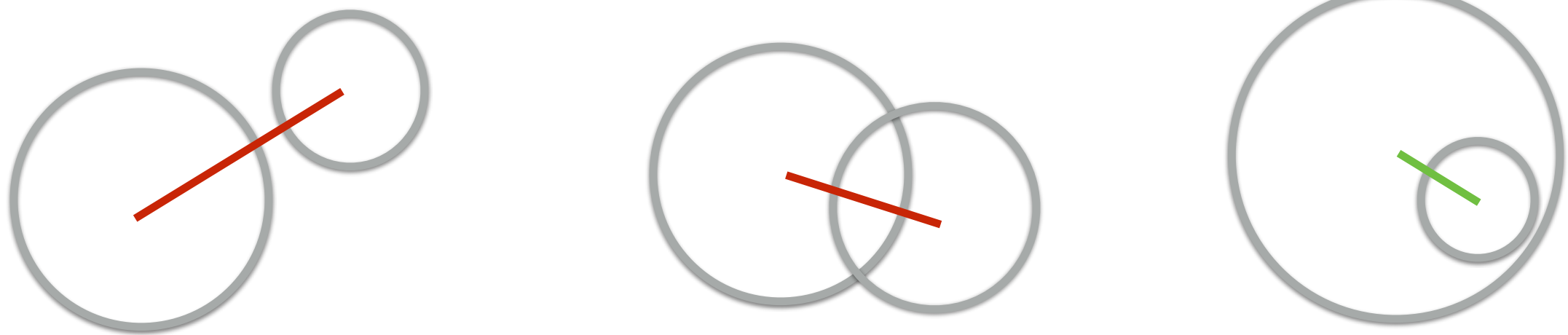
$$|c_Y - c_X| < r_Y - r_X$$

# Example: Containment

The Style directive

**ensure** Y.shape **contains** X.shape

gets automatically translated into the constraint (for circles)

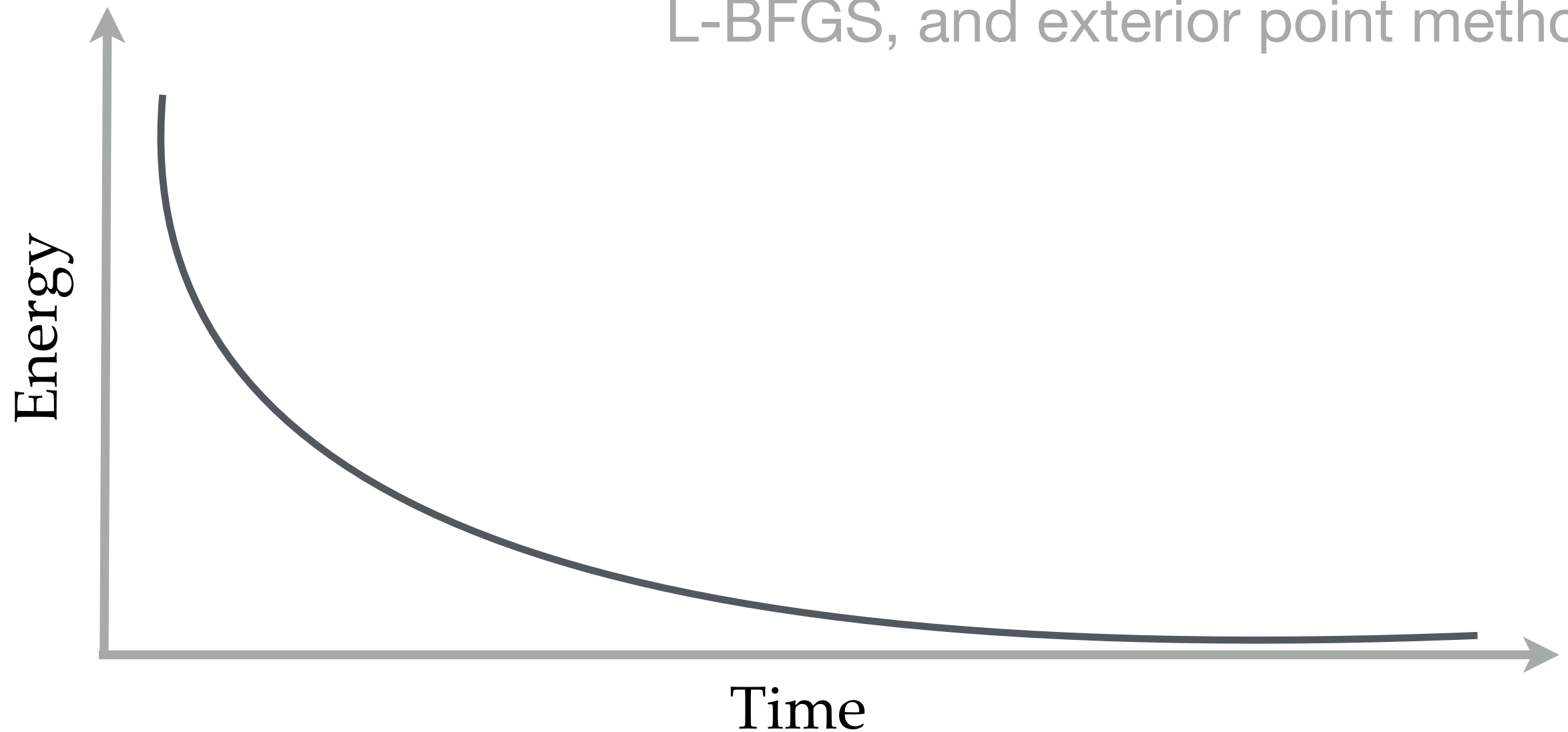


$$|c_Y - c_X| < r_Y - r_X$$

**Key idea:** programmer does not ever have to think this way.

# Automatically laying out a diagram

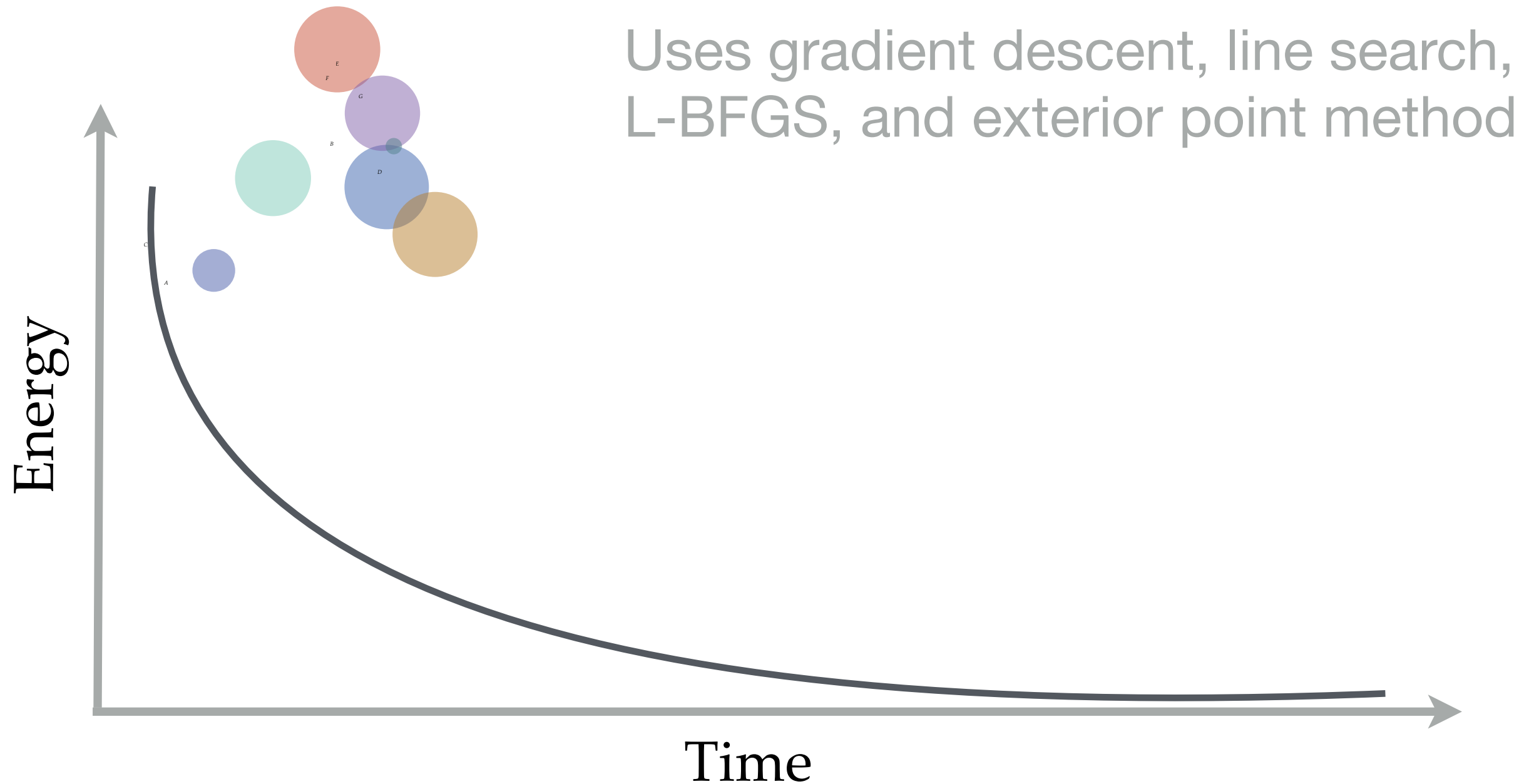
Uses gradient descent, line search, L-BFGS, and exterior point method



