Penrose: from mathematical notation to beautiful diagrams

Katherine Ye*, Nimo Ni*, Max Krieger*, Dor Ma'ayan†, Jenna Wise*, Jonathan Aldrich*, Joshua Sunshine*, and Keenan Crane*

*Carnegie Mellon University
†Technion

SIGGRAPH 2020

Our goal: create high-quality diagrams directly from abstract mathematical statements.

set theory

venn-3d

run

sign in

venn 3d

sub sty dsl

```


AutoLabel All
```

editing fill

Click run to render your diagram.

resample

autostep (off)

download

set theory

venn-3d

run

sign in

venn 3d

sub sty dsl

```

AutoLabel All
```

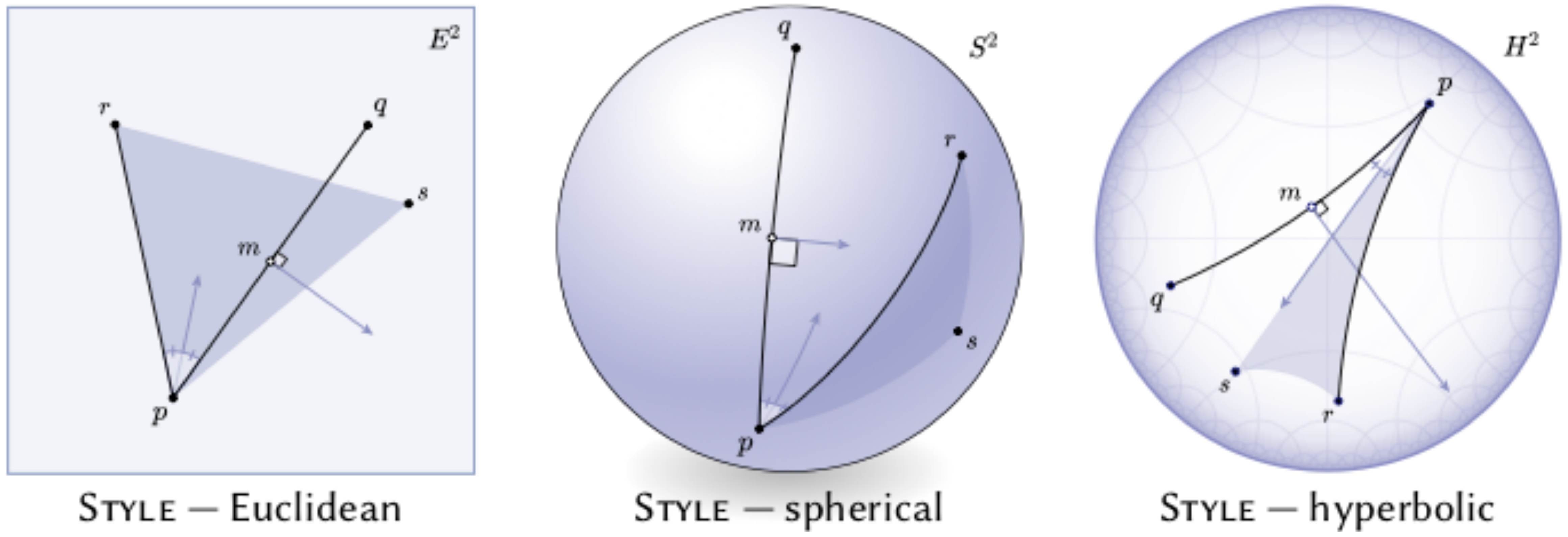
editing fill

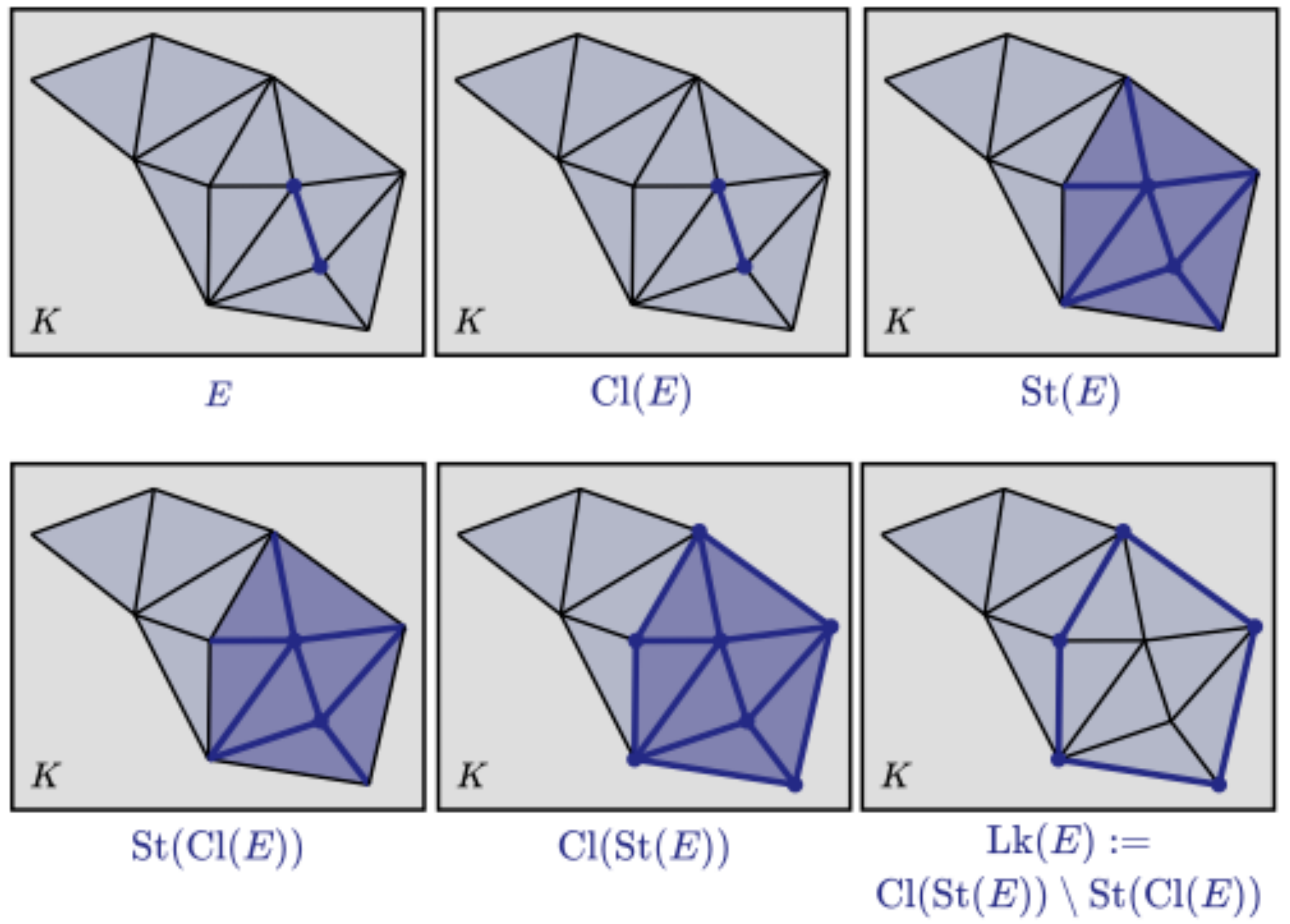
Click run to render your diagram.

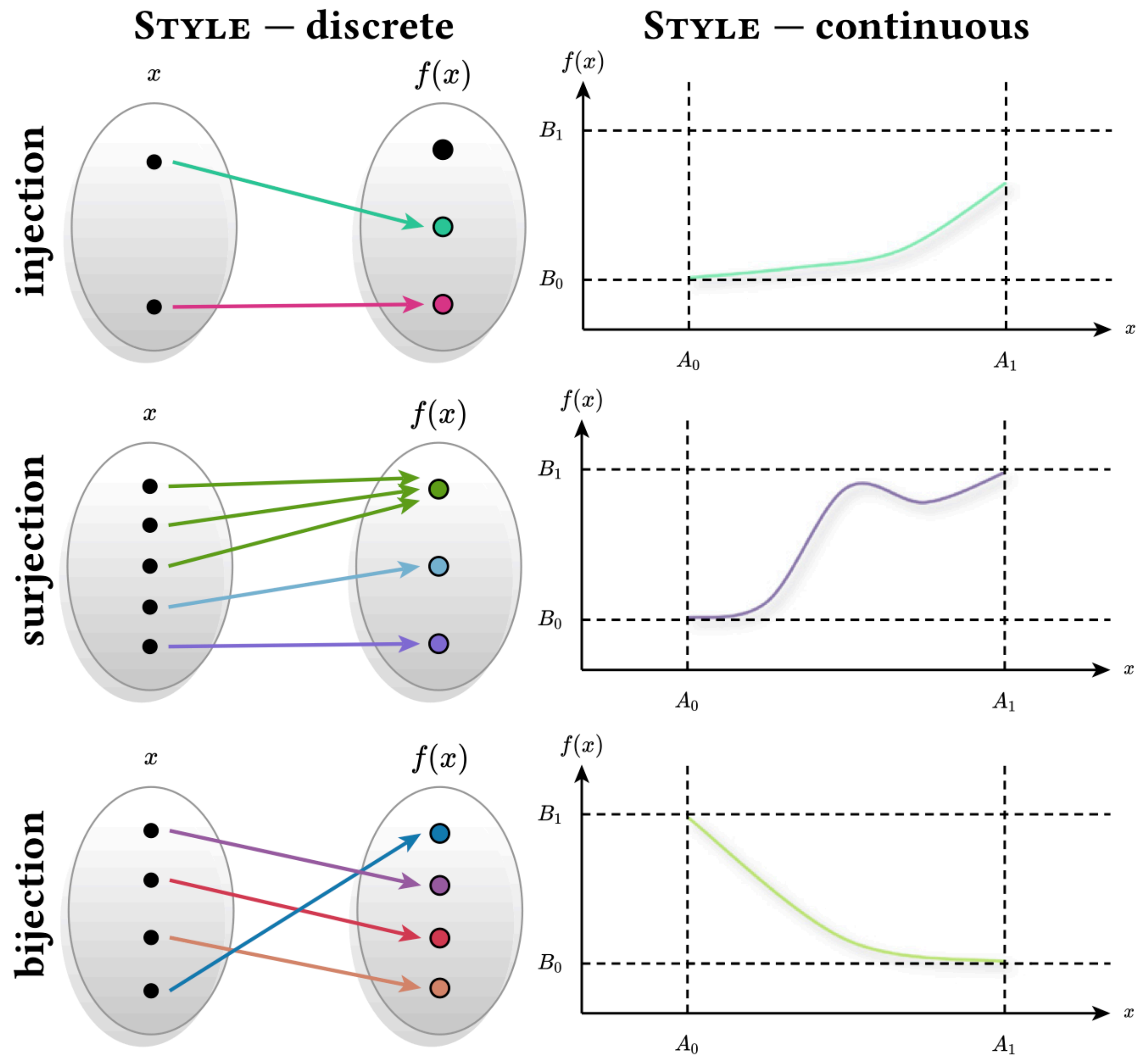
resample autostep (off) download

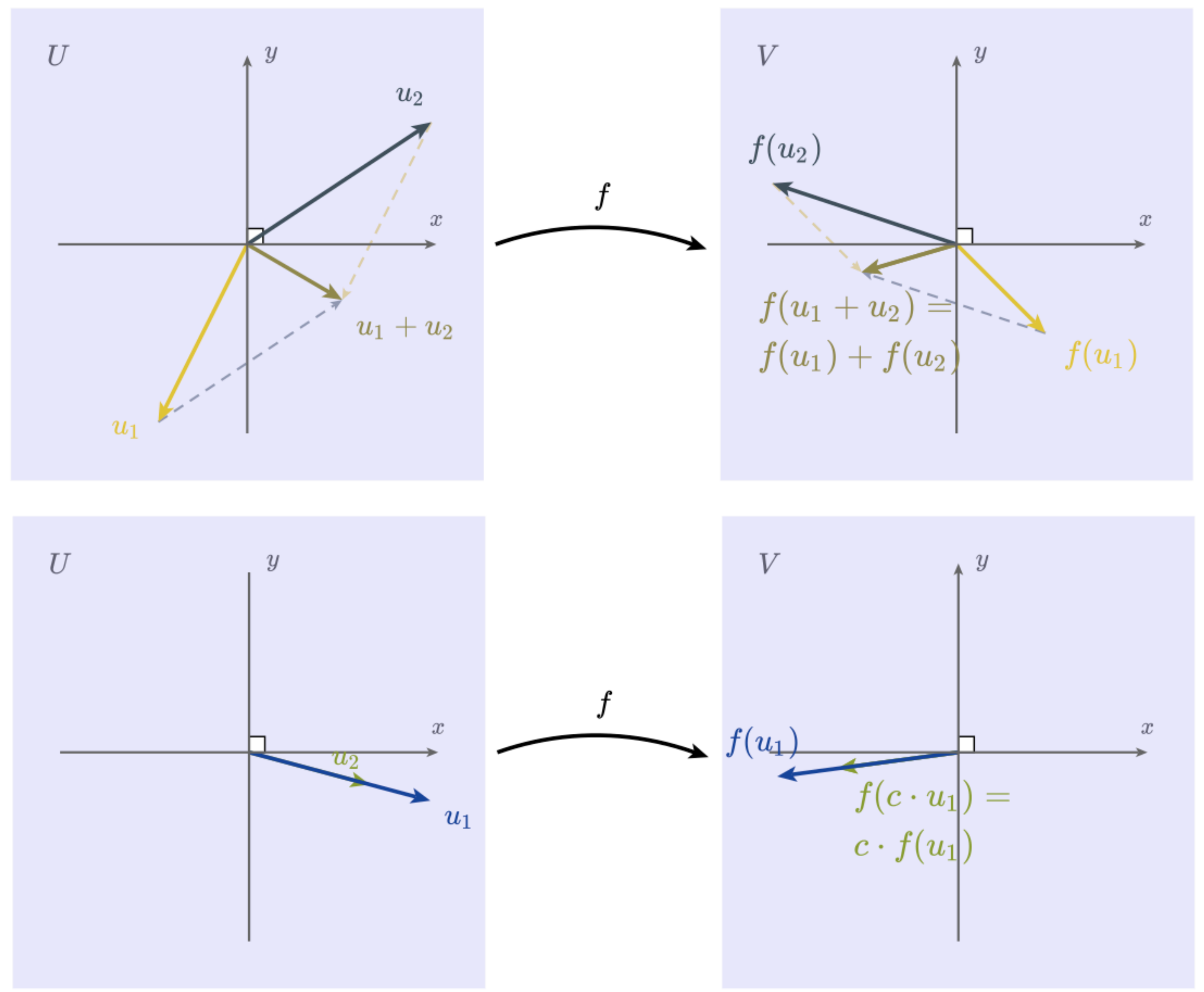
Can draw diagrams from any user-defined domain:

Can draw diagrams from any user-defined domain:





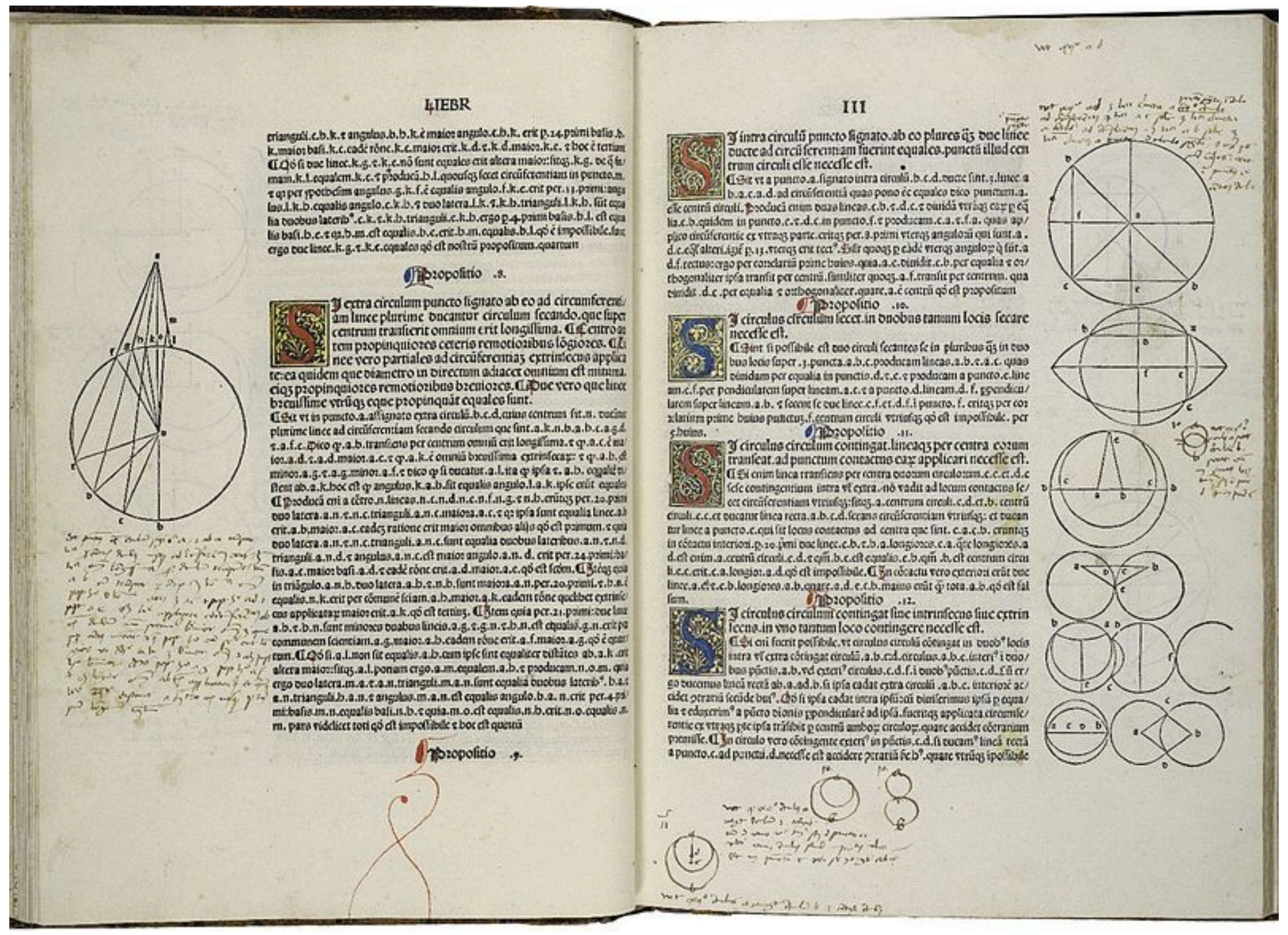




Making pictures has been crucial for making mathematical progress.

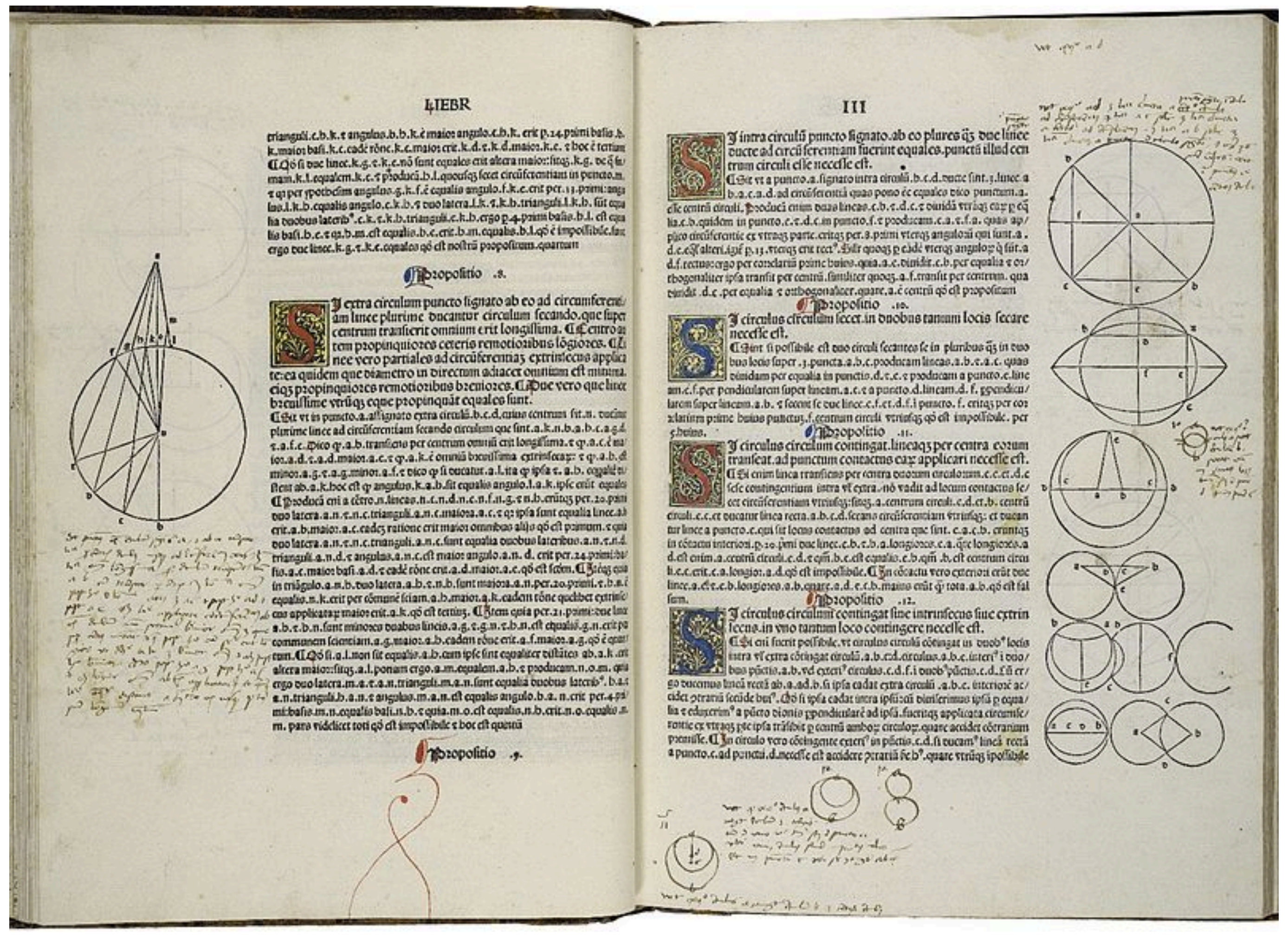
Making pictures has been crucial for making mathematical progress.

Euclid's "Elements"

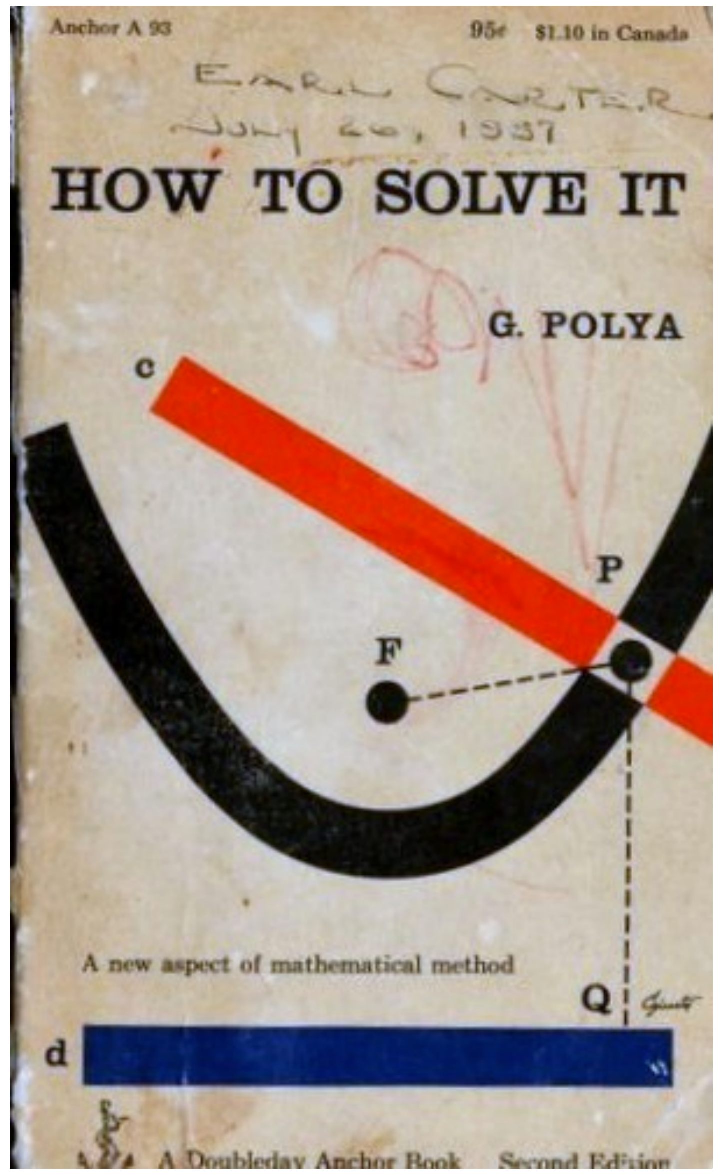


Making pictures has been crucial for making mathematical progress.

Euclid's "Elements"

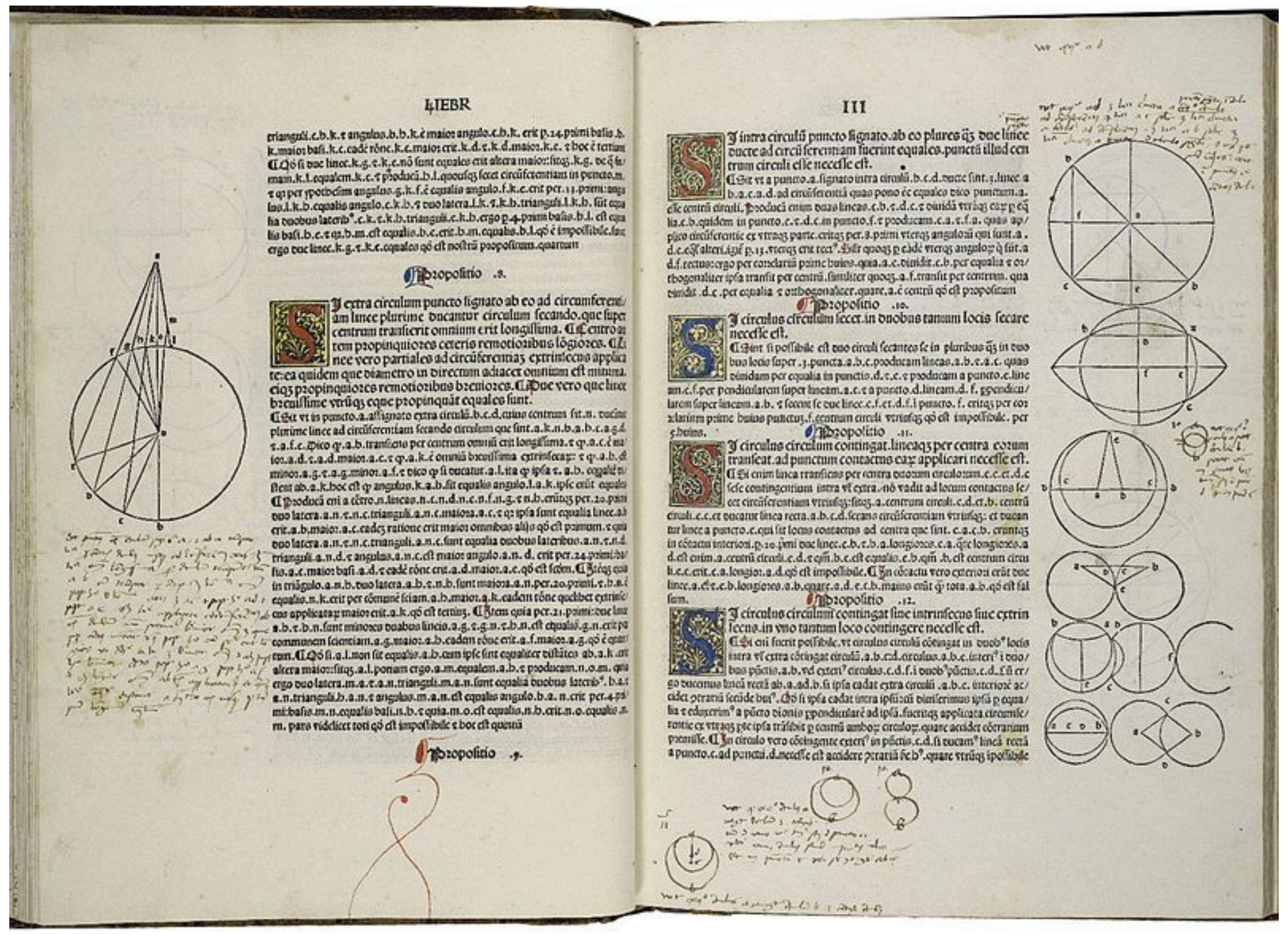


the process of problem-solving

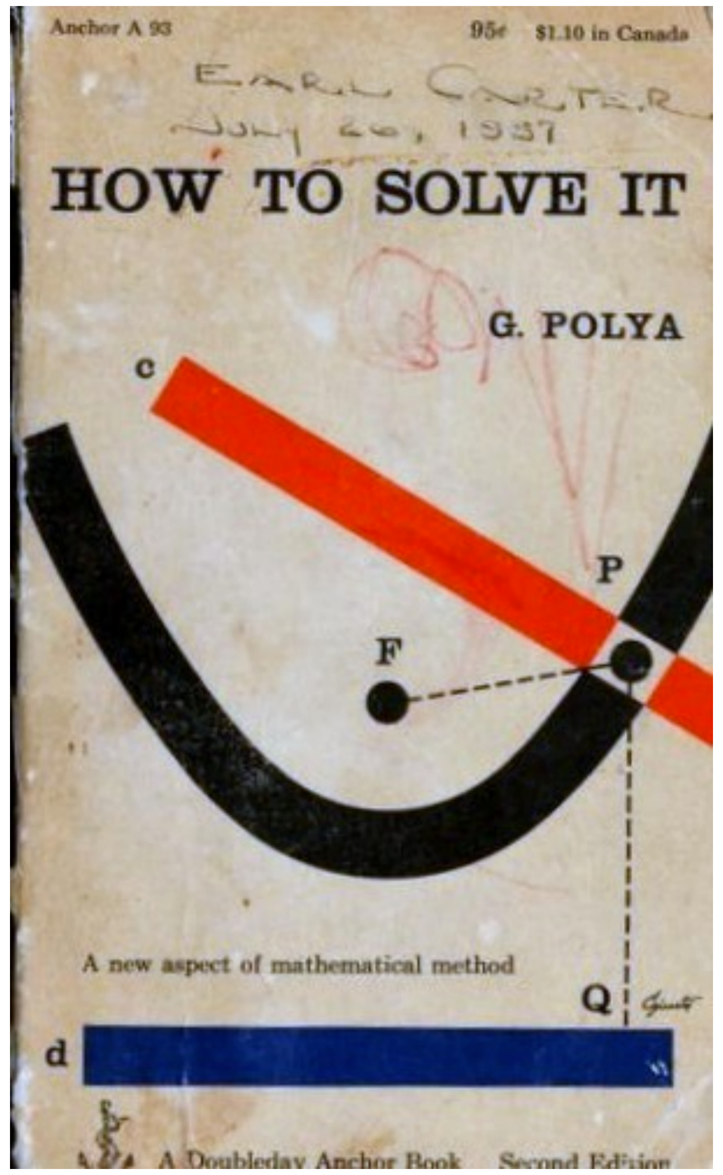


Making pictures has been crucial for making mathematical progress.

Euclid's "Elements"



the process of problem-solving



Feynman diagrams

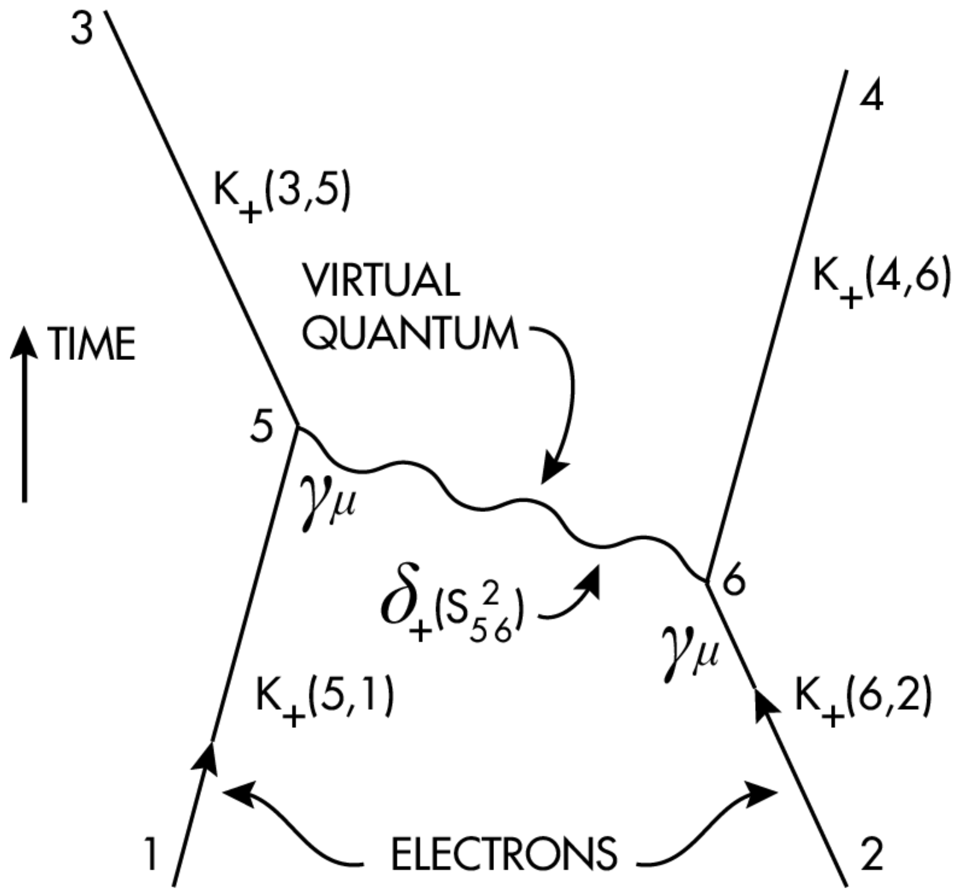
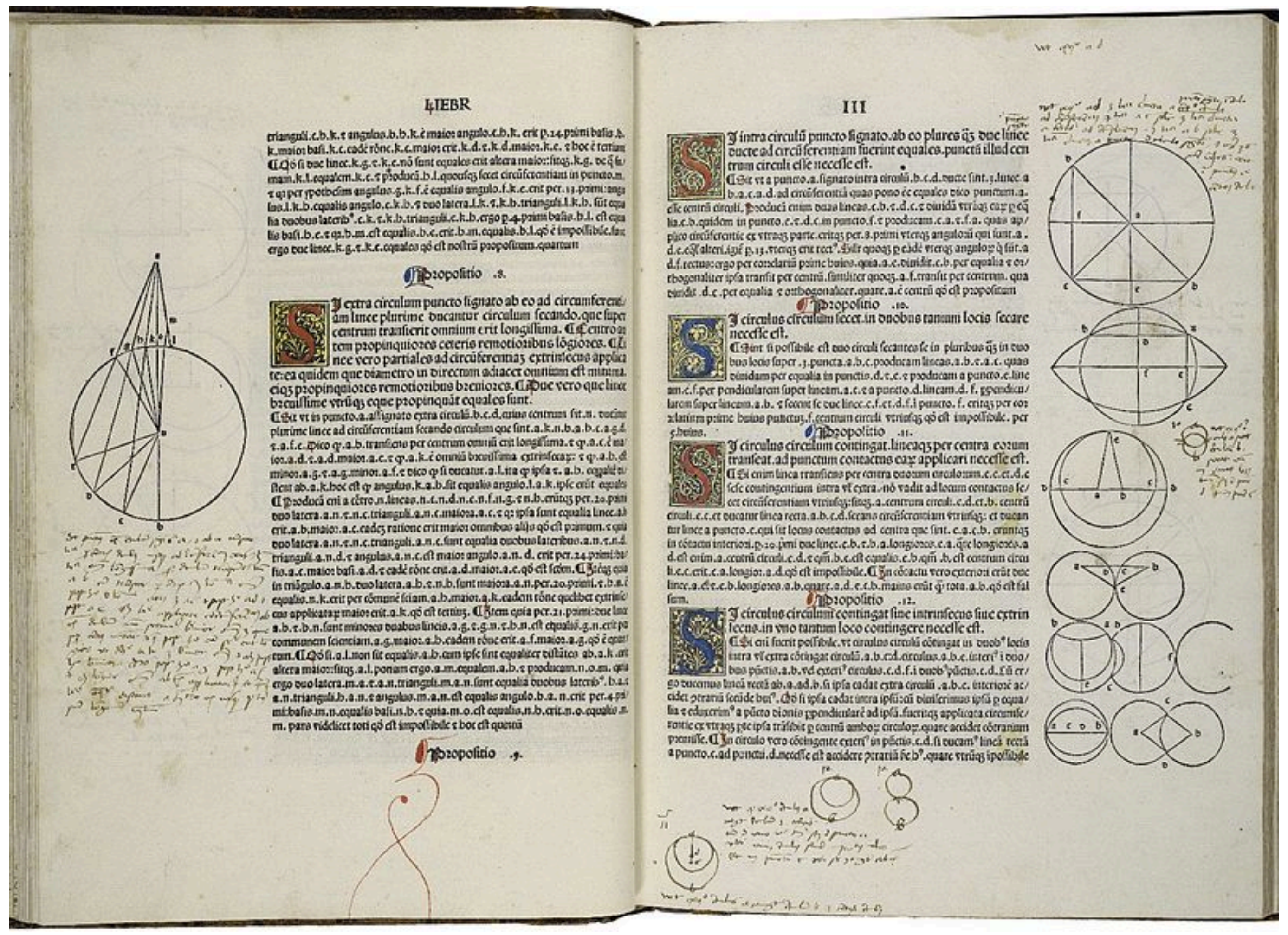


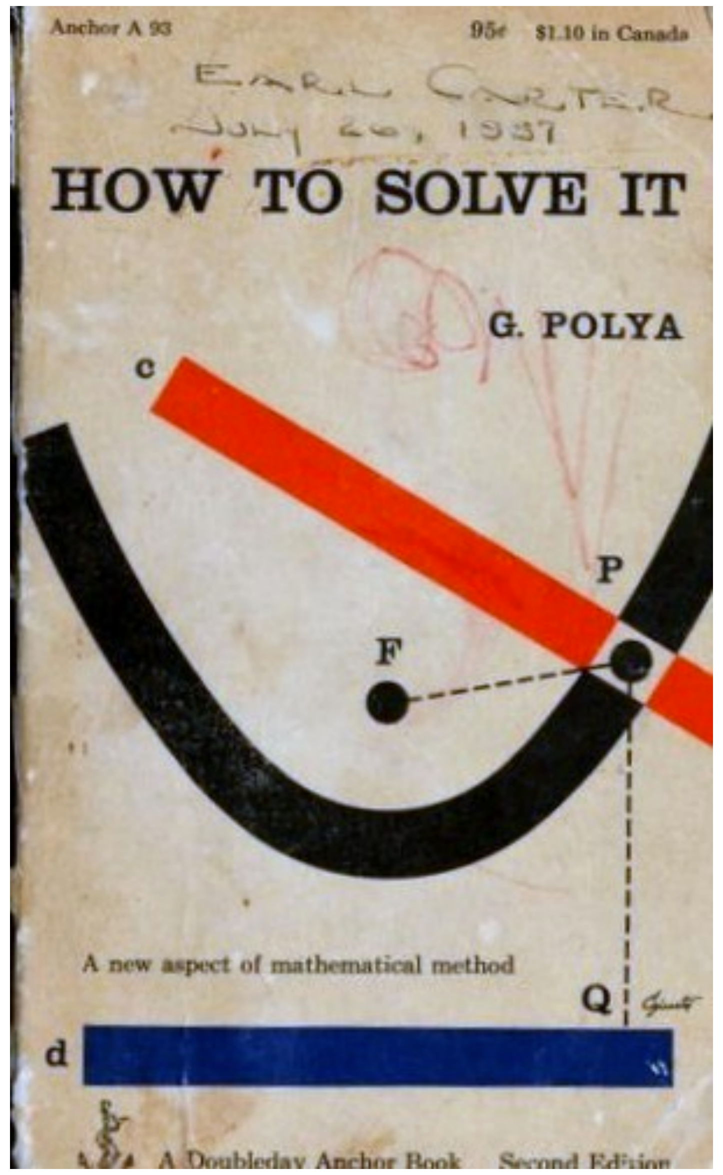
image credit: Olena Shmahala/Quanta Magazine

Making pictures has been crucial for making mathematical progress.

Euclid's "Elements"



the process of problem-solving



Feynman diagrams

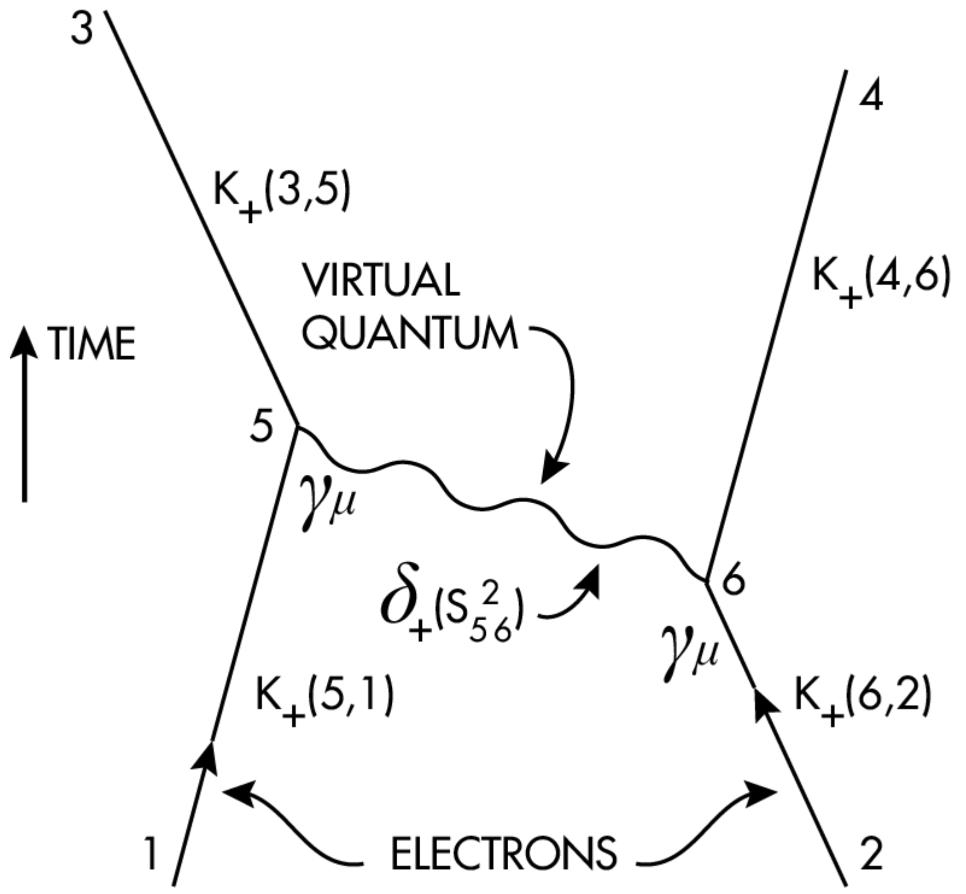


image credit: Olena Shmahala/Quanta Magazine

Cayley graphs

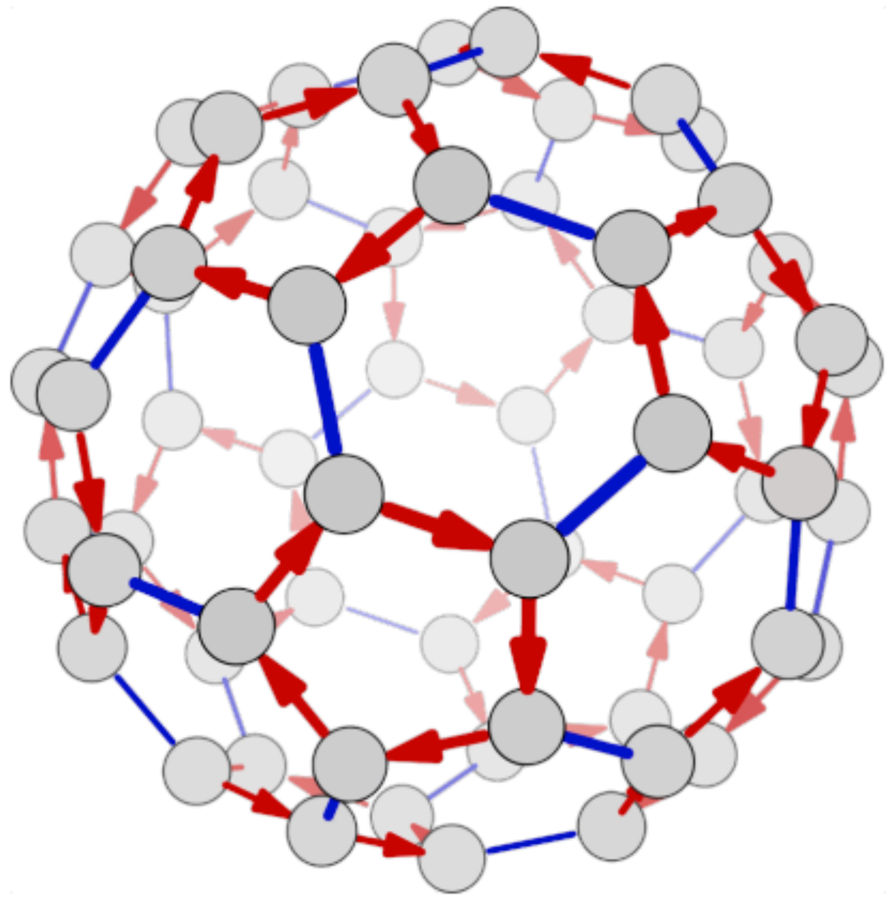
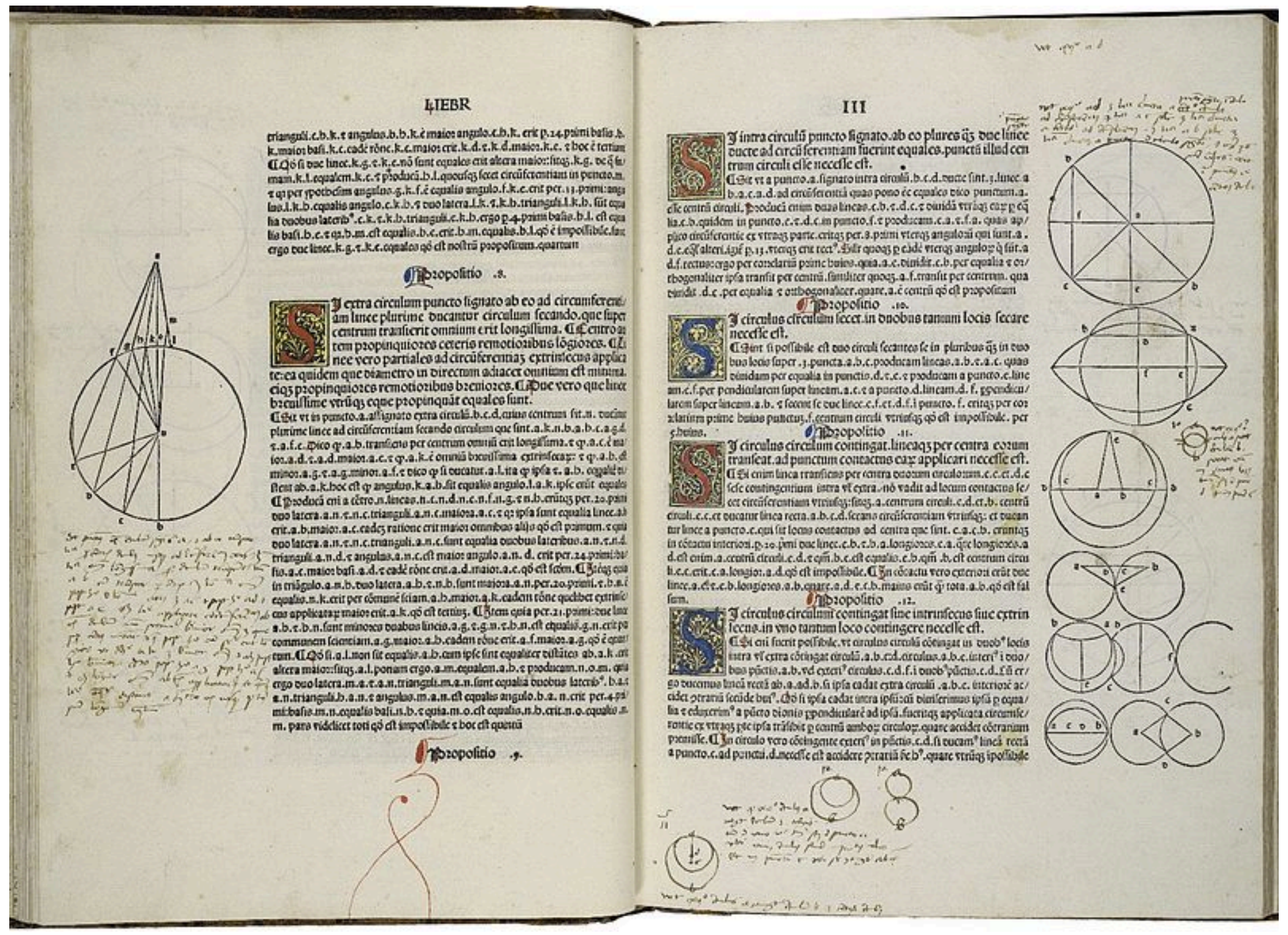


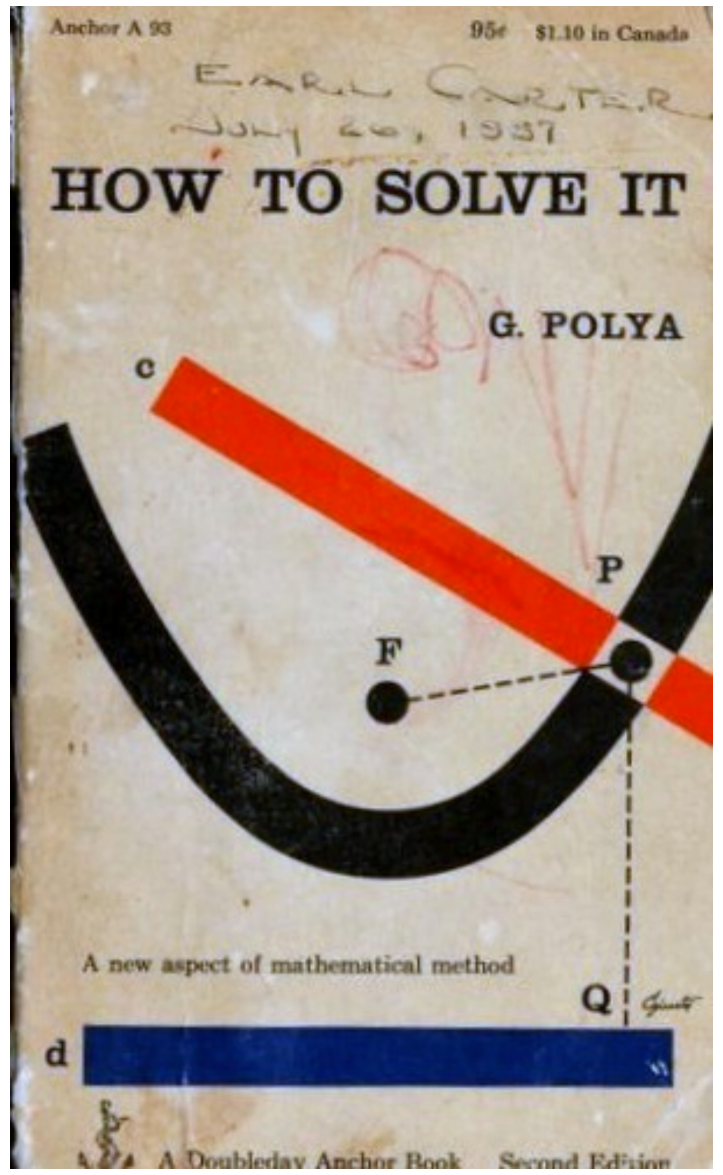
image credit: Nathan Carter

Making pictures has been crucial for making mathematical progress.

Euclid's "Elements"



the process of problem-solving



Feynman diagrams

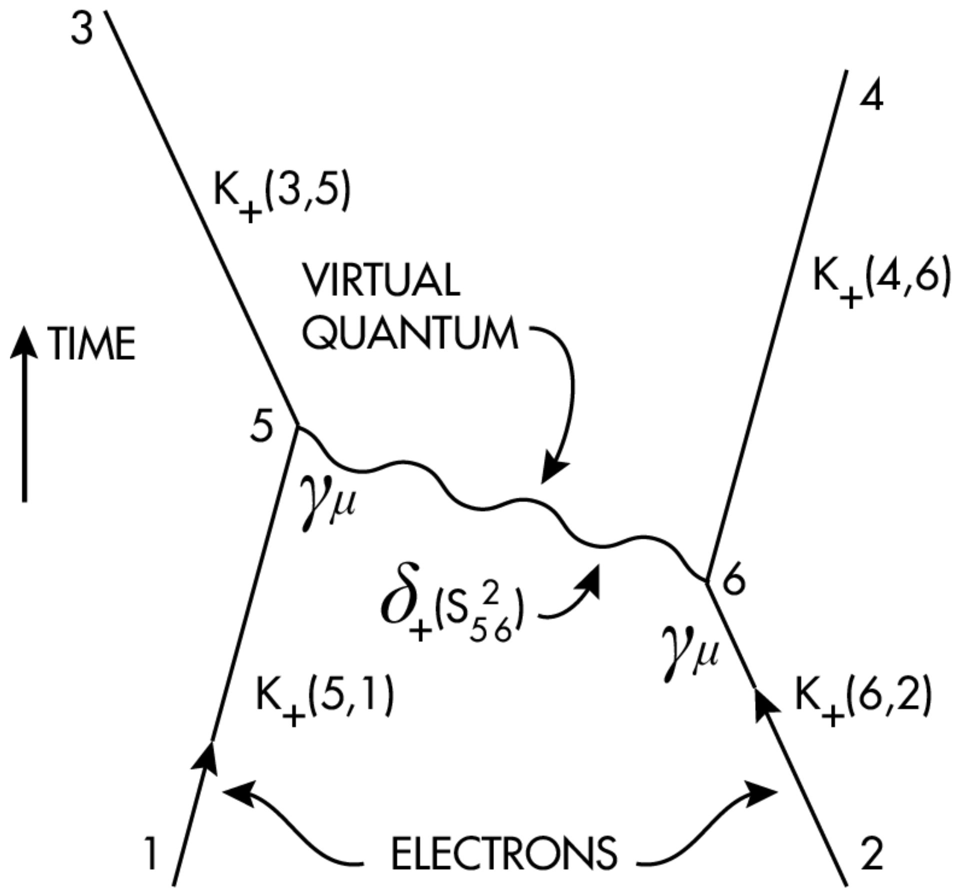


image credit: Olena Shmahala/Quanta Magazine

Cayley graphs

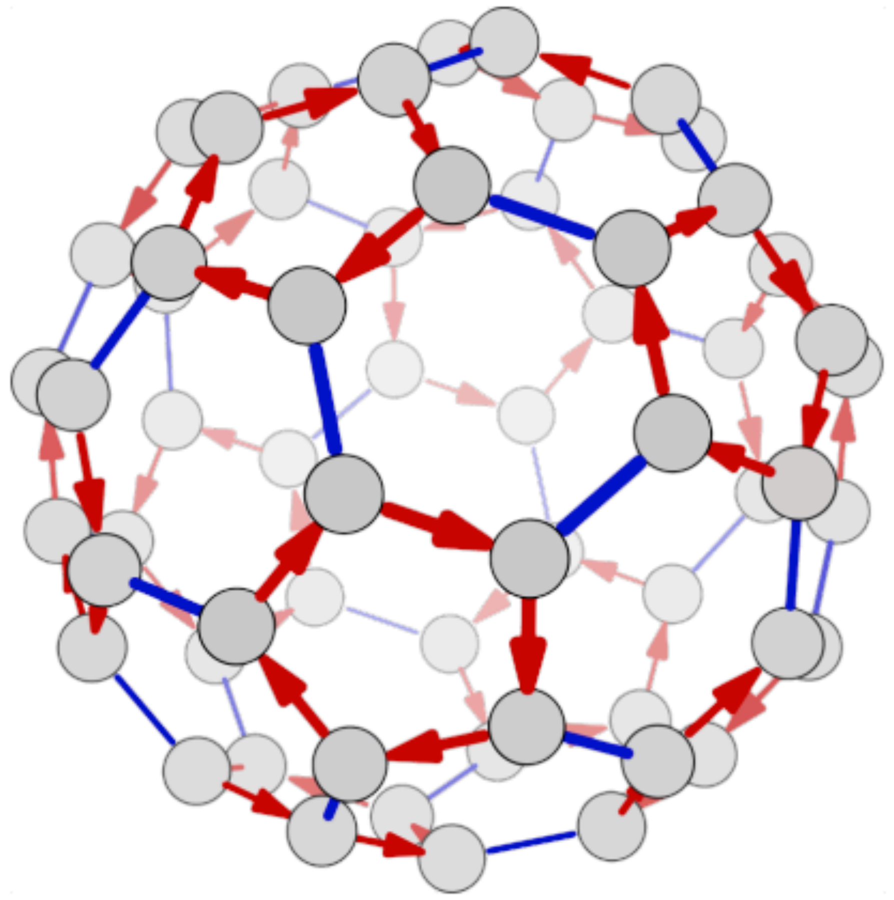


image credit: Nathan Carter

sphere eversion

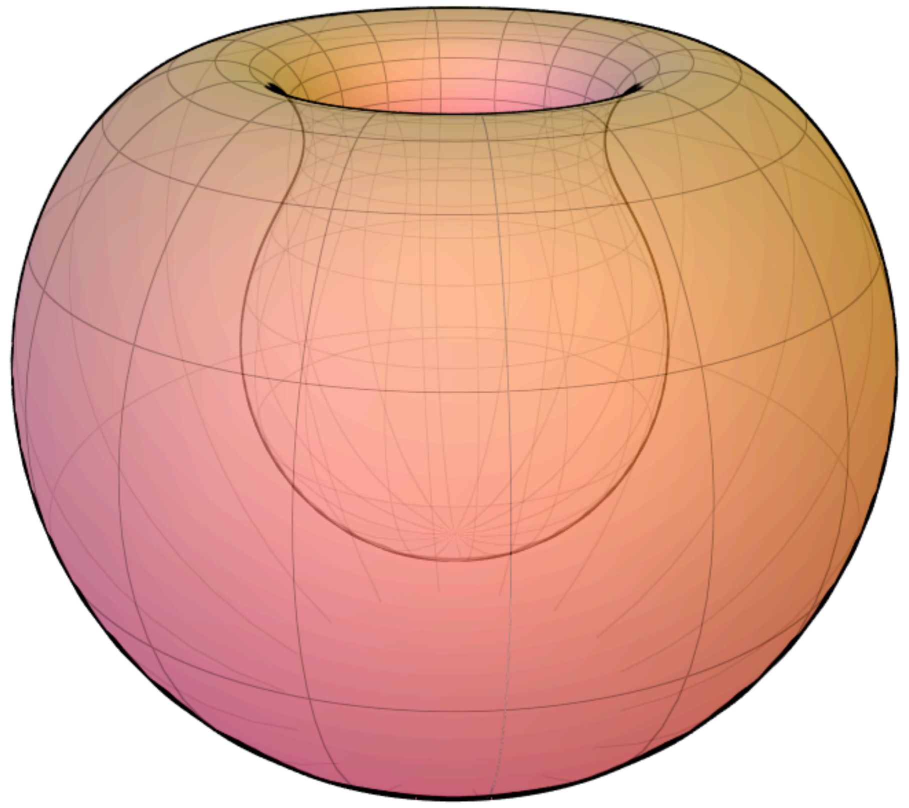
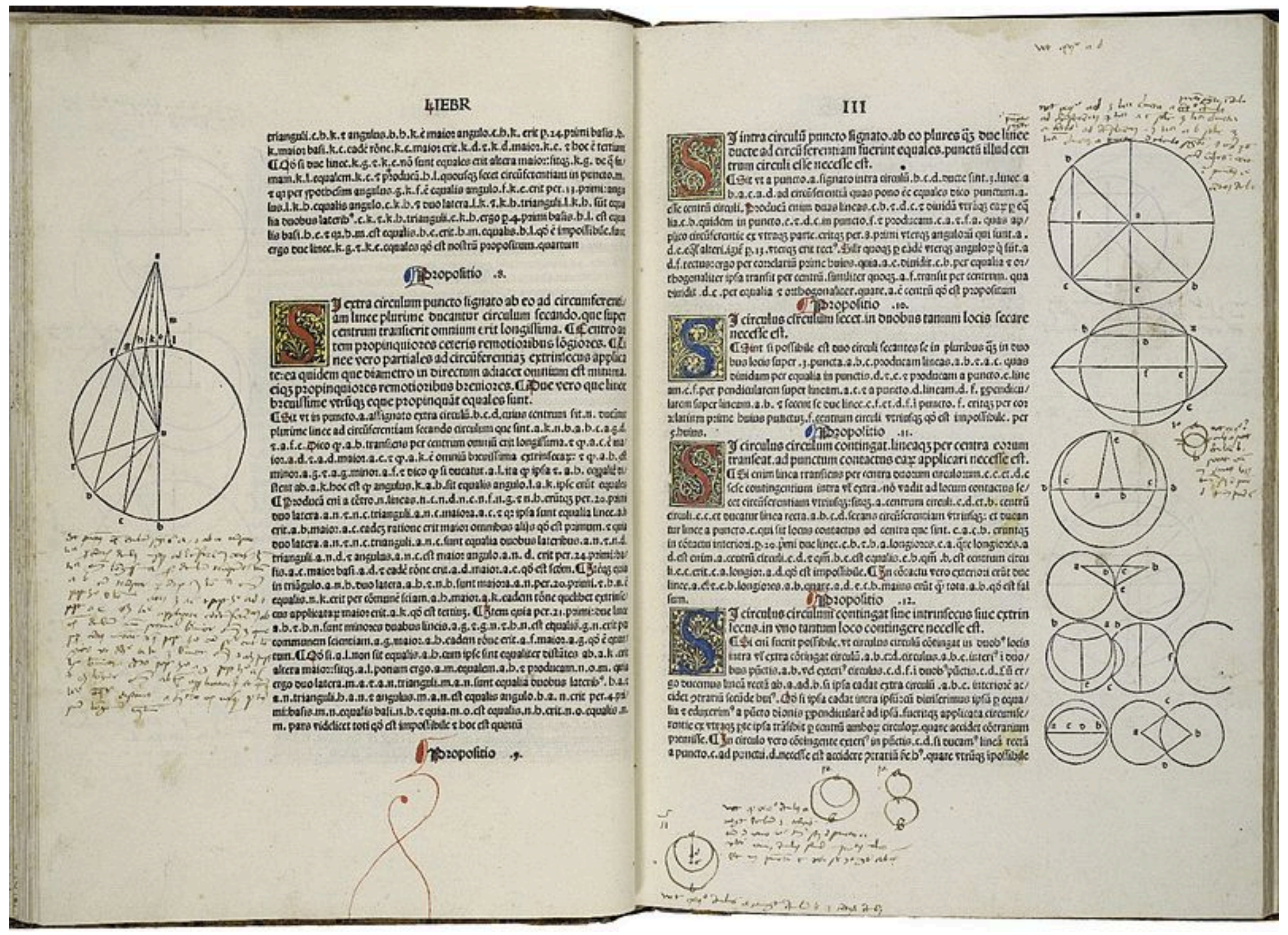


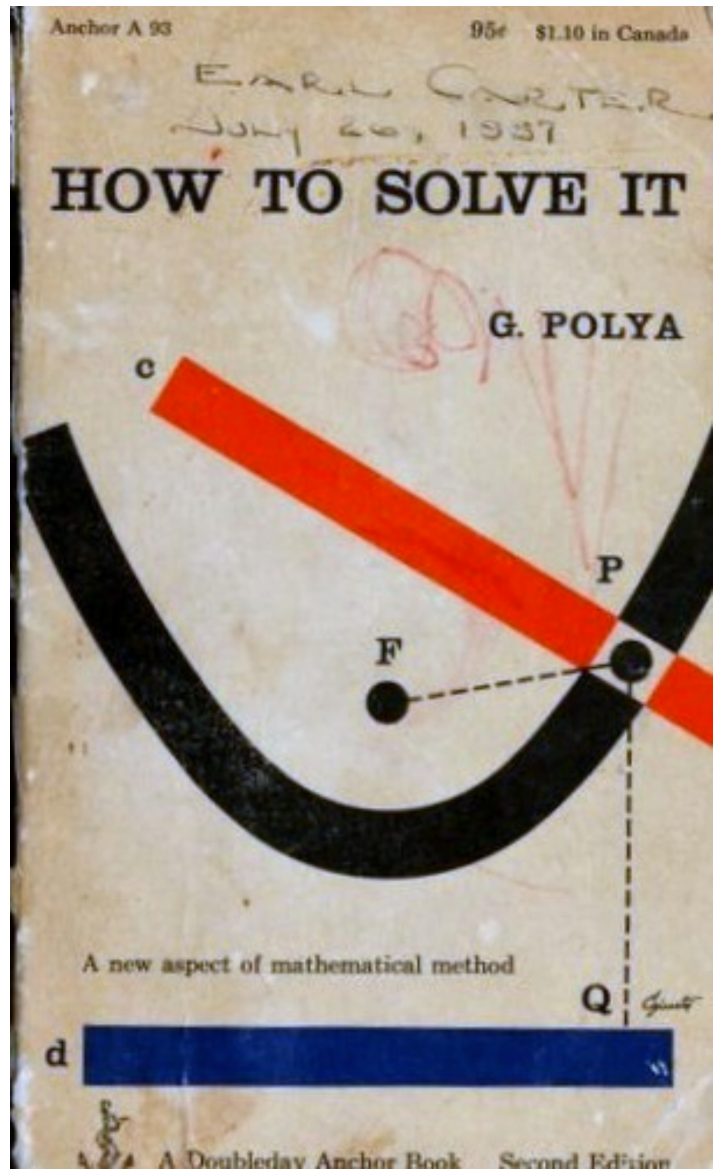
image credit: Ricky Reusser

Making pictures has been crucial for making mathematical progress.

Euclid's "Elements"



the process of problem-solving



Feynman diagrams

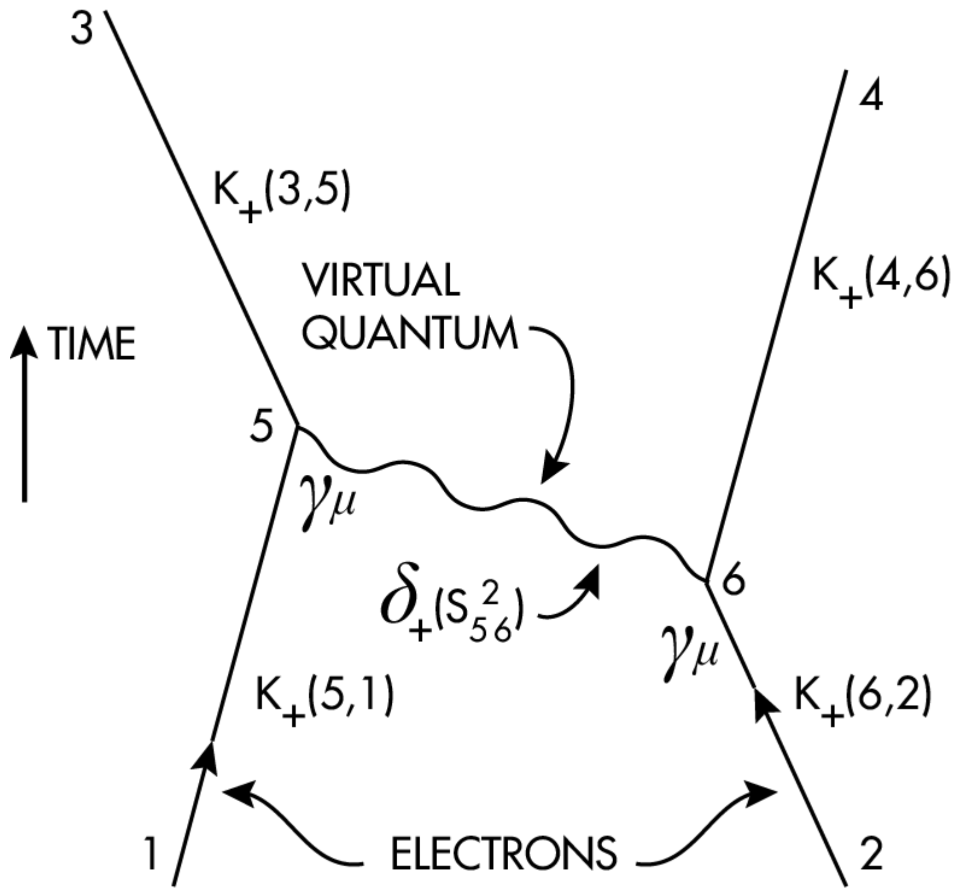


image credit: Olena Shmahala/Quanta Magazine

Cayley graphs

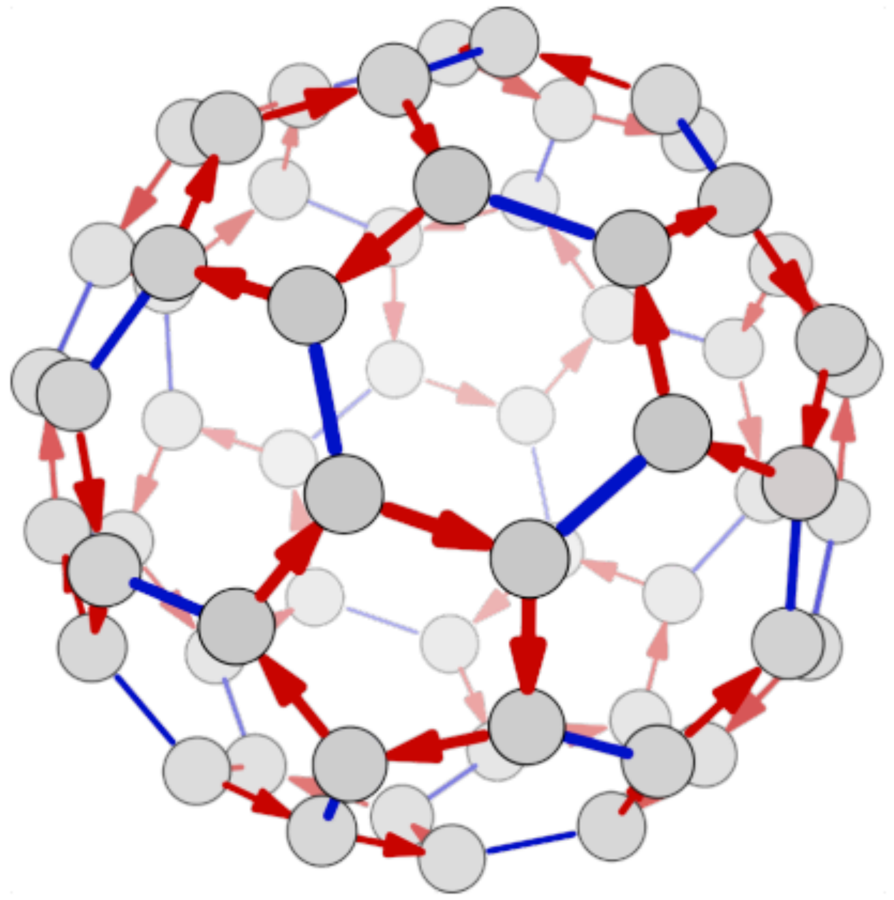


image credit: Nathan Carter

sphere eversion

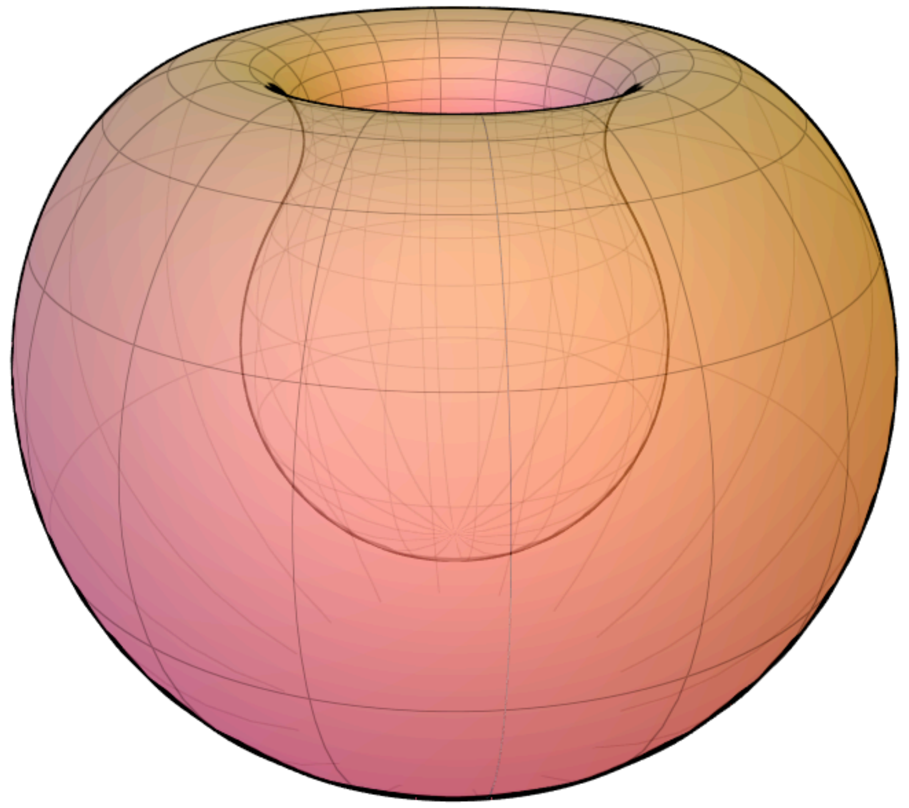


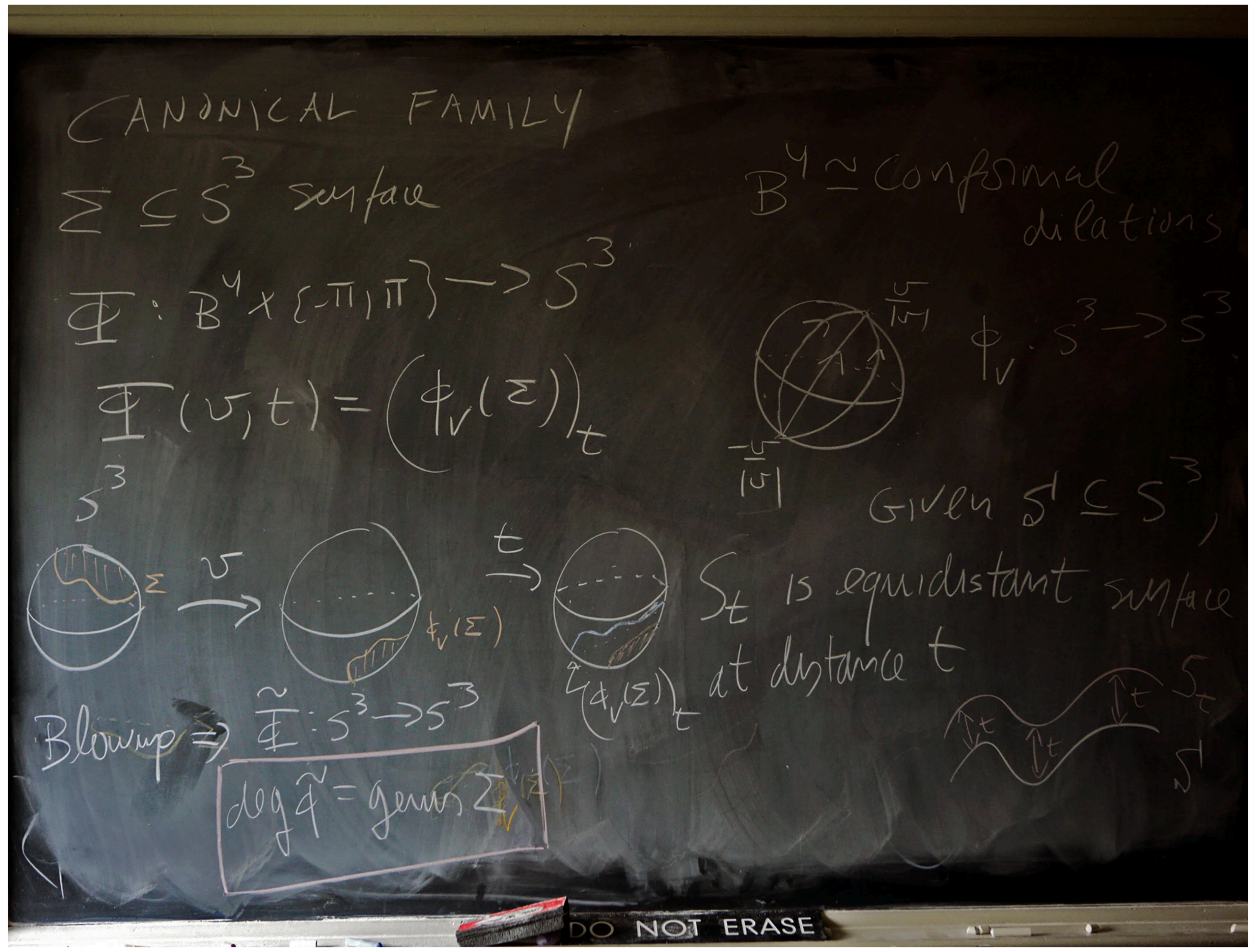
image credit: Ricky Reusser

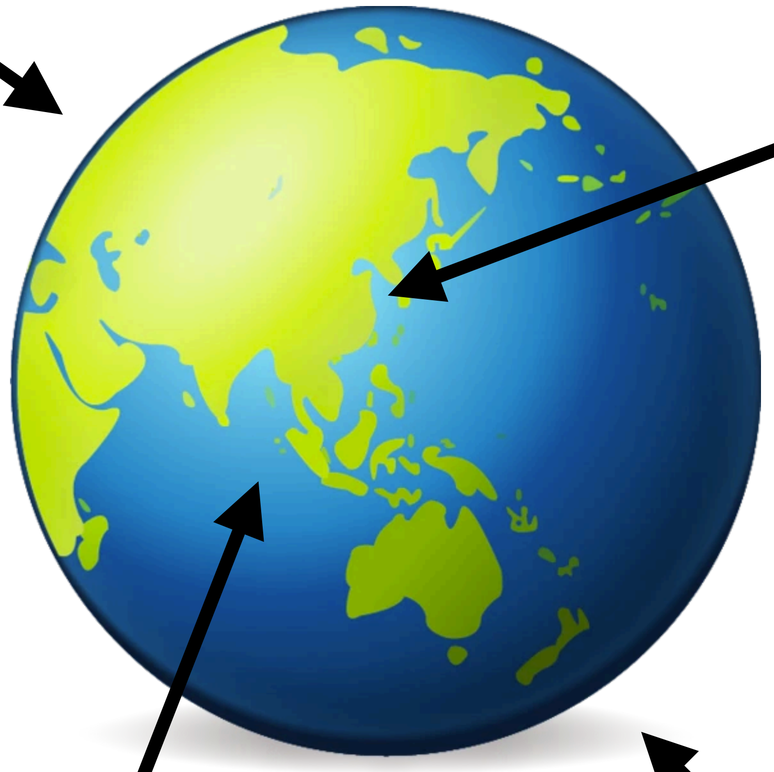
...and many more!

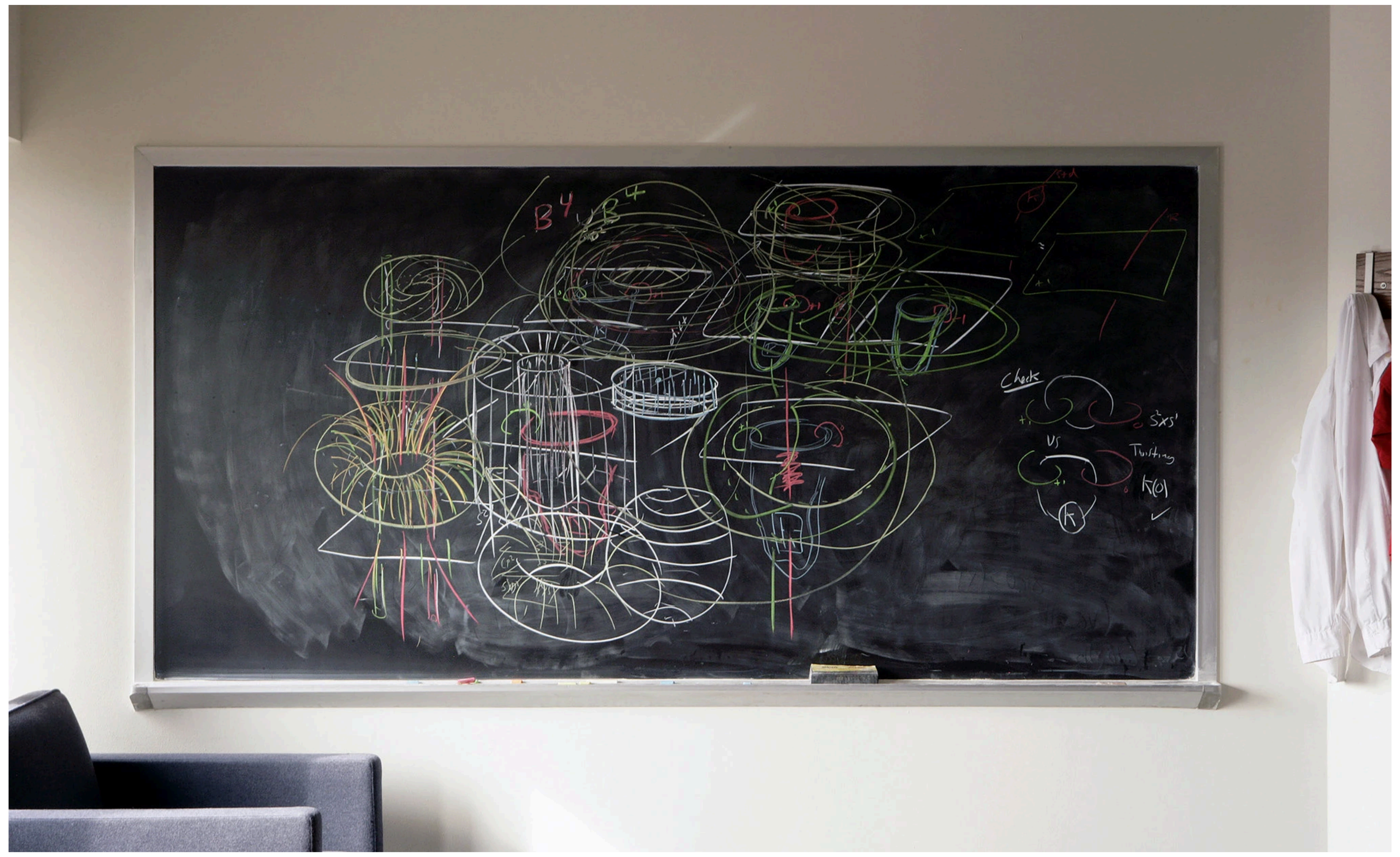
People ~~want~~ need to make diagrams!

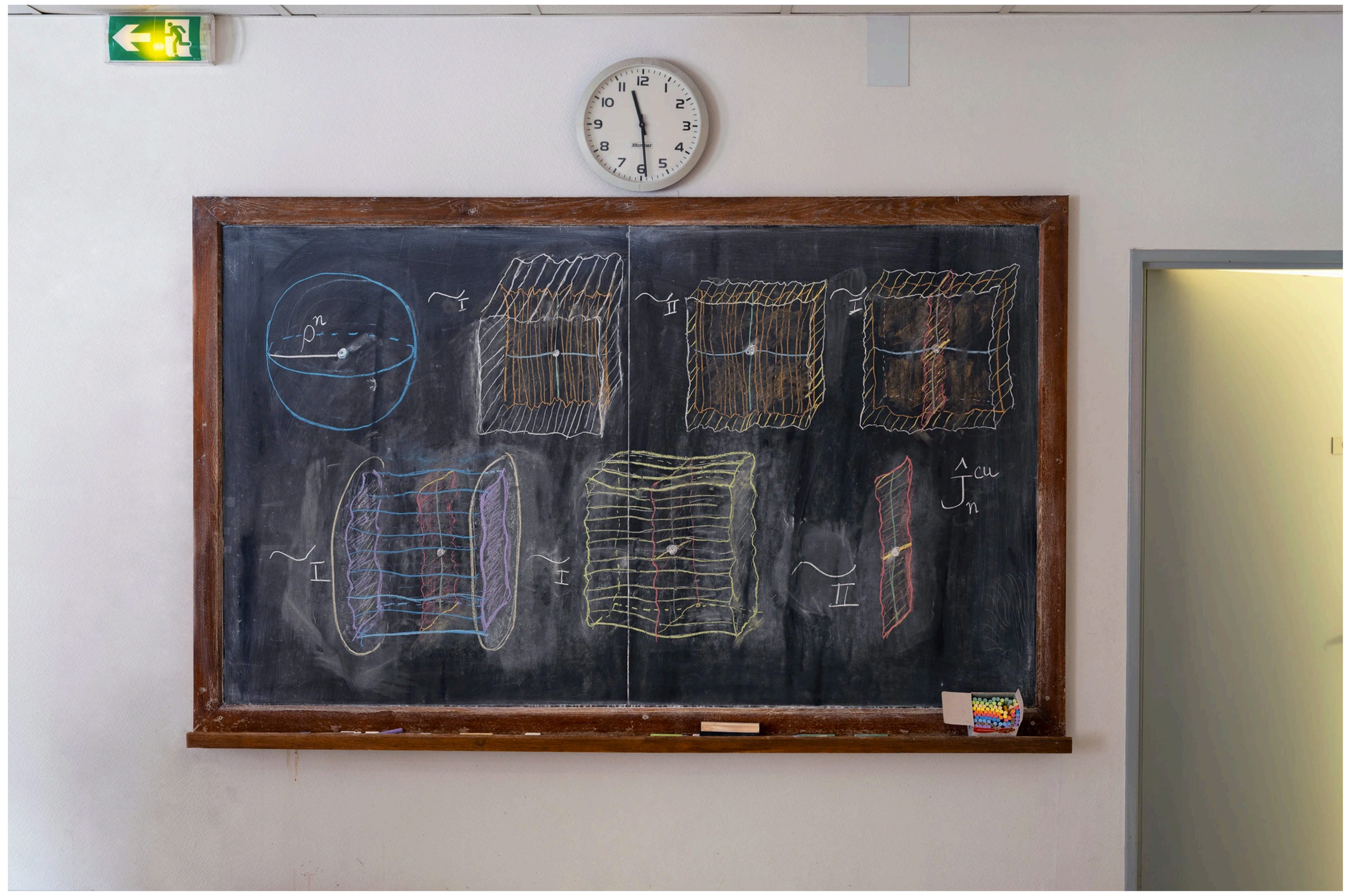
image credit: “Where Theory Meets Chalk, Dust Flies”/NYT

People ~~want~~ need to make diagrams!









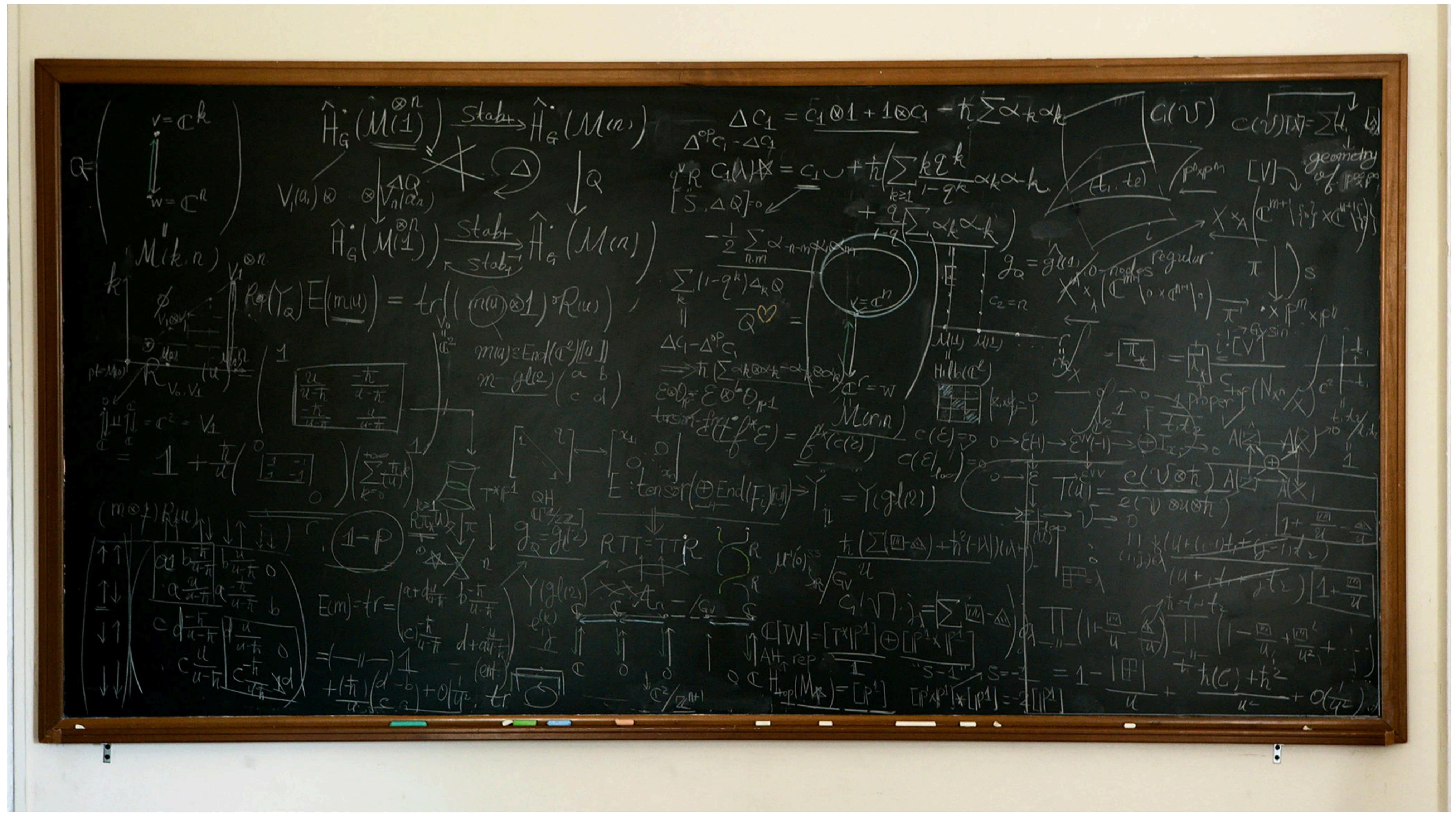


image credit: “Where Theory Meets Chalk, Dust Flies”/NYT

Yet, authoring **digital** diagrams is still hard

Yet, authoring **digital** diagrams is still hard

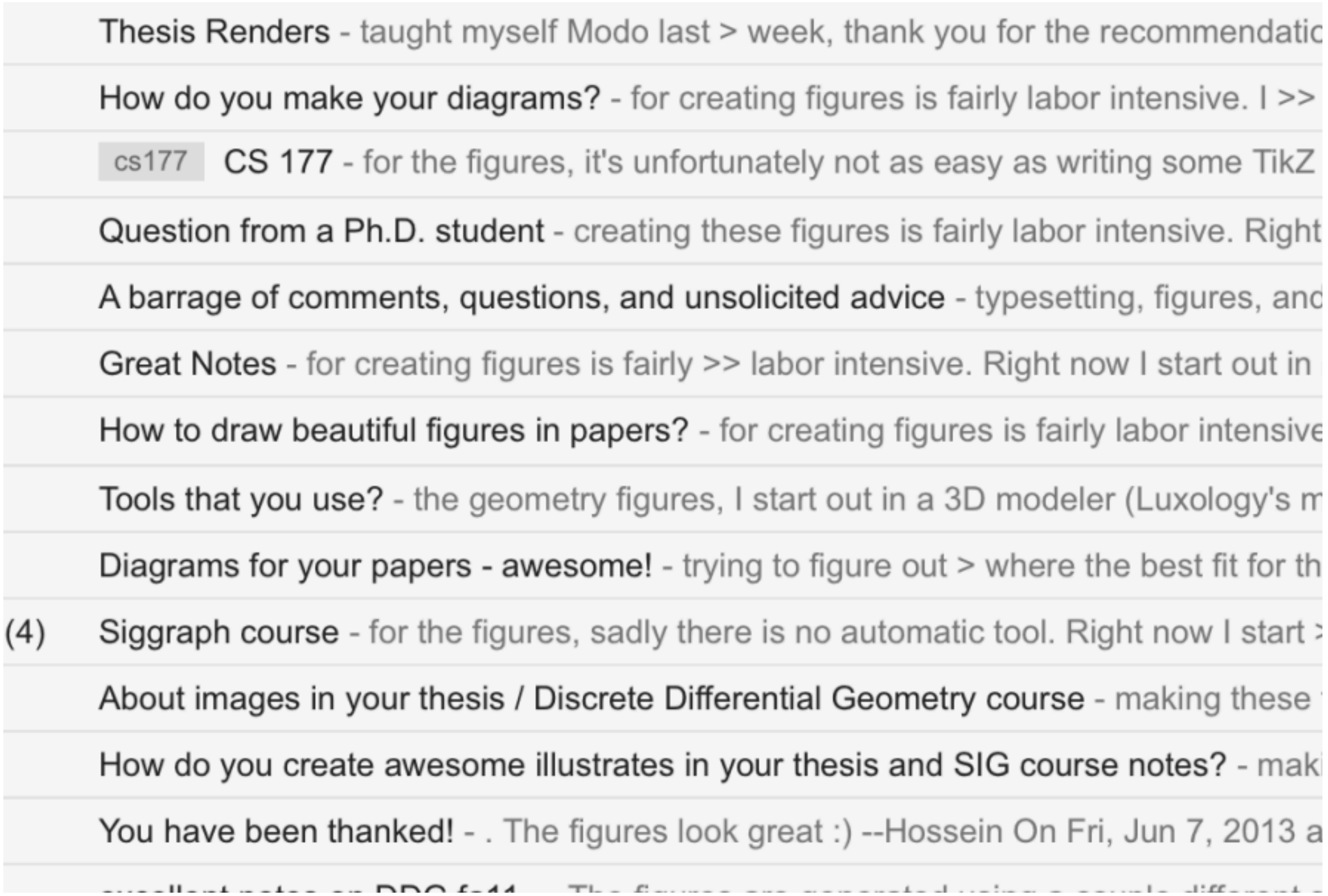
"I can't be the only mathematician who has wasted hours making tiny adjustments to my LaTeX drawings in an effort to make them even minimally attractive."

—Leila Sloman, math PhD candidate (on the AMS blog)

Yet, authoring **digital** diagrams is still hard

“I can't be the only mathematician who has wasted hours making tiny adjustments to my LaTeX drawings in an effort to make them even minimally attractive.“

—Leila Sloman, math PhD candidate (on the AMS blog)



Yet, authoring **digital** diagrams is still hard

"I can't be the only mathematician who has wasted hours making tiny adjustments to my LaTeX drawings in an effort to make them even minimally attractive."

—Leila Sloman, math PhD candidate (on the AMS blog)

Thesis Renders - taught myself Modo last > week, thank you for the recommendatic

How do you make your diagrams? - for creating figures is fairly labor intensive. I >>

cs177 CS 177 - for the figures, it's unfortunately not as easy as writing some TikZ

Question from a Ph.D. student - creating these figures is fairly labor intensive. Right

A barrage of comments, questions, and unsolicited advice - typesetting, figures, and

Great Notes - for creating figures is fairly >> labor intensive. Right now I start out in

How to draw beautiful figures in pap

Tools that you use? - the geometry fi

Diagrams for your papers - awesom

(4) Siggraph course - for the figures, sa

s / Discre

What software is useful for generating diagrams to use in formal proofs?