*the actual energy landscape is much more complicated!*



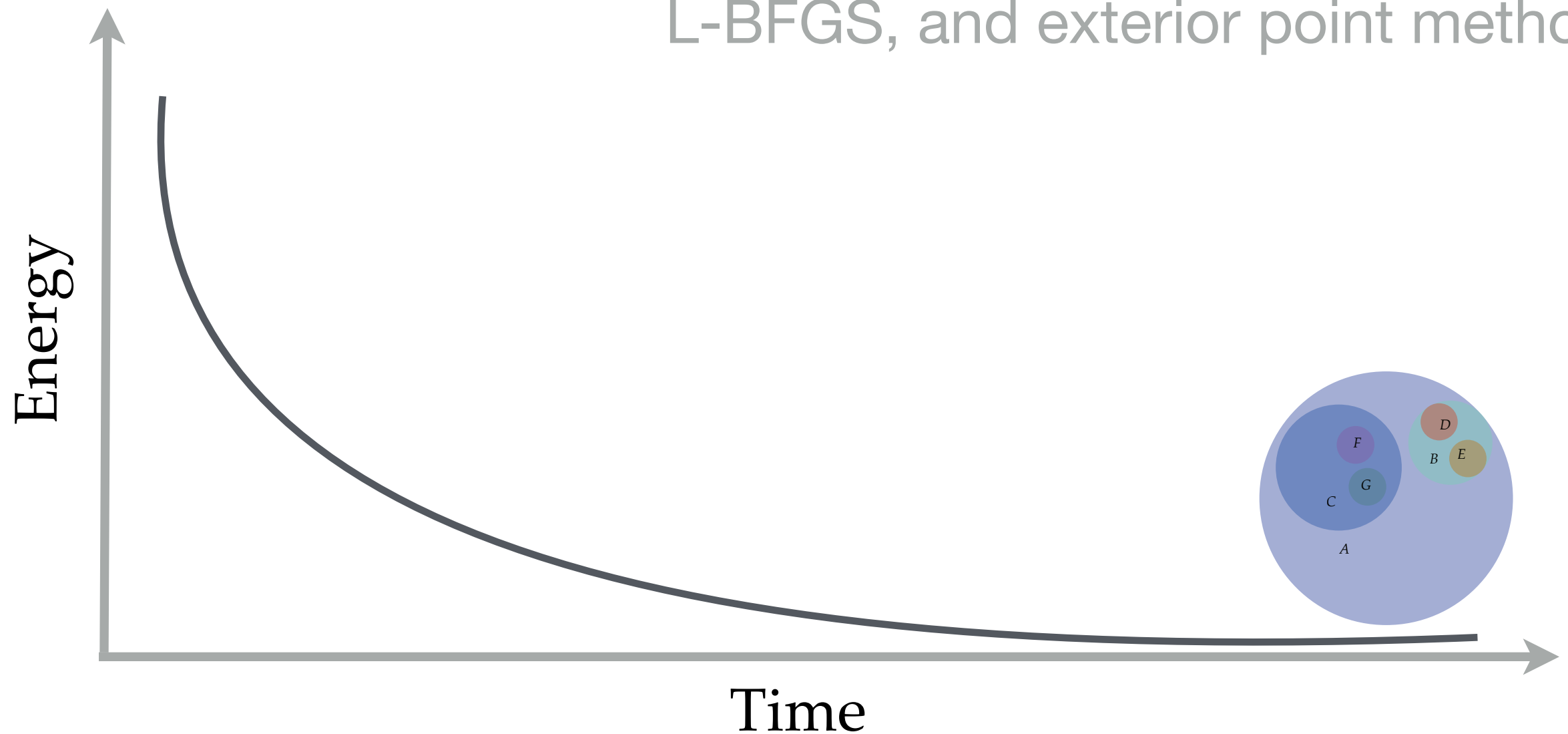
# Automatically laying out a diagram



*the actual energy landscape is much more complicated!*

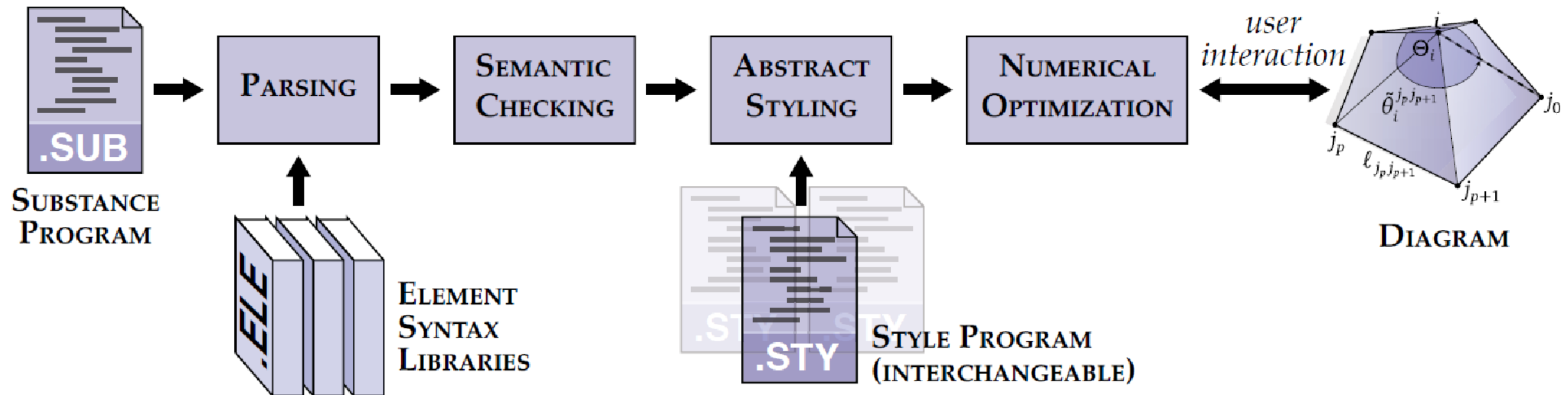
# Automatically laying out a diagram

Uses gradient descent, line search, L-BFGS, and exterior point method



*the actual energy landscape is much more complicated!*

# Opening up the “magical box”



Some implementation details:

*Backend written in Haskell*

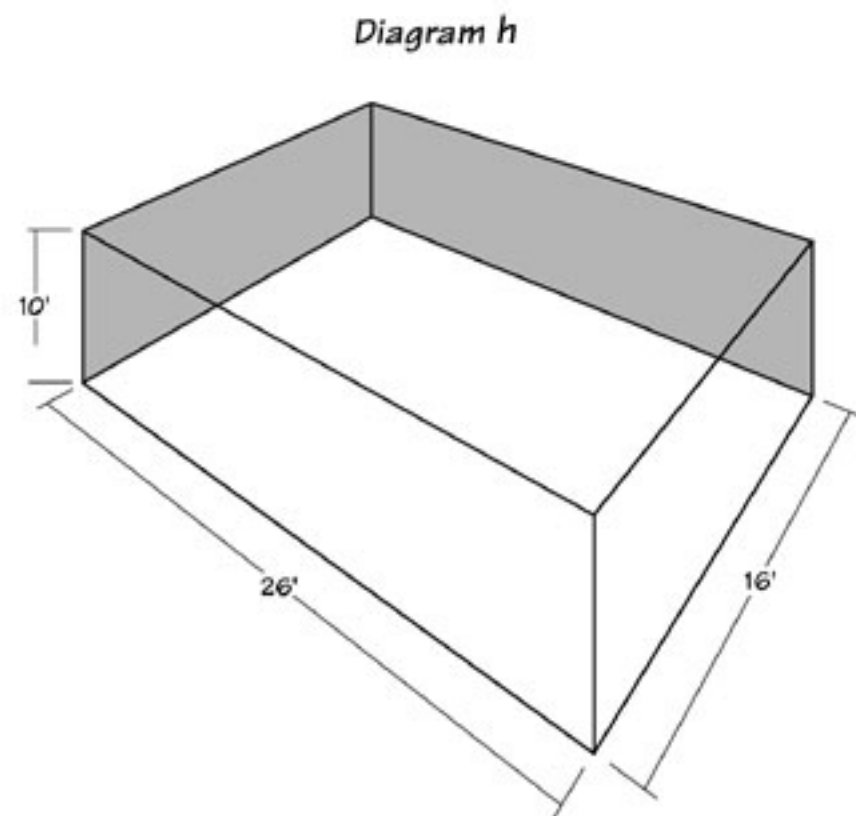
*Frontend written in Typescript + React*

*Outputs diagrams in SVG*

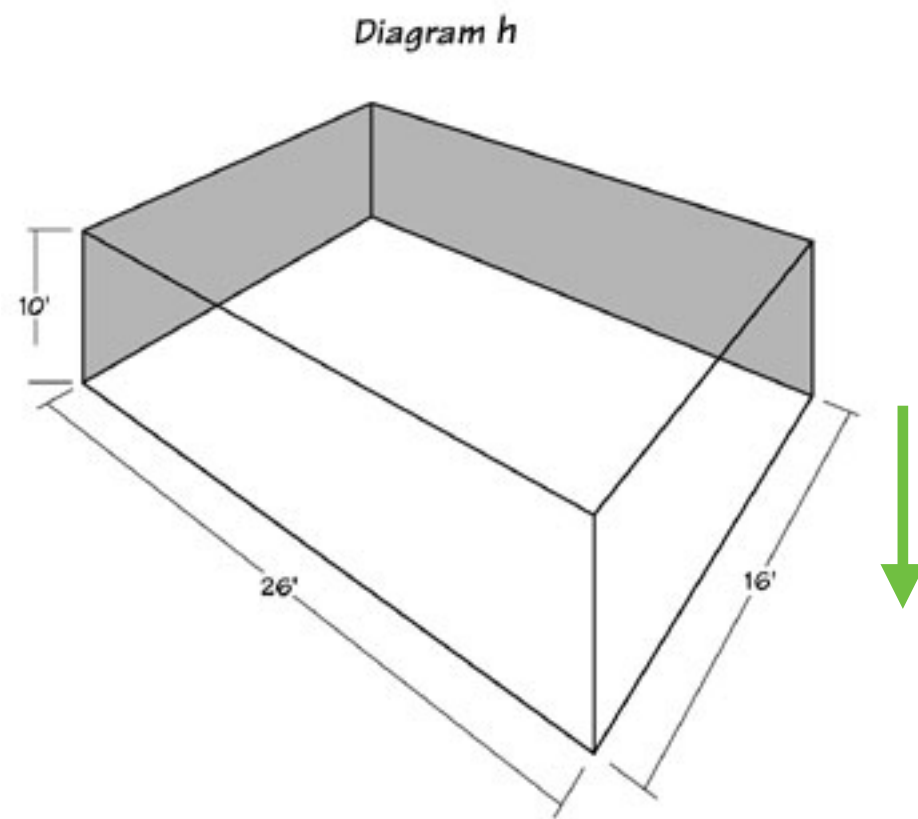
Ask me later if you're really interested!

## **Part IV:** What does language enable?

My goal: lower the floor  
and raise the ceiling for  
making diagrams!

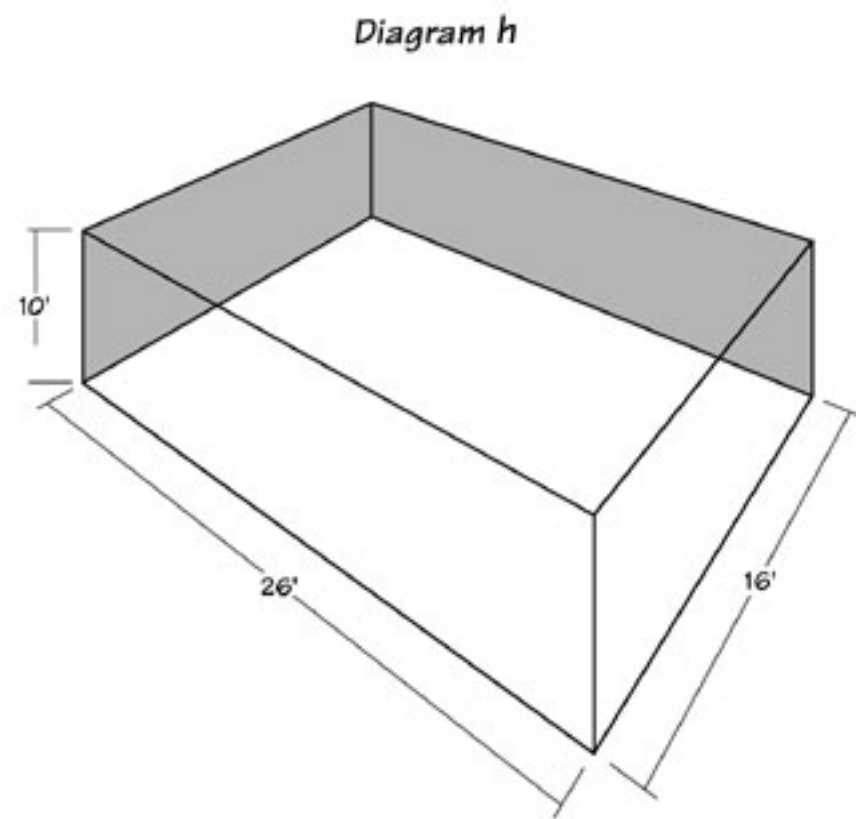


My goal: lower the floor  
and raise the ceiling for  
making diagrams!