Ask Question

Asked 6 years, 7 months ago Active 6 years, 7 months ago Viewed 382 times

2

What software is useful for generating diagrams to use in formal proofs? I am interested in software for geometry diagrams, graphs, plots, and any other useful kinds of diagrams.

proof-writing math-software

Software for drawing geometry diagrams

Asked 9 years, 11 months ago Active 6 months ago Viewed 153k times

147

What software do you use to **accurately draw geometry diagrams**?

geometry big-list math-software

share cite edit follow flag asked Aug 10 '10 at 0:24 community wiki Lucky

Âishik Pyne · 3 years ago

Hi,

I'm actually towards the end of my project and for compilation I need to draw some images of LSTM. Can you tell what software do you use to create the drawings/images?

2 · Reply · Share ›

Ammar Azlan → Âishik Pyne · a year ago

Great question. Do you have the answer yet? I'd like to know too..

Reply · Share ›

Language or package for drawing geometric diagrams

Asked 9 years, 9 months ago Active 4 years, 8 months ago Viewed 6k times

21

I am looking for a language, or package in an existing language, that is good for specifying and drawing geometric diagrams.

For example, I would like to draw a hexagonal grid, with its dual triangular grid superimposed on it. Now, I could sit down, put some elbow grease in to work out the trig by hand and come up with some Postscript or SVG that will display such a grid. But I'm wondering if there are any languages or packages that will help me out with this; that make it easy to specify a hexagonal grid, find the centers, and draw a triangular grid over it.

20

What's the easiest way to do this?

Where can I find a good drawing software?

Asked 5 years, 2 months ago Active 4 years, 6 months ago Viewed 775 times

3

Maybe this is a little off-topic but often, when writing articles, I find myself in need of a good drawing software (for MAC or Windows) that would allow me to draw figures like the one below:

narendraj9 · 4 years ago

Hi! Thanks for this amazing blog post. How did you generate these diagrams?

Reply · Share ›

Open Source Software for Creating Mathematical Diagrams

Ask Question

Asked 5 years, 4 months ago Active 5 months ago Viewed 197 times

1

I work as a software engineer at a company developing navigation systems. As I have a mathematical background I normally get assigned the more mathematical problems and I find myself regularly having to produce diagrams for customers and colleagues to explain how my algorithms work.

I am looking for recommendations of good, open source, and preferably cross-platform software for creating 2- and 3-d diagrams. Any suggestions would be gratefully received.

Publication Quality Mathematics Diagrams

Asked 8 years, 11 months ago Active 8 years, 11 months ago Viewed 2k times

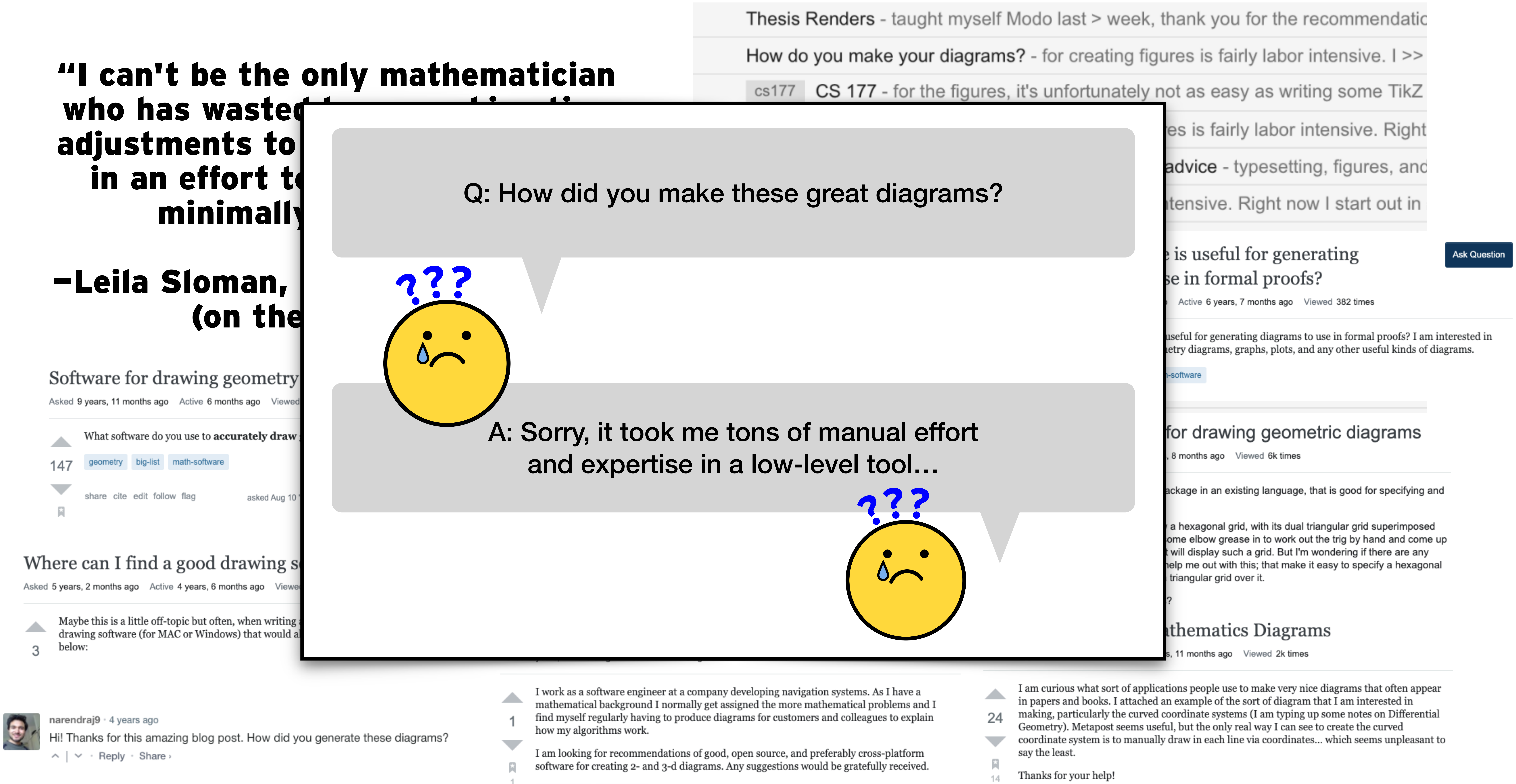
24

I am curious what sort of applications people use to make very nice diagrams that often appear in papers and books. I attached an example of the sort of diagram that I am interested in making, particularly the curved coordinate systems (I am typing up some notes on Differential Geometry). Metapost seems useful, but the only real way I can see to create the curved coordinate system is to manually draw in each line via coordinates... which seems unpleasant to say the least.

14

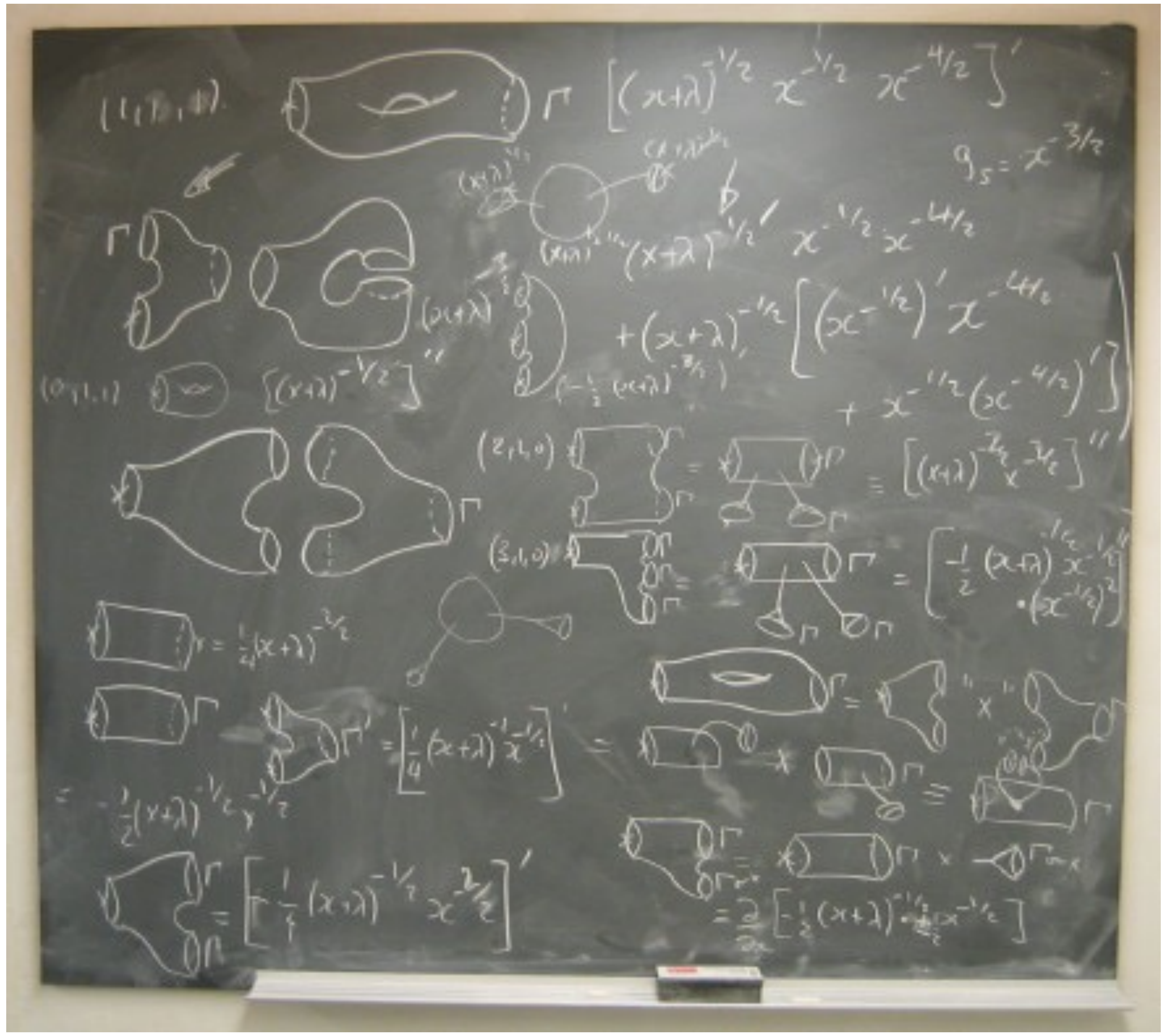
Thanks for your help!

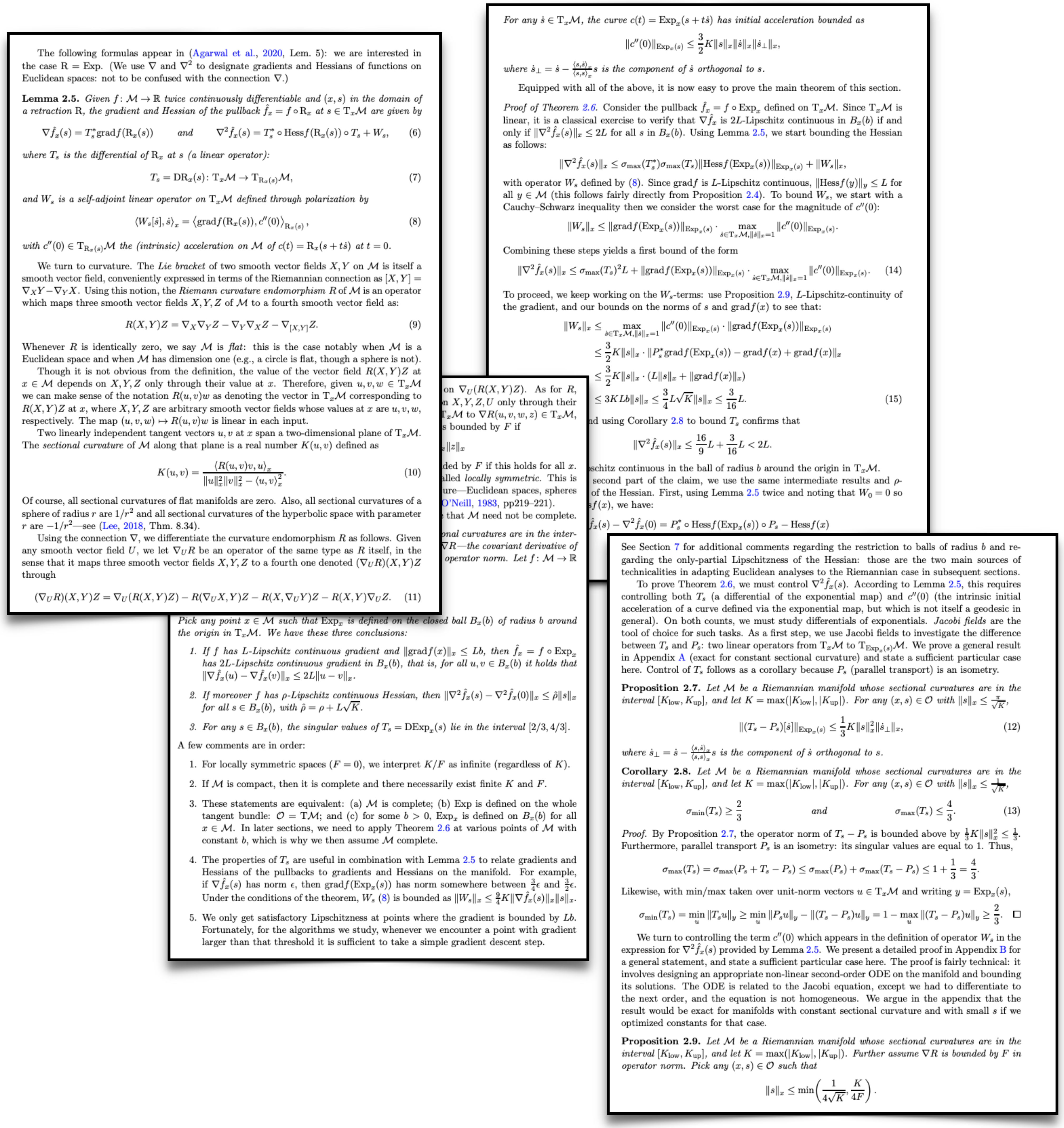
Yet, authoring **digital** diagrams is still hard



Sadly, diagrams don't have as much as impact as they could.

Sadly, diagrams don't have as much as impact as they could.





The following formulas appear in (Agarwal et al., 2020, Lem. 5): we are interested in the case $\mathrm{R} = \mathrm{Exp}$. (We use ∇ and ∇^2 to designate gradients and Hessians of functions on Euclidean spaces: not to be confused with the connection ∇.)

Lemma 2.5. *Given $f \colon \mathcal{M} \to \mathbb{R}$ twice continuously differentiable and (x, s) in the domain of a retraction R, the gradient and Hessian of the pullback $\hat{f}_x = f \circ \mathrm{R}_x$ at $s \in \mathrm{T}_x\mathcal{M}$ are given by*

$$\nabla \hat{f}_x(s) = T_s^* \mathrm{grad} f(\mathrm{R}_x(s)) \quad \text{and} \quad \nabla^2 \hat{f}_x(s) = T_s^* \circ \mathrm{Hess} f(\mathrm{R}_x(s)) \circ T_s + W_s, \tag{6}$$

where T_s is the differential of R_x at s (a linear operator):

$$T_s = \mathrm{DR}_x(s) \colon \mathrm{T}_x\mathcal{M} \to \mathrm{T}_{\mathrm{R}_x(s)}\mathcal{M}, \tag{7}$$

and W_s is a self-adjoint linear operator on $\mathrm{T}_x\mathcal{M}$ defined through polarization by

$$\langle W_s[\dot{s}], \dot{s} \rangle_x = \left\langle \mathrm{grad} f(\mathrm{R}_x(s)), c''(0) \right\rangle_{\mathrm{R}_x(s)}, \tag{8}$$

with $c''(0) \in \mathrm{T}_{\mathrm{R}_x(s)}\mathcal{M}$ the (intrinsic) acceleration on $\mathcal{M}$ of $c(t) = \mathrm{R}_x(s + t\dot{s})$ at $t = 0$.

We turn to curvature. The *Lie bracket* of two smooth vector fields X, Y on $\mathcal{M}$ is itself a smooth vector field, conveniently expressed in terms of the Riemannian connection as $[X, Y] = \nabla_X Y - \nabla_Y X$. Using this notion, the *Riemann curvature endomorphism* R of $\mathcal{M}$ is an operator which maps three smooth vector fields X, Y, Z of $\mathcal{M}$ to a fourth smooth vector field as:

$$R(X, Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z. \tag{9}$$

Whenever R is identically zero, we say $\mathcal{M}$ is *flat*: this is the case notably when $\mathcal{M}$ is a Euclidean space and when $\mathcal{M}$ has dimension one (e.g., a circle is flat, though a sphere is not).

Though it is not obvious from the definition, the value of the vector field $R(X, Y)Z$ at $x \in \mathcal{M}$ depends on X, Y, Z only through their value at x. Therefore, given $u, v, w \in \mathrm{T}_x\mathcal{M}$ we can make sense of the notation $R(u, v)w$ as denoting the vector in $\mathrm{T}_x\mathcal{M}$ corresponding to $R(X, Y)Z$ at x, where X, Y, Z are arbitrary smooth vector fields whose values at x are u, v, w, respectively. The map $(u, v, w) \mapsto R(u, v)w$ is linear in each input.

Two linearly independent tangent vectors u, v at x span a two-dimensional plane of $\mathrm{T}_x\mathcal{M}$. The *sectional curvature* of $\mathcal{M}$ along that plane is a real number $K(u, v)$ defined as

$$K(u, v) = \frac{\langle R(u, v)v, u \rangle_x}{\|u\|_x^2 \|v\|_x^2 - \langle u, v \rangle_x^2}. \tag{10}$$

Of course, all sectional curvatures of flat manifolds are zero. Also, all sectional curvatures of a sphere of radius r are $1/r^2$ and all sectional curvatures of the hyperbolic space with parameter r are $-1/r^2$—see (Lee, 2018, Thm. 8.34).

Using the connection ∇, we differentiate the curvature endomorphism R as follows. Given any smooth vector field U, we let $\nabla_U R$ be an operator of the same type as R itself, in the sense that it maps three smooth vector fields X, Y, Z to a fourth one denoted $(\nabla_U R)(X, Y)Z$ through

$$(\nabla_U R)(X, Y)Z = \nabla_U(R(X, Y)Z) - R(\nabla_U X, Y)Z - R(X, \nabla_U Y)Z - R(X, Y)\nabla_U Z. \tag{11}$$

Pick any point $x \in \mathcal{M}$ such that Exp_x is defined on the closed ball $B_x(b)$ of radius b around the origin in $\mathrm{T}_x\mathcal{M}$. We have these three conclusions:

1. *If f has L-Lipschitz continuous gradient and $\|\mathrm{grad} f(x)\|_x \leq Lb$, then $\hat{f}_x = f \circ \mathrm{Exp}_x$ has $2L$-Lipschitz continuous gradient in $B_x(b)$, that is, for all $u, v \in B_x(b)$ it holds that $\|\nabla \hat{f}_x(u) - \nabla \hat{f}_x(v)\|_x \leq 2L\|u - v\|_x$.*
2. *If moreover f has ρ-Lipschitz continuous Hessian, then $\|\nabla^2 \hat{f}_x(s) - \nabla^2 \hat{f}_x(0)\|_x \leq \hat{\rho}\|s\|_x$ for all $s \in B_x(b)$, with $\hat{\rho} = \rho + L\sqrt{K}$.*
3. *For any $s \in B_x(b)$, the singular values of $T_s = \mathrm{DExp}_x(s)$ lie in the interval $[2/3, 4/3]$.*

A few comments are in order:

1. For locally symmetric spaces ($F = 0$), we interpret K/F as infinite (regardless of K).
2. If $\mathcal{M}$ is compact, then it is complete and there necessarily exist finite K and F.
3. These statements are equivalent: (a) $\mathcal{M}$ is complete; (b) Exp is defined on the whole tangent bundle: $\mathcal{O} = \mathrm{T}\mathcal{M}$; and (c) for some $b > 0$, Exp_x is defined on $B_x(b)$ for all $x \in \mathcal{M}$. In later sections, we need to apply Theorem 2.6 at various points of $\mathcal{M}$ with constant b, which is why we then assume $\mathcal{M}$ complete.
4. The properties of T_s are useful in combination with Lemma 2.5 to relate gradients and Hessians of the pullbacks to gradients and Hessians on the manifold. For example, if $\nabla \hat{f}_x(s)$ has norm ϵ, then $\mathrm{grad} f(\mathrm{Exp}_x(s))$ has norm somewhere between $\frac{3}{4}\epsilon$ and $\frac{3}{2}\epsilon$. Under the conditions of the theorem, W_s (8) is bounded as $\|W_s\|_x \leq \frac{9}{4}K\|\nabla \hat{f}_x(s)\|_x\|s\|_x$.
5. We only get satisfactory Lipschitzness at points where the gradient is bounded by Lb. Fortunately, for the algorithms we study, whenever we encounter a point with gradient larger than that threshold it is sufficient to take a simple gradient descent step.

For any $\dot{s} \in \mathrm{T}_x\mathcal{M}$, the curve $c(t) = \mathrm{Exp}_x(s + t\dot{s})$ has initial acceleration bounded as

$$\|c''(0)\|_{\mathrm{Exp}_x(s)} \leq \frac{3}{2}K\|s\|_x\|\dot{s}\|_x\|\dot{s}_\perp\|_x,$$

where $\dot{s}_\perp = \dot{s} - \frac{\langle s, \dot{s} \rangle_x}{\langle s, s \rangle_x}s$ is the component of $\dot{s}$ orthogonal to s.

Equipped with all of the above, it is now easy to prove the main theorem of this section.

Proof of Theorem 2.6. Consider the pullback $\hat{f}_x = f \circ \mathrm{Exp}_x$ defined on $\mathrm{T}_x\mathcal{M}$. Since $\mathrm{T}_x\mathcal{M}$ is linear, it is a classical exercise to verify that $\nabla \hat{f}_x$ is $2L$-Lipschitz continuous in $B_x(b)$ if and only if $\|\nabla^2 \hat{f}_x(s)\|_x \leq 2L$ for all s in $B_x(b)$. Using Lemma 2.5, we start bounding the Hessian as follows:

$$\|\nabla^2 \hat{f}_x(s)\|_x \leq \sigma_{\max}(T_s^*)\sigma_{\max}(T_s)\|\mathrm{Hess} f(\mathrm{Exp}_x(s))\|_{\mathrm{Exp}_x(s)} + \|W_s\|_x,$$

with operator W_s defined by (8). Since $\mathrm{grad} f$ is L-Lipschitz continuous, $\|\mathrm{Hess} f(y)\|_y \leq L$ for all $y \in \mathcal{M}$ (this follows fairly directly from Proposition 2.4). To bound W_s, we start with a Cauchy–Schwarz inequality then we consider the worst case for the magnitude of $c''(0)$:

$$\|W_s\|_x \leq \|\mathrm{grad} f(\mathrm{Exp}_x(s))\|_{\mathrm{Exp}_x(s)} \cdot \max_{\dot{s} \in \mathrm{T}_x\mathcal{M}, \|\dot{s}\|_x = 1} \|c''(0)\|_{\mathrm{Exp}_x(s)}.$$

Combining these steps yields a first bound of the form

$$\|\nabla^2 \hat{f}_x(s)\|_x \leq \sigma_{\max}(T_s)^2 L + \|\mathrm{grad} f(\mathrm{Exp}_x(s))\|_{\mathrm{Exp}_x(s)} \cdot \max_{\dot{s} \in \mathrm{T}_x\mathcal{M}, \|\dot{s}\|_x = 1} \|c''(0)\|_{\mathrm{Exp}_x(s)}. \tag{14}$$

To proceed, we keep working on the W_s-terms: use Proposition 2.9, L-Lipschitz-continuity of the gradient, and our bounds on the norms of s and $\mathrm{grad} f(x)$ to see that:

$$\begin{aligned} \|W_s\|_x &\leq \max_{\dot{s} \in \mathrm{T}_x\mathcal{M}, \|\dot{s}\|_x = 1} \|c''(0)\|_{\mathrm{Exp}_x(s)} \cdot \|\mathrm{grad} f(\mathrm{Exp}_x(s))\|_{\mathrm{Exp}_x(s)} \\ &\leq \frac{3}{2}K\|s\|_x \cdot \|P_s^* \mathrm{grad} f(\mathrm{Exp}_x(s)) - \mathrm{grad} f(x) + \mathrm{grad} f(x)\|_x \\ &\leq \frac{3}{2}K\|s\|_x \cdot (L\|s\|_x + \|\mathrm{grad} f(x)\|_x) \\ &\leq 3KLb\|s\|_x \leq \frac{3}{4}L\sqrt{K}\|s\|_x \leq \frac{3}{16}L. \end{aligned} \tag{15}$$

nd using Corollary 2.8 to bound T_s confirms that

$$\|\nabla^2 \hat{f}_x(s)\|_x \leq \frac{16}{9}L + \frac{3}{16}L < 2L.$$

See Section 7 for additional comments regarding the restriction to balls of radius b and regarding the only-partial Lipschitzness of the Hessian: those are the two main sources of technicalities in adapting Euclidean analyses to the Riemannian case in subsequent sections.

To prove Theorem 2.6, we must control $\nabla^2 \hat{f}_x(s)$. According to Lemma 2.5, this requires controlling both T_s (a differential of the exponential map) and $c''(0)$ (the intrinsic initial acceleration of a curve defined via the exponential map, but which is not itself a geodesic in general). On both counts, we must study differentials of exponentials. *Jacobi fields* are the tool of choice for such tasks. As a first step, we use Jacobi fields to investigate the difference between T_s and P_s: two linear operators from $\mathrm{T}_x\mathcal{M}$ to $\mathrm{T}_{\mathrm{Exp}_x(s)}\mathcal{M}$. We prove a general result in Appendix A (exact for constant sectional curvature) and state a sufficient particular case here. Control of T_s follows as a corollary because P_s (parallel transport) is an isometry.

Proposition 2.7. *Let $\mathcal{M}$ be a Riemannian manifold whose sectional curvatures are in the interval $[K_{\mathrm{low}}, K_{\mathrm{up}}]$, and let $K = \max(|K_{\mathrm{low}}|, |K_{\mathrm{up}}|)$. For any $(x, s) \in \mathcal{O}$ with $\|s\|_x \leq \frac{\pi}{\sqrt{K}}$,*

$$\|(T_s - P_s)[\dot{s}]\|_{\mathrm{Exp}_x(s)} \leq \frac{1}{3}K\|s\|_x^2\|\dot{s}_\perp\|_x, \tag{12}$$

where $\dot{s}_\perp = \dot{s} - \frac{\langle s, \dot{s} \rangle_x}{\langle s, s \rangle_x}s$ is the component of $\dot{s}$ orthogonal to s.

Corollary 2.8. *Let $\mathcal{M}$ be a Riemannian manifold whose sectional curvatures are in the interval $[K_{\mathrm{low}}, K_{\mathrm{up}}]$, and let $K = \max(|K_{\mathrm{low}}|, |K_{\mathrm{up}}|)$. For any $(x, s) \in \mathcal{O}$ with $\|s\|_x \leq \frac{1}{\sqrt{K}}$,*

$$\sigma_{\min}(T_s) \geq \frac{2}{3} \quad \text{and} \quad \sigma_{\max}(T_s) \leq \frac{4}{3}. \tag{13}$$

Proof. By Proposition 2.7, the operator norm of $T_s - P_s$ is bounded above by $\frac{1}{3}K\|s\|_x^2 \leq \frac{1}{3}$. Furthermore, parallel transport P_s is an isometry: its singular values are equal to 1. Thus,

$$\sigma_{\max}(T_s) = \sigma_{\max}(P_s + T_s - P_s) \leq \sigma_{\max}(P_s) + \sigma_{\max}(T_s - P_s) \leq 1 + \frac{1}{3} = \frac{4}{3}.$$

Likewise, with min/max taken over unit-norm vectors $u \in \mathrm{T}_x\mathcal{M}$ and writing $y = \mathrm{Exp}_x(s)$,

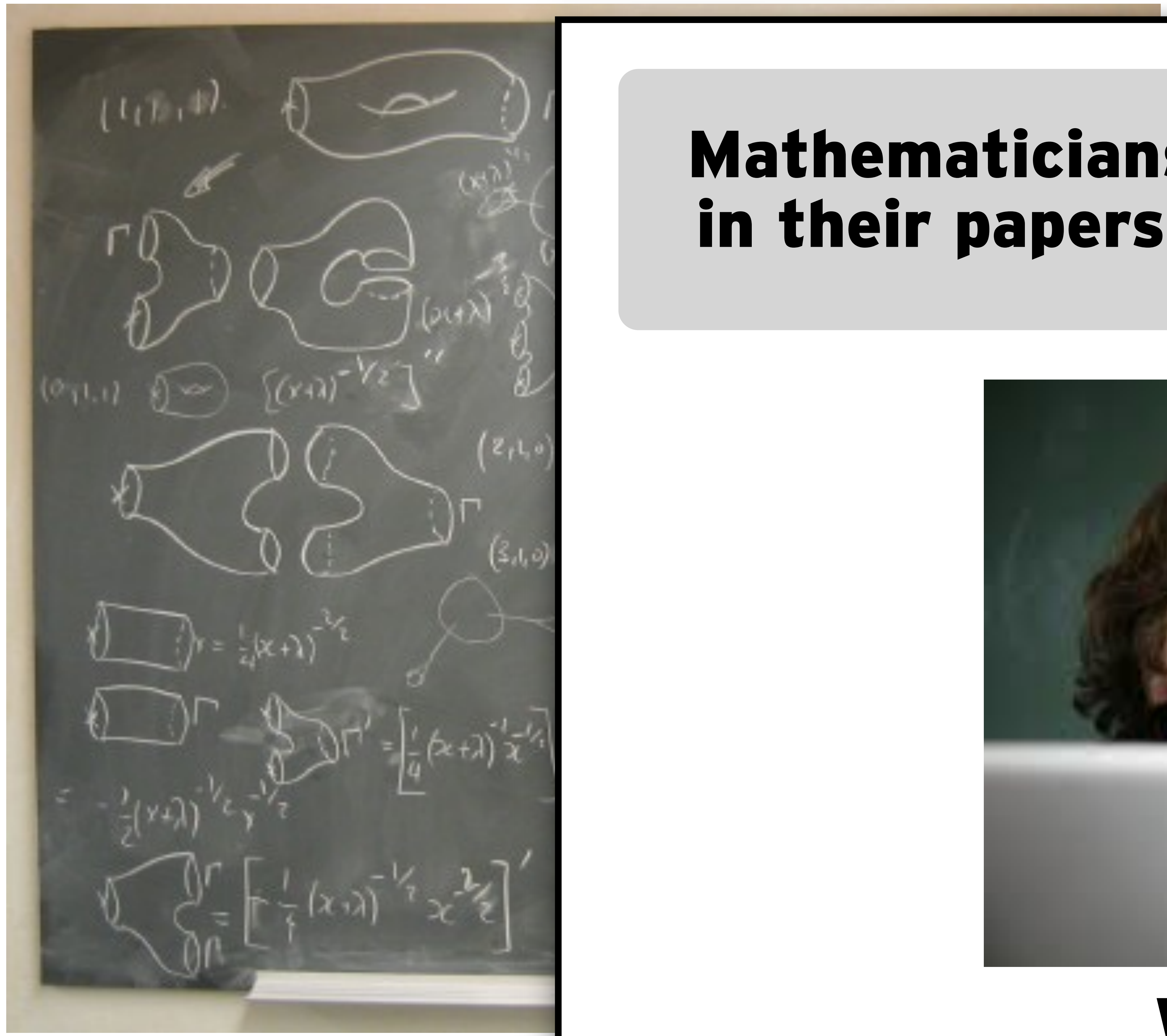
$$\sigma_{\min}(T_s) = \min_u \|T_s u\|_y \geq \min_u \|P_s u\|_y - \|(T_s - P_s)u\|_y = 1 - \max_u \|(T_s - P_s)u\|_y \geq \frac{2}{3}. \quad \square$$

We turn to controlling the term $c''(0)$ which appears in the definition of operator W_s in the expression for $\nabla^2 \hat{f}_x(s)$ provided by Lemma 2.5. We present a detailed proof in Appendix B for a general statement, and state a sufficient particular case here. The proof is fairly technical: it involves designing an appropriate non-linear second-order ODE on the manifold and bounding its solutions. The ODE is related to the Jacobi equation, except we had to differentiate to the next order, and the equation is not homogeneous. We argue in the appendix that the result would be exact for manifolds with constant sectional curvature and with small s if we optimized constants for that case.

Proposition 2.9. *Let $\mathcal{M}$ be a Riemannian manifold whose sectional curvatures are in the interval $[K_{\mathrm{low}}, K_{\mathrm{up}}]$, and let $K = \max(|K_{\mathrm{low}}|, |K_{\mathrm{up}}|)$. Further assume ∇R is bounded by F in operator norm. Pick any $(x, s) \in \mathcal{O}$ such that*

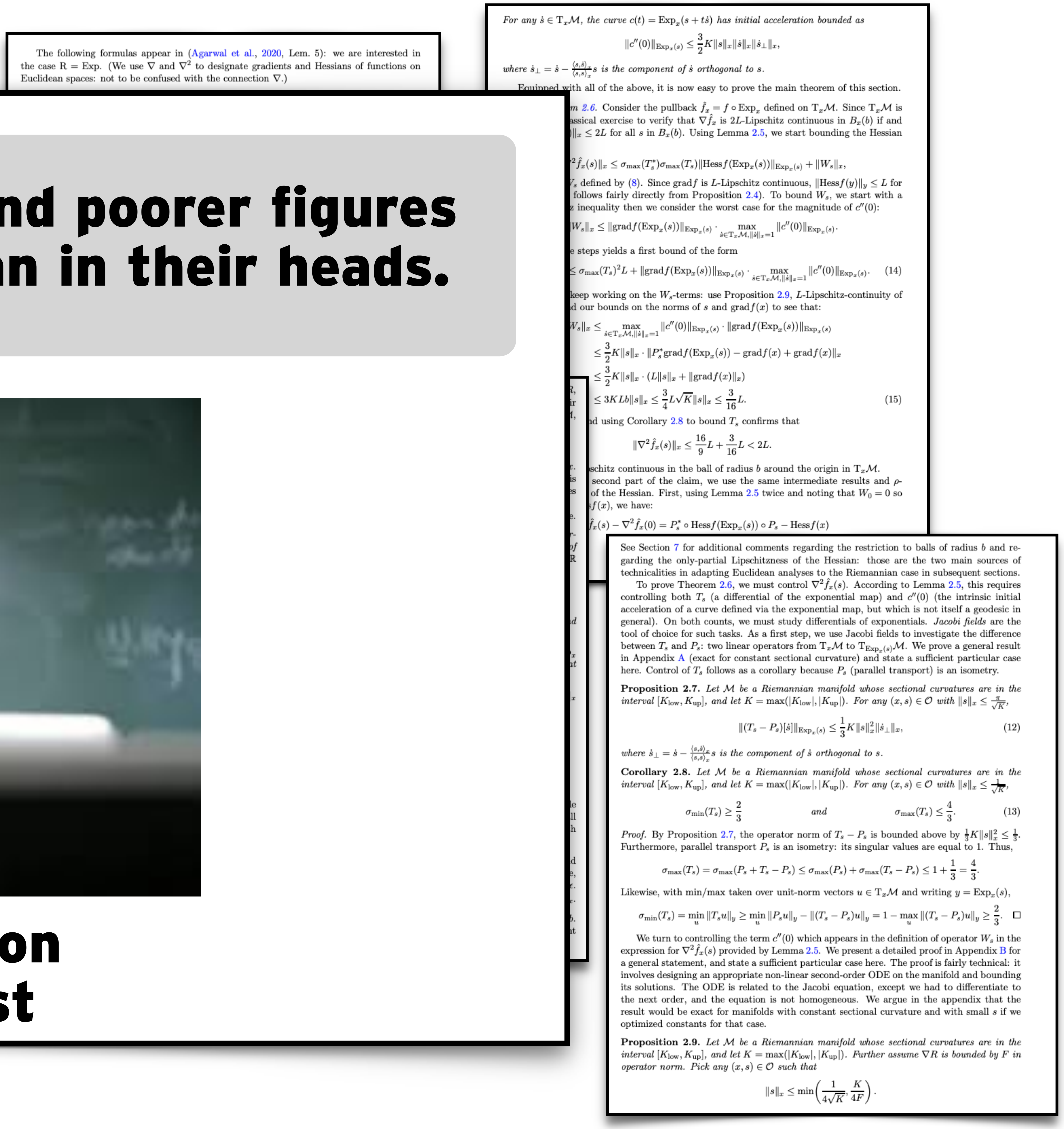
$$\|s\|_x \leq \min\left(\frac{1}{4\sqrt{K}}, \frac{K}{4F}\right).$$

Sadly, diagrams don't have as much as impact as they could.



Mathematicians have fewer and poorer figures in their papers and books than in their heads.

William Thurston
Fields medalist



Why is mathematical diagramming hard?

Must solve **two** problems:

Why is mathematical diagramming hard?

Must solve **two** problems:

mathematical idea

↓

visual representation

Why is mathematical diagramming hard?

Must solve **two** problems:

mathematical idea

visual representation

high-quality image

Why is mathematical diagramming hard?

Must solve **two** problems:

mathematical idea

$\downarrow$

visual representation

$\downarrow$

high-quality image

Most digital illustration tools just consider the second part

It's still (very) hard to turn abstract logical ideas into a visual representation!

3.1 THE NOTION OF A TOPOLOGY

The fundamental properties of open subsets of a metric space are outlined in Proposition 2 of Chapter 2. Mathematicians have found from experience that families of subsets having these same properties arise in contexts other than those of metric spaces; hence it is reasonable to study these properties in their own right, abstracted from the limitations that metric spaces impose. In particular, the properties of open sets in metric spaces inspire the following definition.

Definition 1. Let X be any set. A collection τ of subsets of X is said to be a *topology* on X if the following axioms are satisfied:

i) X and ϕ are members of τ.

ii) The intersection of any two members of τ is a member of τ.

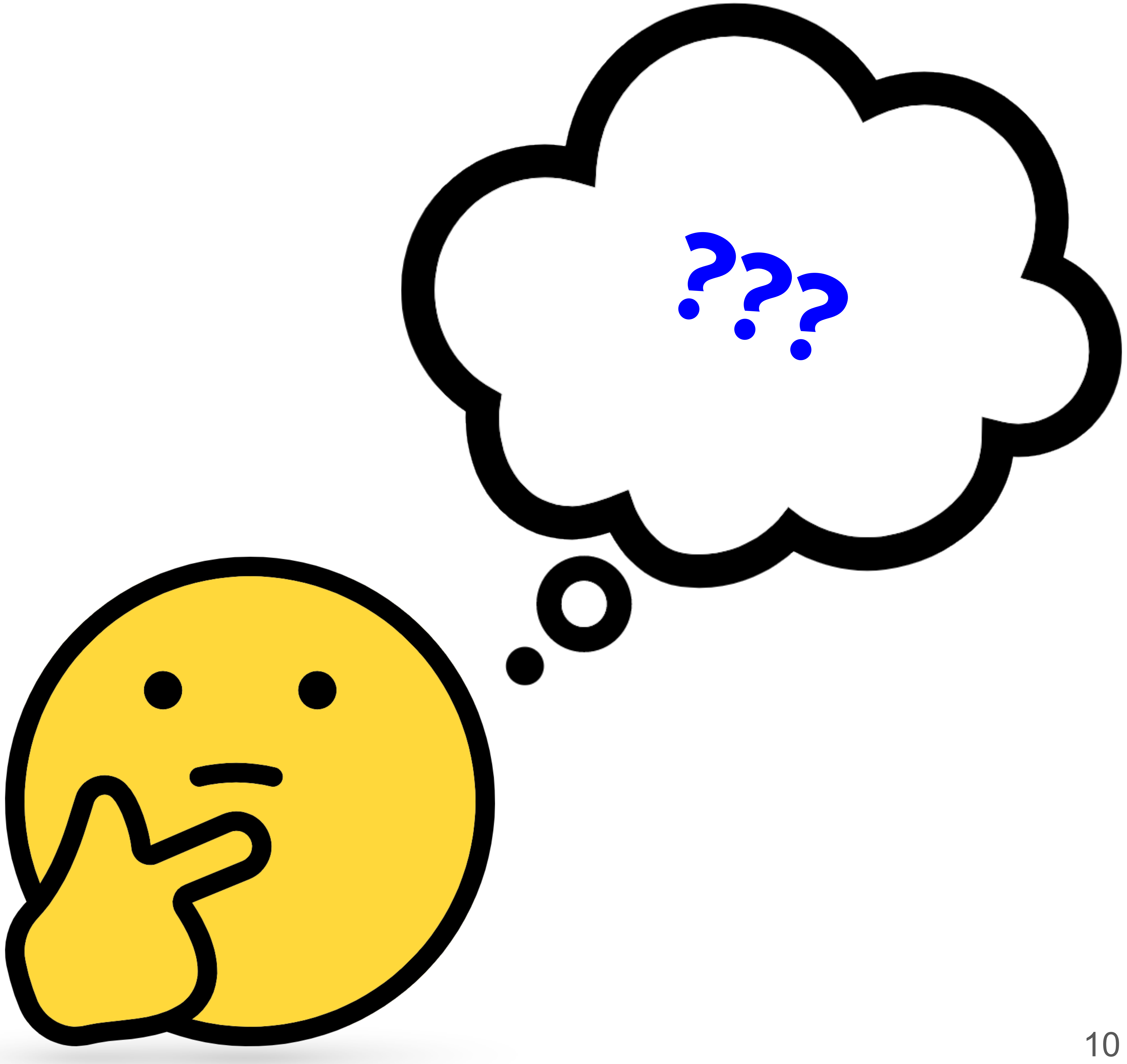
iii) The union of any family of members of τ is again in τ.

The members of τ are then said to be *τ-open* subsets of X, or merely *open* subsets of X if no confusion may result.

Example 1. If X, D is a metric space, then the D-open subsets of X form a topology on X. This topology is called the *metric topology* induced on X by D. It was, of course, this topology that we studied in Chapter 2.

Example 2. Let X be any set. Then the family of all subsets of X forms a topology on X. This topology consisting of all of the subsets of X is called the *discrete topology* on X. The discrete topology contains the maximum possible number of open sets since, relative to the discrete topology, every subset of X is open.

Example 3. If X is any set, then the collection $\{X, \phi\}$ of subsets of X also forms a topology on X. This topology is called the *trivial* (by some, the



It's still (very) hard to turn abstract logical ideas into a visual representation!

3.1 THE NOTION OF A TOPOLOGY

The fundamental properties of open subsets of a metric space are outlined in Proposition 2 of Chapter 2. Mathematicians have found from experience that families of subsets having these same properties arise in contexts other than those of metric spaces; hence it is reasonable to study these properties in their own right, abstracted from the limitations that metric spaces impose. In particular, the properties of open sets in metric spaces inspire the following definition.

Definition 1. Let X be any set. A collection τ of subsets of X is said to be a *topology* on X if the following axioms are satisfied:

i) X and ϕ are members of τ.

ii) The intersection of any two members of τ is a member of τ.

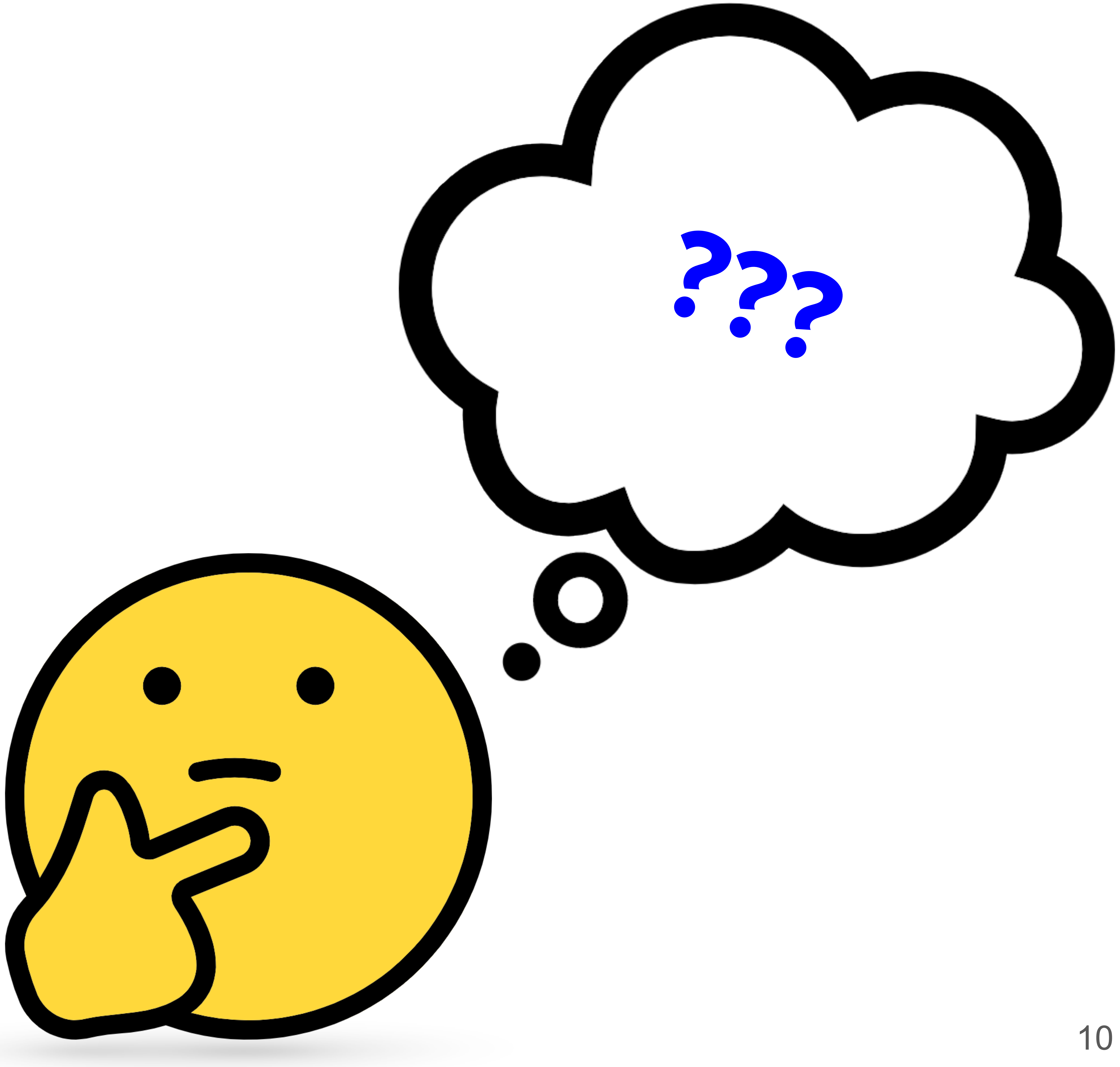
iii) The union of any family of members of τ is again in τ.

The members of τ are then said to be *τ-open* subsets of X, or merely *open* subsets of X if no confusion may result.

Example 1. If X, D is a metric space, then the D-open subsets of X form a topology on X. This topology is called the *metric topology* induced on X by D. It was, of course, this topology that we studied in Chapter 2.