reduce the amount of work  
and expertise  
needed to make a diagram

# My goal: lower the floor and raise the ceiling for making diagrams!



empower people to create new kinds of diagrams



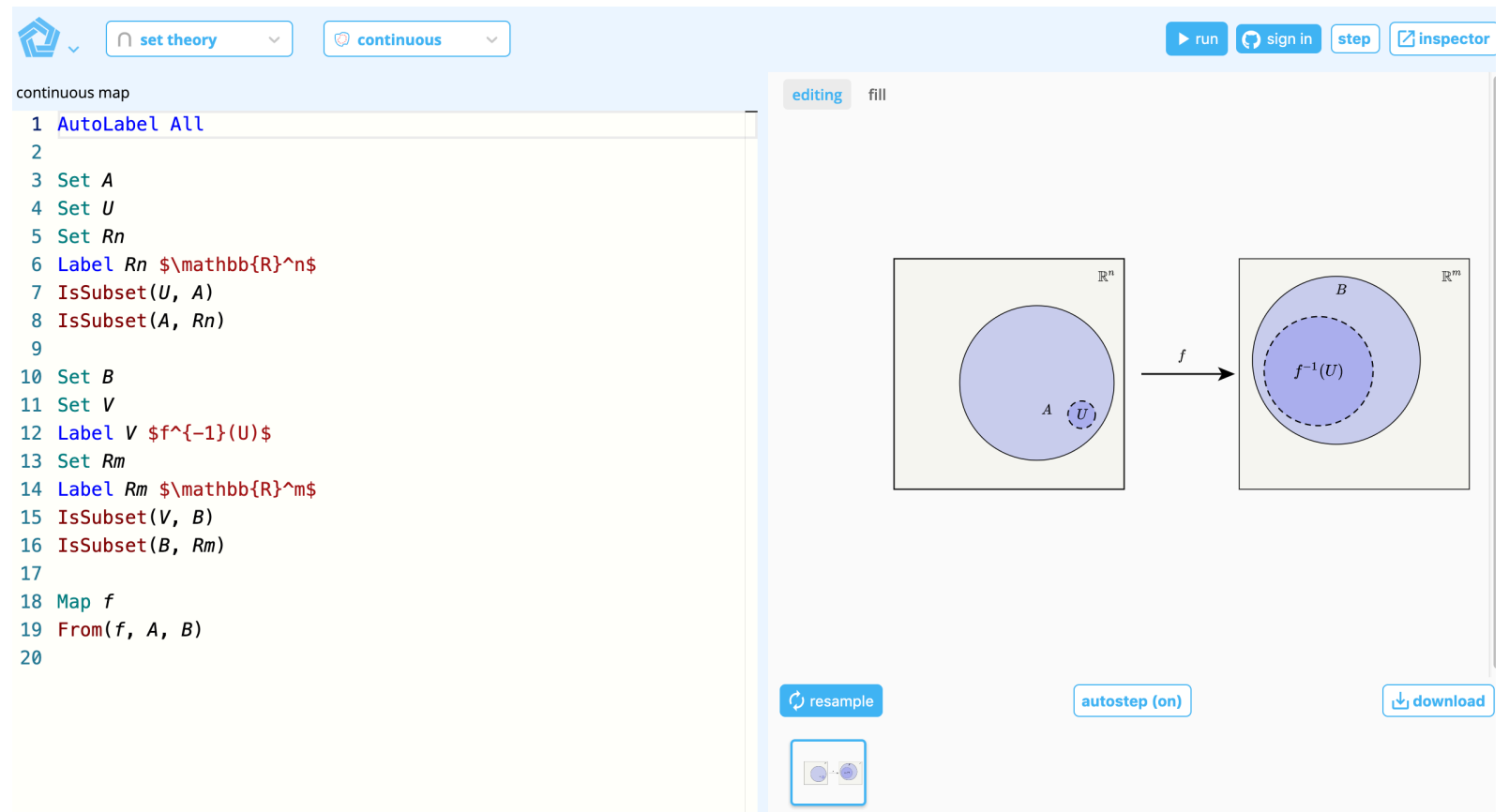
reduce the amount of work and expertise needed to make a diagram

# Some live examples

Sets

Functions

Vectors



[use.penrose.ink](https://use.penrose.ink)  
(ALPHA)



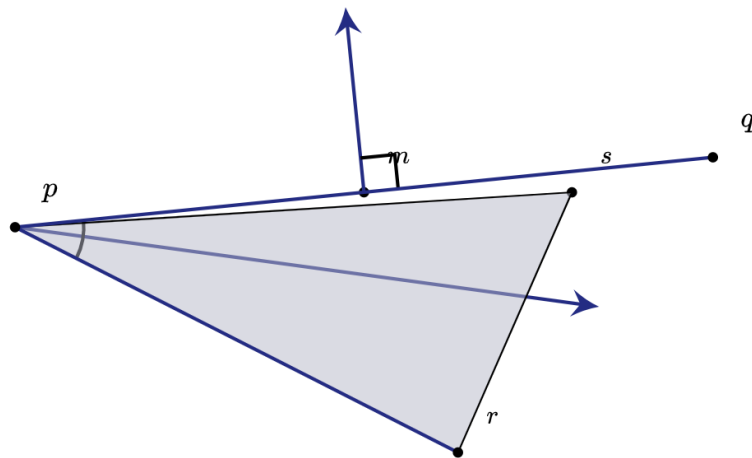
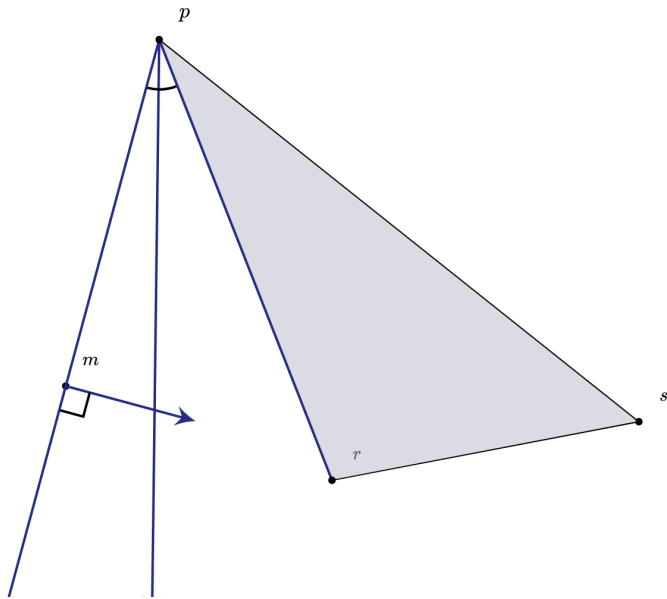
# More examples made in Penrose

A yellow equilateral triangle with a thin black border.

**Work  
in  
progress!**

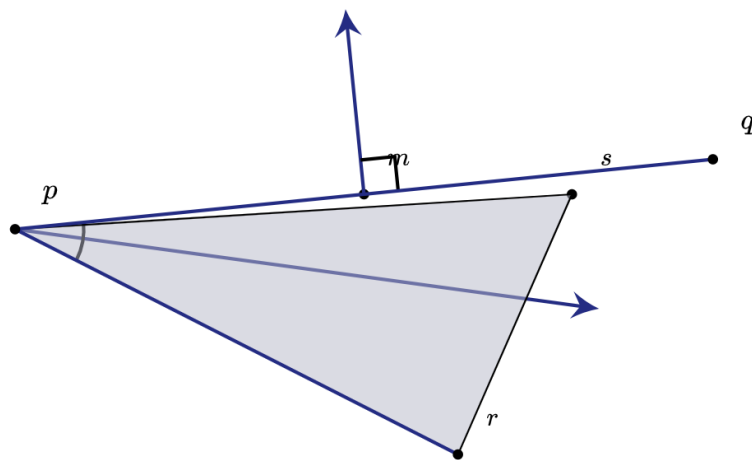
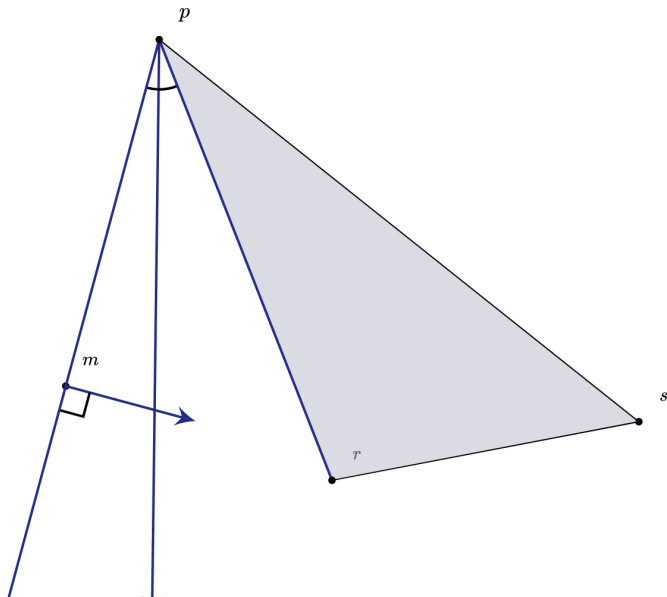
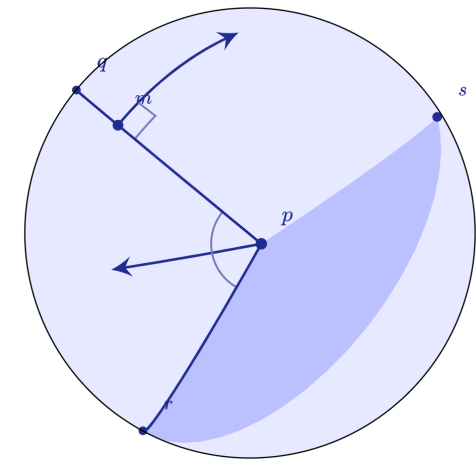
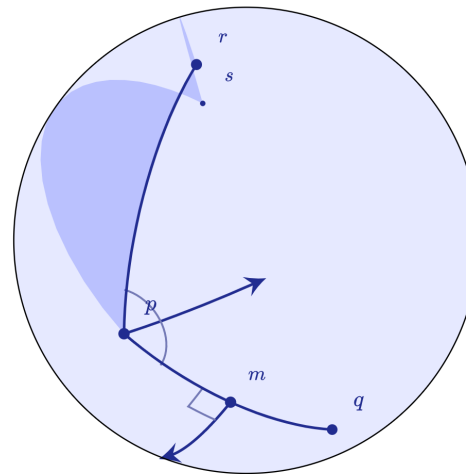
# More examples made in Penrose

**Work  
in  
progress!**



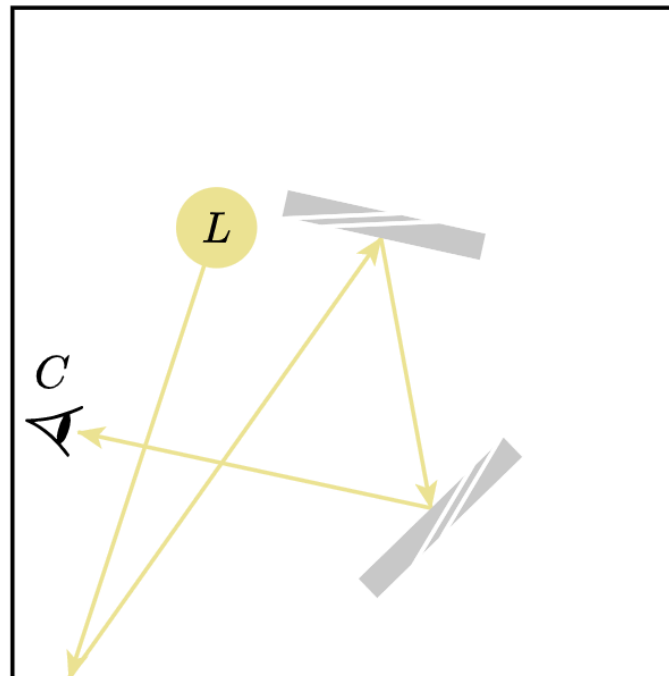
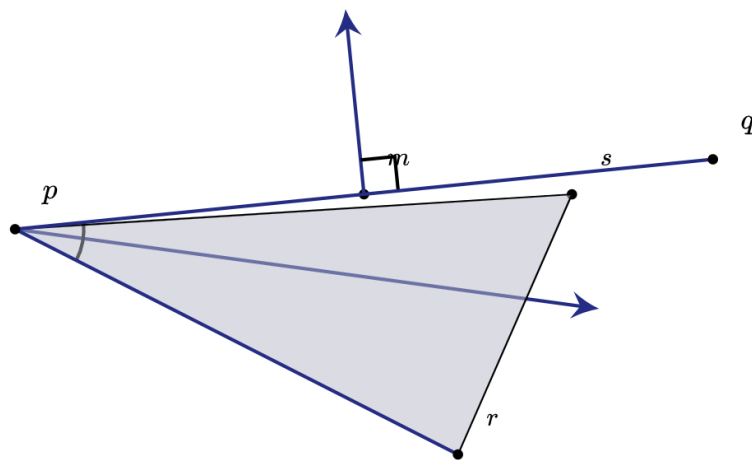
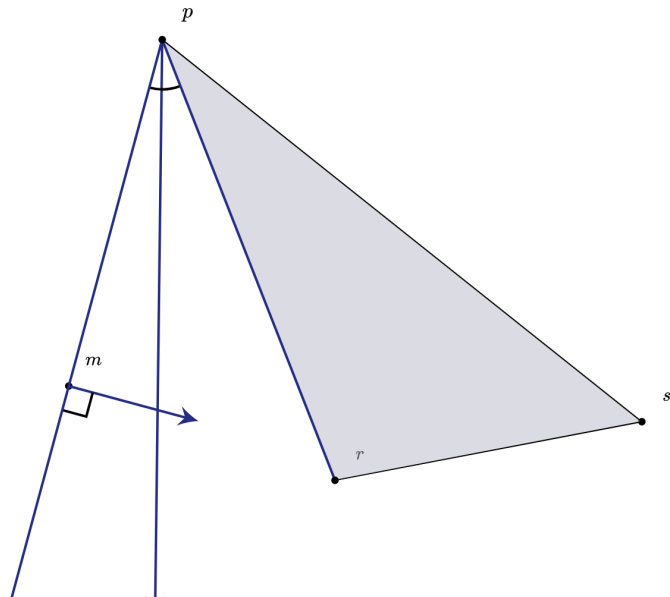
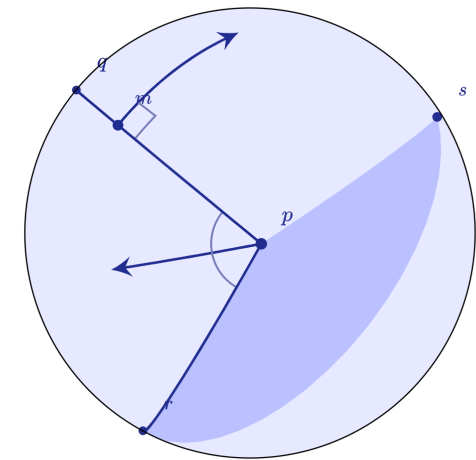
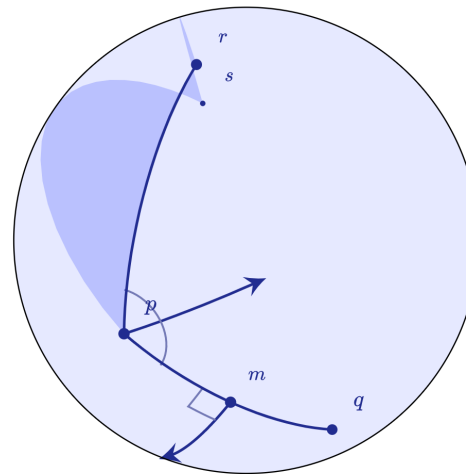
# More examples made in Penrose