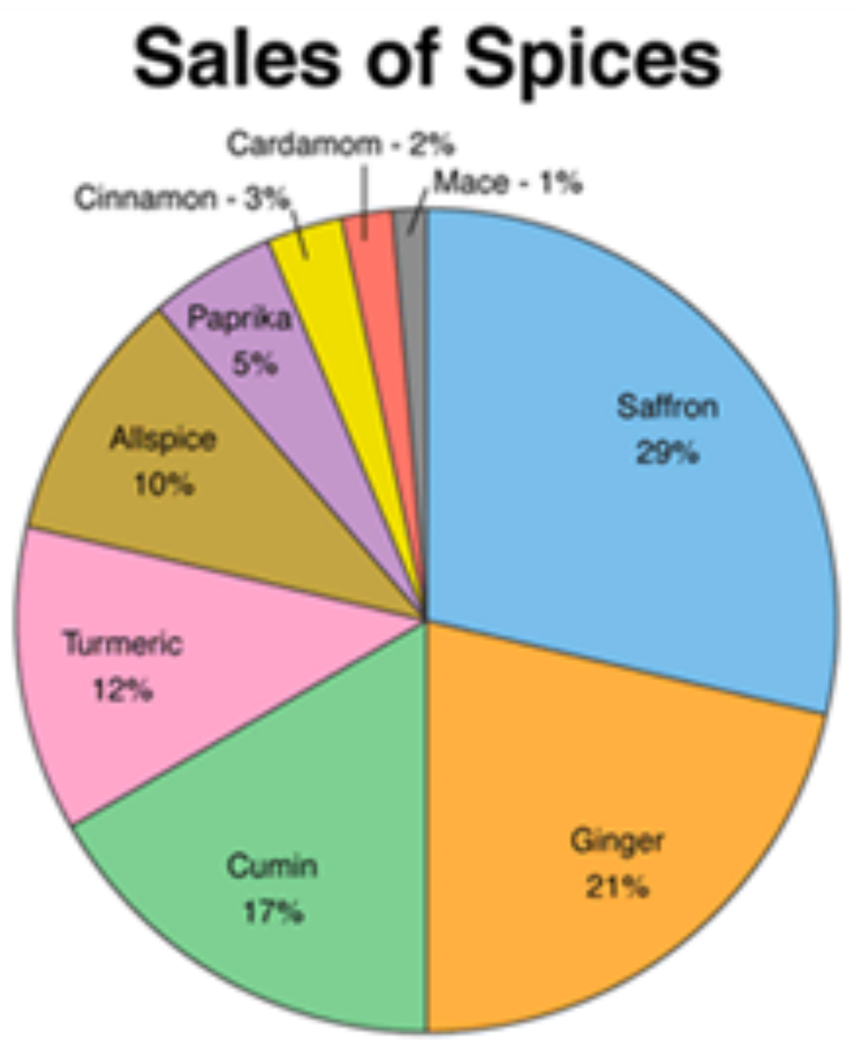
Example 2. Let X be any set. Then the family of all subsets of X forms a topology on X. This topology consisting of all of the subsets of X is called the *discrete topology* on X. The discrete topology contains the maximum possible number of open sets since, relative to the discrete topology, every subset of X is open.

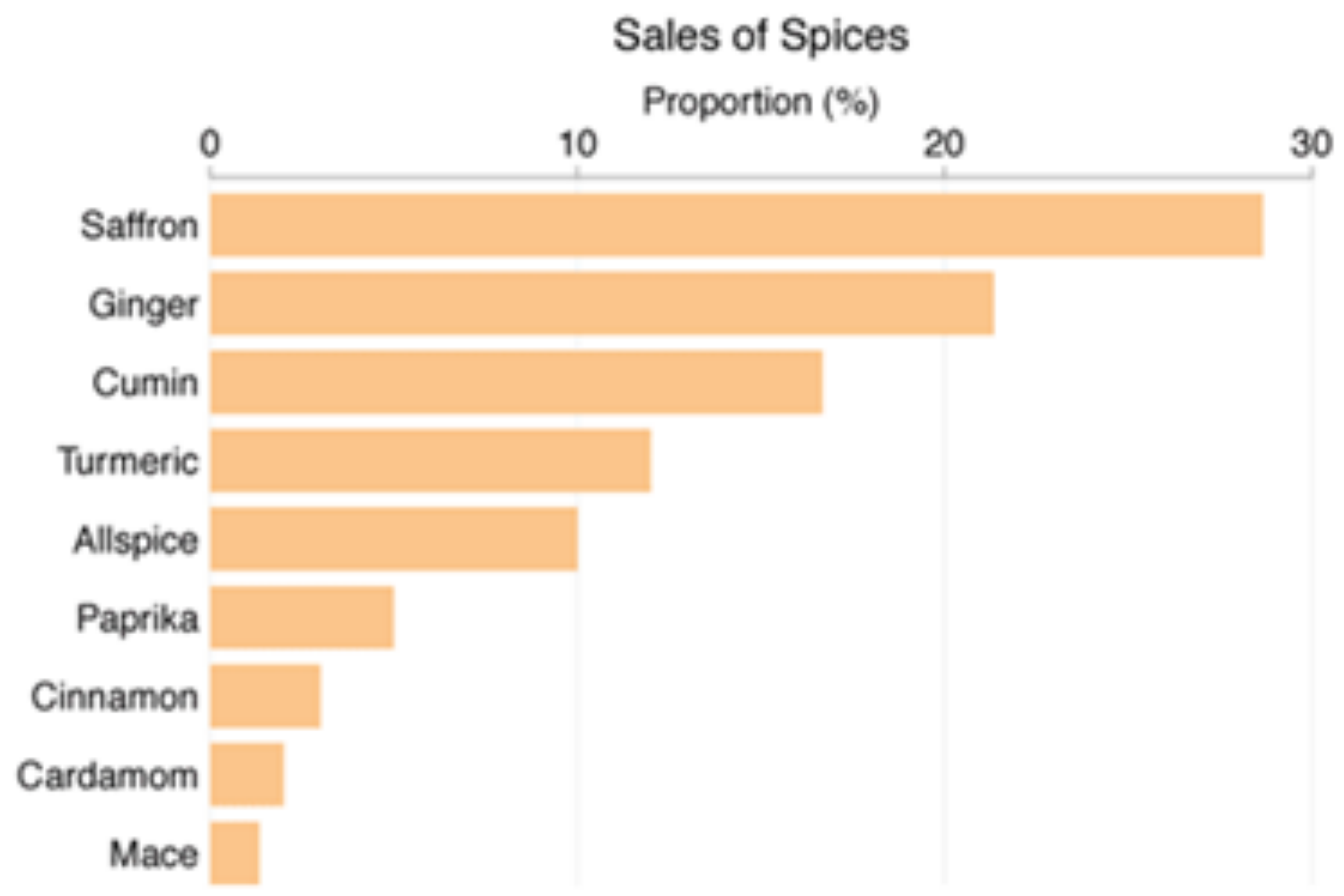
Example 3. If X is any set, then the collection $\{X, \phi\}$ of subsets of X also forms a topology on X. This topology is called the *trivial* (by some, the



However, we can learn a lot from data visualization…

```
Saffron 58
Ginger 42
Cumin 34
Turmeric 24
Allspice 2
Paprika 1
Cinnamon 0.6
Cardamom 0.4
Mace 0.2
```

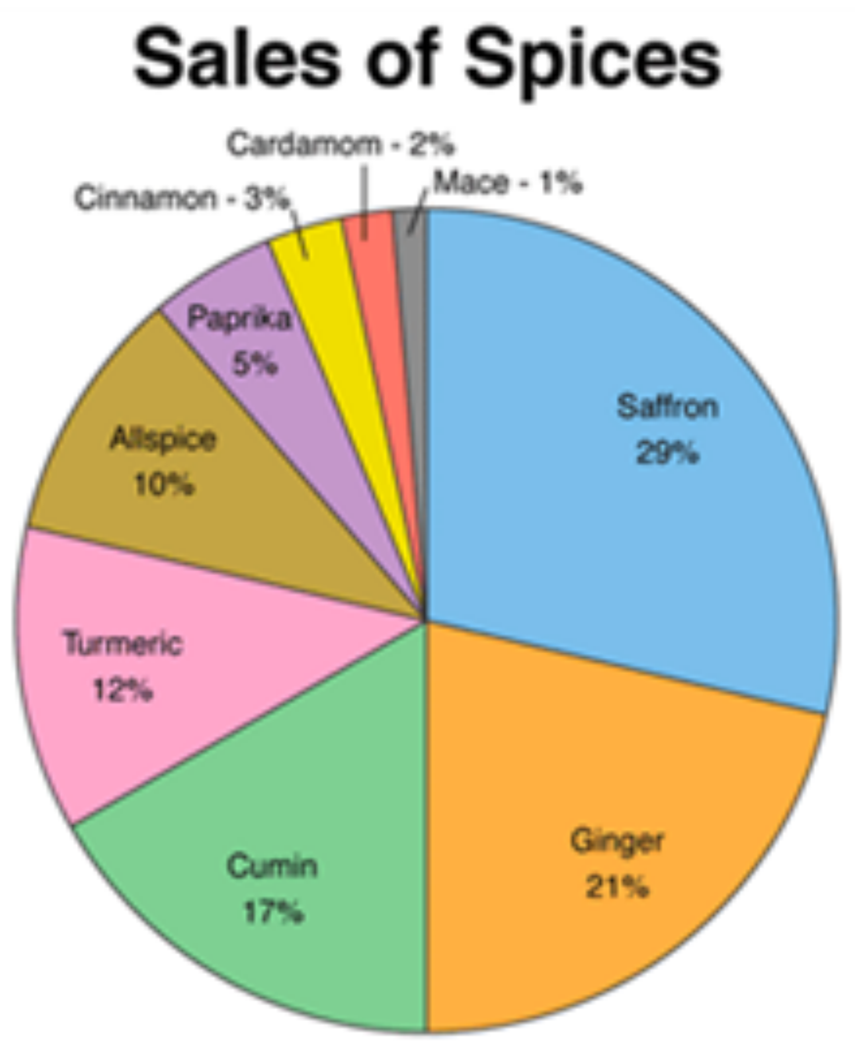


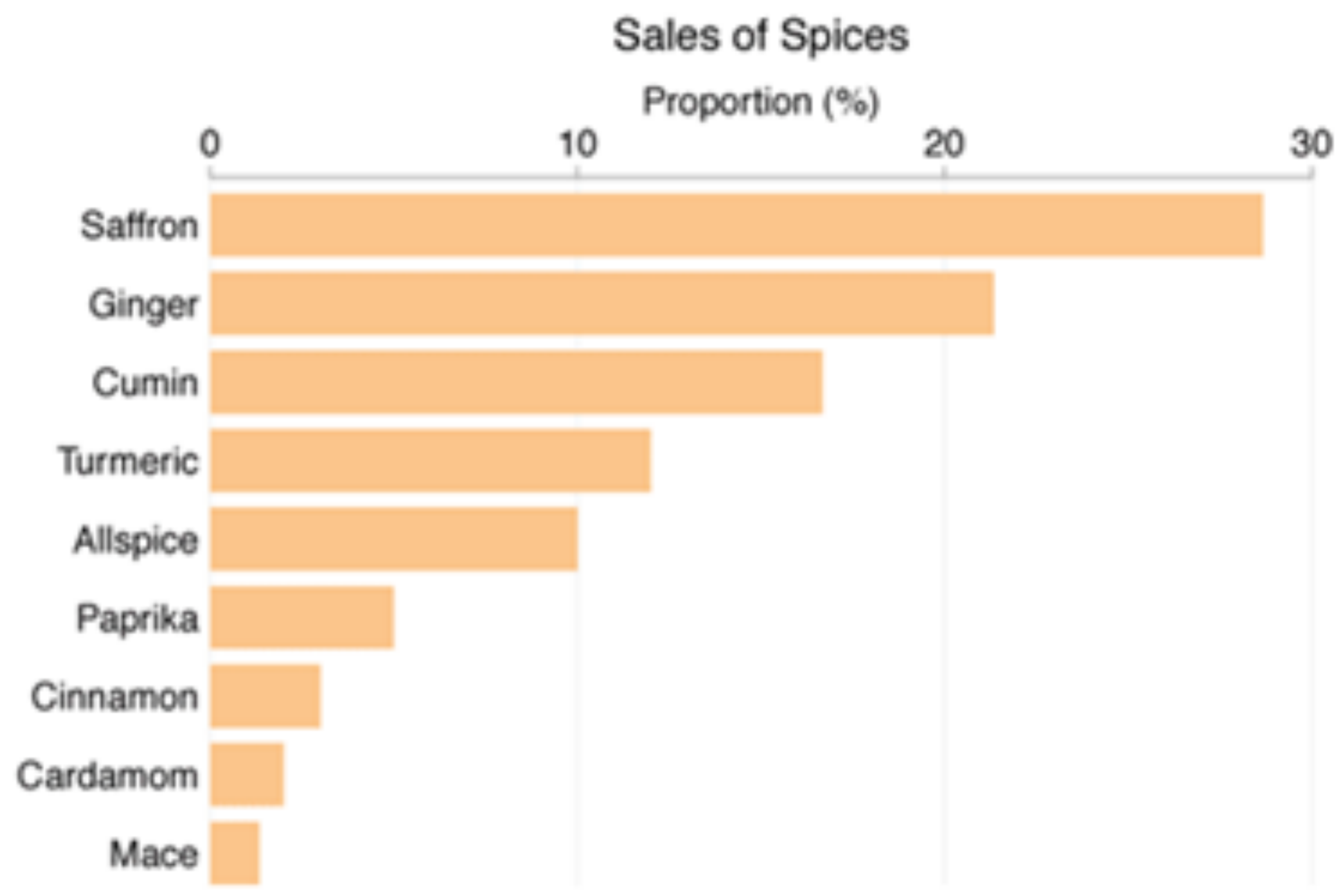


Data

However, we can learn a lot from data visualization...

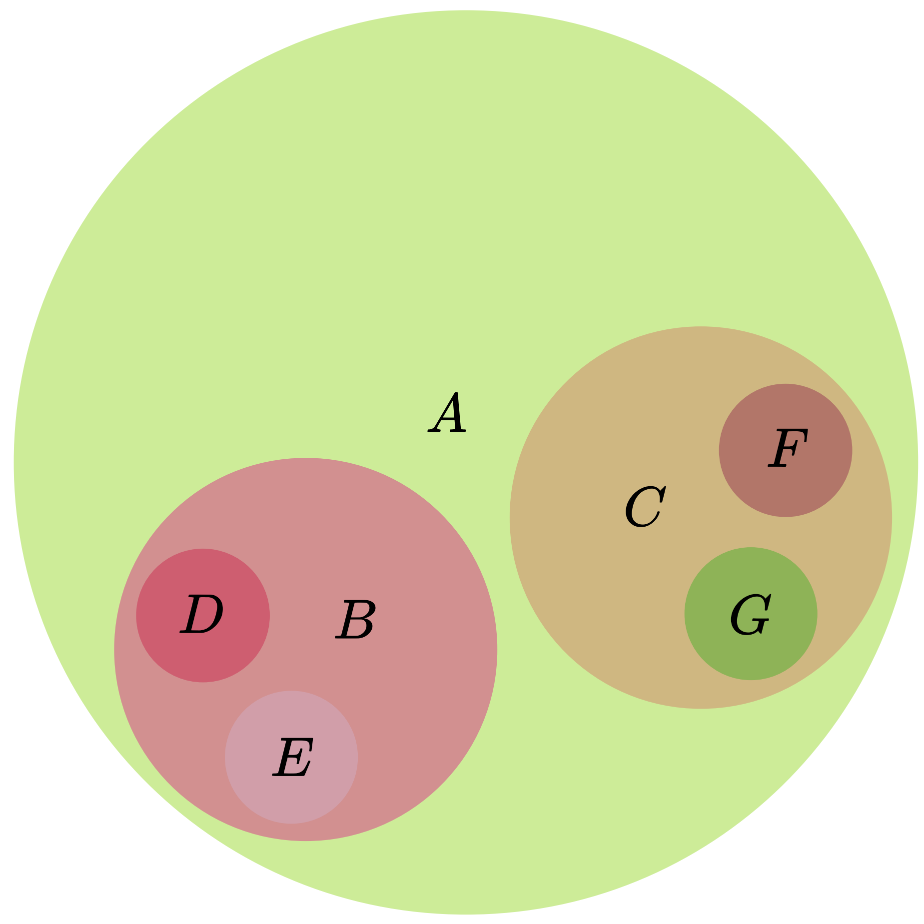
```
Saffron 58
Ginger 42
Cumin 34
Turmeric 24
Allspice 2
Paprika 1
Cinnamon 0.6
Cardamom 0.4
Mace 0.2
```

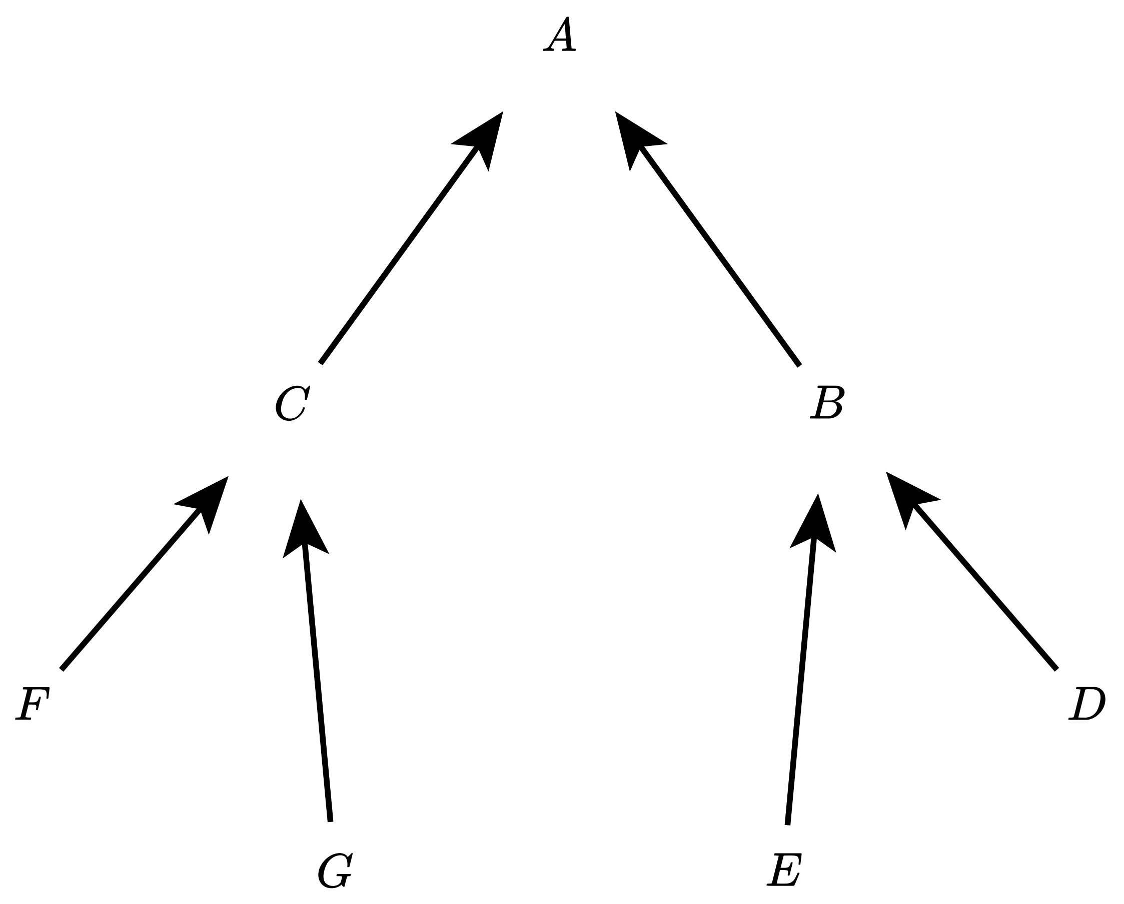




Data

```
Set A, B, C, D, E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

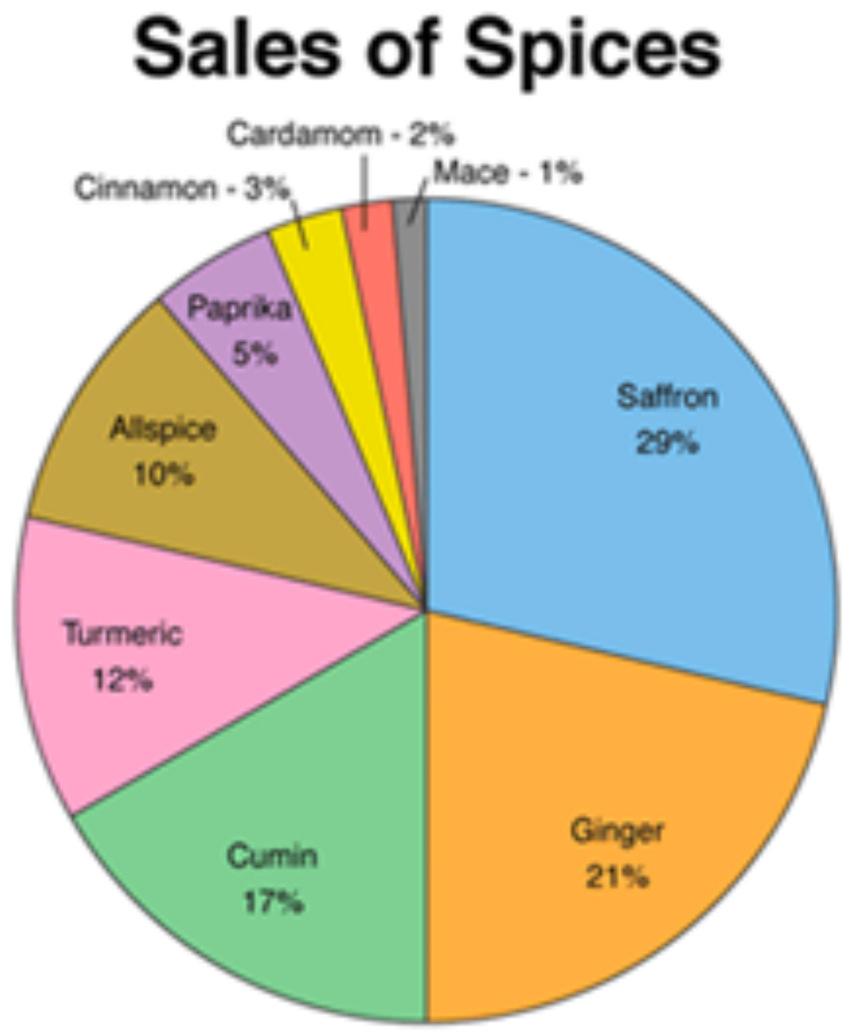


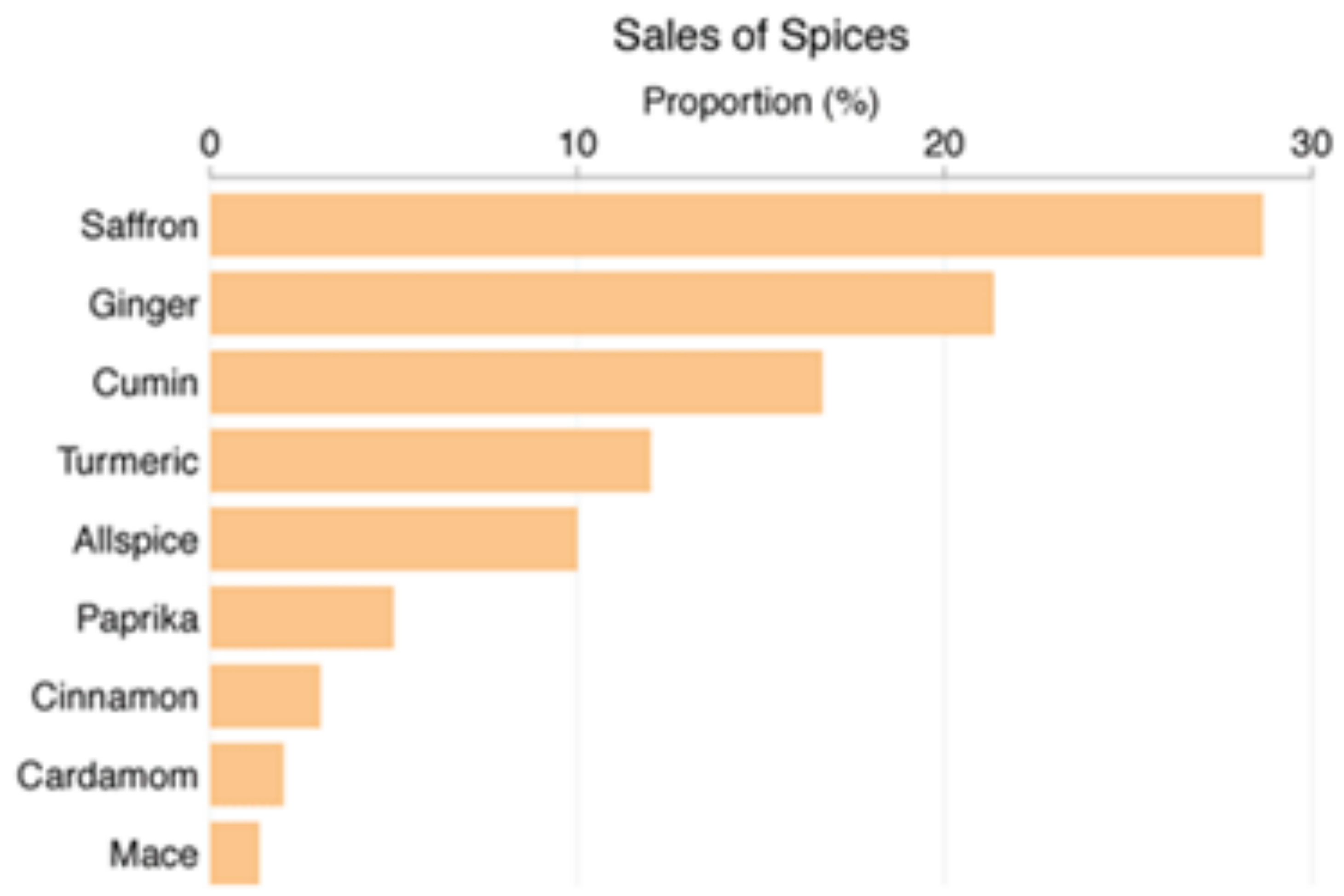


A new kind of data

However, we can learn a lot from data visualization...

```
Saffron 58
Ginger 42
Cumin 34
Turmeric 24
Allspice 2
Paprika 1
Cinnamon 0.6
Cardamom 0.4
Mace 0.2
```

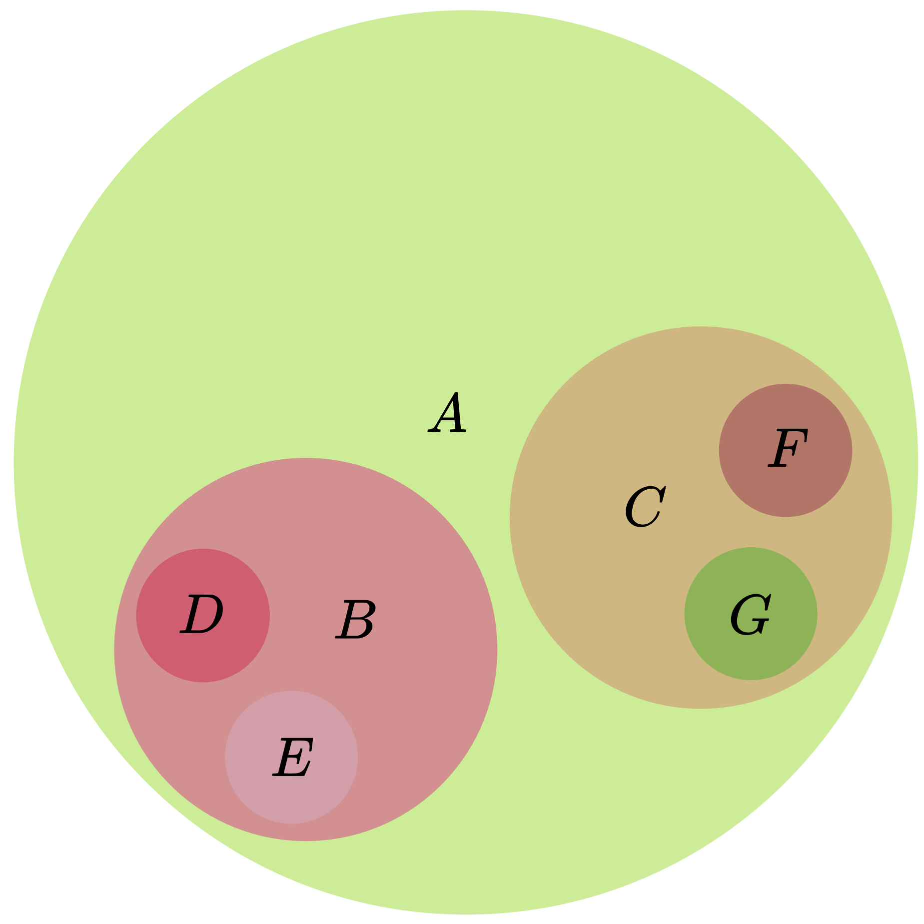


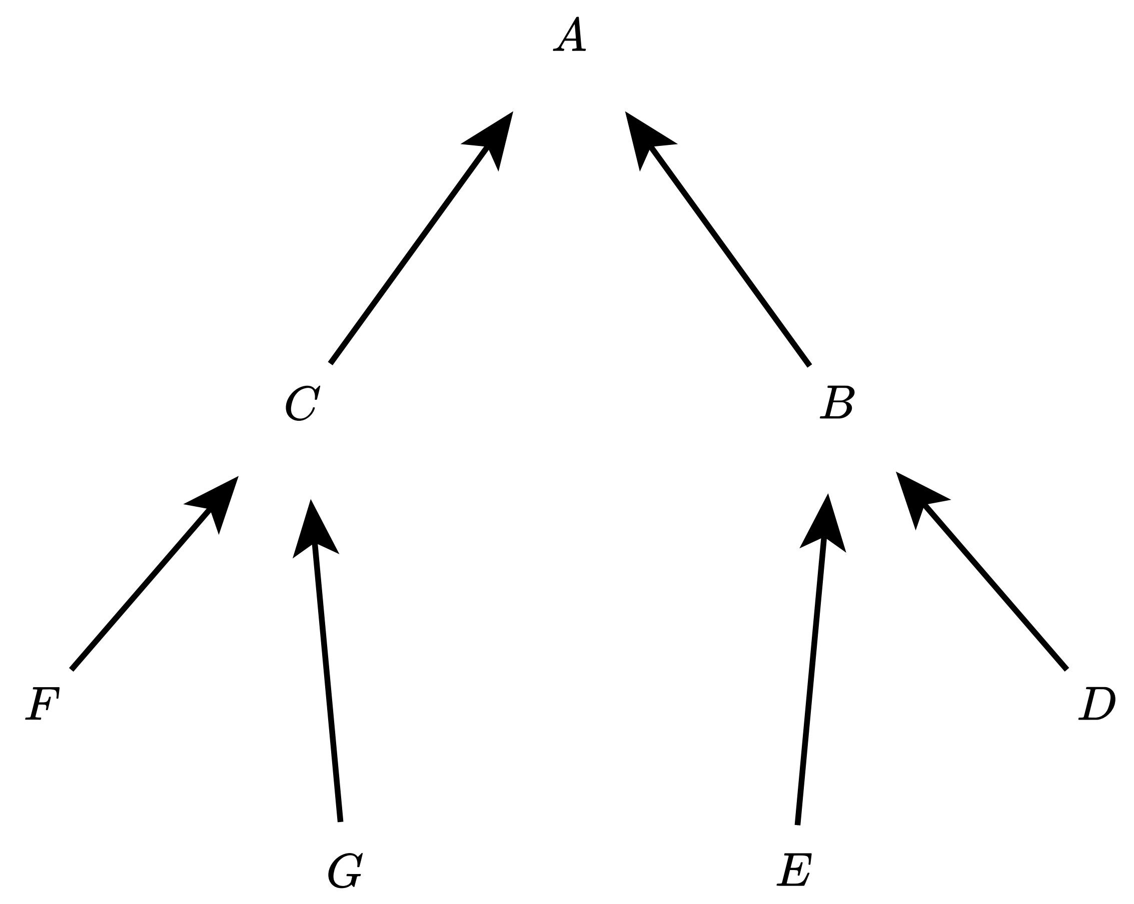


Data

```
Set A, B, C, D, E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

Penrose





A new kind of data

Organizing principles

Organizing principles

Organizing principles

- (Specification) A diagram is a **mapping** from abstract mathematical statements to a concrete visual representation.

Organizing principles

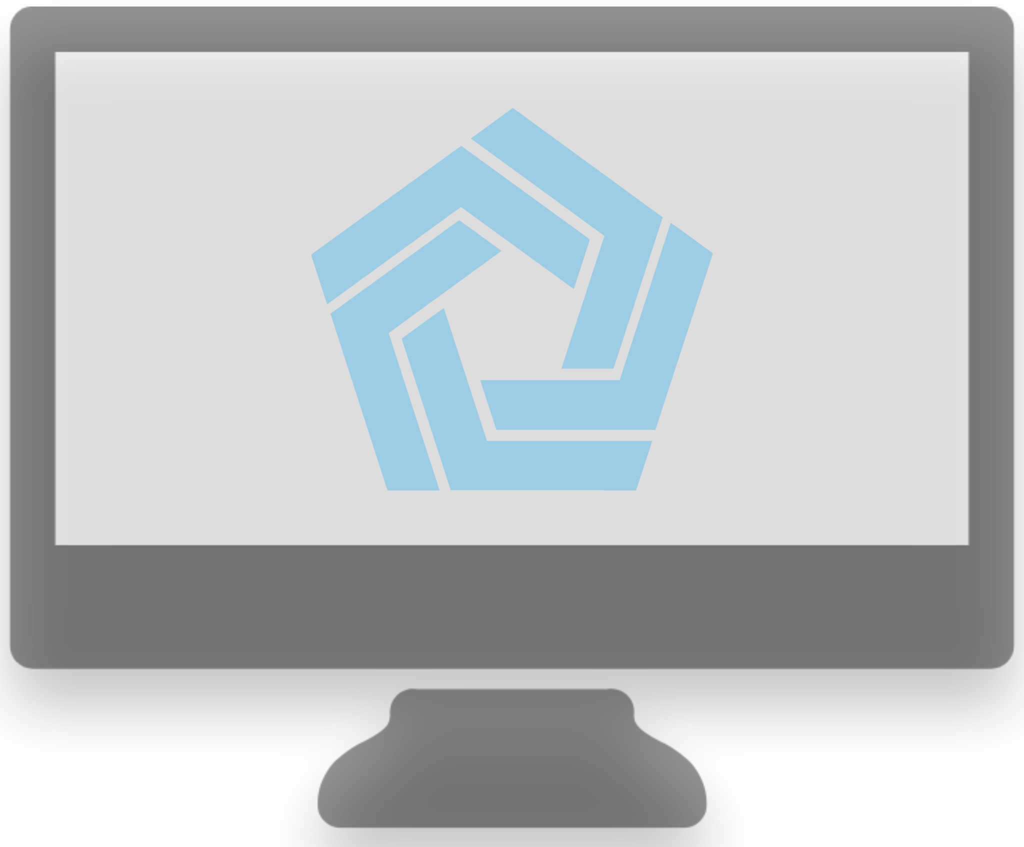
- (Specification) A diagram is a **mapping** from abstract mathematical statements to a concrete visual representation.
- (Synthesis) Given a specification, specific diagrams are realized via numerical **optimization**.

Organizing principles

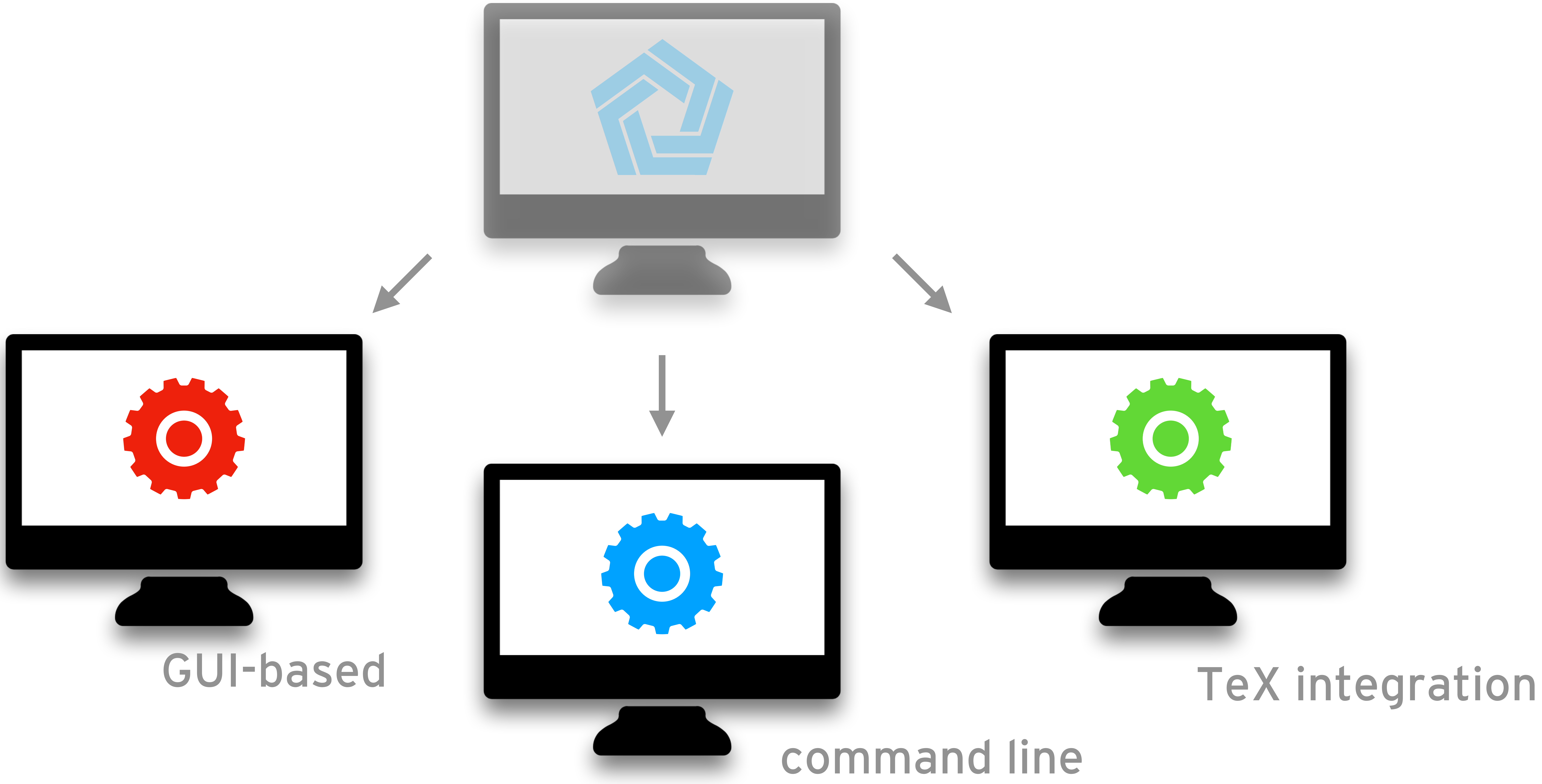
- (Specification) A diagram is a **mapping** from abstract mathematical statements to a concrete visual representation.
- (Synthesis) Given a specification, specific diagrams are realized via numerical **optimization**.

Specification is encoded via domain-specific languages

Provide a platform for building diagramming tools.



Provide a platform for building diagramming tools.



Benefits

Benefits

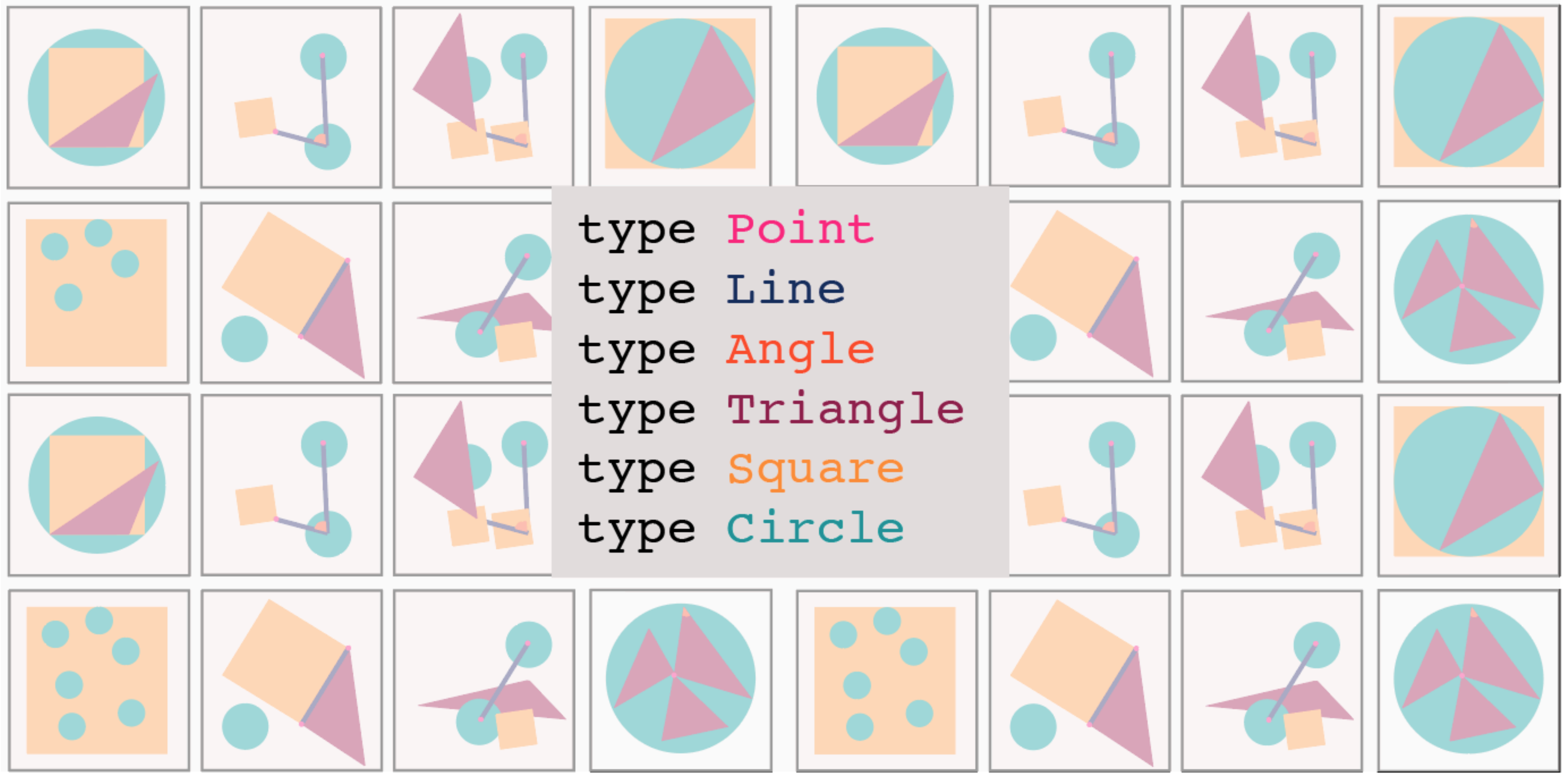
- Greater accessibility to novices

Benefits

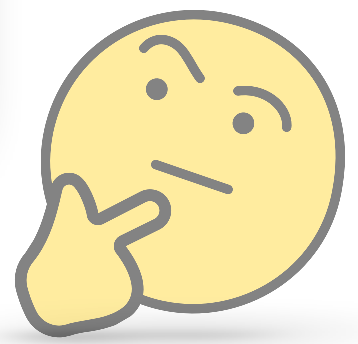
- Greater accessibility to novices
- Separation of content and presentation

Benefits

- Greater accessibility to novices
- Separation of content and presentation
- Large-scale generation and evolution of collections

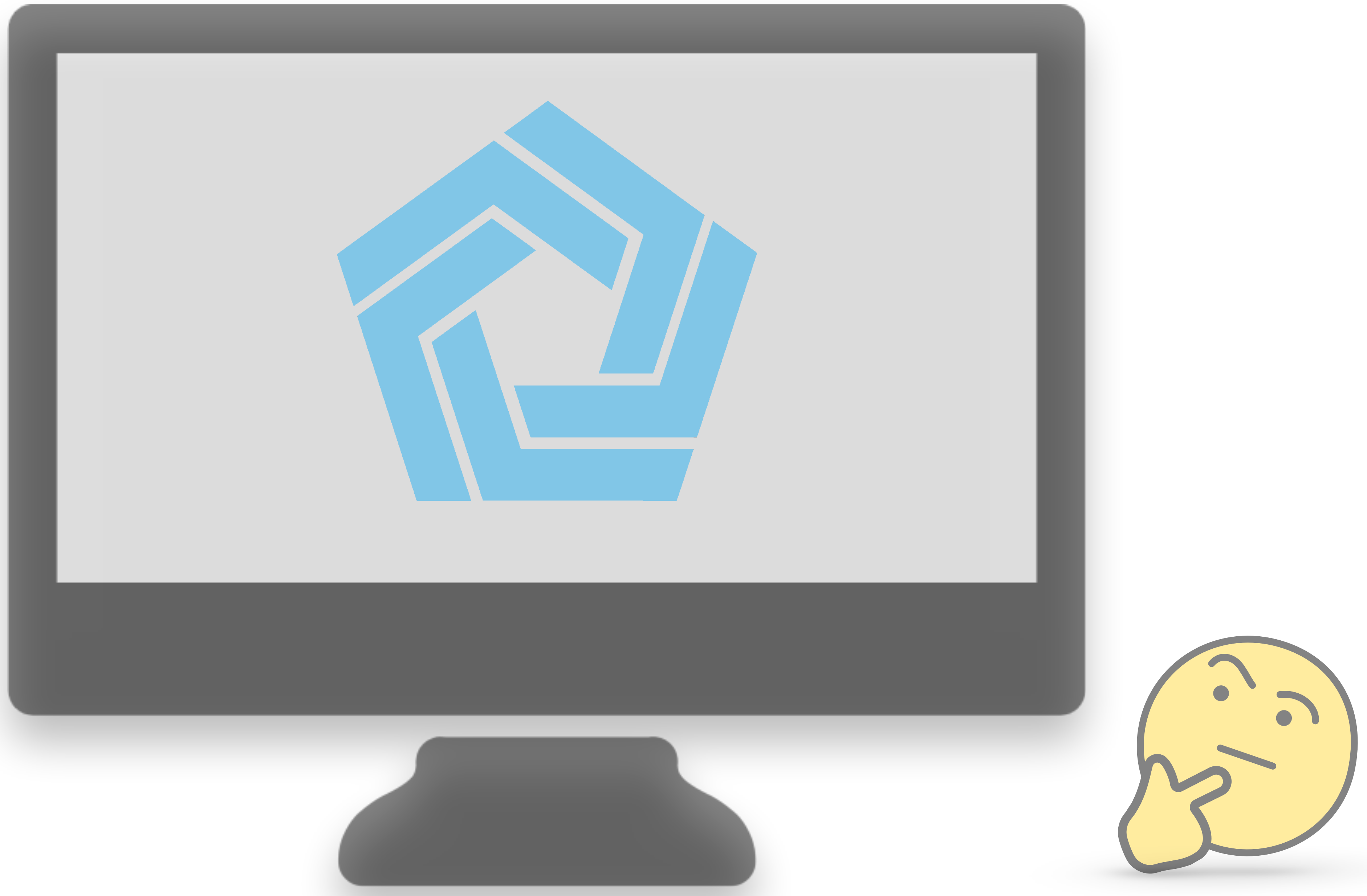


Talk outline



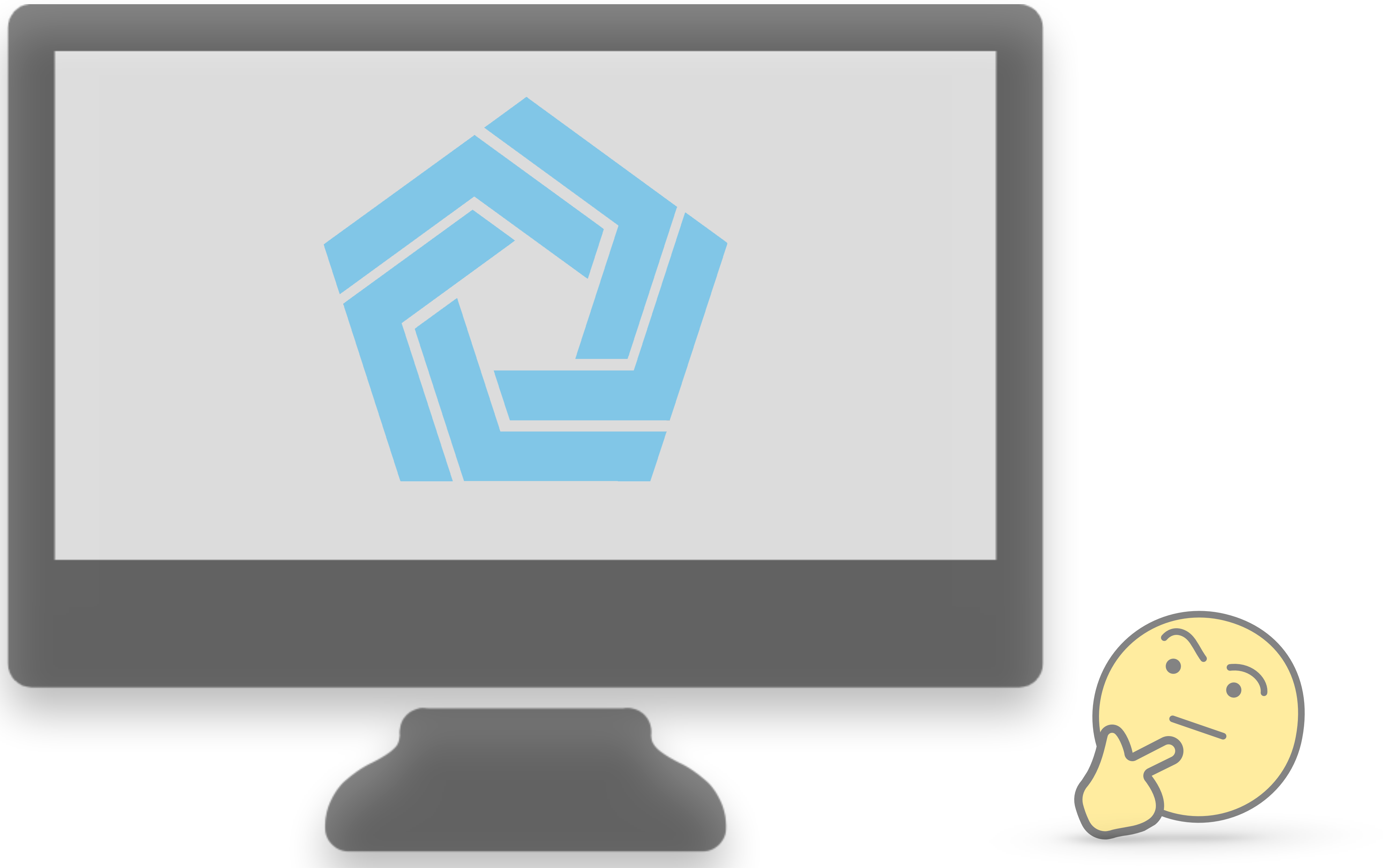
Talk outline

- Related tools



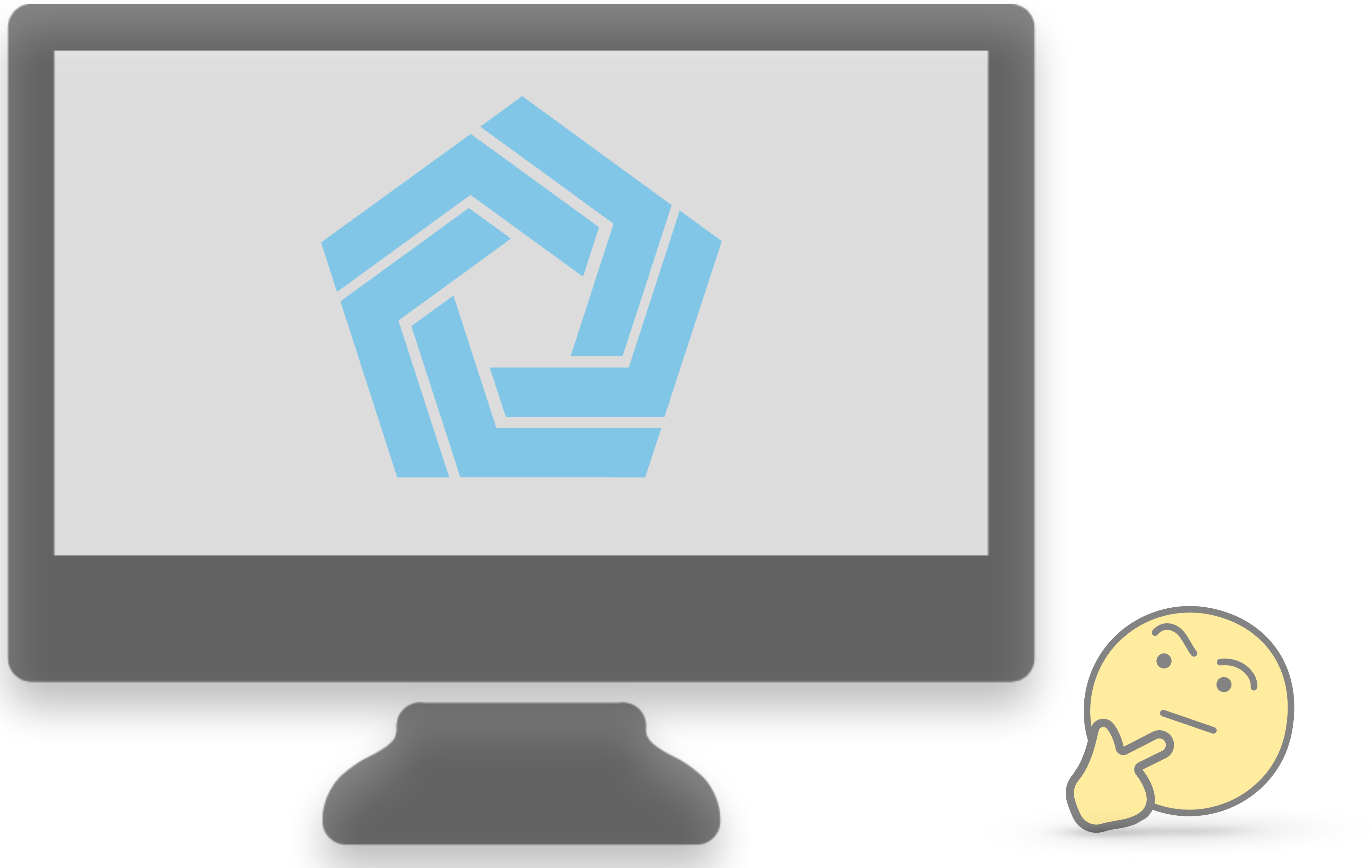
Talk outline

- Related tools
- System design principles



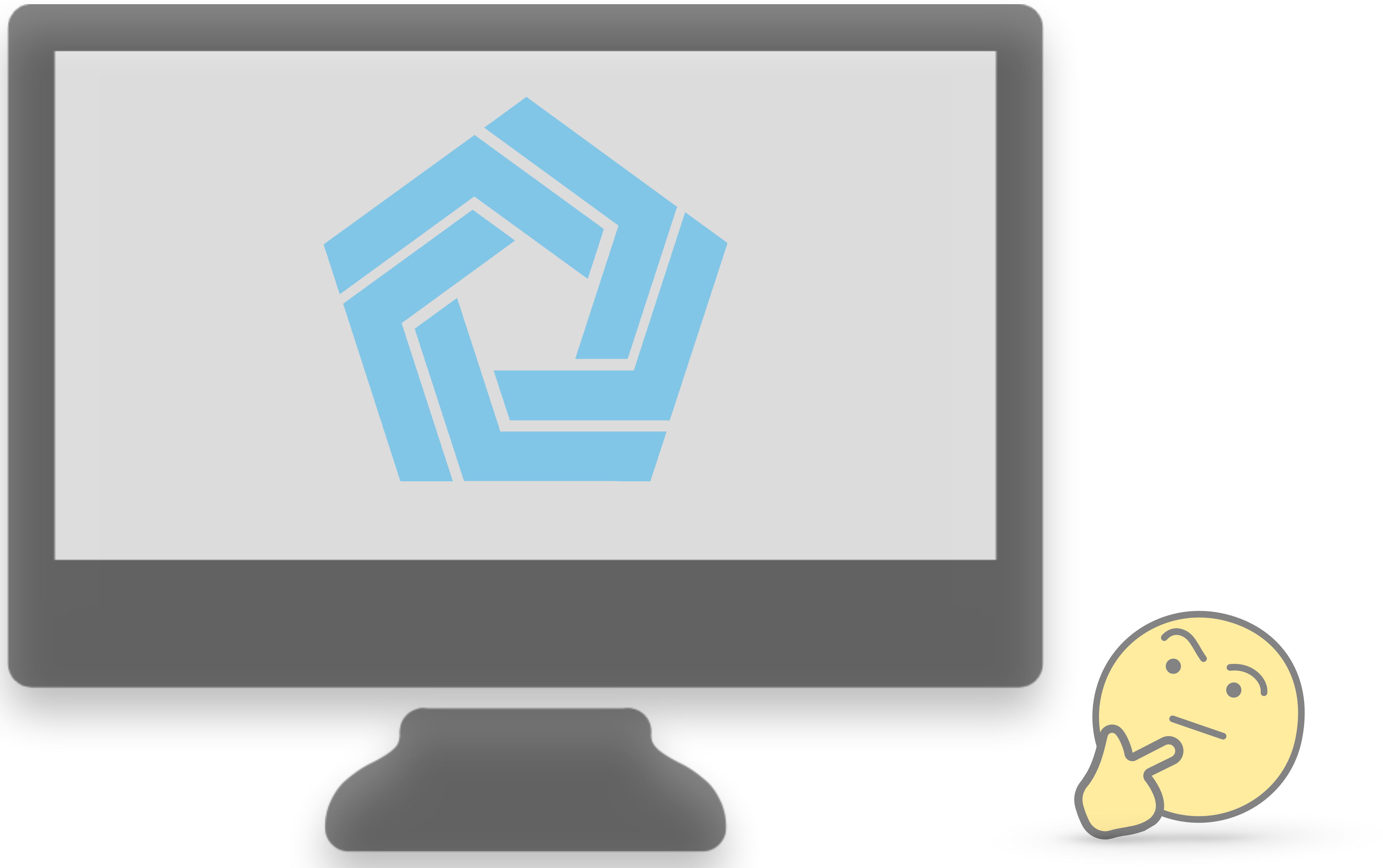
Talk outline

- Related tools
- System design principles
- System walkthrough



Talk outline

- Related tools
- System design principles
- System walkthrough
- **The system in action**



Goals & related work

What would an ideal diagramming tool look like?

Design goals for a mathematical diagramming tool

Design goals for a mathematical diagramming tool

- Σ **Familiarity** – input should look like ordinary mathematics

Design goals for a mathematical diagramming tool

- Σ **Familiarity** — input should look like ordinary mathematics
- **Universality** — should be possible to talk about any domain

Design goals for a mathematical diagramming tool

- Σ **Familiarity** – input should look like ordinary mathematics
- **Universality** – should be possible to talk about any domain
- **Extensibility** – should be possible to use any visualization

Design goals for a mathematical diagramming tool

- **Familiarity** — input should look like ordinary mathematics
- **Universality** — should be possible to talk about any domain
- **Extensibility** — should be possible to use any visualization
- **Beauty** — no inherent limitation of visual sophistication

Design goals for a mathematical diagramming tool

- **Familiarity** – input should look like ordinary mathematics
- **Universality** – should be possible to talk about any domain
- **Extensibility** – should be possible to use any visualization
- **Beauty** – no inherent limitation of visual sophistication
- **Productivity** – should be easy to iterate on diagrams

Design goals for a mathematical diagramming tool

- **Familiarity** – input should look like ordinary mathematics
- **Universality** – should be possible to talk about any domain
- **Extensibility** – should be possible to use any visualization
- **Beauty** – no inherent limitation of visual sophistication
- **Productivity** – should be easy to iterate on diagrams
- **Reusability** – effort should be amortized across diagrams

Broad categories of diagramming tools

Broad categories of diagramming tools

drawing tools / GUIs

examples: chalkboard / pen-and-paper, Illustrator, Inkscape
tradeoffs: universal and familiar vs. tedious and difficult

Broad categories of diagramming tools

drawing tools / GUIs

examples: chalkboard / pen-and-paper, Illustrator, Inkscape
tradeoffs: universal and familiar vs. tedious and difficult

domain-specific packages

examples: GraphViz, GroupExplorer
tradeoffs: concise and automatic vs. narrow

Broad categories of diagramming tools

drawing tools / GUIs

examples: chalkboard / pen-and-paper, Illustrator, Inkscape
tradeoffs: universal and familiar vs. tedious and difficult

domain-specific packages

examples: GraphViz, GroupExplorer
tradeoffs: concise and automatic vs. narrow

graphical specification languages

examples: PostScript, SVG, OpenGL, PGF / TikZ
tradeoffs: universal and reusable vs. tedious and brittle

Broad categories of diagramming tools

drawing tools / GUIs

examples: chalkboard / pen-and-paper, Illustrator, Inkscape
tradeoffs: universal and familiar vs. tedious and difficult

domain-specific packages

examples: GraphViz, GroupExplorer
tradeoffs: concise and automatic vs. narrow

graphical specification languages

examples: PostScript, SVG, OpenGL, PGF / TikZ
tradeoffs: universal and reusable vs. tedious and brittle

graphing / plotting tools

examples: graphing calculator, Mathematica, MATLAB, matplotlib
tradeoffs: convenient and straightforward vs. literal and brittle

Existing tools don't consider the larger process

specify the mathematical idea

↓

translate it into a visual interpretation

↓

make a high-quality drawing

Existing tools don't consider the larger process

specify the mathematical idea

↓

translate it into a visual interpretation

↓

make a high-quality drawing

Existing tools don't consider the larger process

specify the mathematical idea

↓

translate it into a visual interpretation

GUI tools (e.g. Illustrator)

↓

make a high-quality drawing

Existing tools don't consider the larger process

specify the mathematical idea

translate it into a visual interpretation


GUI tools (e.g. Illustrator)


plotting tools (e.g. Mathematica)

make a high-quality drawing

Existing tools don't consider the larger process

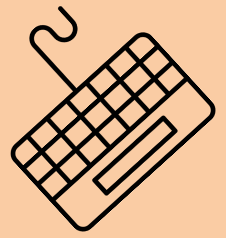
specify the mathematical idea

↓

translate it into a visual interpretation


GUI tools (e.g. Illustrator)


plotting tools (e.g. Mathematica)


graphical specification languages
(e.g. TikZ)

↓

make a high-quality drawing

Existing tools don't consider the larger process

specify the mathematical idea

↓

translate it into a visual interpretation

GUI tools (e.g. Illustrator)

plotting tools (e.g. Mathematica)

graphical specification languages (e.g. TikZ)

↓

make a high-quality drawing

Existing tools don't consider the larger process

specify the mathematical idea

domain-specific packages
(e.g. GraphViz)

translate it into a visual interpretation

GUI tools (e.g. Illustrator)

plotting tools (e.g. Mathematica)

graphical specification languages
(e.g. TikZ)

make a high-quality drawing

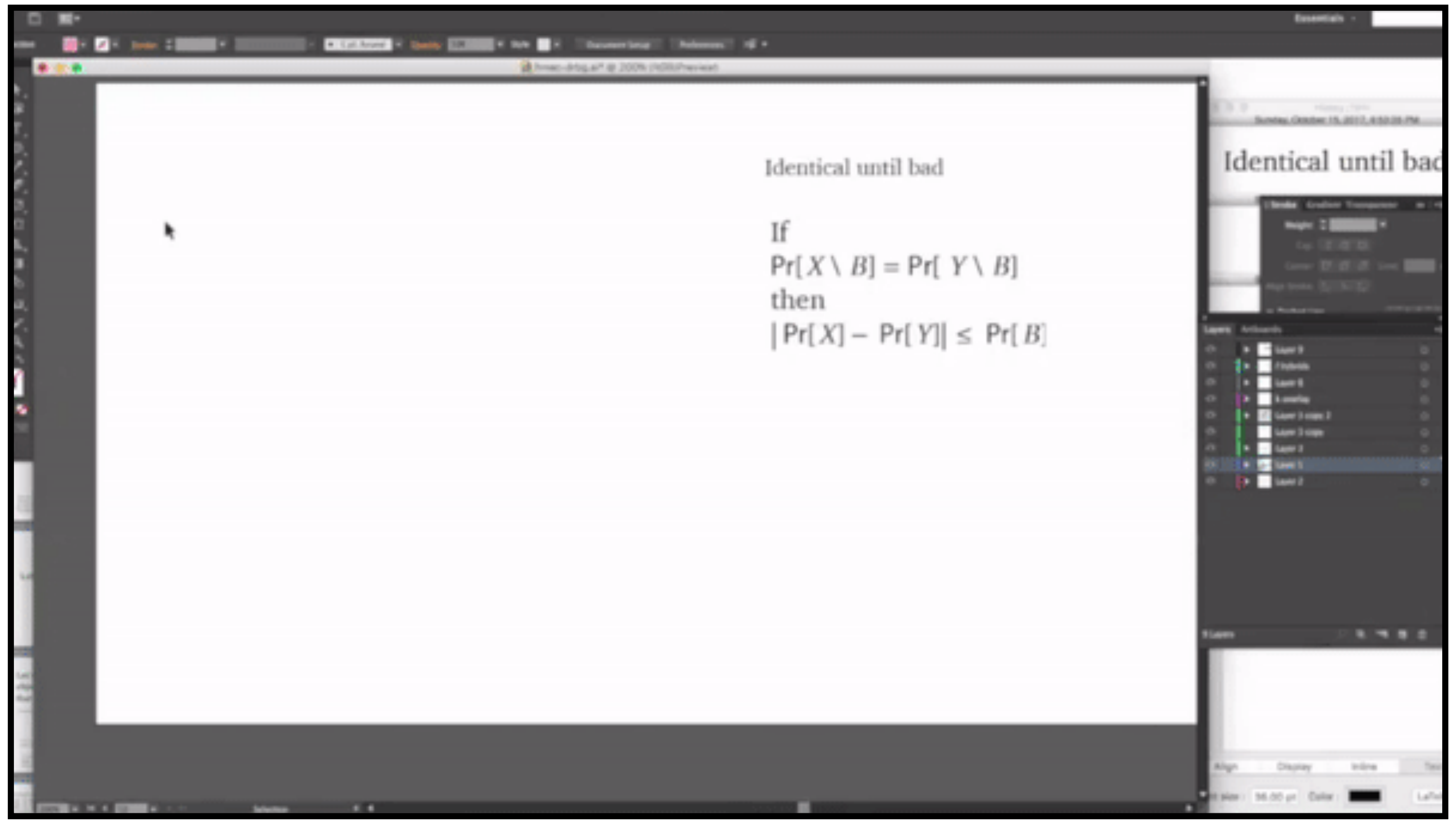
Goal: provide translation; avoid tedium

Goal: provide translation; avoid tedium

Less of this:

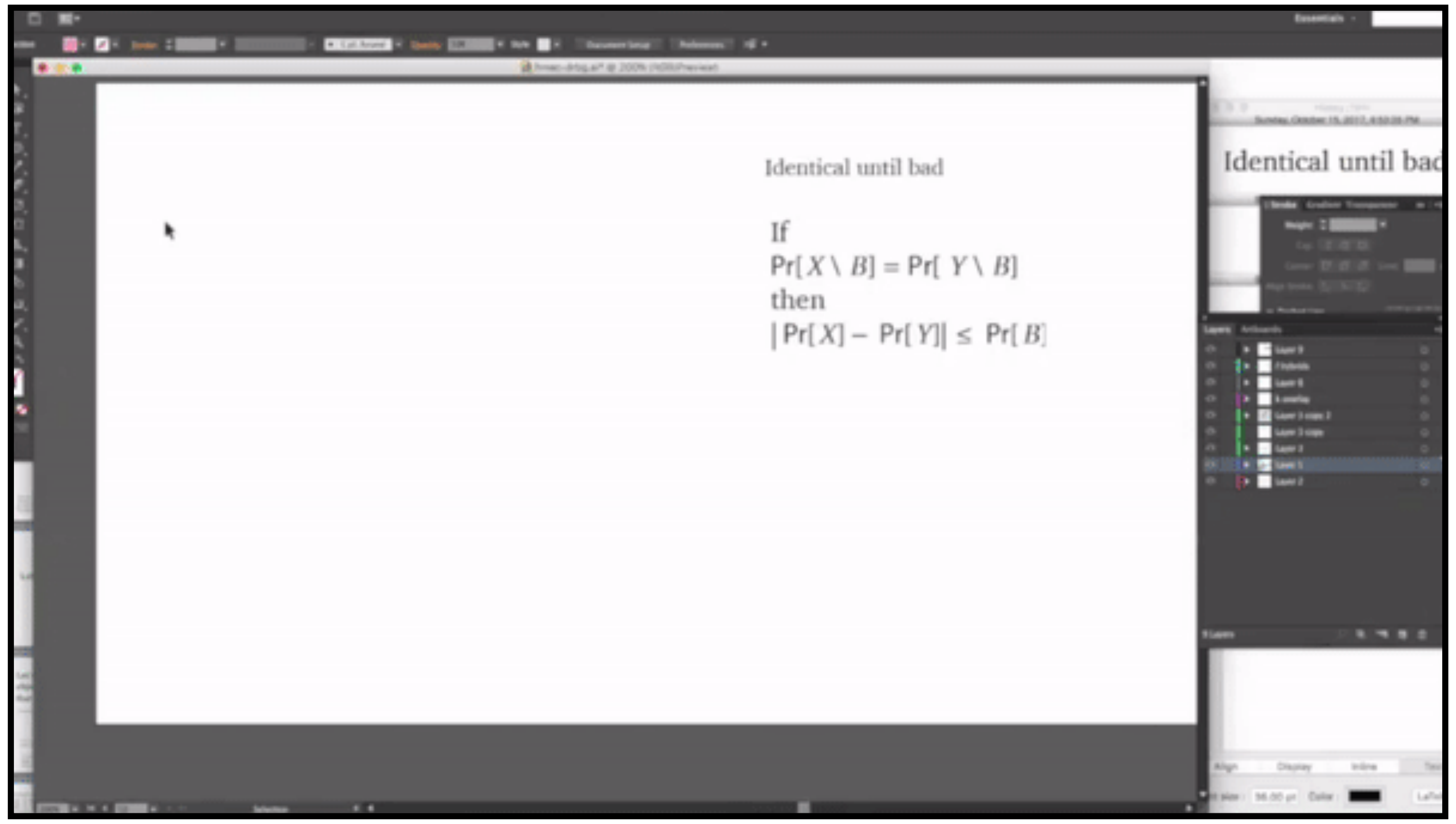
Goal: provide translation; avoid tedium

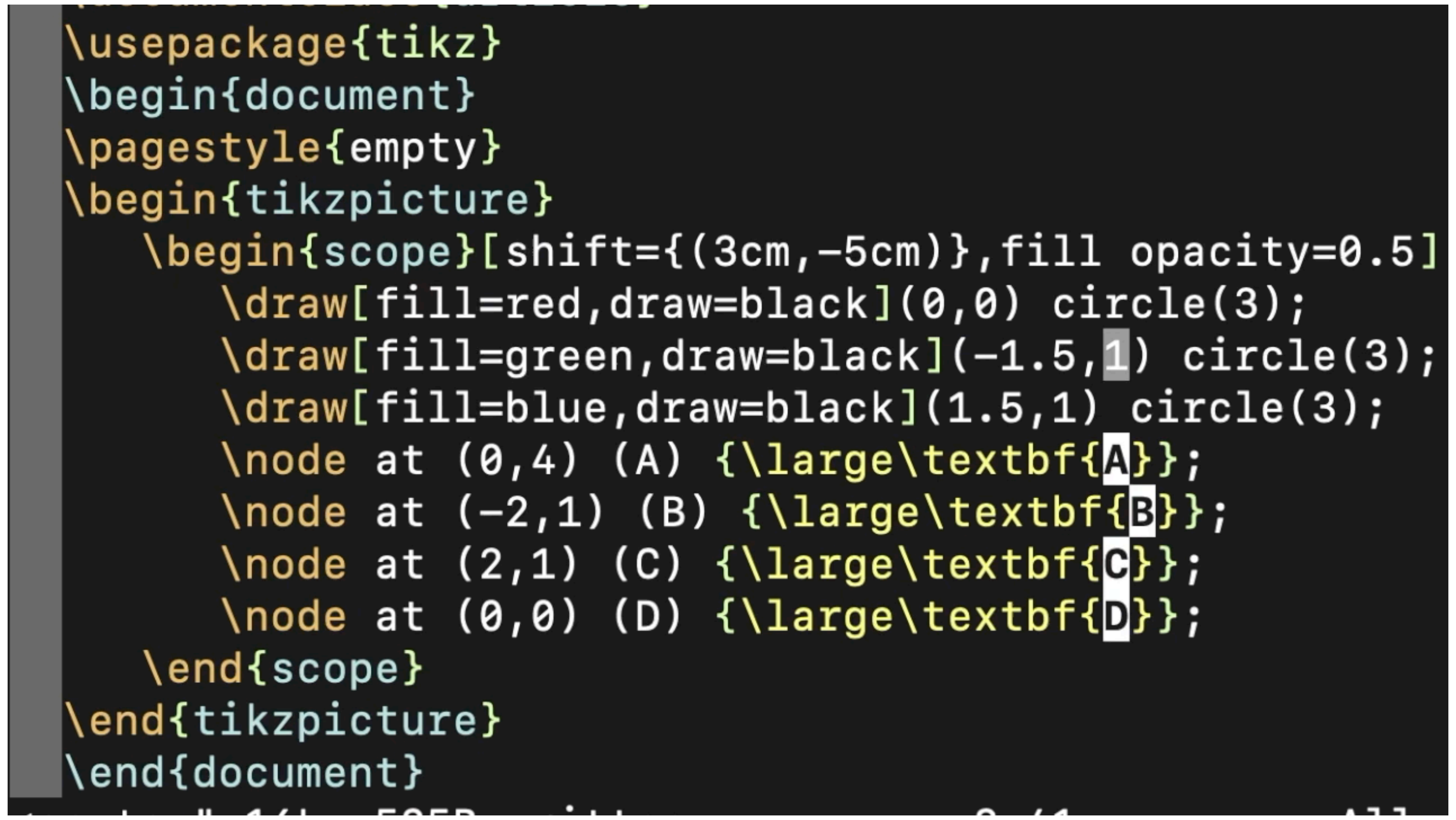
Less of this:



Goal: provide translation; avoid tedium

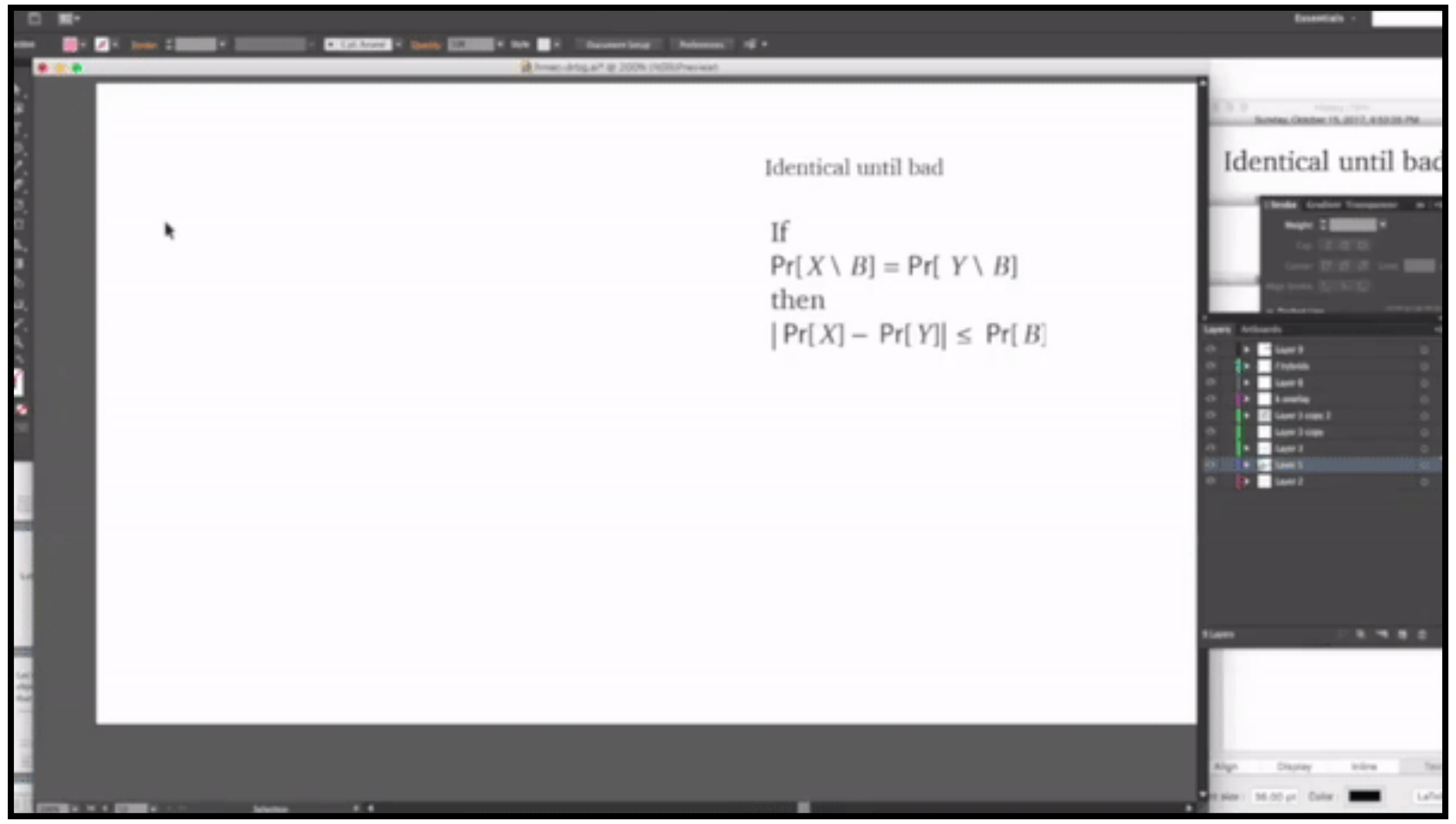
Less of this:

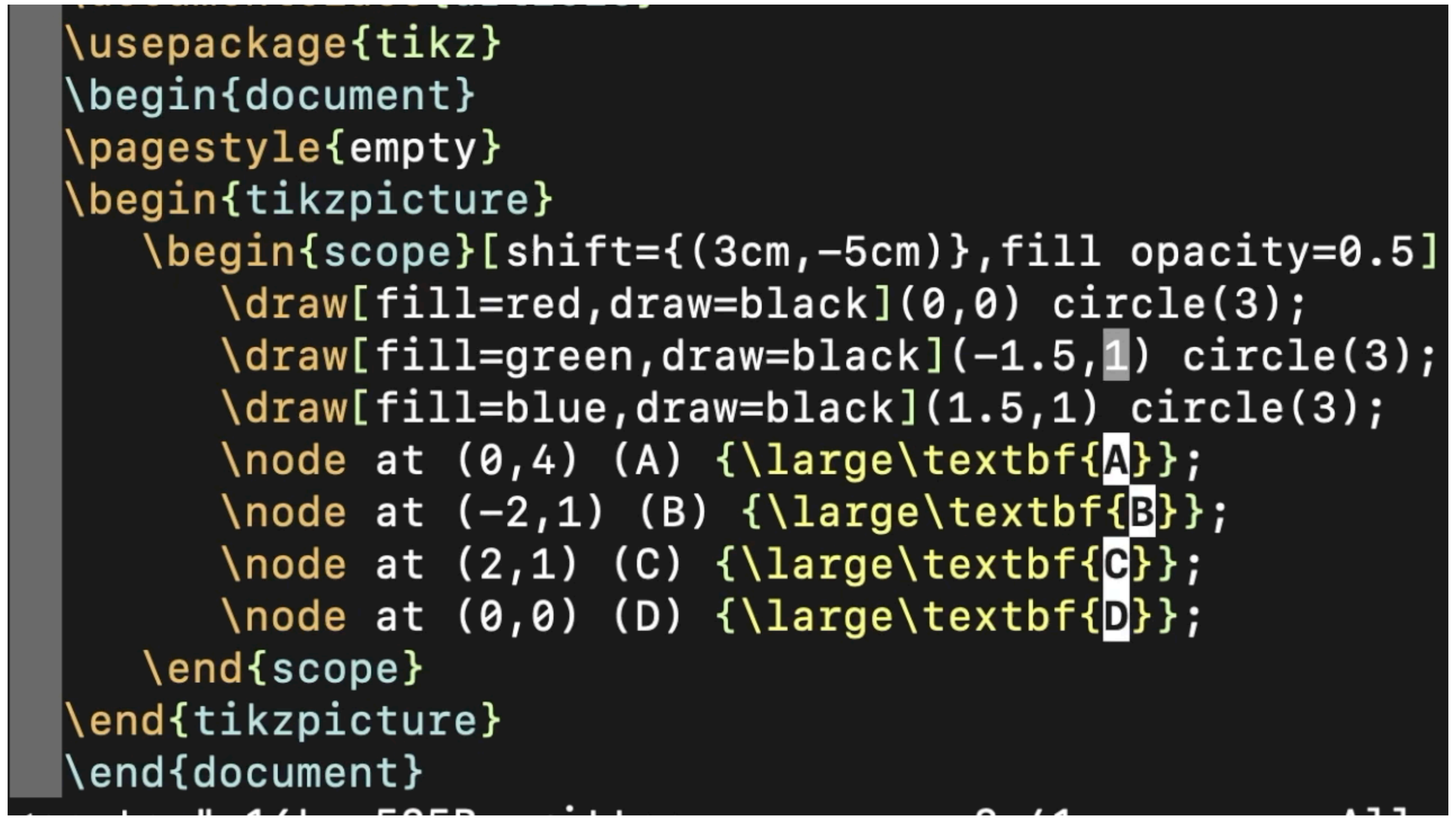




Goal: provide translation; avoid tedium

Less of this:

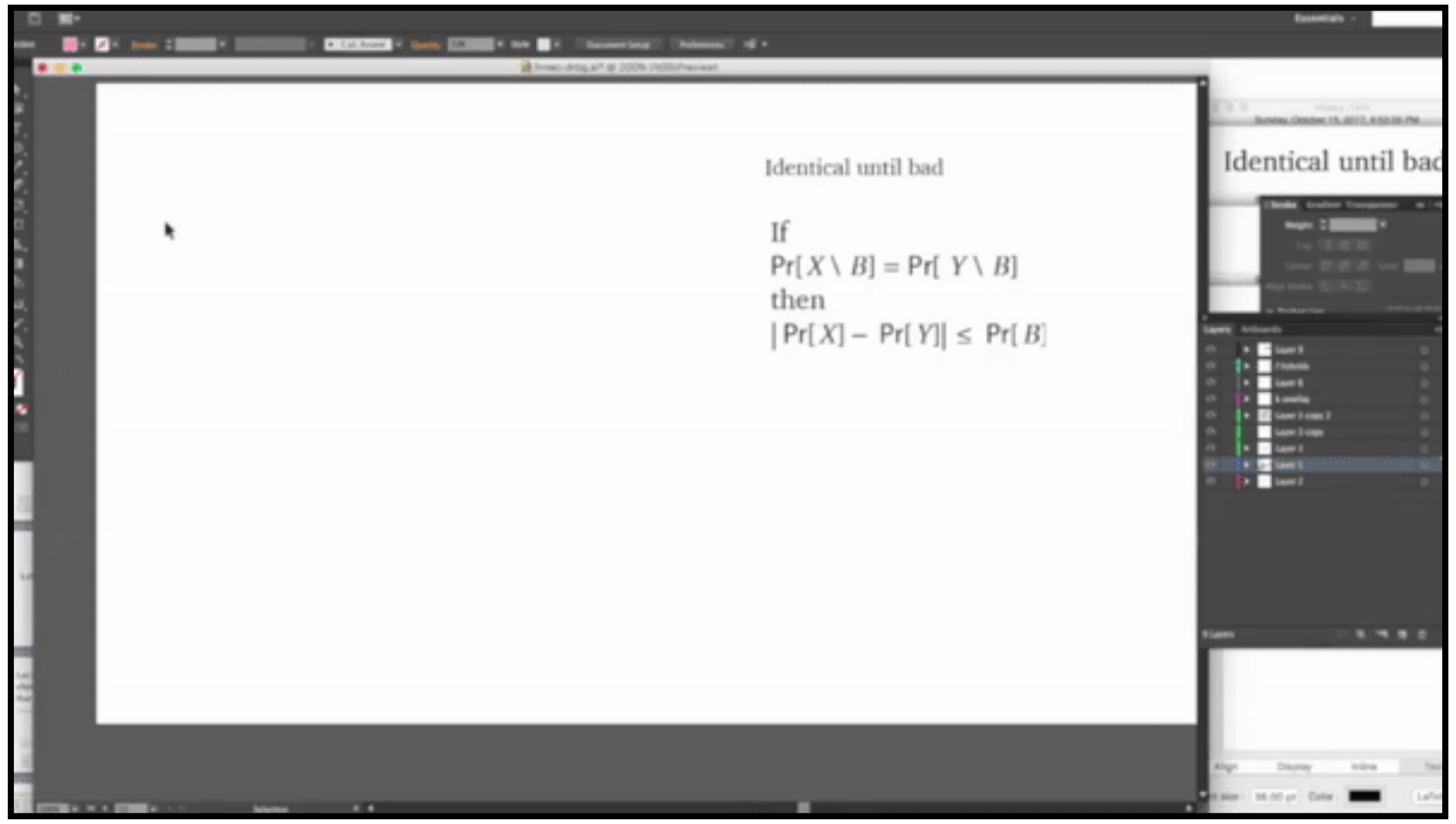


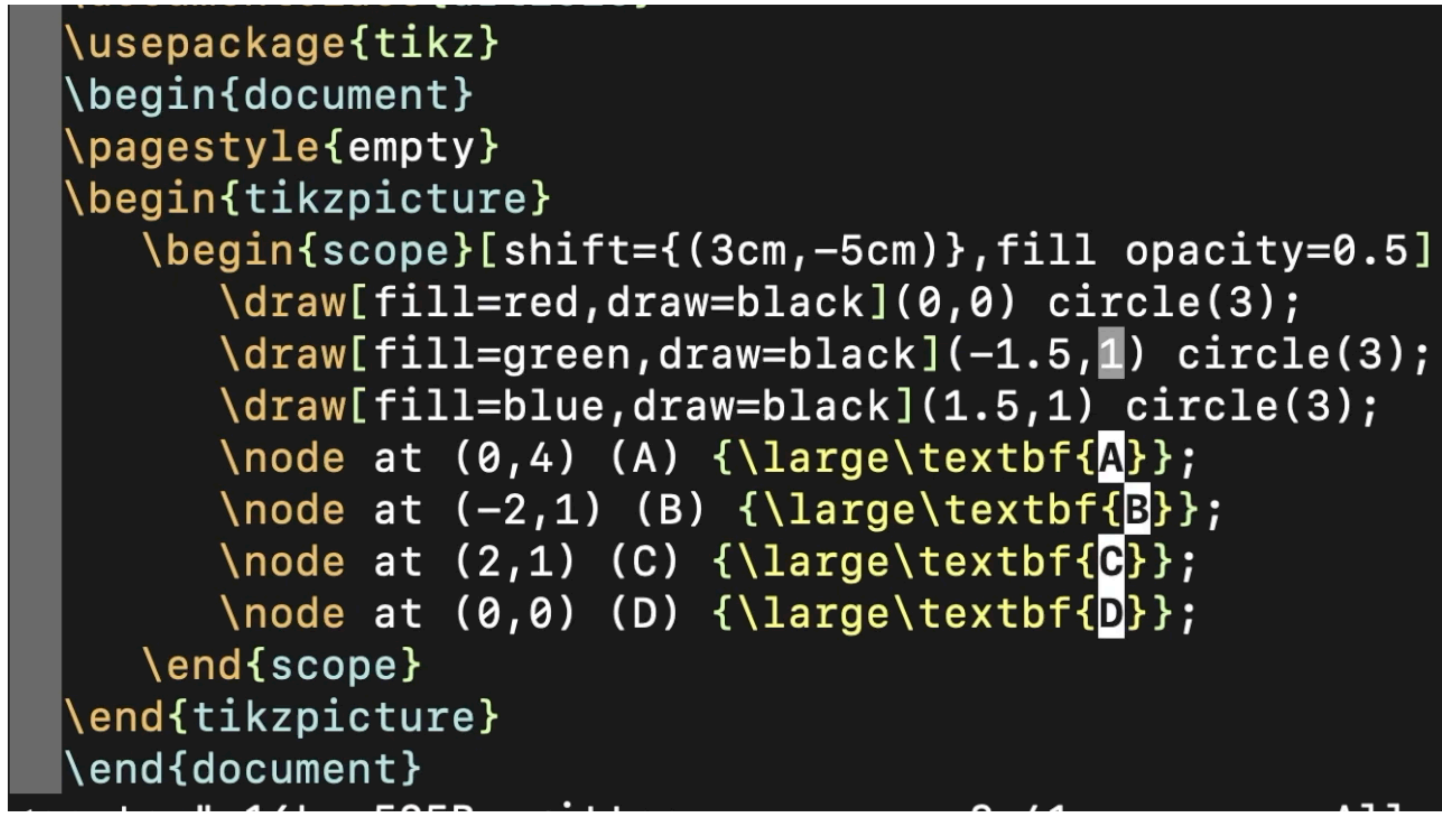


More of this:

Goal: provide translation; avoid tedium

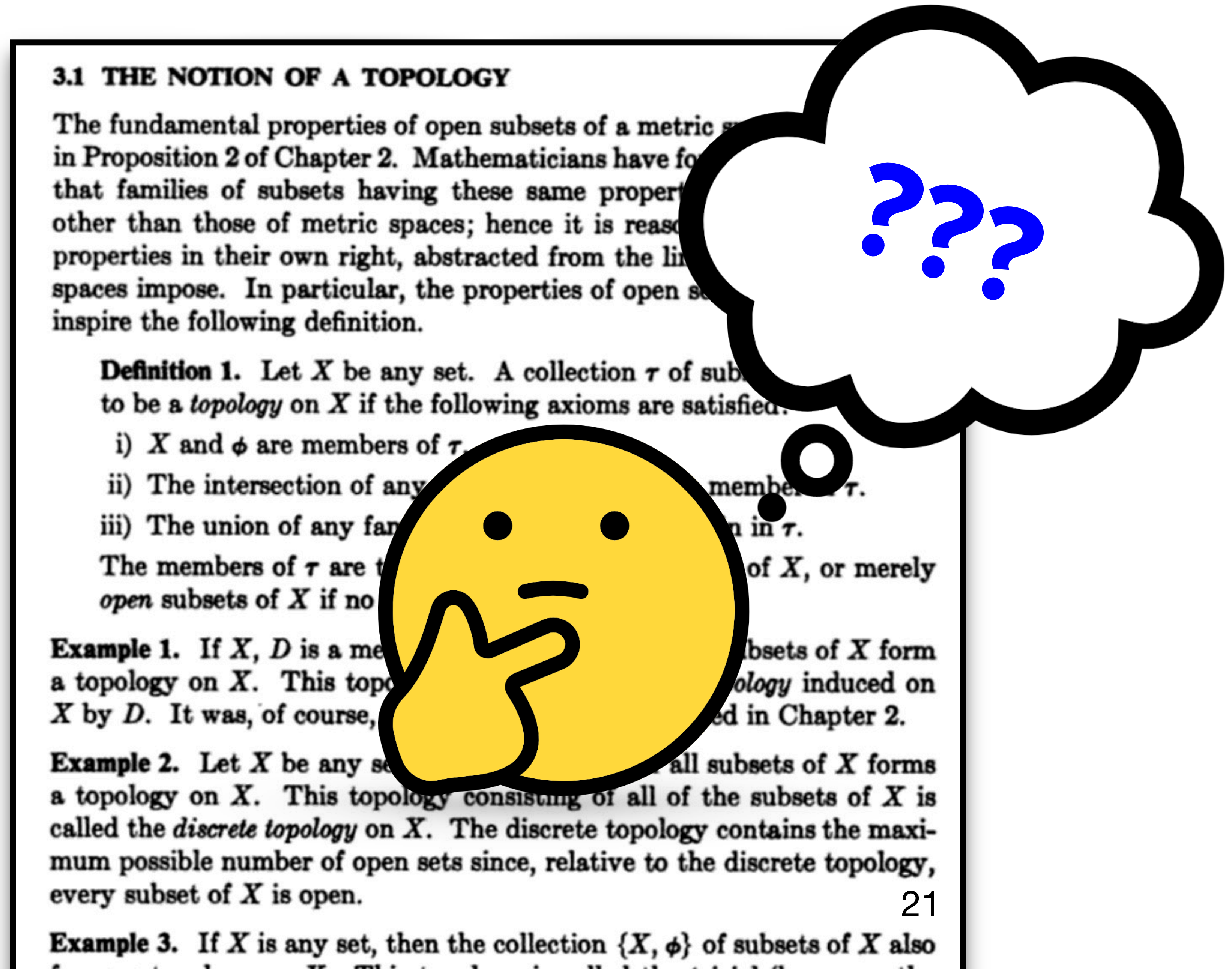
Less of this:





```
\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(3);
      \draw[fill=green,draw=black](-1.5,1) circle(3);
      \draw[fill=blue,draw=black](1.5,1) 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}
```

More of this:



3.1 THE NOTION OF A TOPOLOGY

The fundamental properties of open subsets of a metric s
in Proposition 2 of Chapter 2. Mathematicians have fo
that families of subsets having these same propert
other than those of metric spaces; hence it is reaso
properties in their own right, abstracted from the li
spaces impose. In particular, the properties of open s
inspire the following definition.

Definition 1. Let X be any set. A collection τ of sub
to be a *topology* on X if the following axioms are satisfied

i) X and ϕ are members of τ.

ii) The intersection of any … membe … τ.

iii) The union of any fa … n in τ.

The members of τ are … of X, or merely *open* subsets of X if no

Example 1. If X, D is a me … bsets of X form a topology on X. This top … *ology* induced on X by D. It was, of course, … ed in Chapter 2.

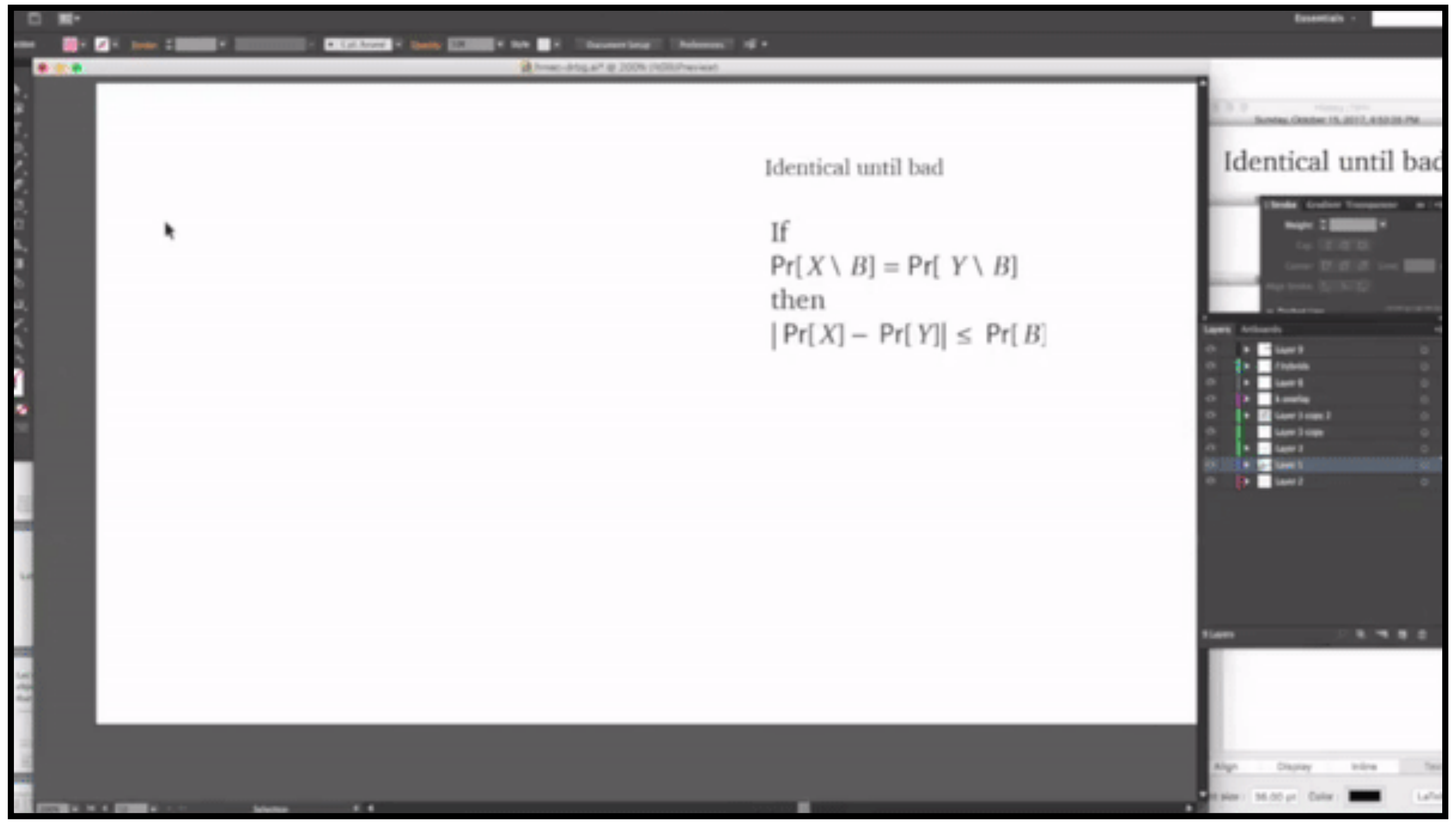
Example 2. Let X be any s … all subsets of X forms a topology on X. This topology consisting of all of the subsets of X is called the *discrete topology* on X. The discrete topology contains the maximum possible number of open sets since, relative to the discrete topology, every subset of X is open.