**Work  
in  
progress!**



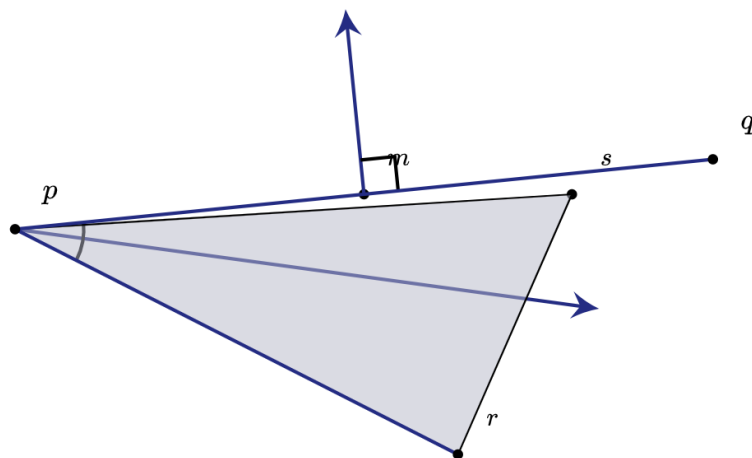
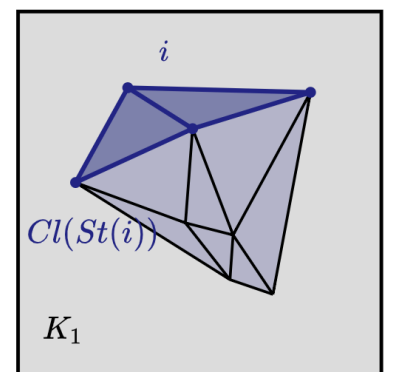
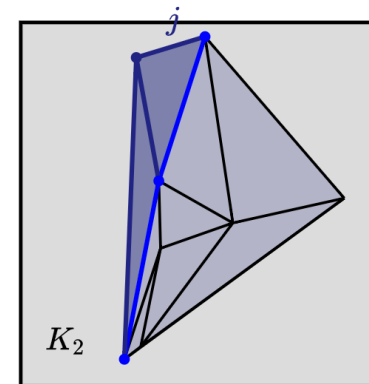
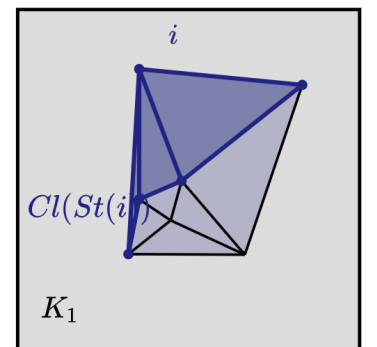
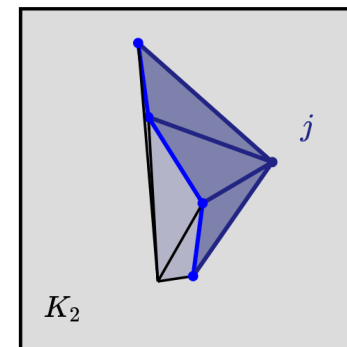
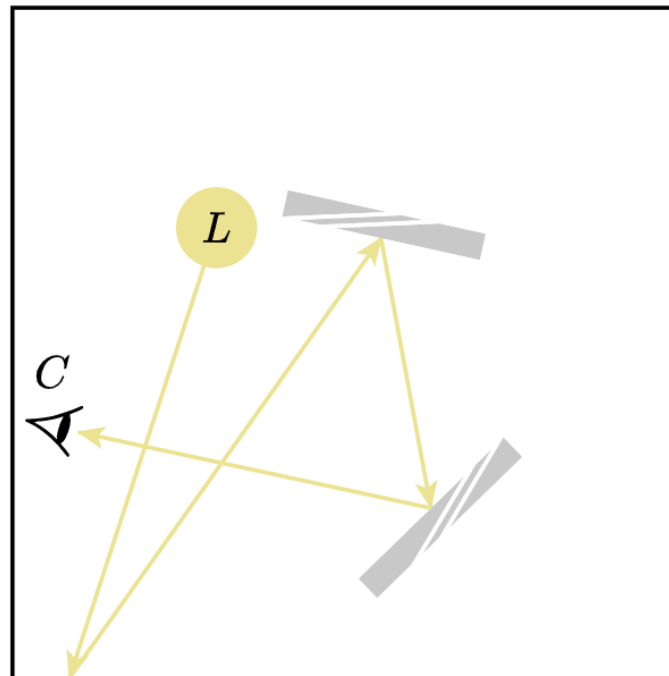
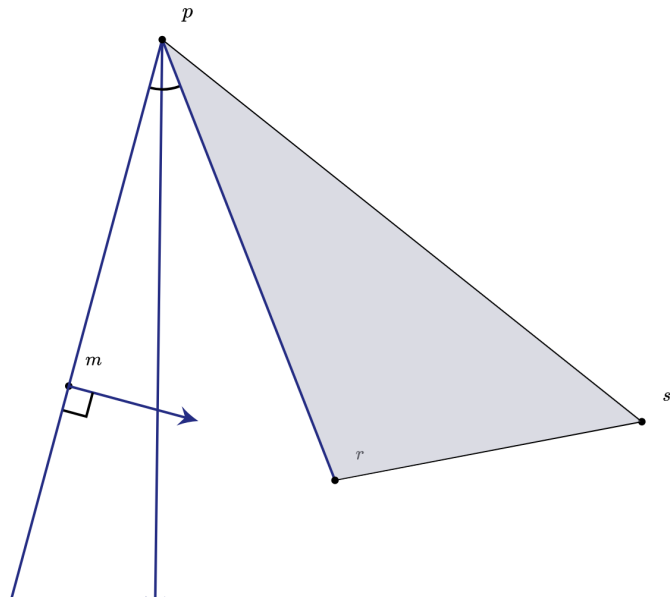
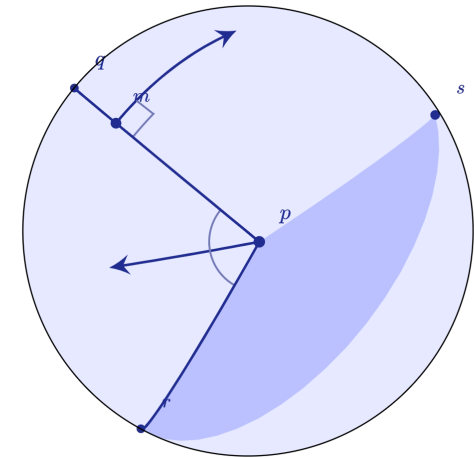
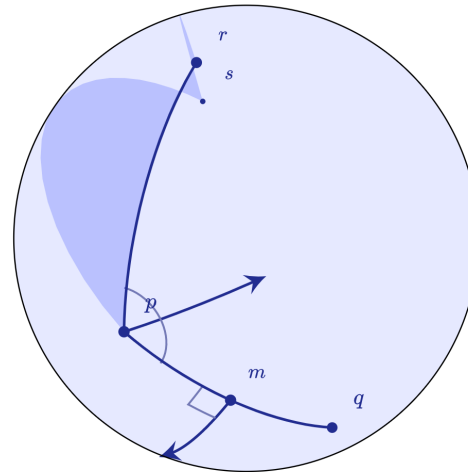
# More examples made in Penrose