21

Example 3. If X is any set, then the collection $\{X, \phi\}$ of subsets of X also

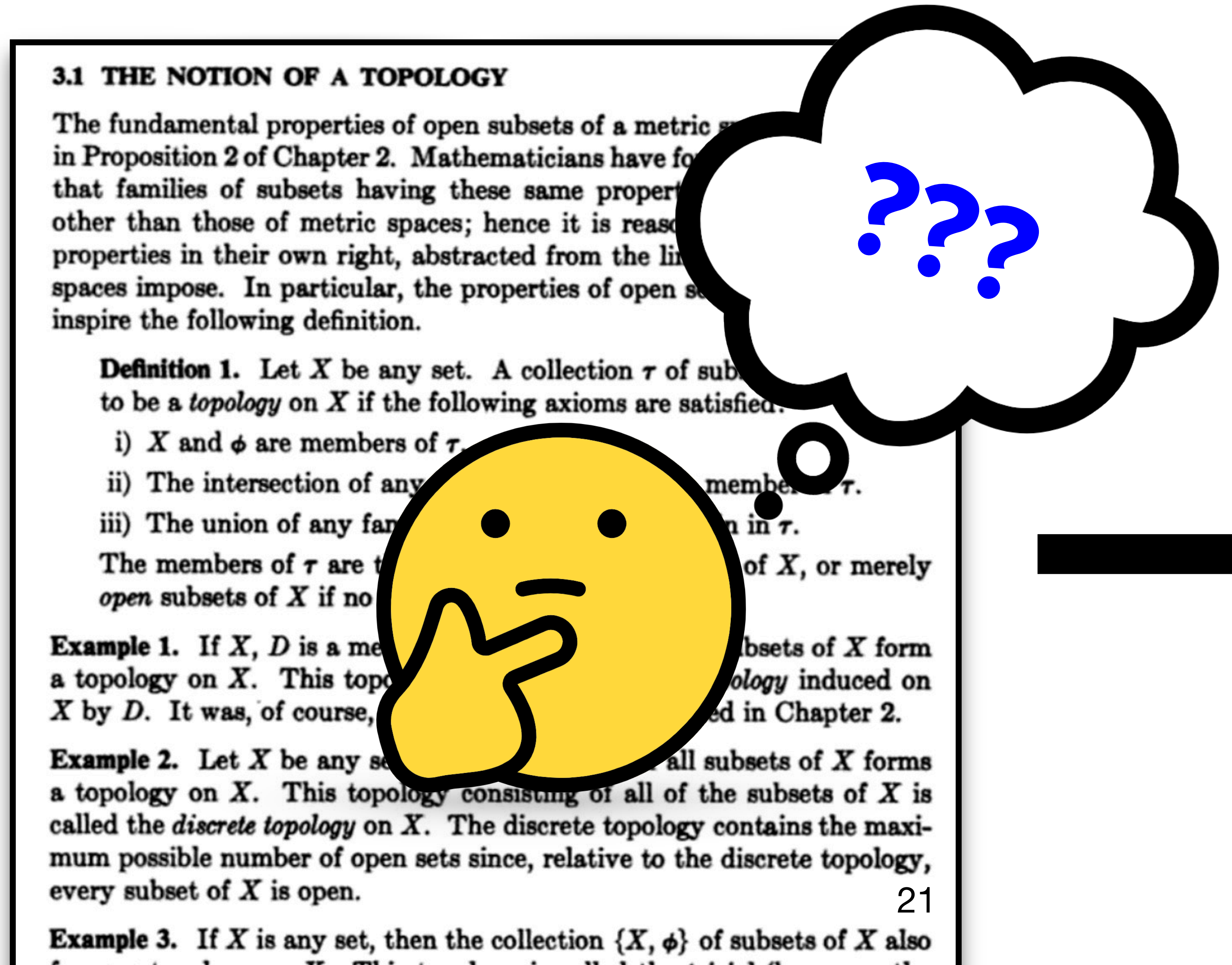
Goal: provide translation; avoid tedium

Less of this:



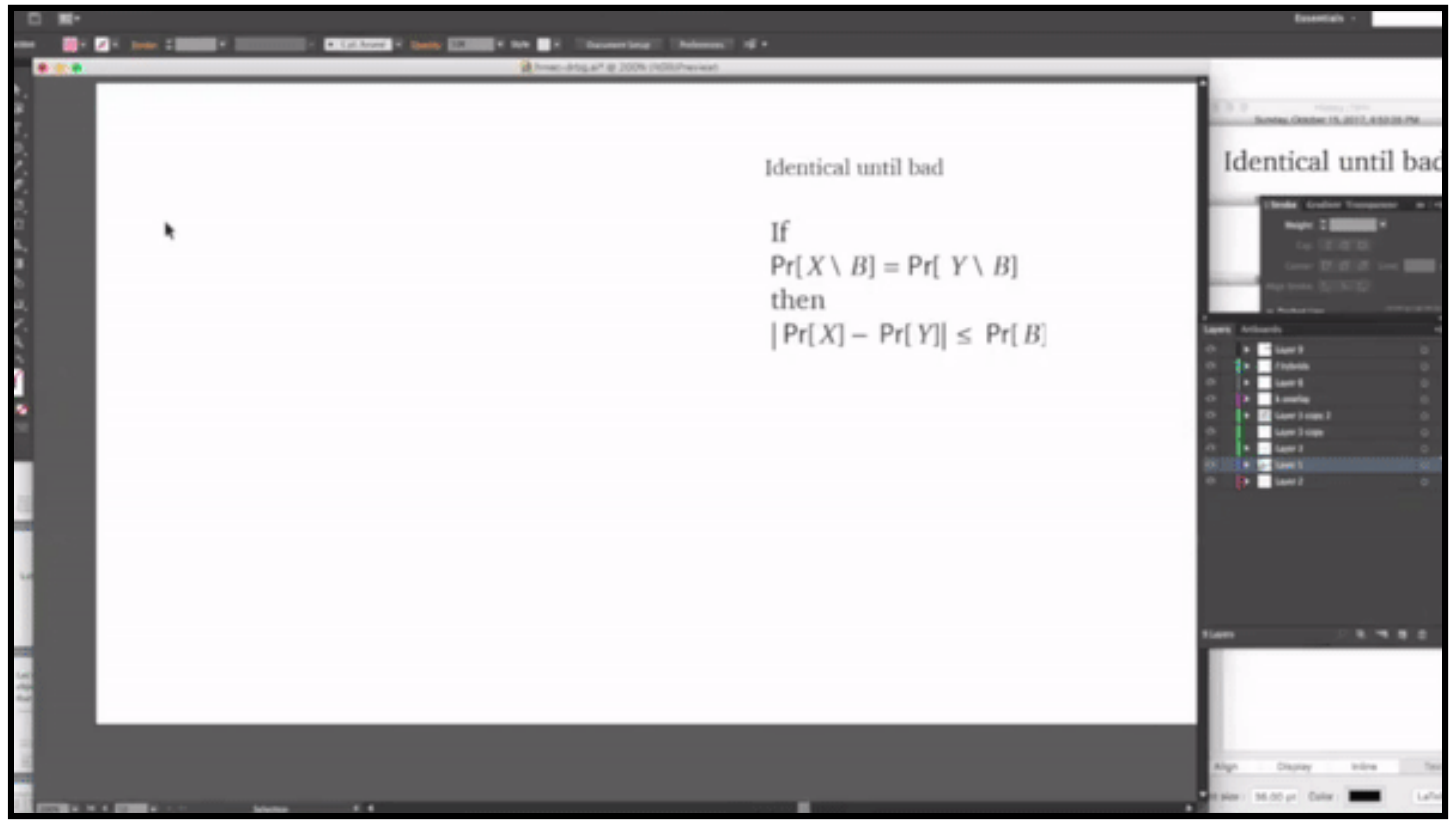

```
\usepackage{tikz}
\begin{document}
\pagestyle{empty}
\begin{tikzpicture}
  \begin{scope}[shift={(3cm,-5cm)},fill opacity=0.5]
    \draw[fill=red,draw=black](0,0) circle(3);
    \draw[fill=green,draw=black](-1.5,1) circle(3);
    \draw[fill=blue,draw=black](1.5,1) circle(3);
    \node at (0,4) (A)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

More of this:



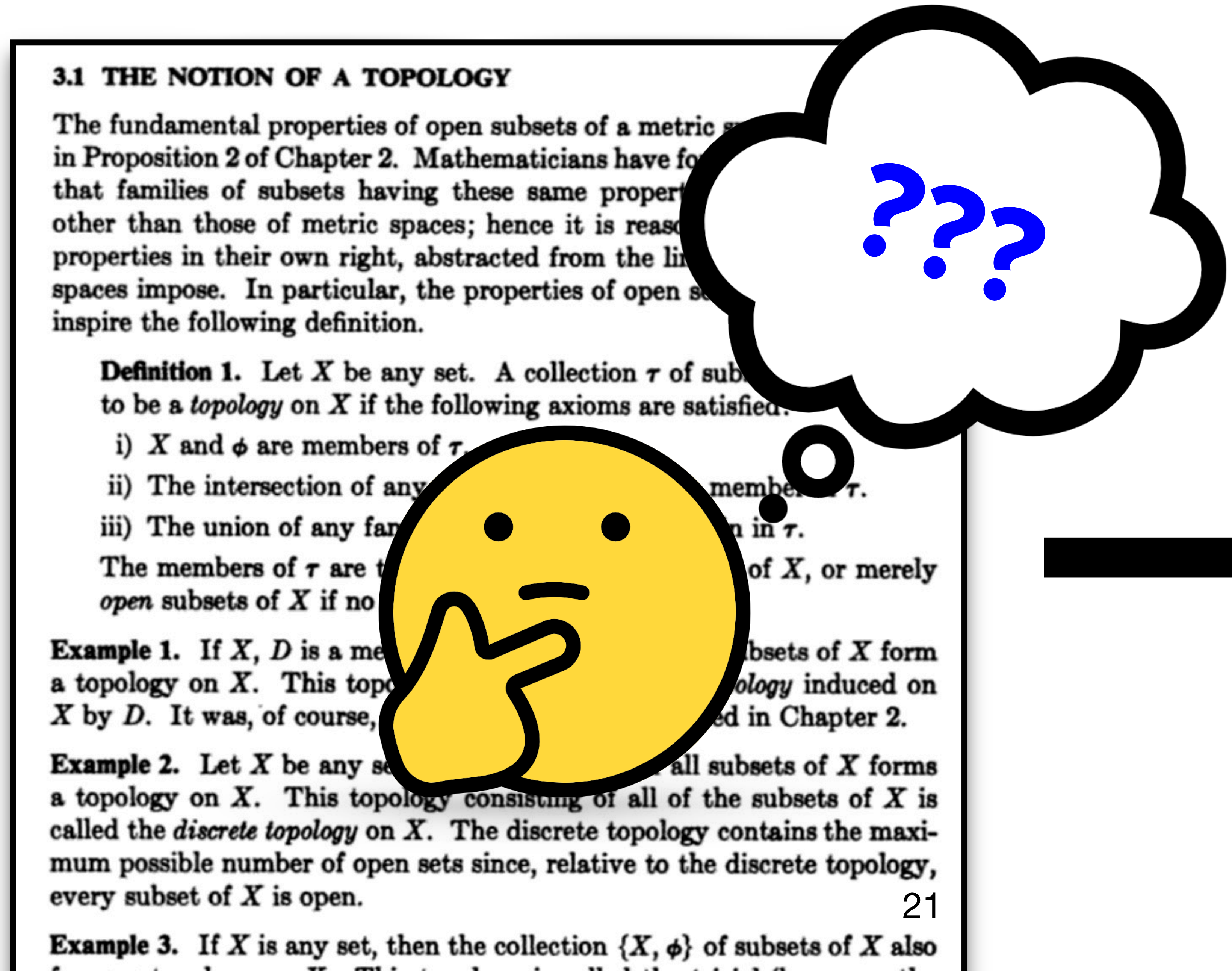
Goal: provide translation; avoid tedium

Less of this:




```
\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(3);
    \draw[fill=green,draw=black](-1.5,1) circle(3);
    \draw[fill=blue,draw=black](1.5,1) 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}
```

More of this:



3.1 THE NOTION OF A TOPOLOGY

The fundamental properties of open subsets of a metric s… in Proposition 2 of Chapter 2. Mathematicians have fo… that families of subsets having these same propert… other than those of metric spaces; hence it is reas… properties in their own right, abstracted from the li… spaces impose. In particular, the properties of open s… inspire the following definition.

Definition 1. Let X be any set. A collection τ of sub… to be a *topology* on X if the following axioms are satisfied:

i) X and ϕ are members of τ.
ii) The intersection of any … members of τ.
iii) The union of any fa… in τ.

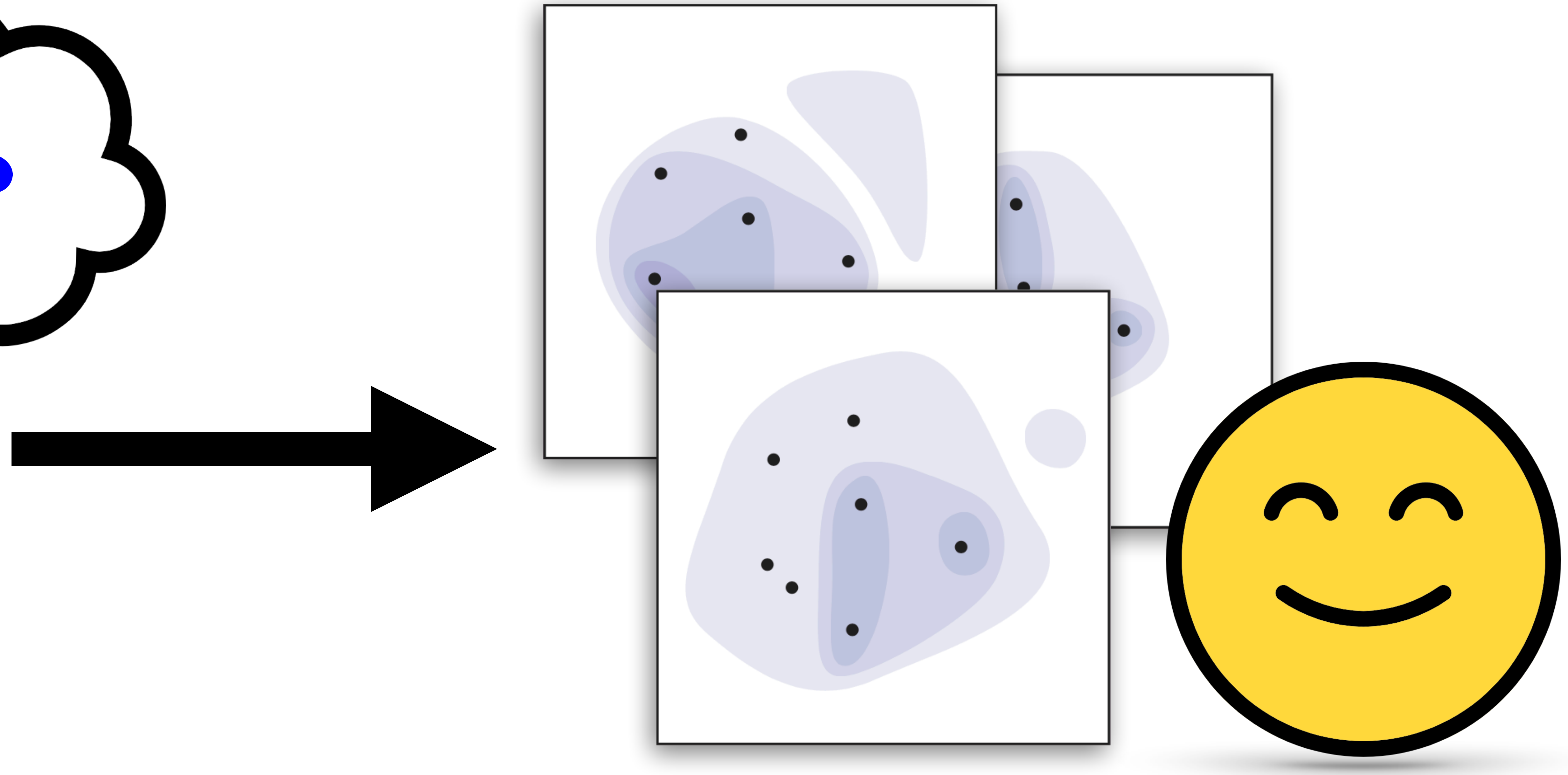
The members of τ are … of X, or merely *open* subsets of X if no…

Example 1. If X, D is a me… subsets of X form a topology on X. This topo… *ology* induced on X by D. It was, of course, … ed in Chapter 2.

Example 2. Let X be any se… all subsets of X forms a topology on X. This topology consisting of all of the subsets of X is called the *discrete topology* on X. The discrete topology contains the maximum possible number of open sets since, relative to the discrete topology, every subset of X is open.

21

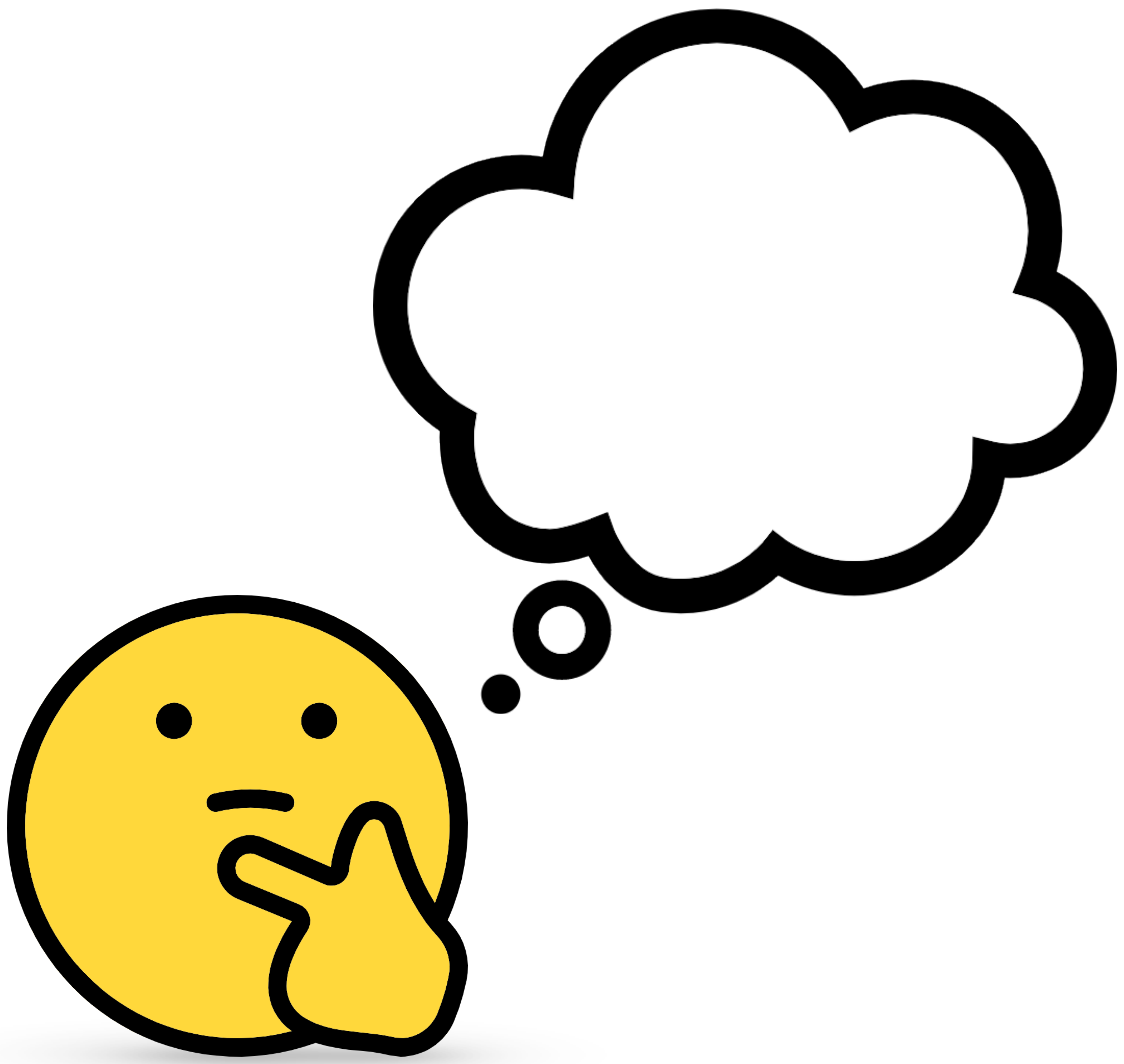
Example 3. If X is any set, then the collection $\{X, \phi\}$ of subsets of X also

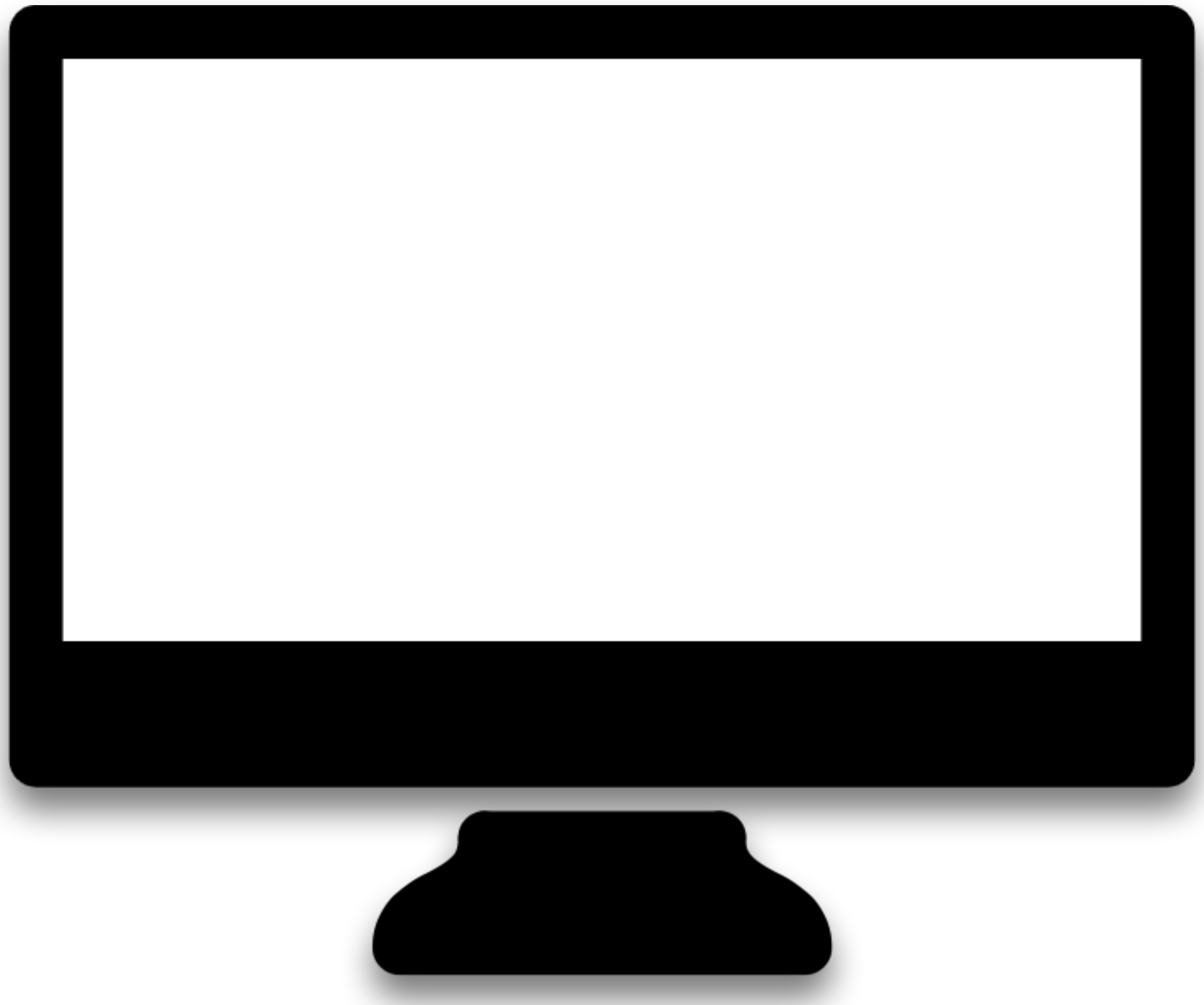


System design principles

Modeling how people visualize abstract ideas by hand

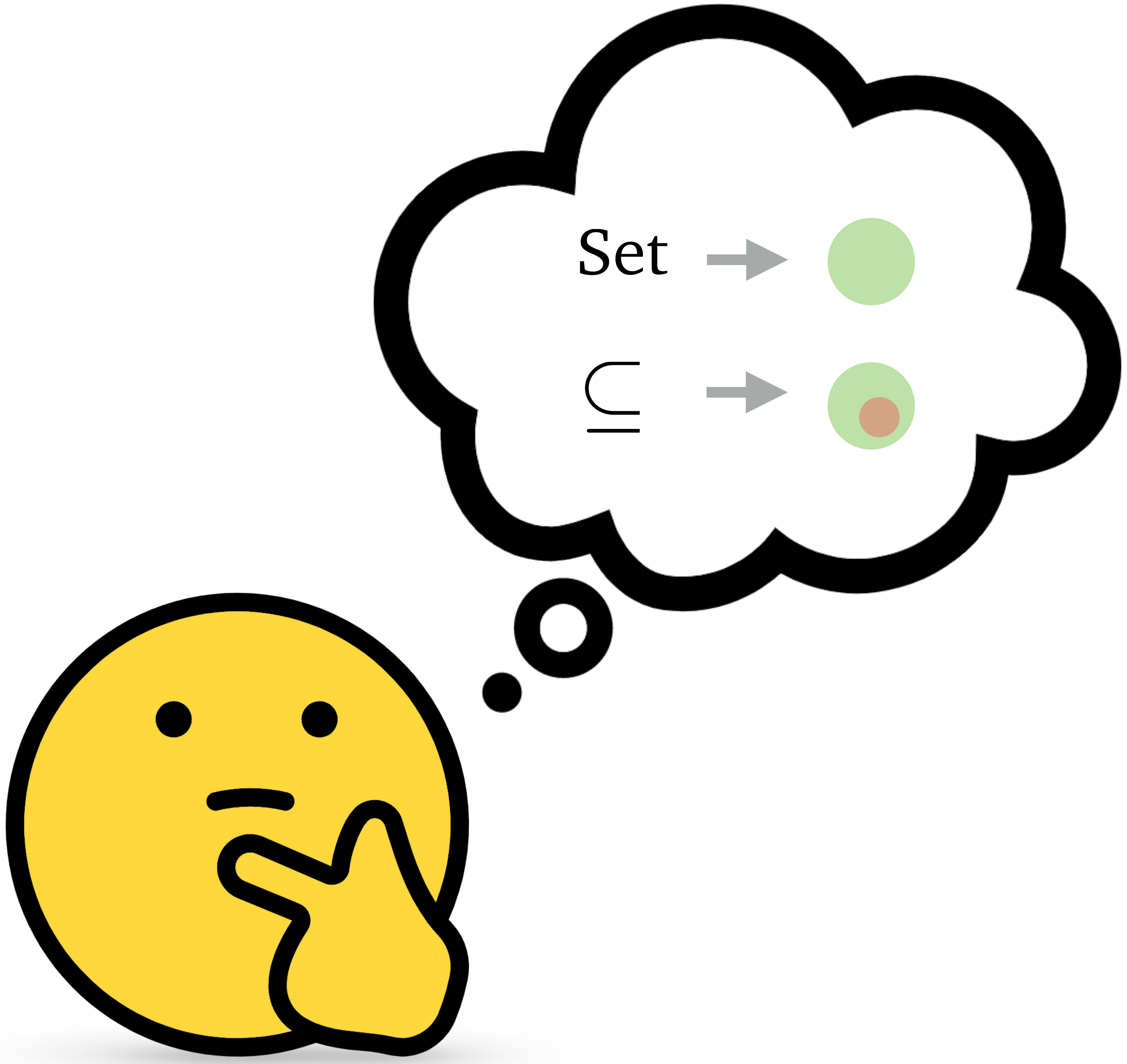
There exist sets A, B such that $B \subseteq A$.

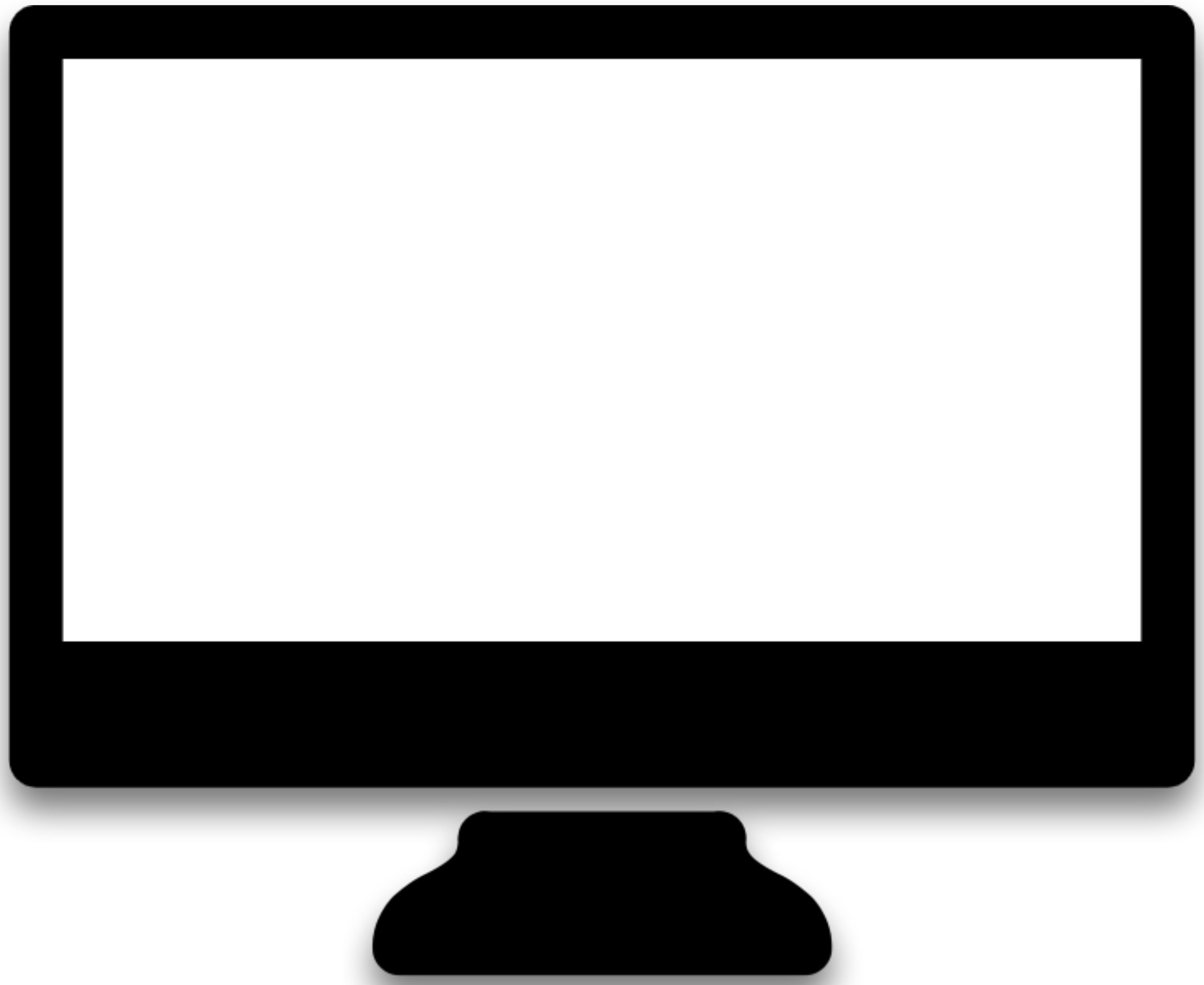




Modeling how people visualize abstract ideas by hand

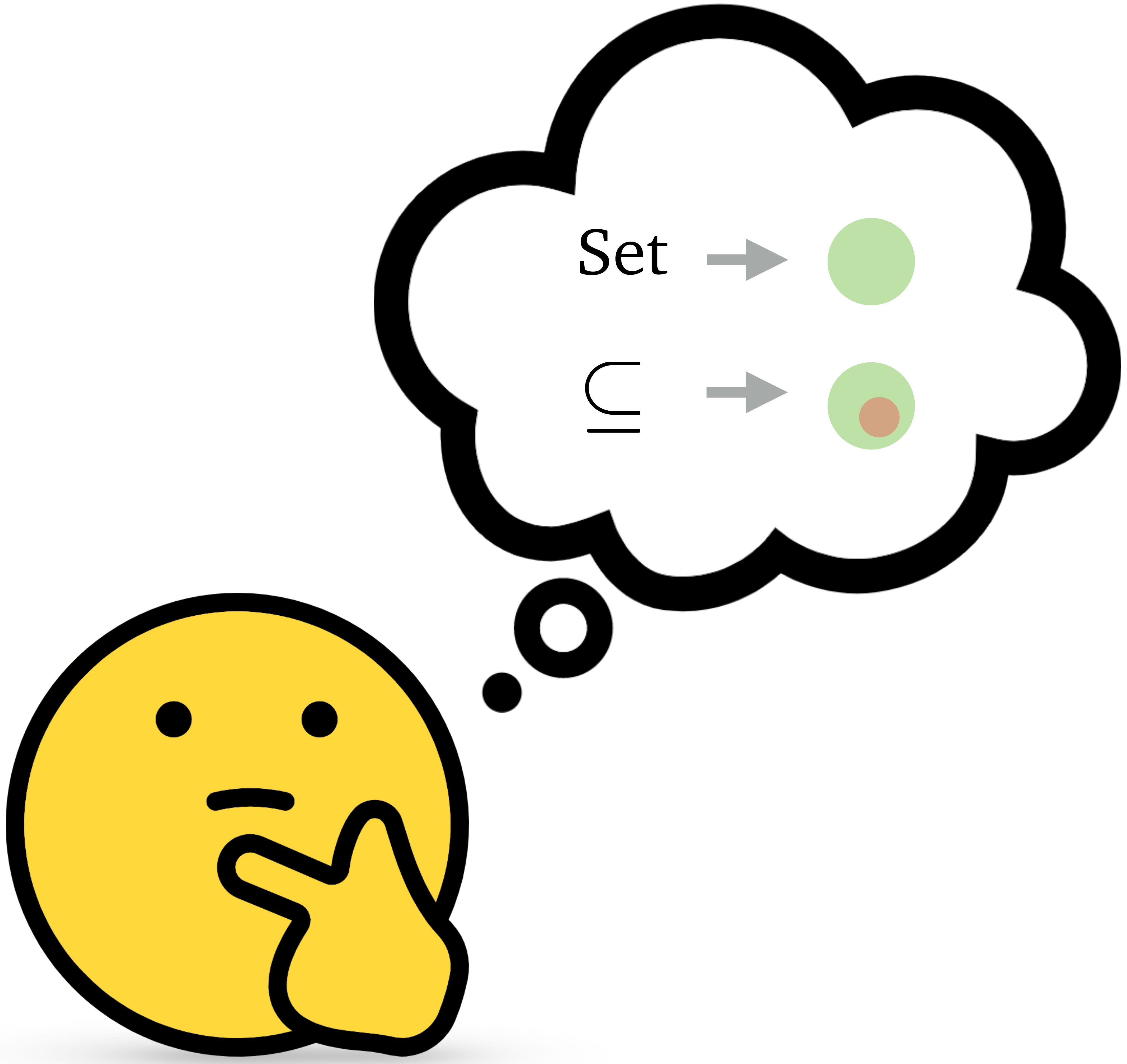
There exist sets A, B such that $B \subseteq A$.

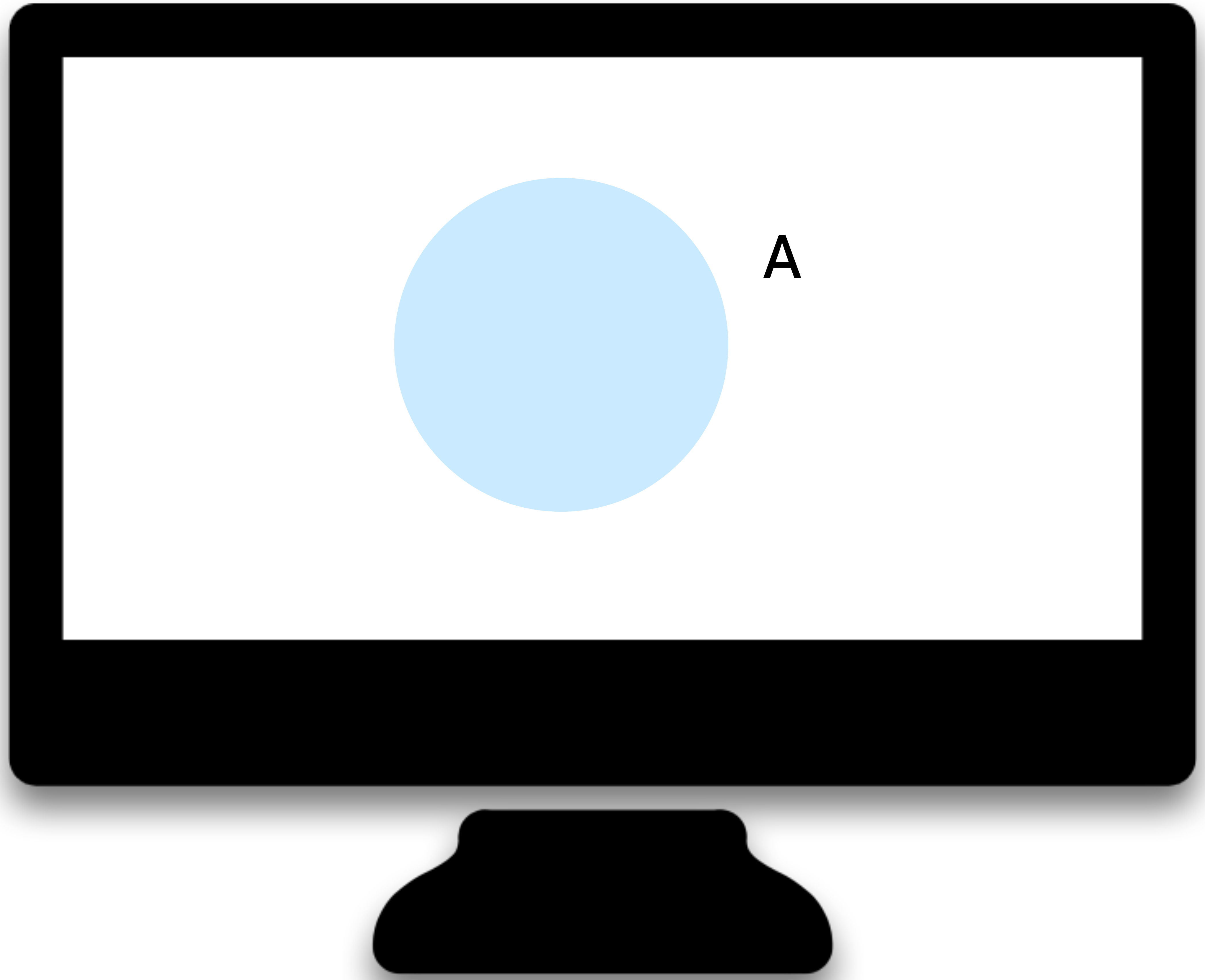




Modeling how people visualize abstract ideas by hand

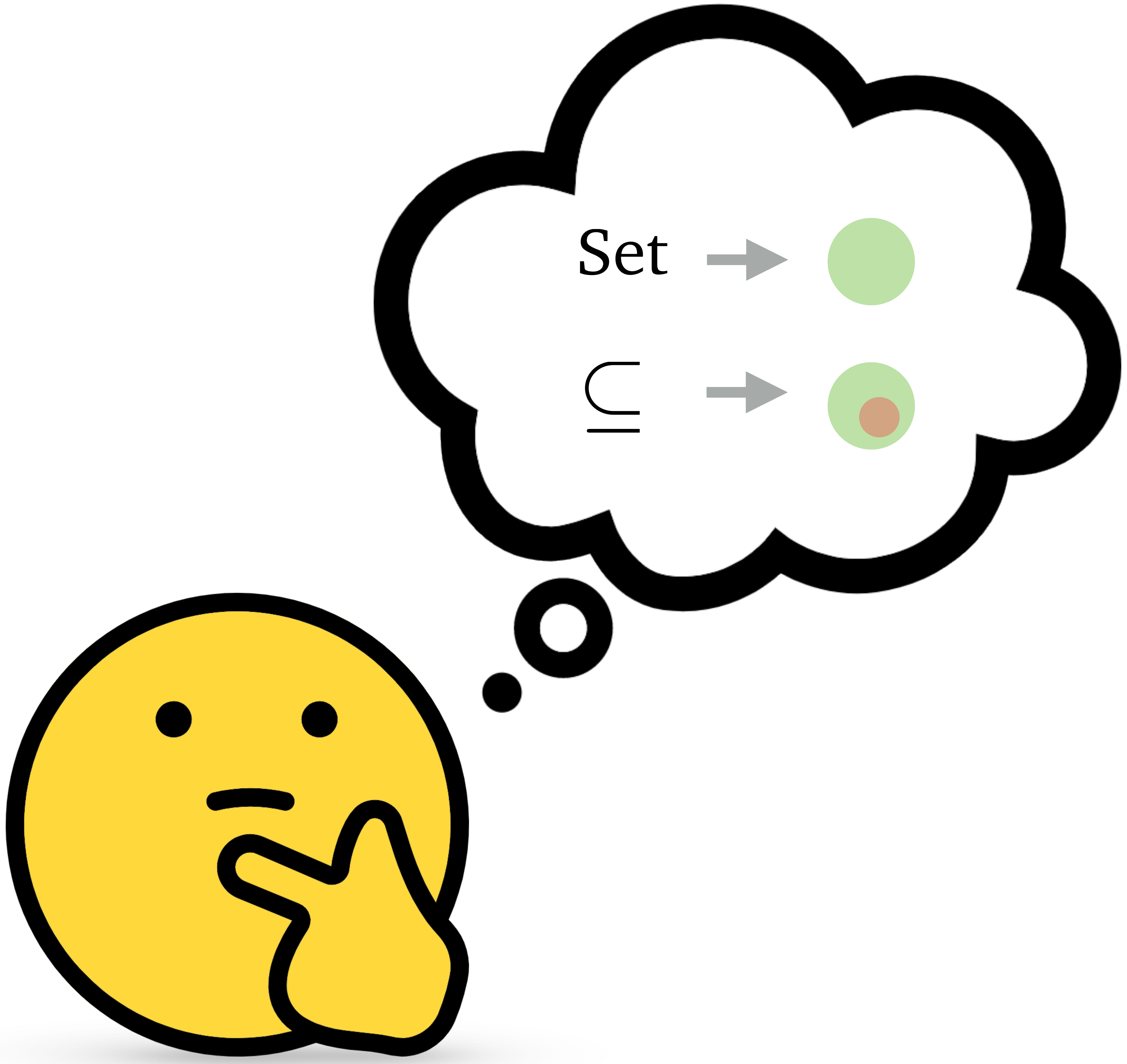
There exist sets A, B such that $B \subseteq A$.

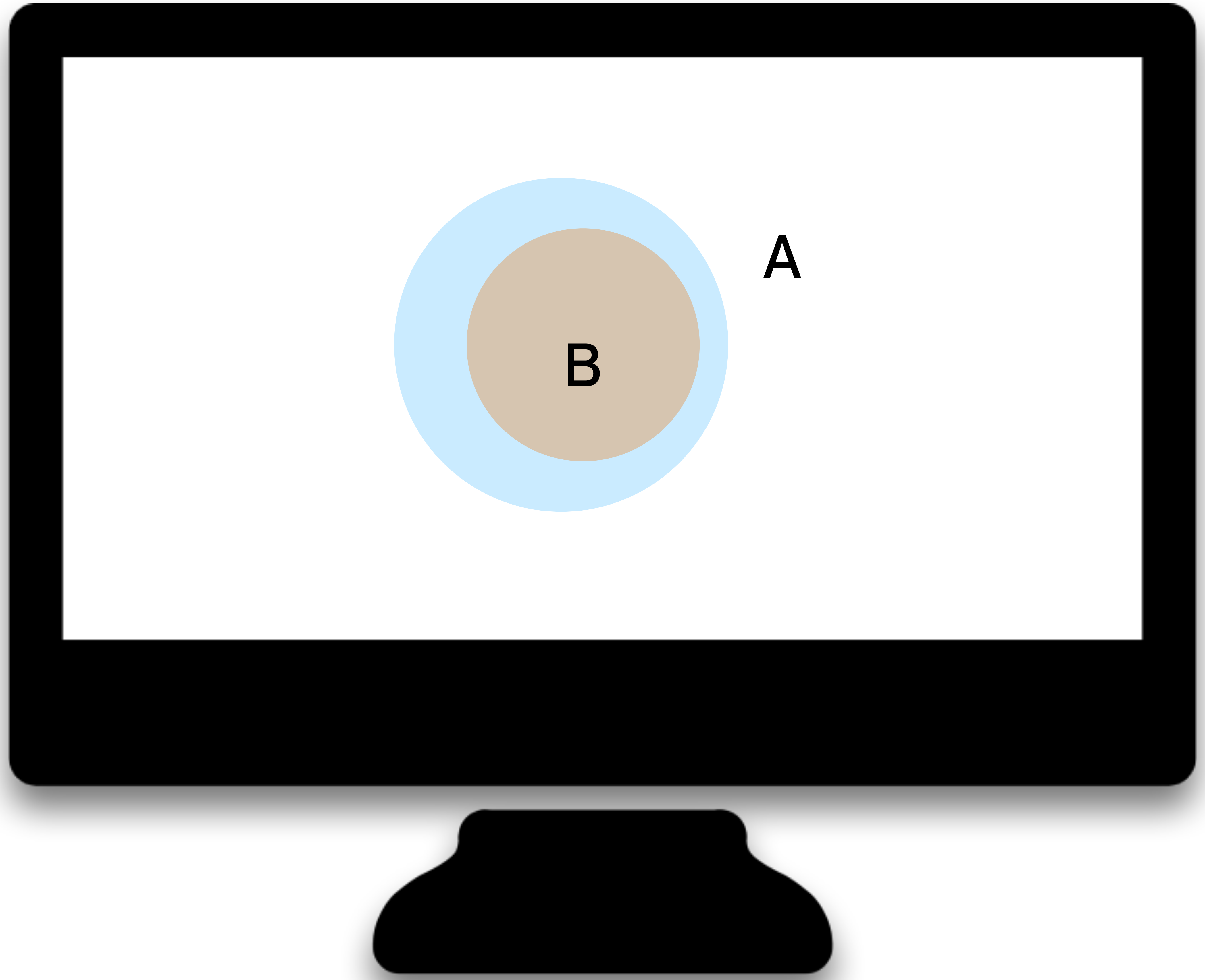




Modeling how people visualize abstract ideas by hand

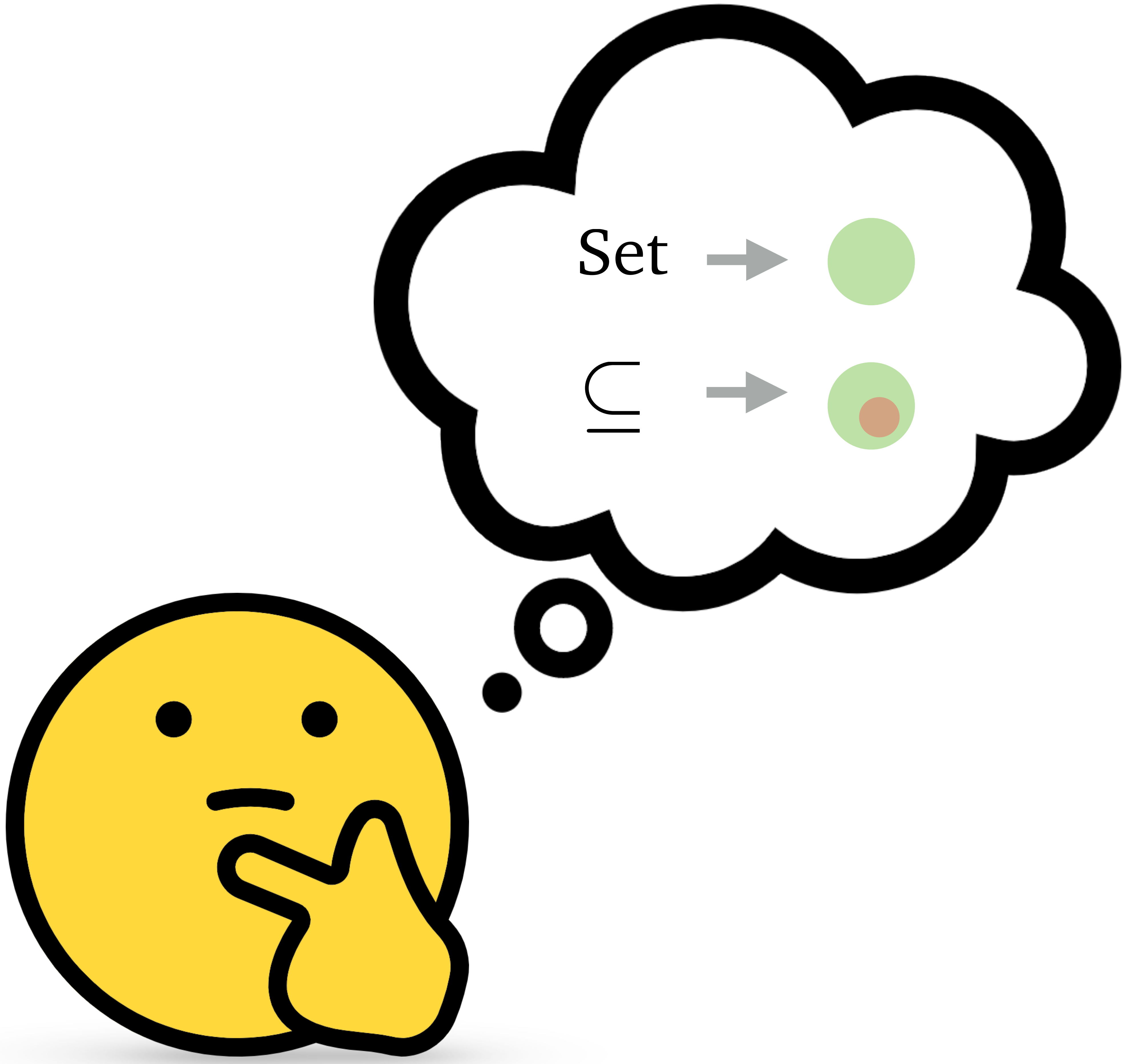
There exist sets A, B such that $B \subseteq A$.

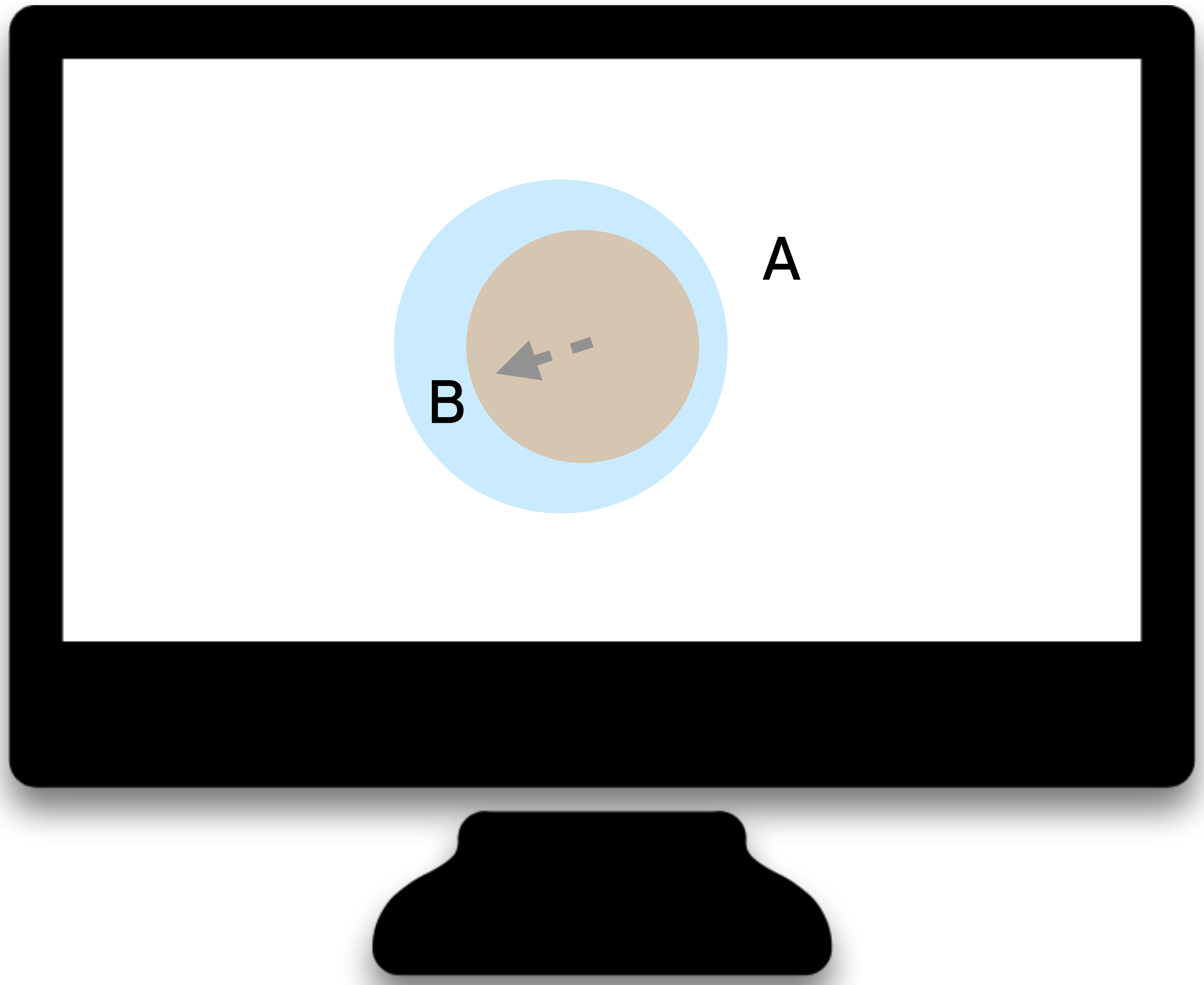




Modeling how people visualize abstract ideas by hand

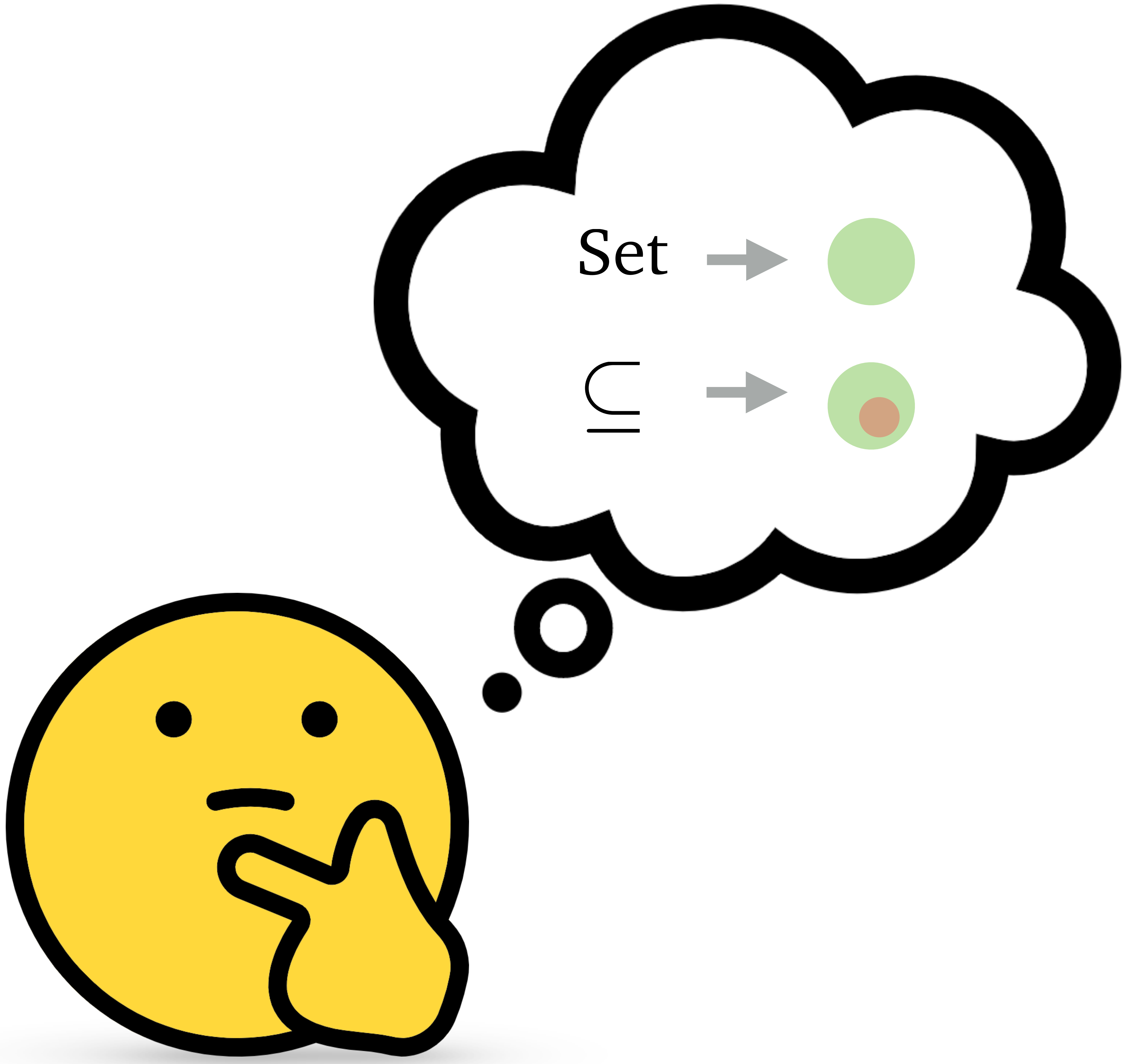
There exist sets A, B such that $B \subseteq A$.

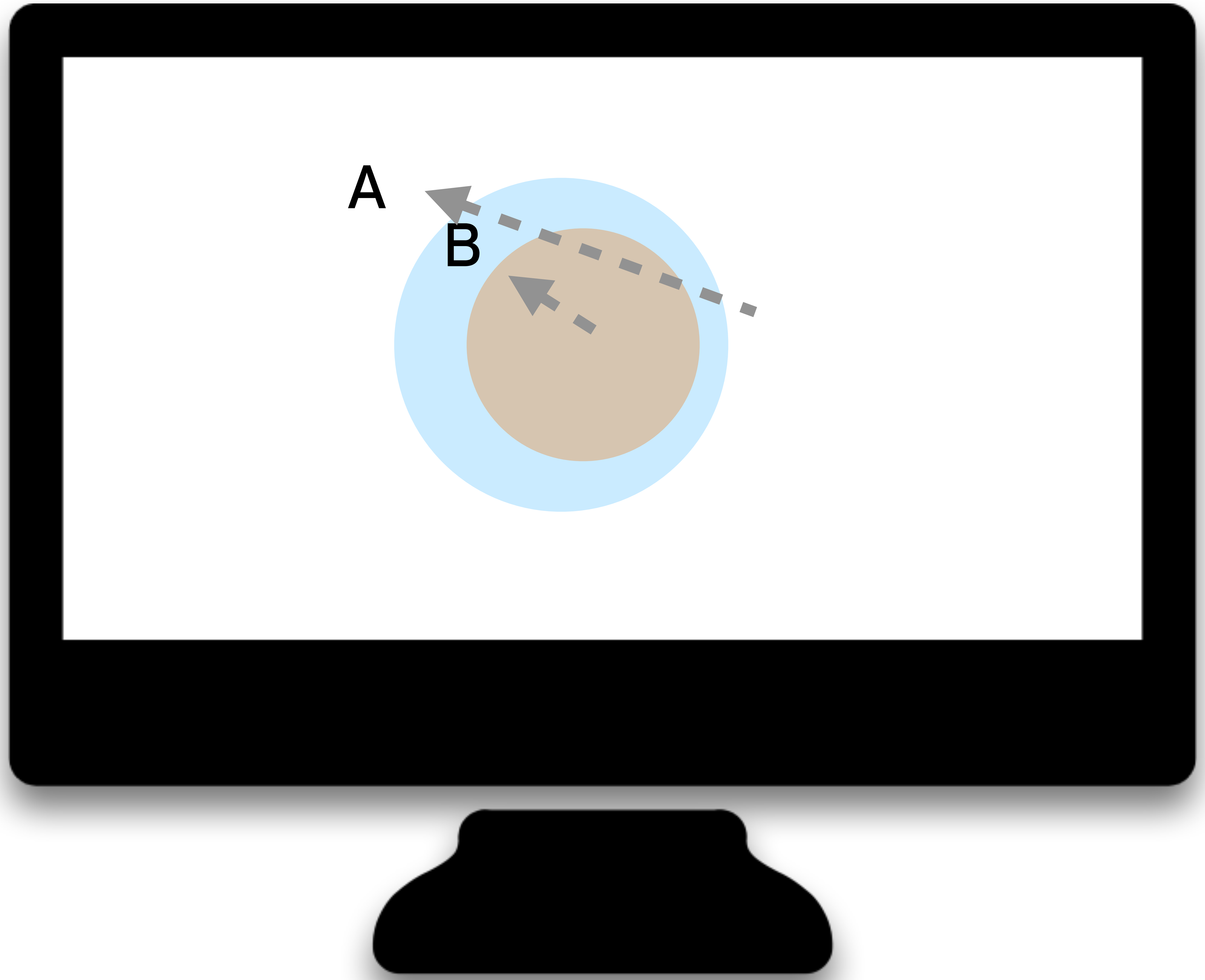




Modeling how people visualize abstract ideas by hand

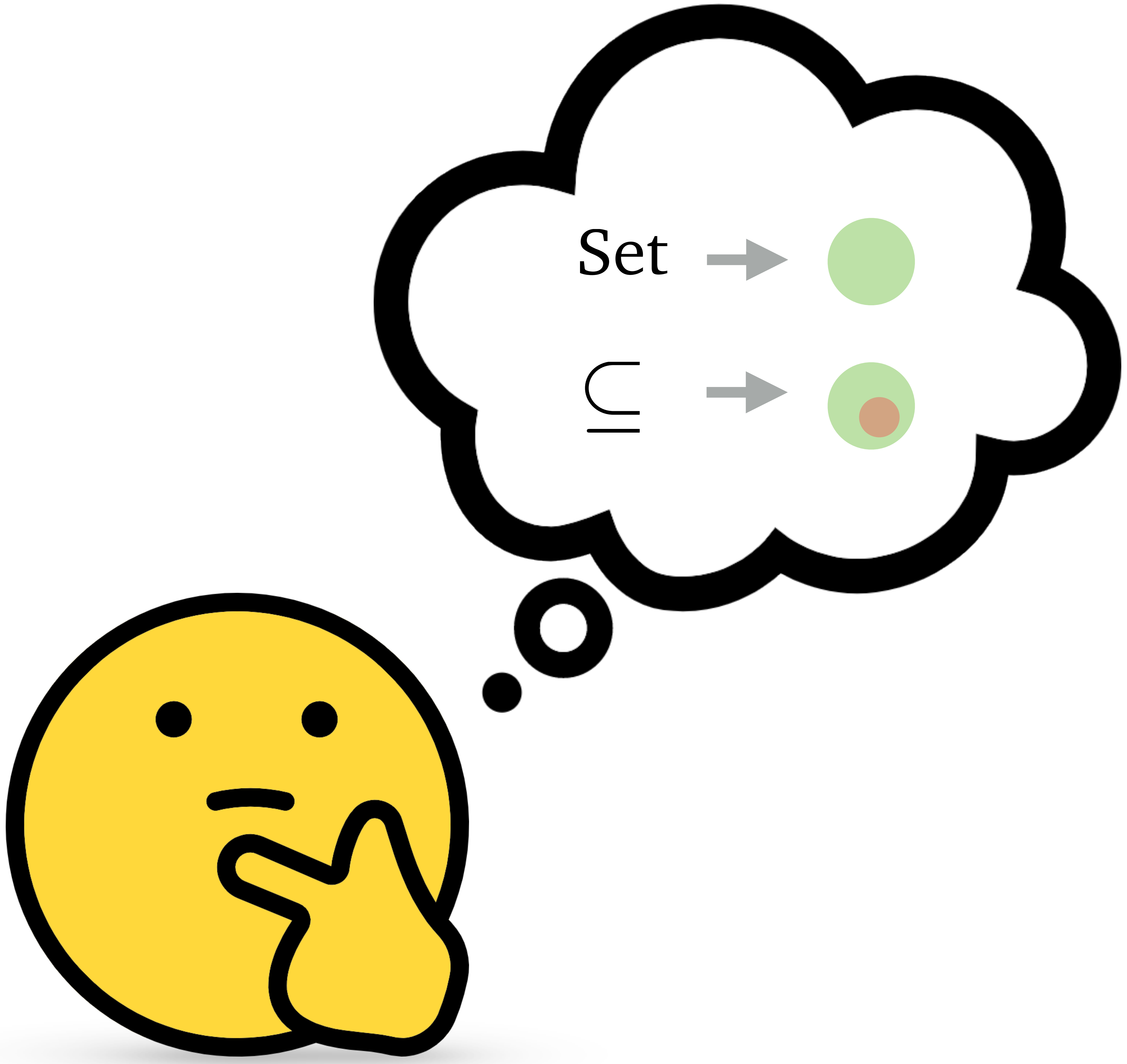
There exist sets A, B such that $B \subseteq A$.

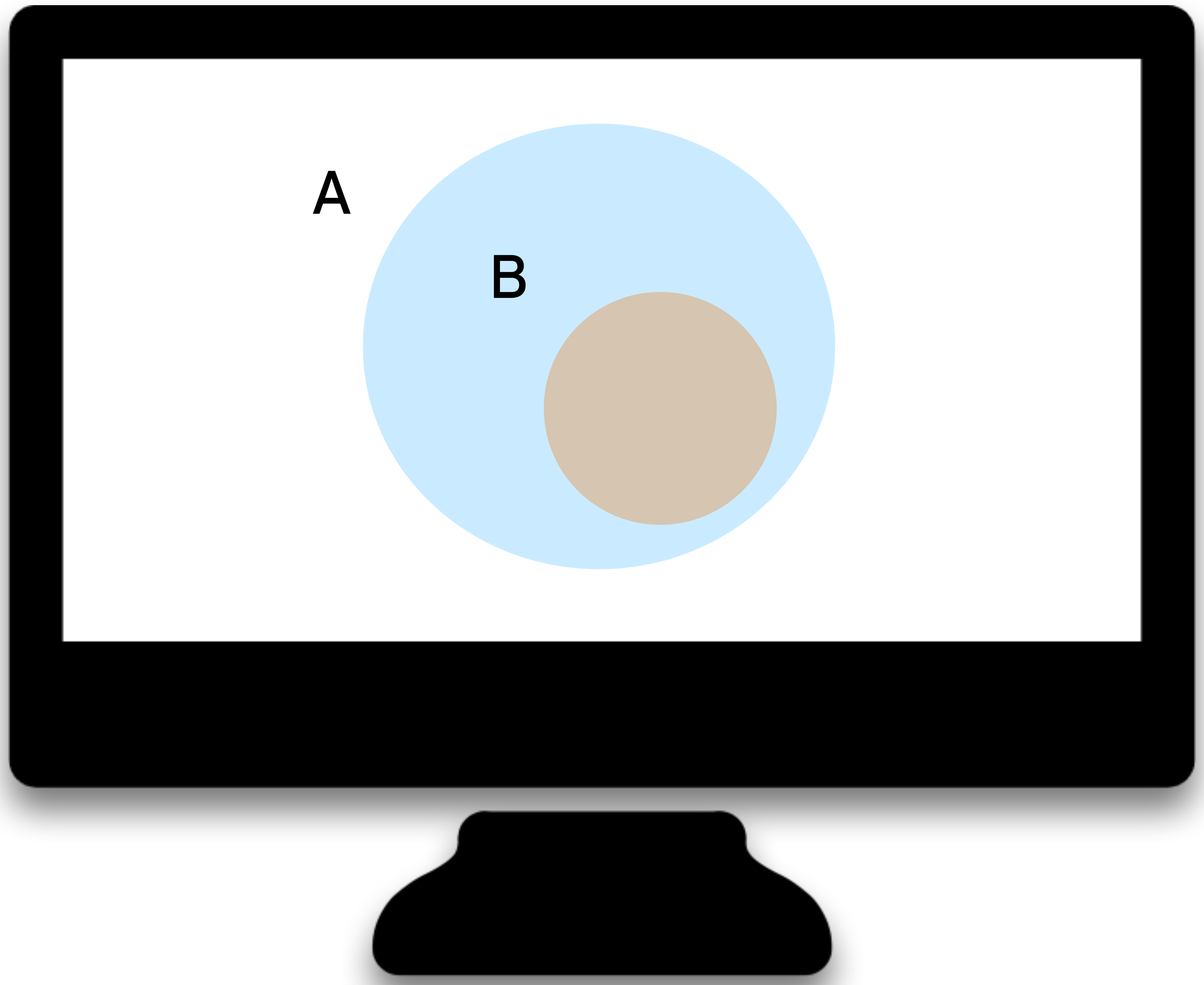




Modeling how people visualize abstract ideas by hand

There exist sets A, B such that $B \subseteq A$.





General design principles for math diagramming systems

There exist sets A, B such that $B \subseteq A$.

Specification

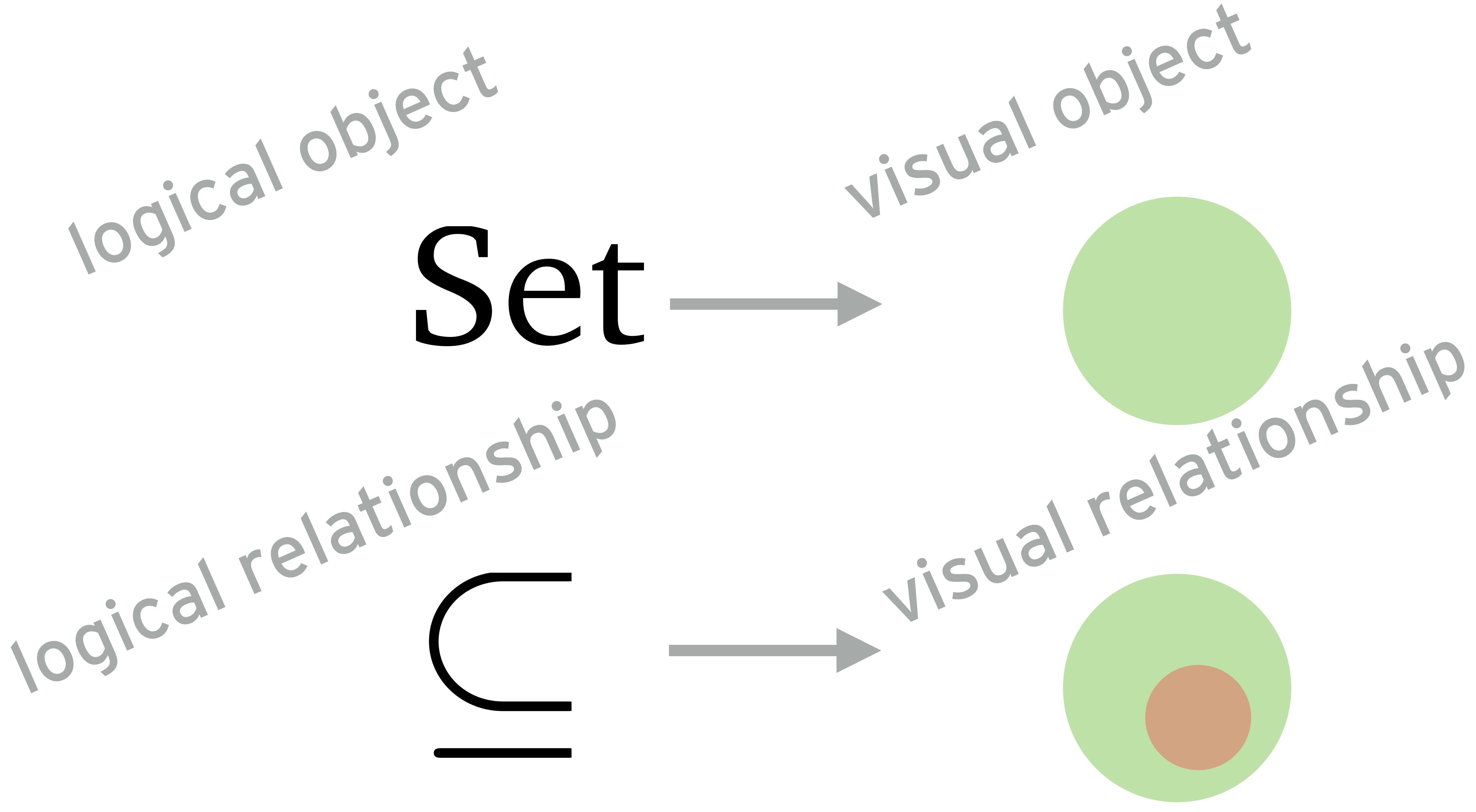
Synthesis

General design principles for math diagramming systems

There exist sets A, B such that $B \subseteq A$.

Specification

applying a formal mapping



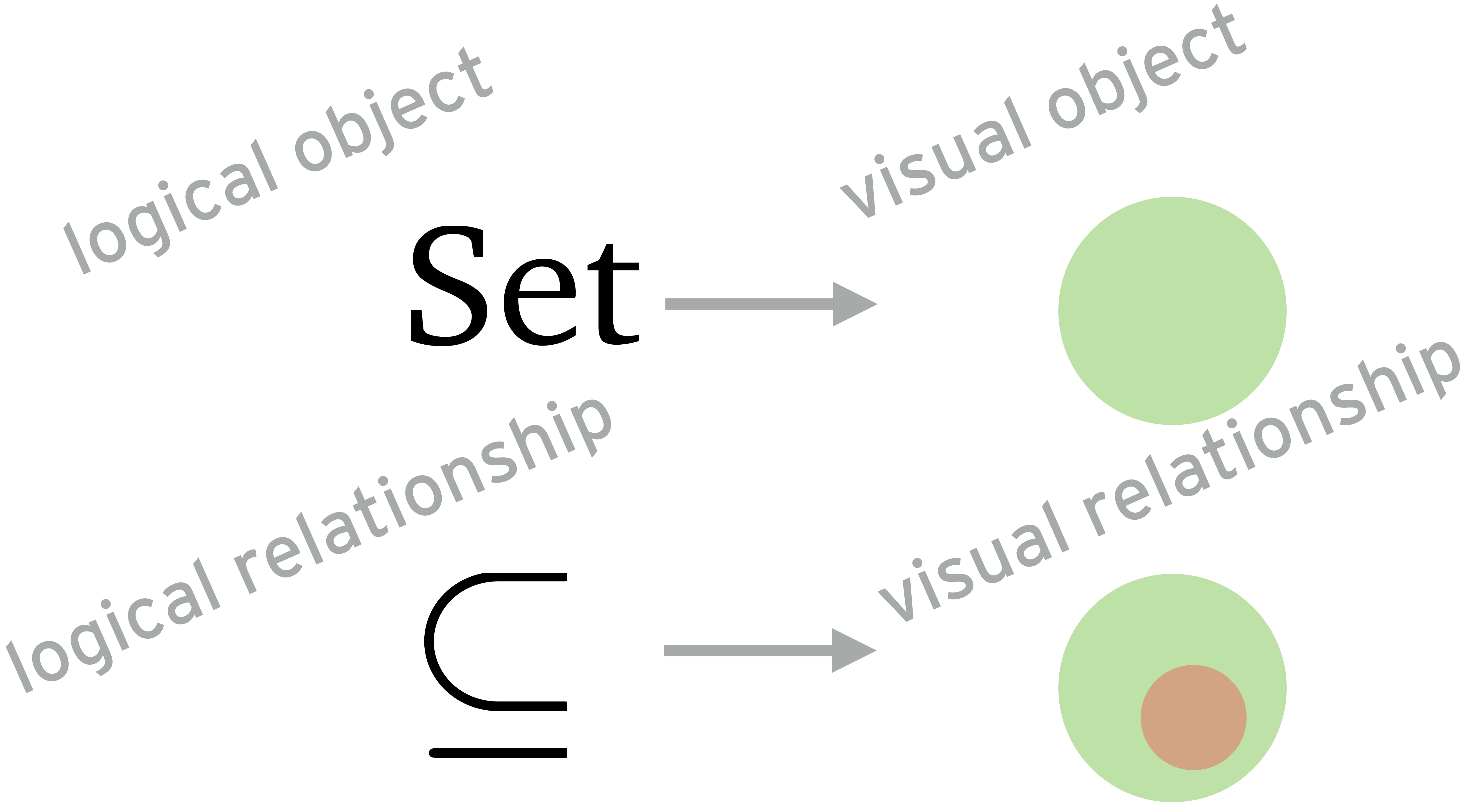
Synthesis

General design principles for math diagramming systems

There exist sets A, B such that $B \subseteq A$.

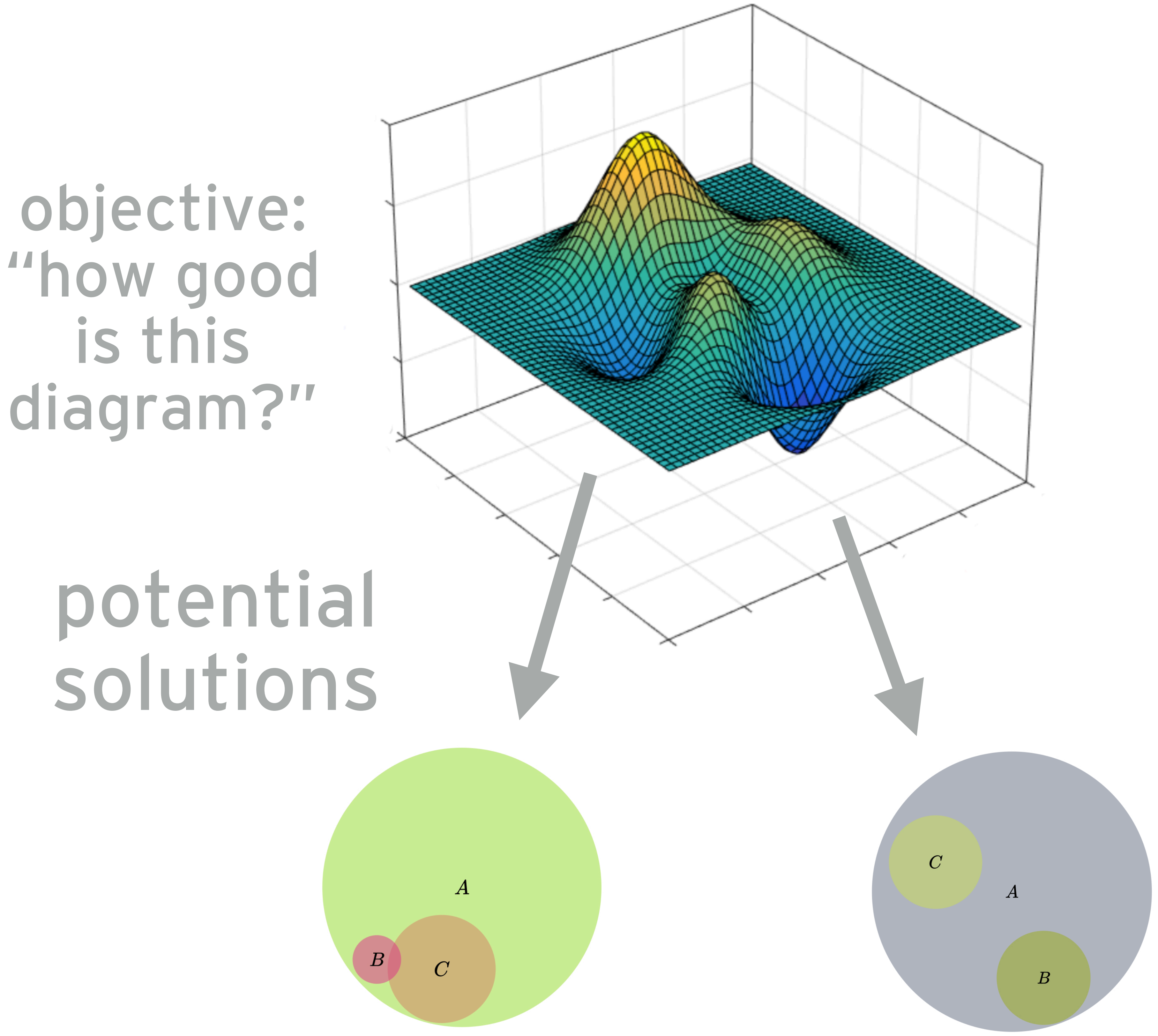
Specification

applying a formal mapping



Synthesis

finding a satisfying diagram



General design principles for math diagramming systems

There exist sets A, B such that $B \subseteq A$.

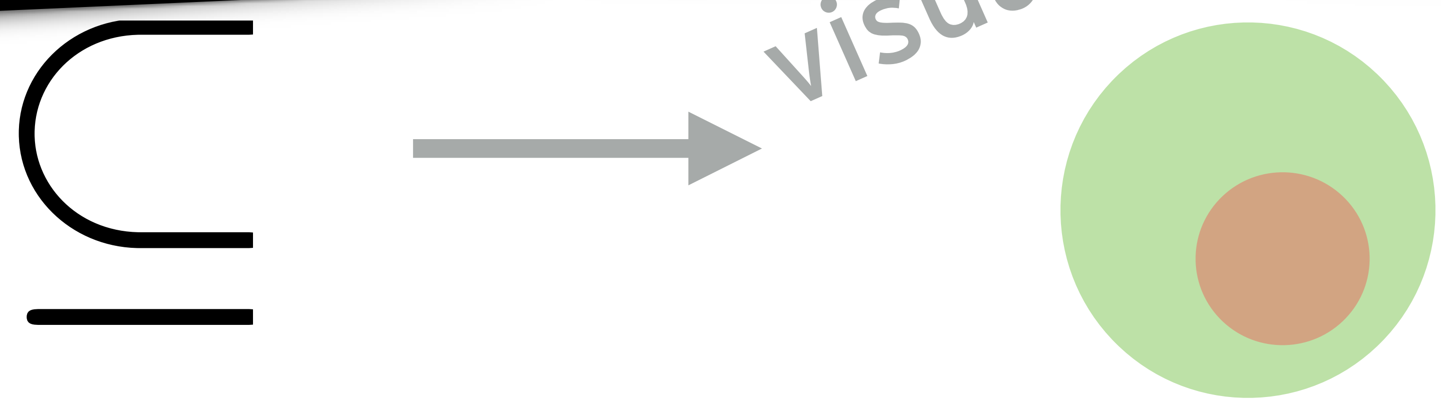
Specification

applying a formal mapping

good tool for the job:
domain-specific languages
+ compiler

logical

visual

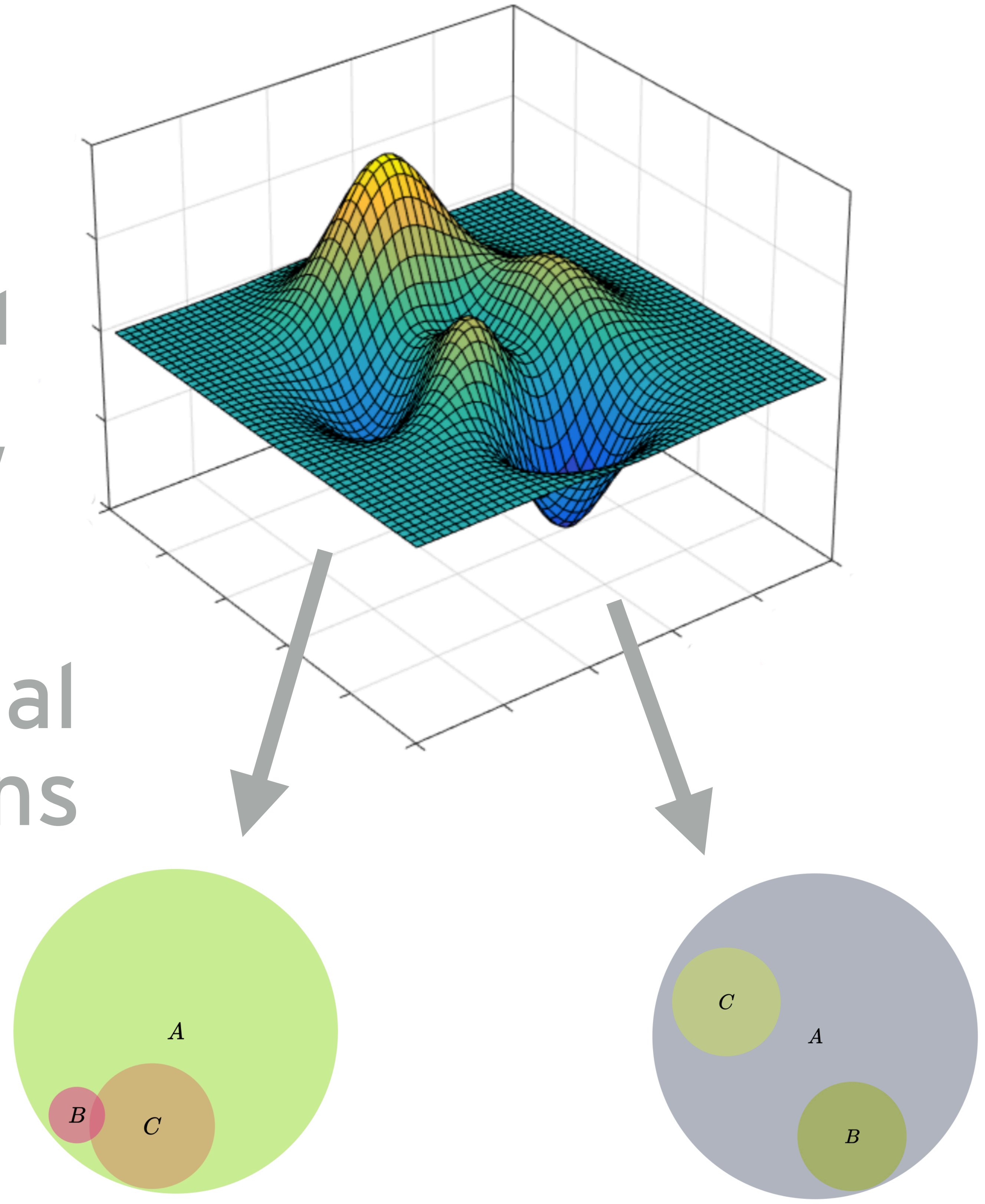


Synthesis

finding a satisfying diagram

objective:
"how good
is this
diagram?"

potential
solutions



General design principles for math diagramming systems

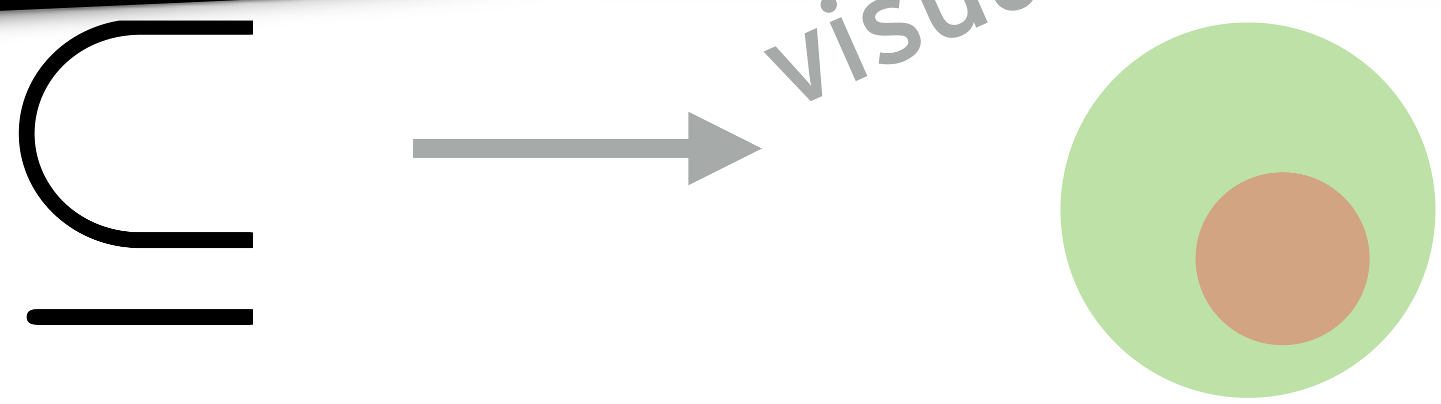
There exist sets A, B such that $B \subseteq A$.

Specification

applying a formal mapping

good tool for the job: domain-specific languages + compiler

logical

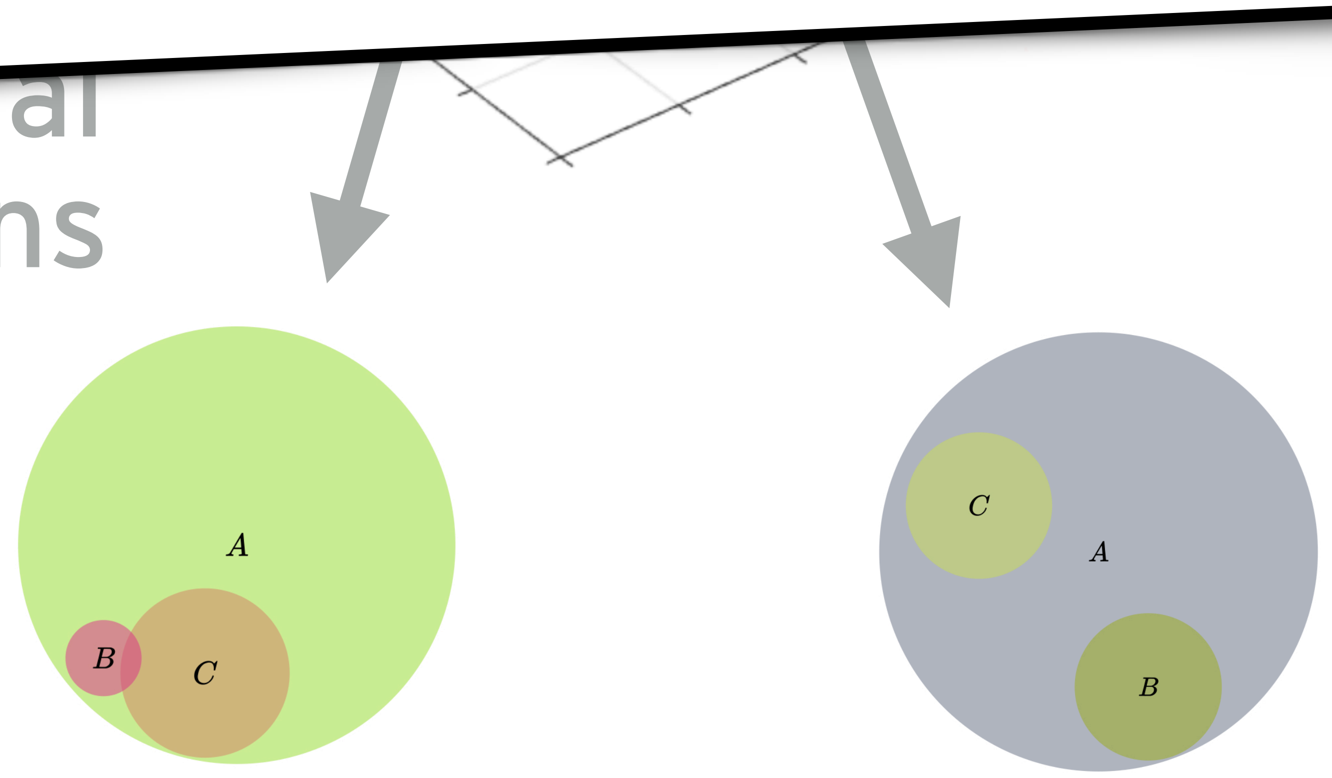


Synthesis

finding a satisfying diagram

good tool for the job: constrained optimization + numerical solver

potential solutions



Building these design principles into a system

Specification

Synthesis

Building these design principles into a system

Specification

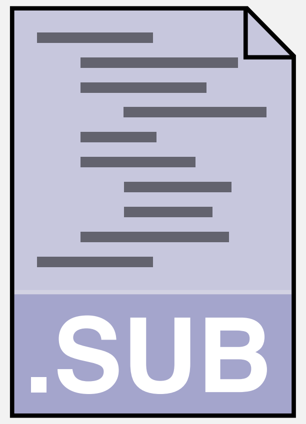
Synthesis

source code

Building these design principles into a system

Specification

source code



Substance
mathematical
content

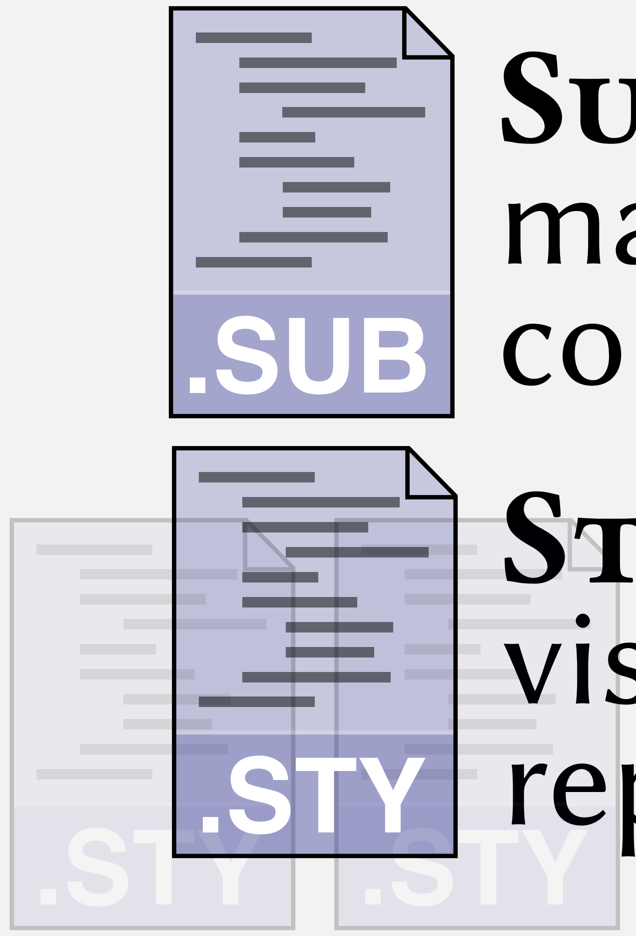
Synthesis

Building these design principles into a system

Specification

Synthesis

source code



Substance
mathematical content

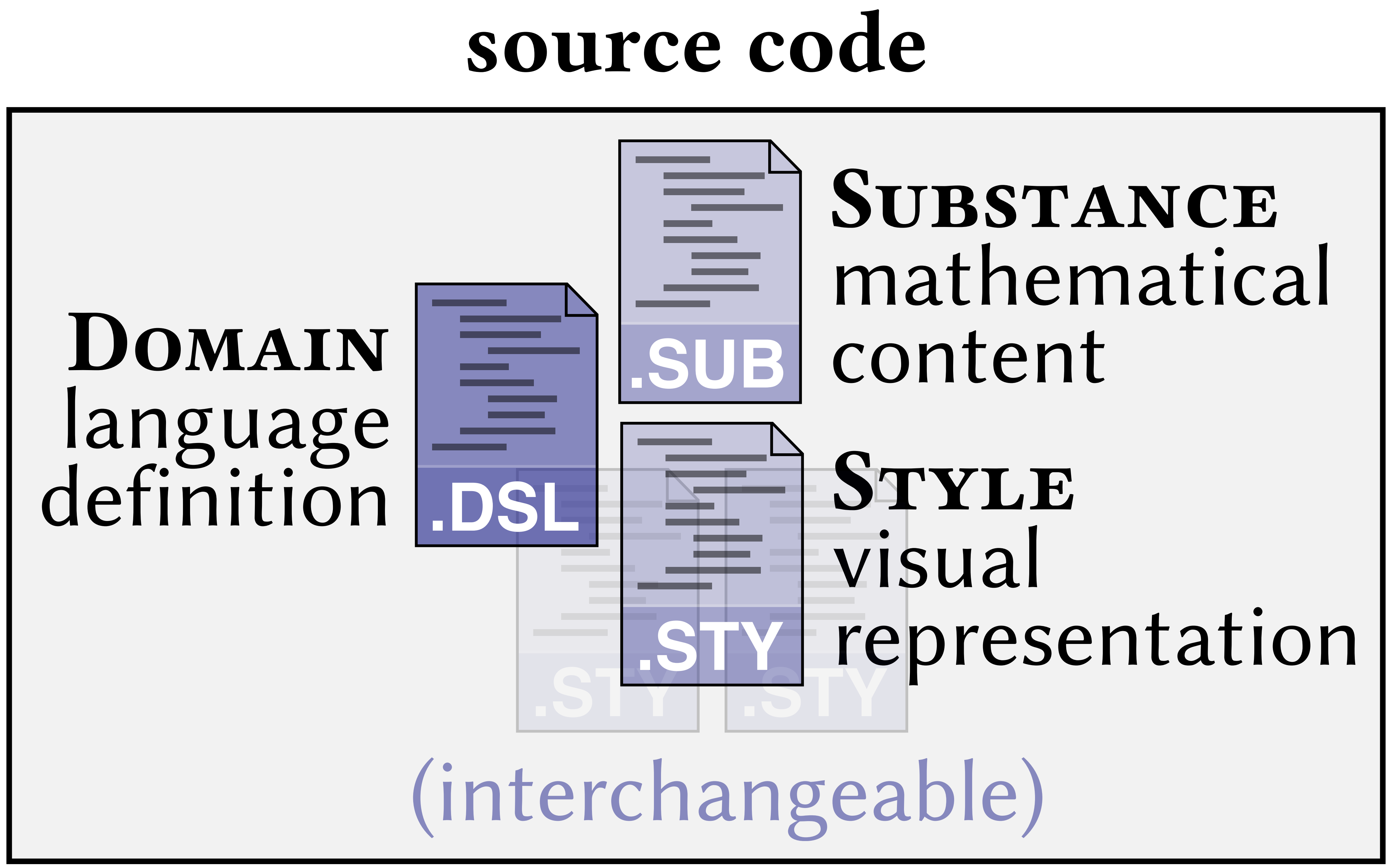
Style
visual representation

(interchangeable)

Building these design principles into a system

Specification

Synthesis

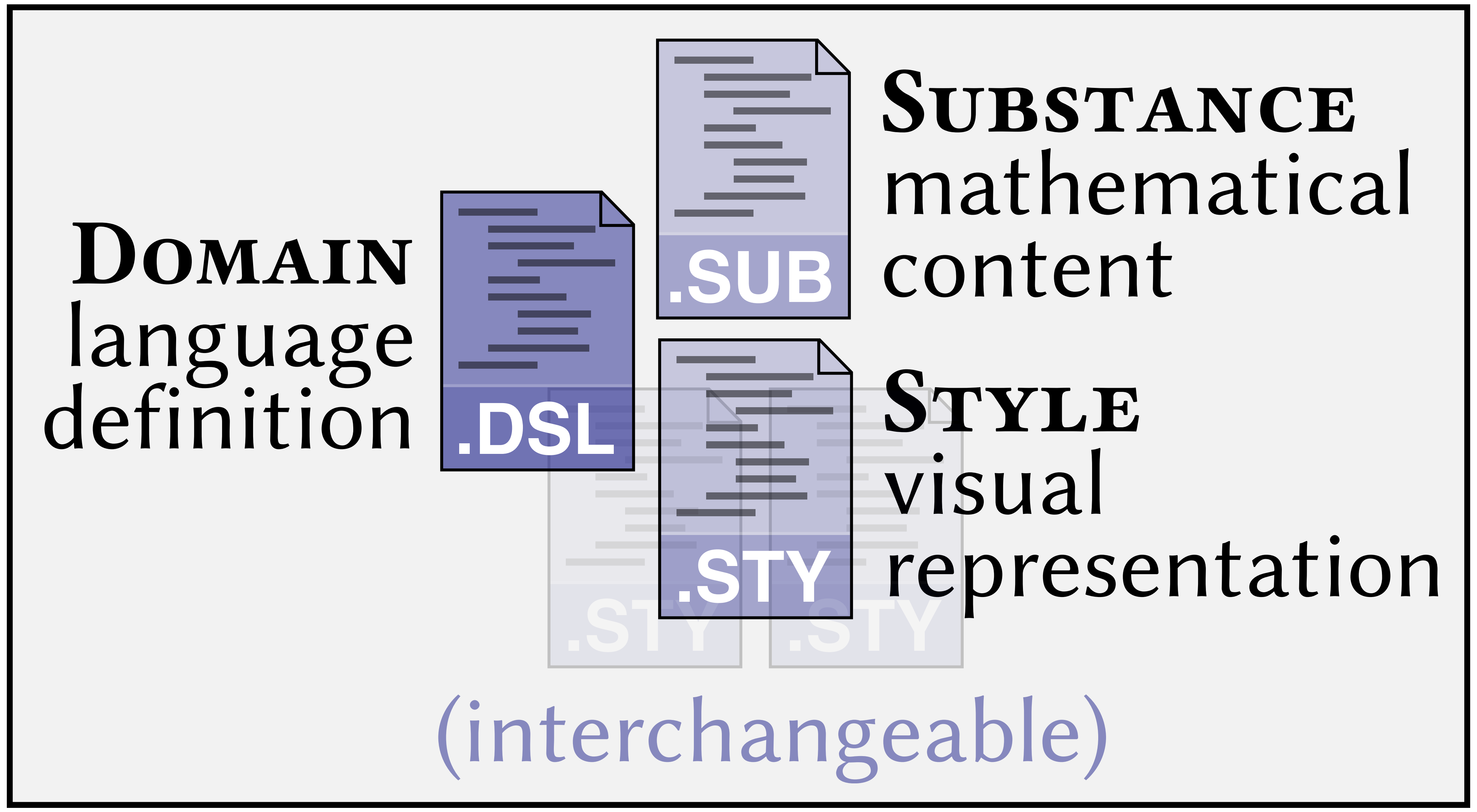


Building these design principles into a system

Specification

Synthesis

source code



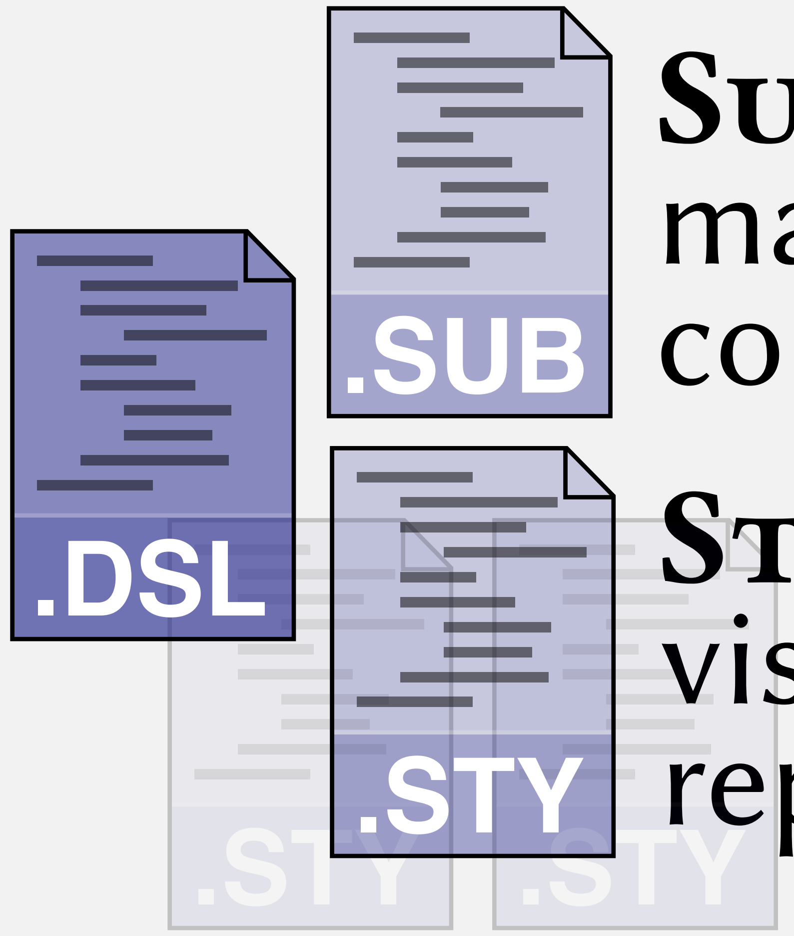
compiler

Building these design principles into a system

Specification

source code

Domain language definition



Substance mathematical content

Style visual representation

(interchangeable)

compiler →

Synthesis

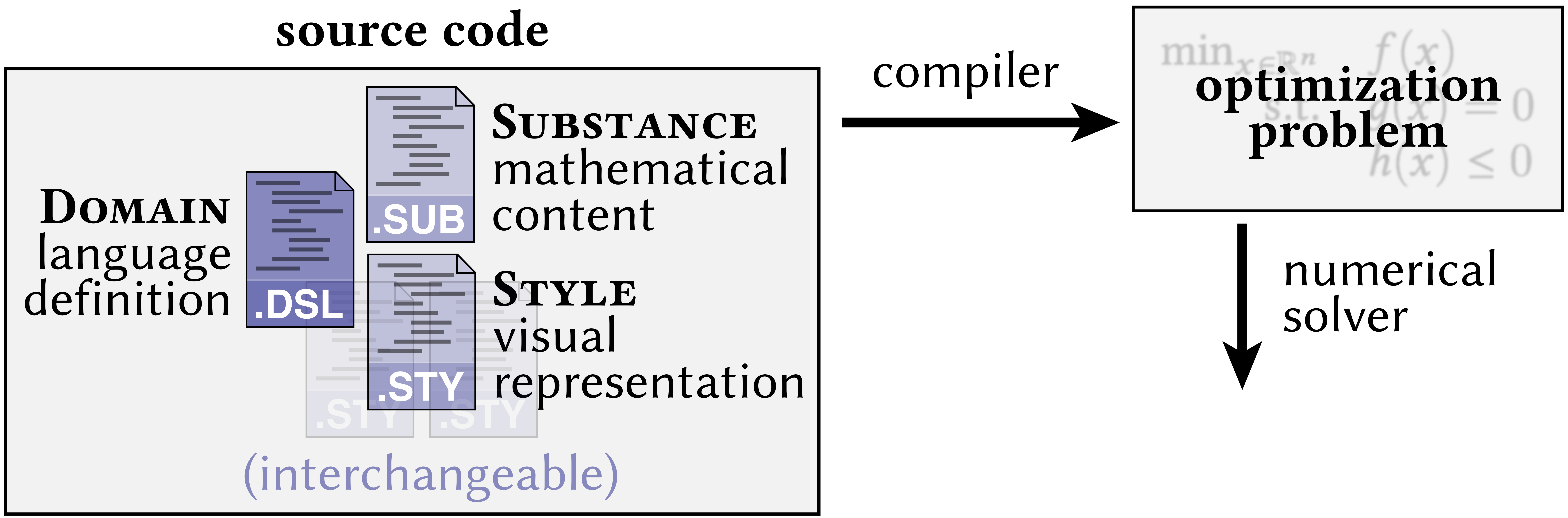
optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) \quad \text{s.t. } g(x) = 0,\; h(x) \leq 0$$

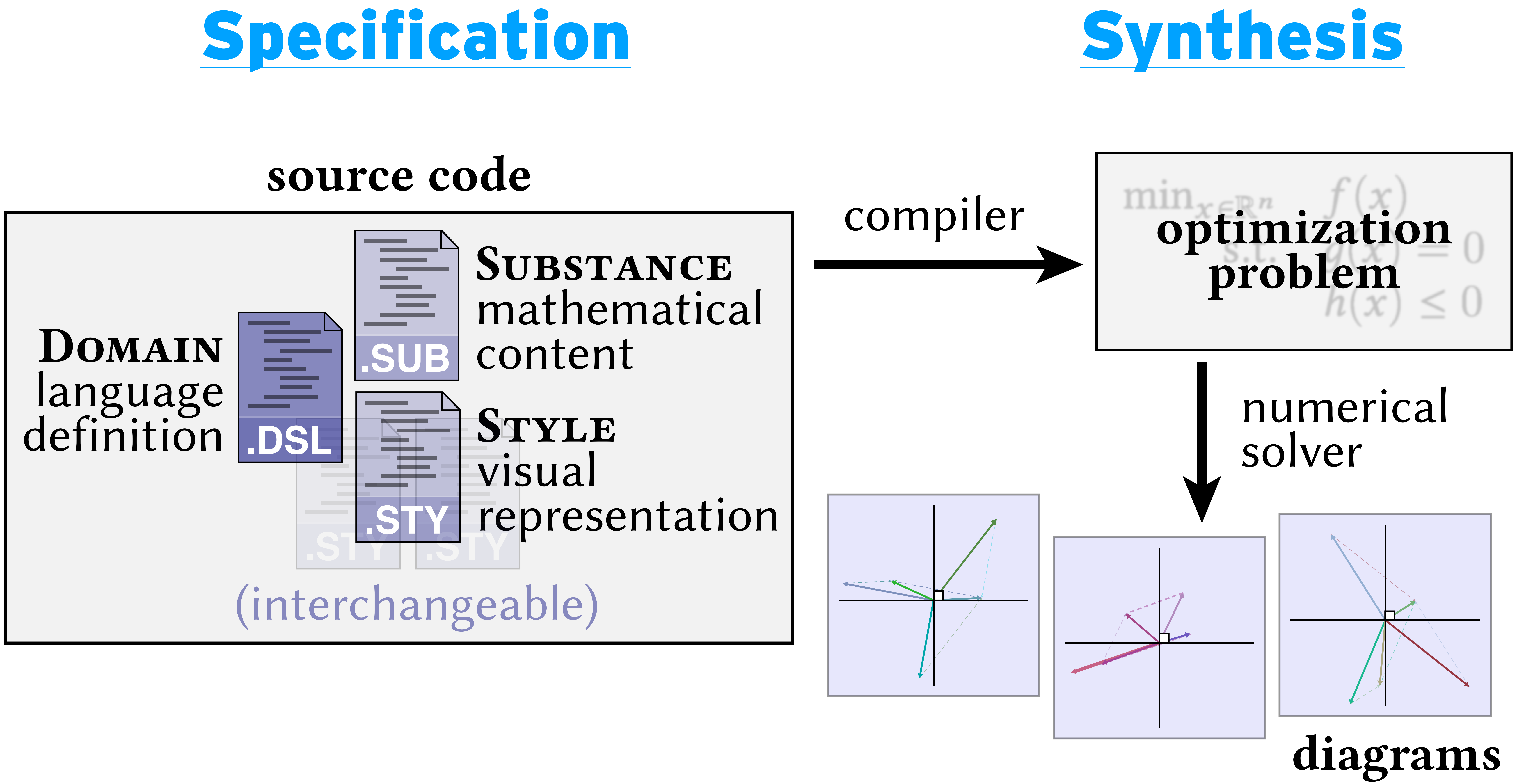
Building these design principles into a system

Specification

Synthesis



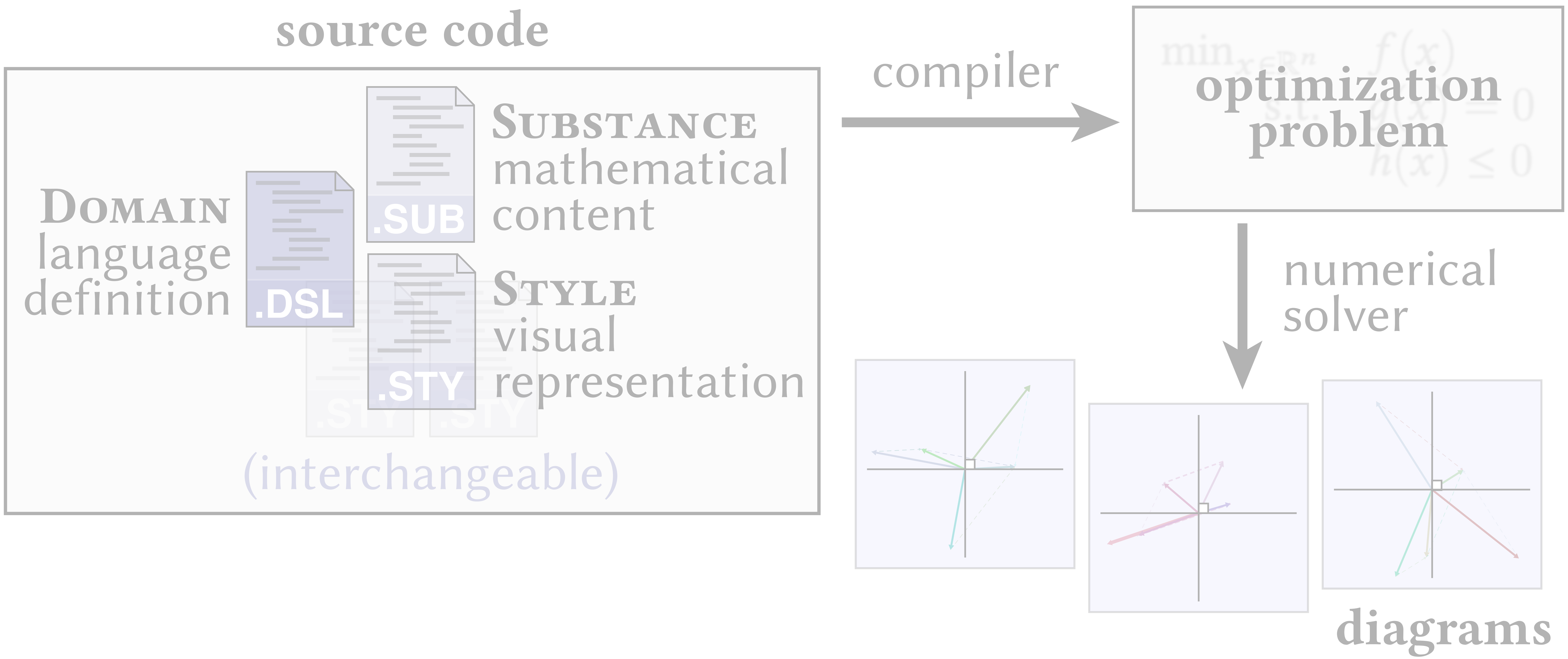
Building these design principles into a system



Webpages follow similar design principles

Specification

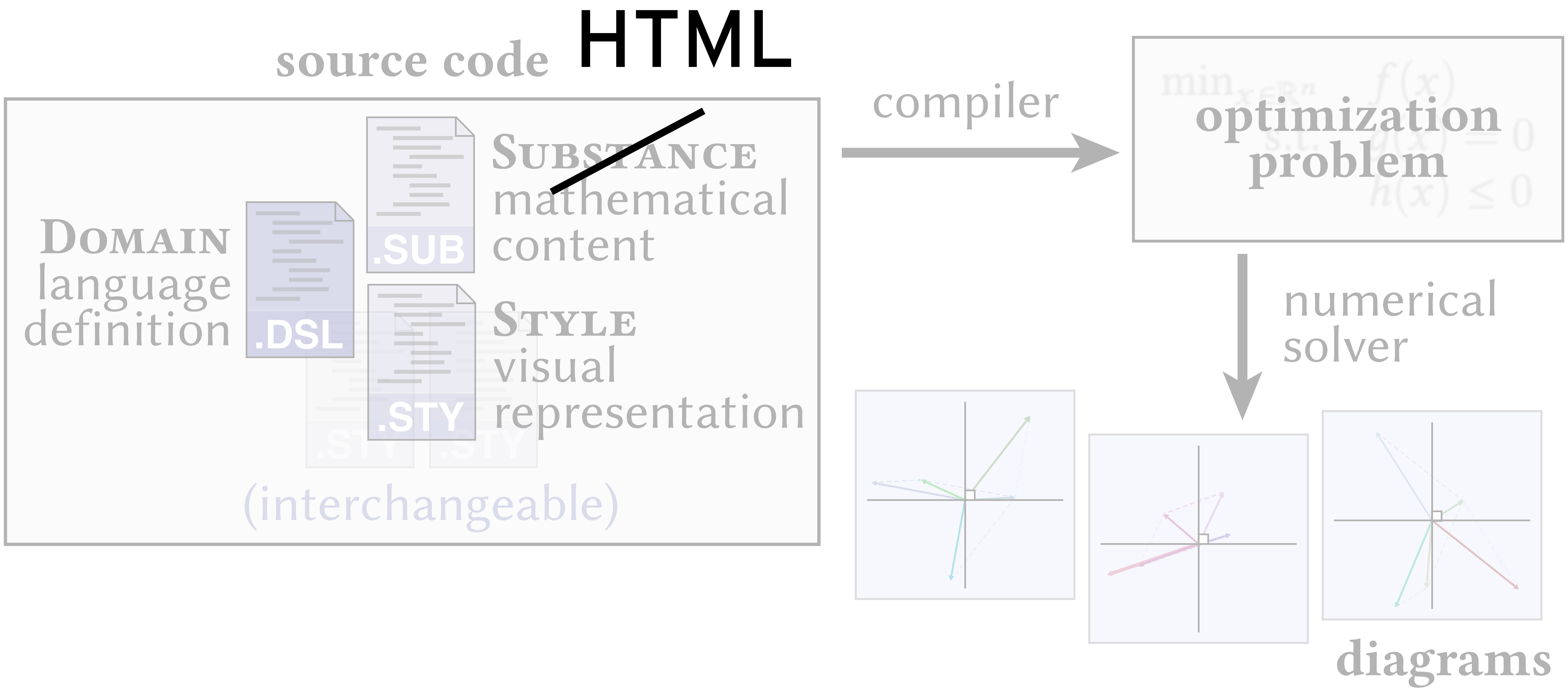
Synthesis



Webpages follow similar design principles

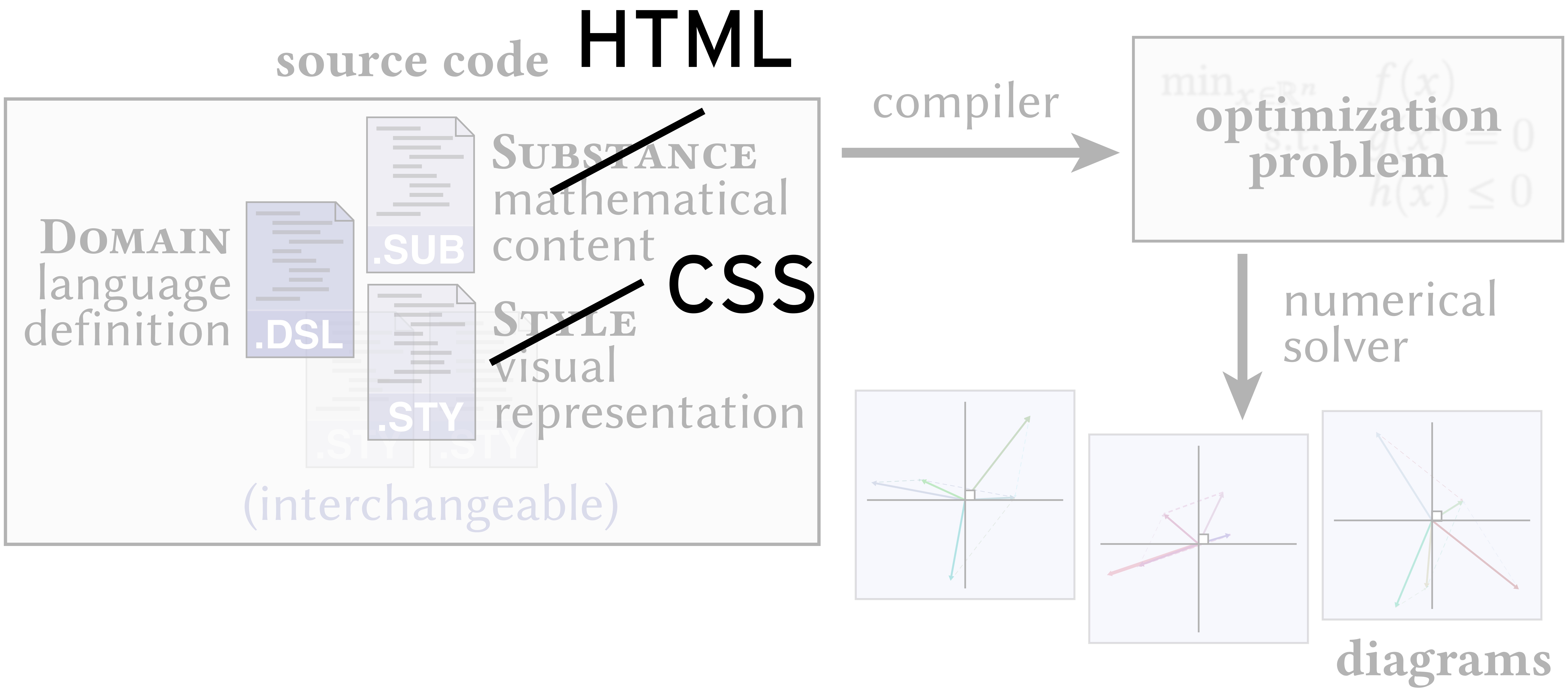
Specification

Synthesis



Webpages follow similar design principles

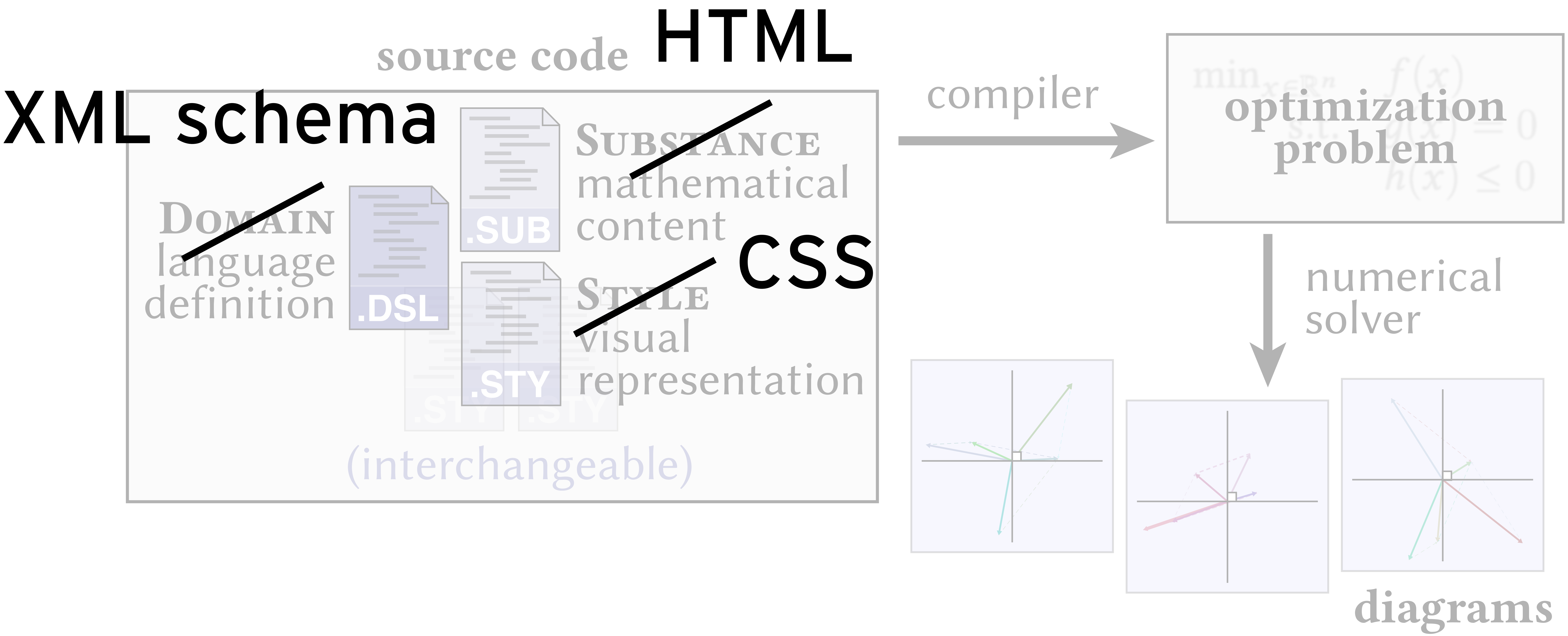
Specification **Synthesis**



Webpages follow similar design principles

Specification

Synthesis



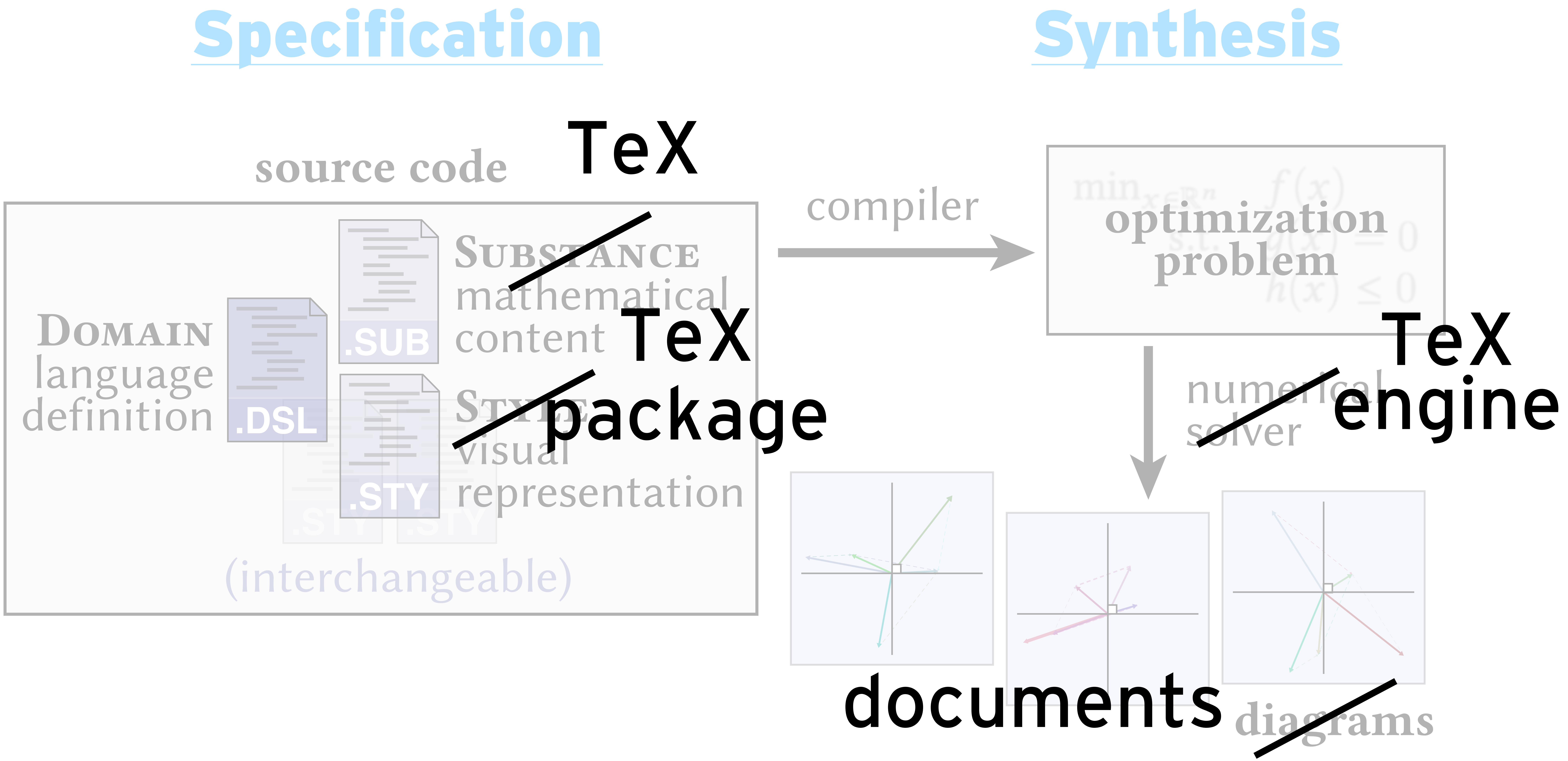
Webpages follow similar design principles

Webpages follow similar design principles

Specification

Synthesis

TeX/LaTeX follows similar design principles

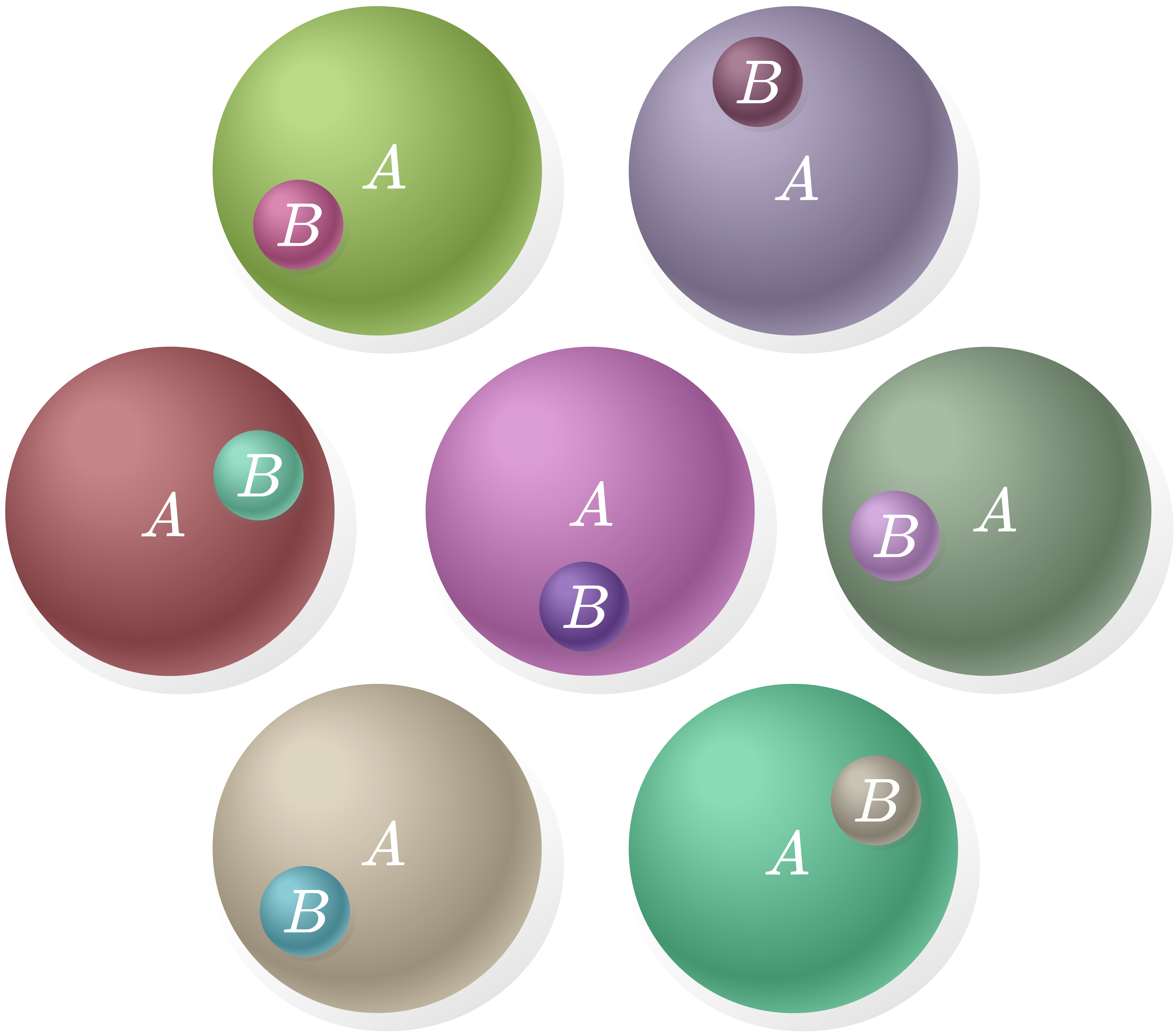


System walkthrough

A simple example

A simple example