**Work  
in  
progress!**



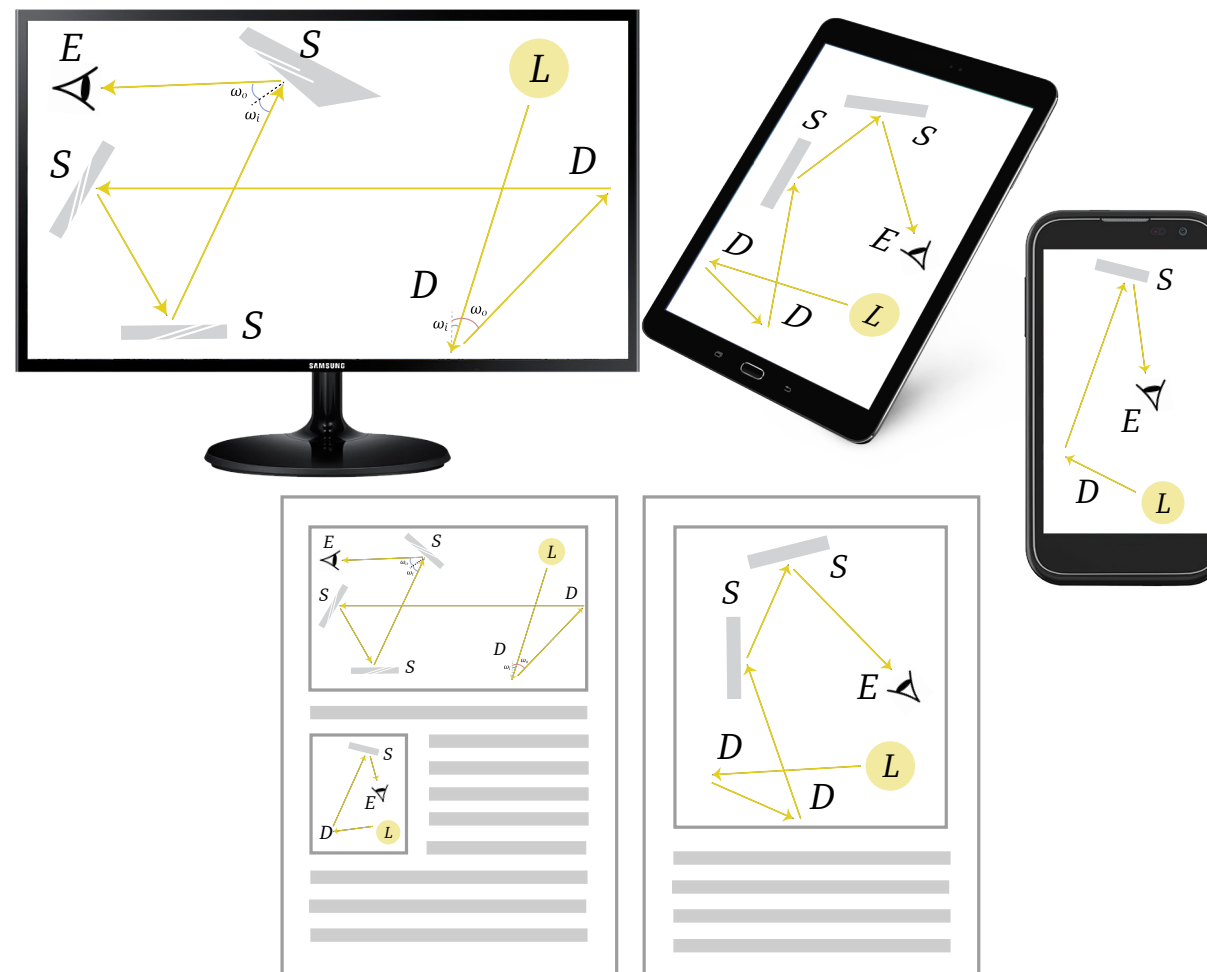
# More examples made in Penrose

**Work  
in  
progress!**



# More abilities that language enables (mockups)

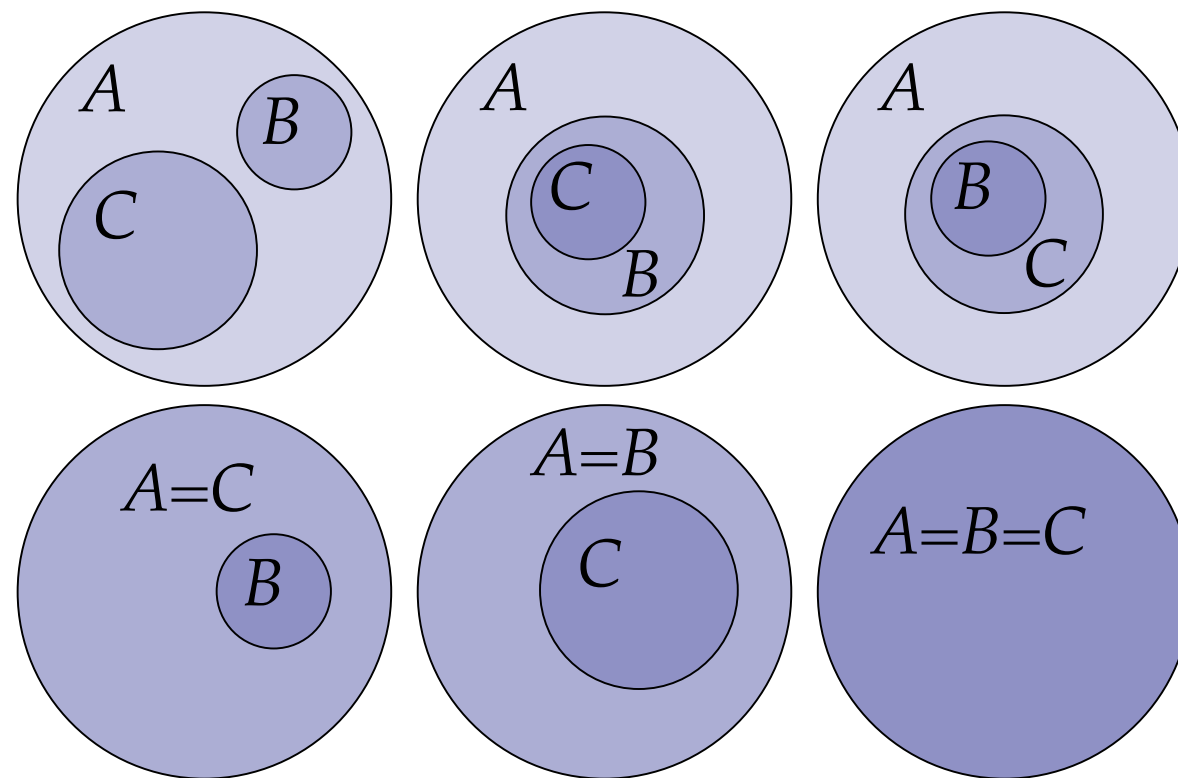
responsive diagrams



# More abilities that language enables (mockups)

intelligently exploring the diagram space  
by finding different cases in the notation

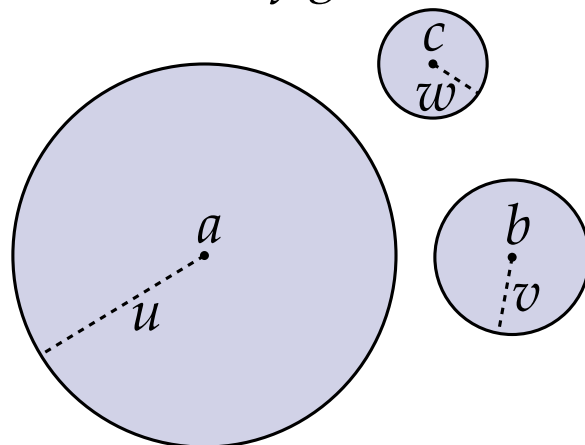
$$B, C \subset A$$



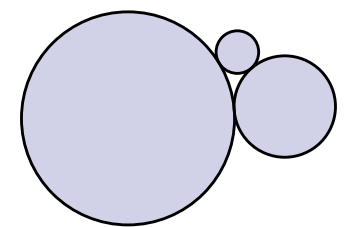
# More abilities that language enables (mockups)

intelligently exploring the diagram space  
by finding different cases in the visualization

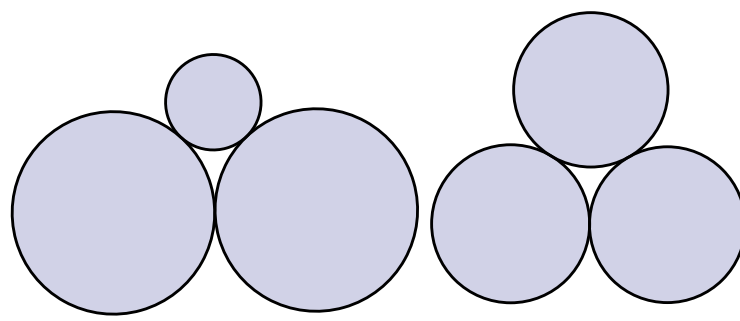
*(initial configuration)*



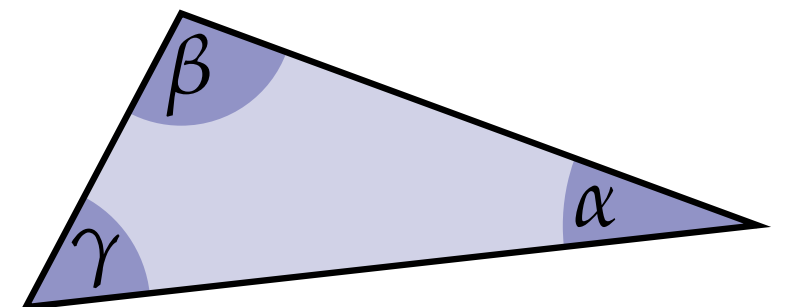
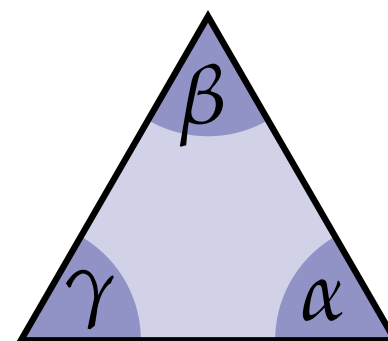
constraints
$ b - a ^2 = (u + v)^2$
$ c - b ^2 = (v + w)^2$
$ a - c ^2 = (w + u)^2$



*satisfies  
constraint (all  
circles tangent)*




*satisfies constraint &  
constraint Jacobian is  
degenerate*





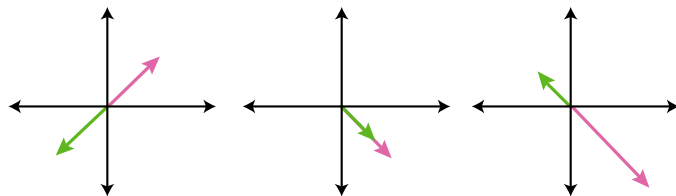
# More abilities that language enables (mockups)

creating visual exercises for students

 Linear Algebra  $\vec{v}$

1


$M = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$



Check share ↓


**Objective**


Write expressions that generate the diagrams above




# More abilities that language enables (mockups)


creating visual exercises for students

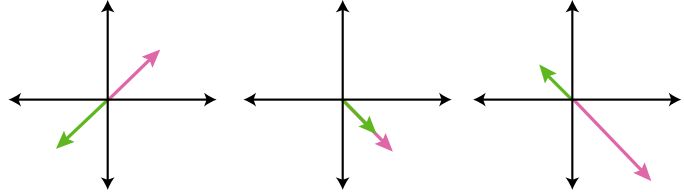



Linear Algebra 

1 **Vector** **c1** = <**a**, **b**>  
2 **Vector** **c2** = <**c**, **d**>  
3 **Matrix** **M** = **columns**(**c1**, **c2**)  
4 **LinearlyDependent**(**c1**, **c2**)



Cartesian 

$$M = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$$


Checkshare

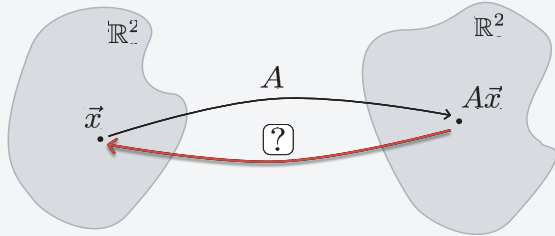
Success!

Write expressions that generate the diagrams above

# More abilities that language enables (mockups)

creating visual exercises for students

Write the math expression that describes the red arrow



Write the math expression that describes the red arrow

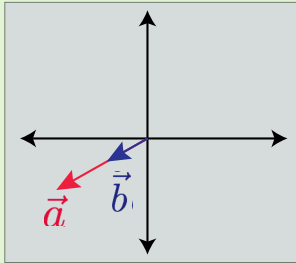
$A$   $^{-1}$   $\mathbb{R}$   $\vec{x}$   $^2$

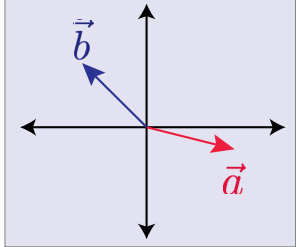
Check

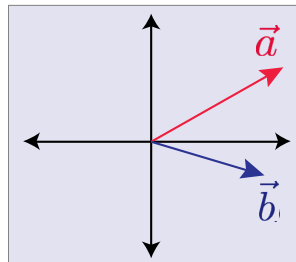
# More abilities that language enables (mockups)

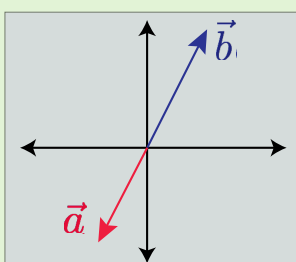
creating visual exercises for students


Select diagrams that correspond to the following expression  
 $\text{LinearlyDependent}(\vec{a}, \vec{b})$

1 ☒ 

2 ☐ 

3 ☐ 

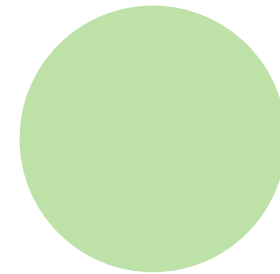
4 ☒ 

 Correct! [Continue](#)

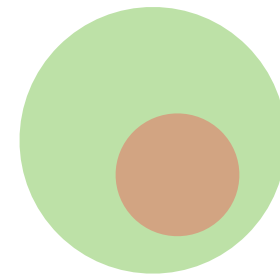
## Key idea

Sets  $A, B, C$  such that  $B \subseteq A$  and  $C \subseteq A$ .

Set



$\subseteq$



It is powerful to formally encode the **mapping** from abstract objects and relationships to their visual representations.

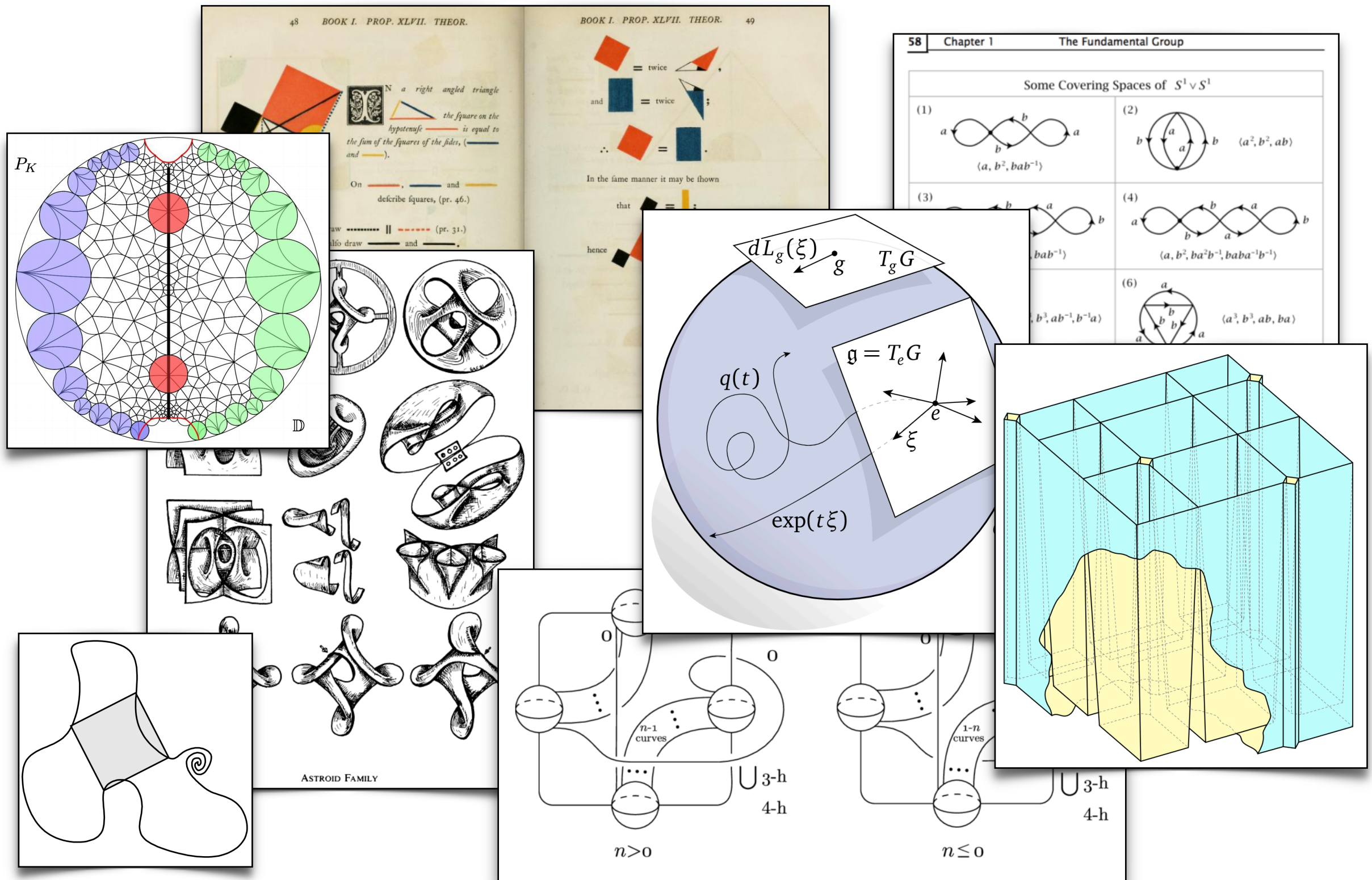
# Thanks to my collaborators!

Wode Ni, Max Krieger, Rain Du, Dor Ma'ayan,  
Lily Shellhammer, Jenna Wise

advised by Keenan Crane, Jonathan Aldrich,  
and Joshua Sunshine



We want to make diagrams like these the norm—  
not the exception!



# We want your input!

Come talk to Katherine ([kqy@cs.cmu.edu](mailto:kqy@cs.cmu.edu))





turning mathematical notation into beautiful diagrams

<http://penrose.ink>  
<http://use.penrose.ink>