```
Set A, B
B ⊂ A
```



Substance defines the content of a diagram

There exist sets A *and* B
such that B *is a subset of* A.

Substance defines the content of a diagram

specify objects

There exist sets A *and* B
such that B *is a subset of* A.

Substance defines the content of a diagram

specify objects

There exist sets A *and* B
such that B *is a subset of* A.

specify a relationship

Substance defines the content of a diagram

There exist sets A *and* B
such that B *is a subset of* A.

specify a
relationship

Substance defines the content of a diagram

There exist sets A *and* B
such that B *is a subset of* A.

Substance defines the content of a diagram

There exist sets A *and* B
such that B *is a subset of* A.

Substance defines the content of a diagram

There exist sets A *and* B
such that B *is a subset of* A.

Substance defines the content of a diagram

There exist sets A *and* B *such that* B *is a subset of* A.

```
Set A, B
B ⊂ A
```

Substance defines the content of a diagram

There exist sets A *and* B
such that B *is a subset of* A.

specify objects

```
Set A, B
B ⊂ A
```

Substance defines the content of a diagram

There exist sets A *and* B *such that* B *is a subset of* A.

specify objects

```
Set A, B
B ⊂ A
```

specify a relationship

Substance is compositional

"Consider sets A, B, and C such that B is a subset of A, C is a subset of A, and B and C don't intersect, and …"

Substance is compositional

"Consider sets *A*, *B*, and *C* such that *B* is a subset of *A*, *C* is a subset of *A*, and *B* and *C* don't intersect, and …"

```
Set A, B, C
B ⊂ A
C ⊂ A
B ∩ C = ∅
…
```

Style defines a domain's visual representation

data

```
Saffron 58
Ginger 42
Cumin 34
Turmeric 24
Allspice 2
Paprika 1
Cinnamon 0.6
Cardamom 0.4
Mace 0.2
```

logical statements

```
Set A, B, C, D, E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

Style defines a domain's visual representation

data

```
Saffron 58
Ginger 42
Cumin 34
Turmeric 24
Allspice 2
Paprika 1
Cinnamon 0.6
Cardamom 0.4
Mace 0.2
```

logical statements

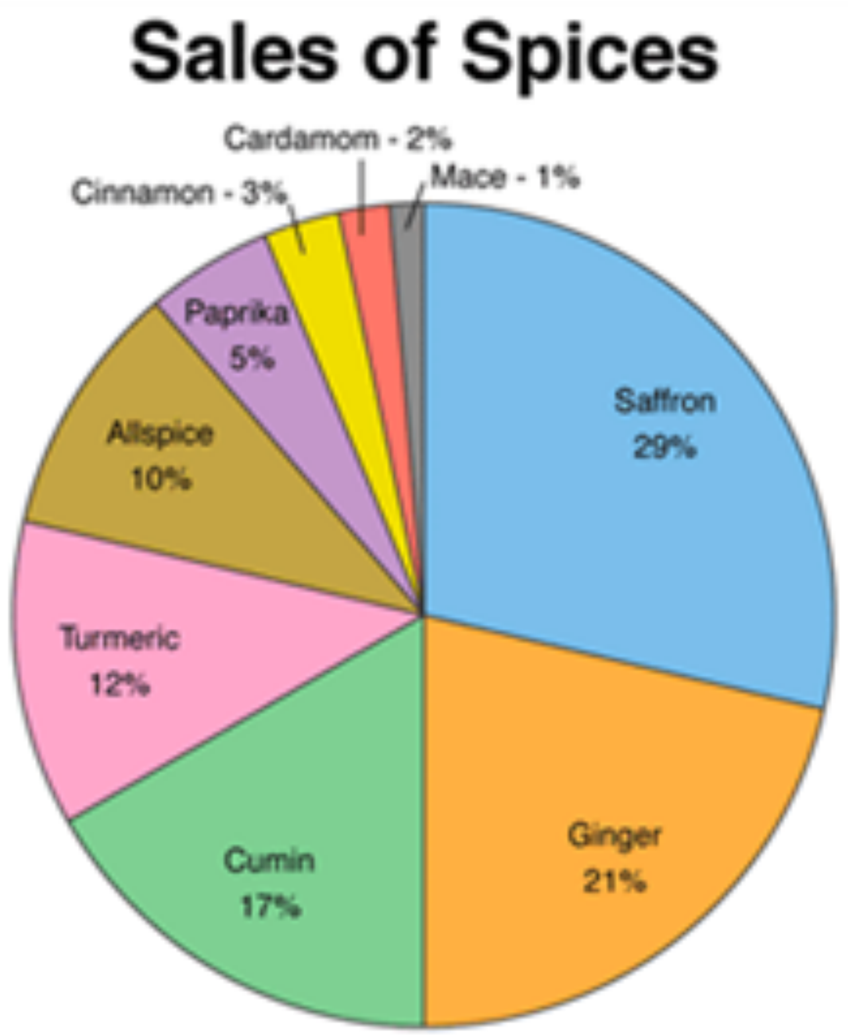
```
Set A, B, C, D, E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

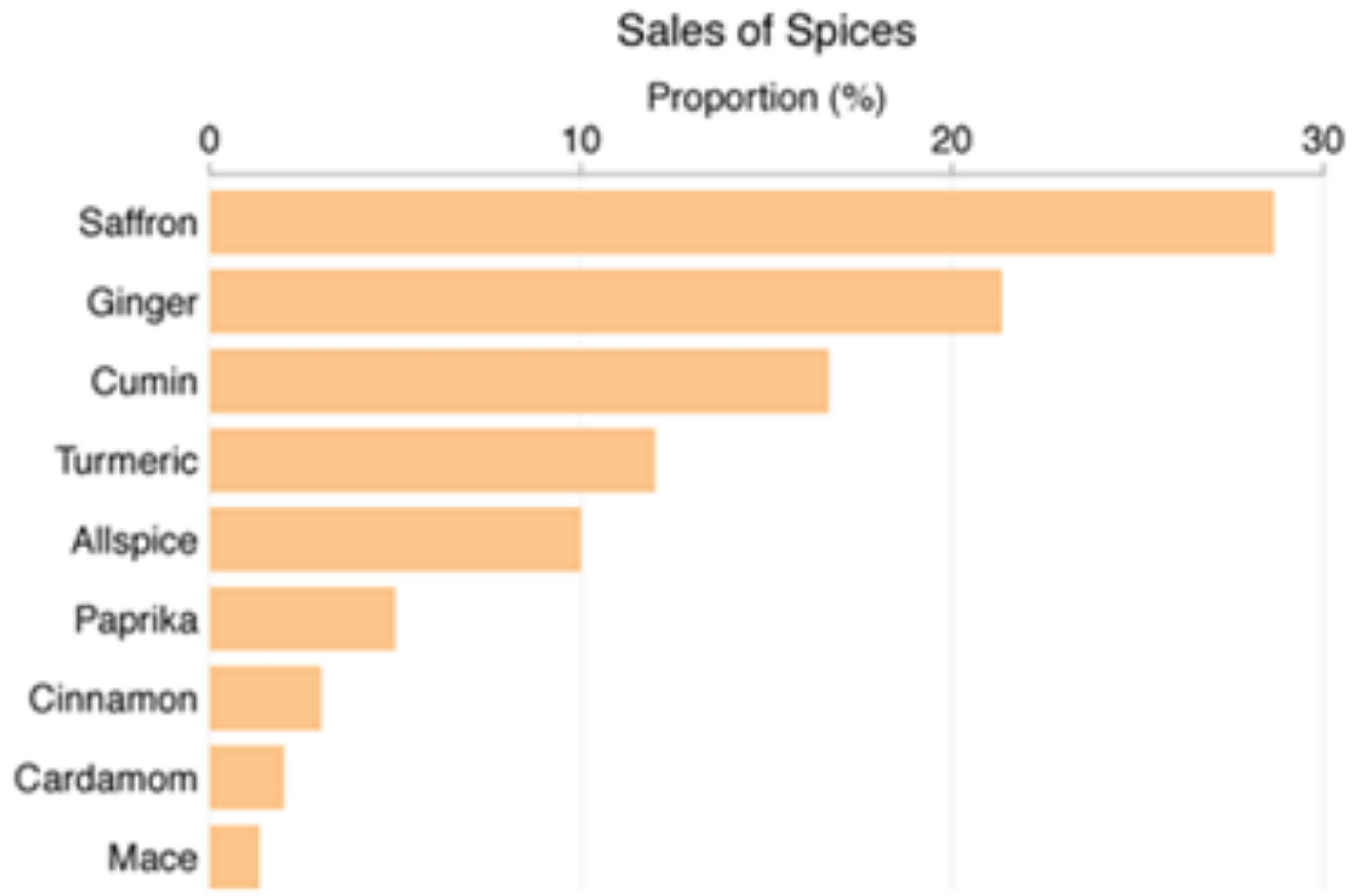
What visual representation best conveys the information?
(It's not inherent in the information!)

Style defines a domain's visual representation

data

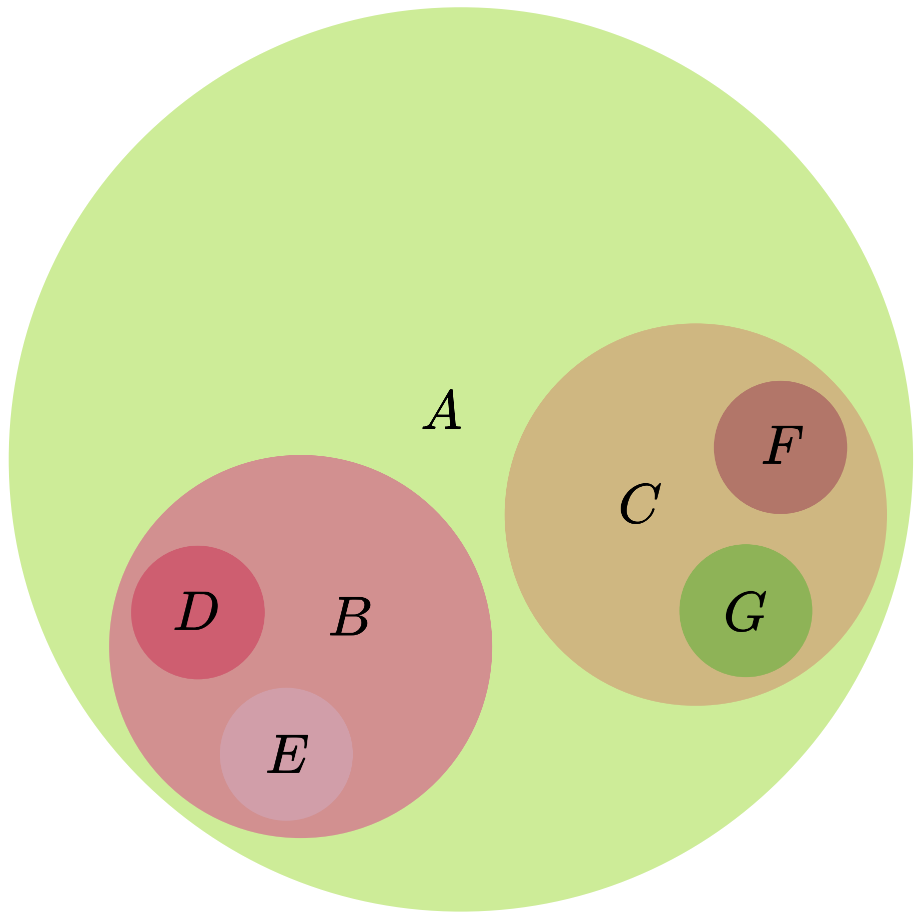
```
Saffron 58
Ginger 42
Cumin 34
Turmeric 24
Allspice 2
Paprika 1
Cinnamon 0.6
Cardamom 0.4
Mace 0.2
```

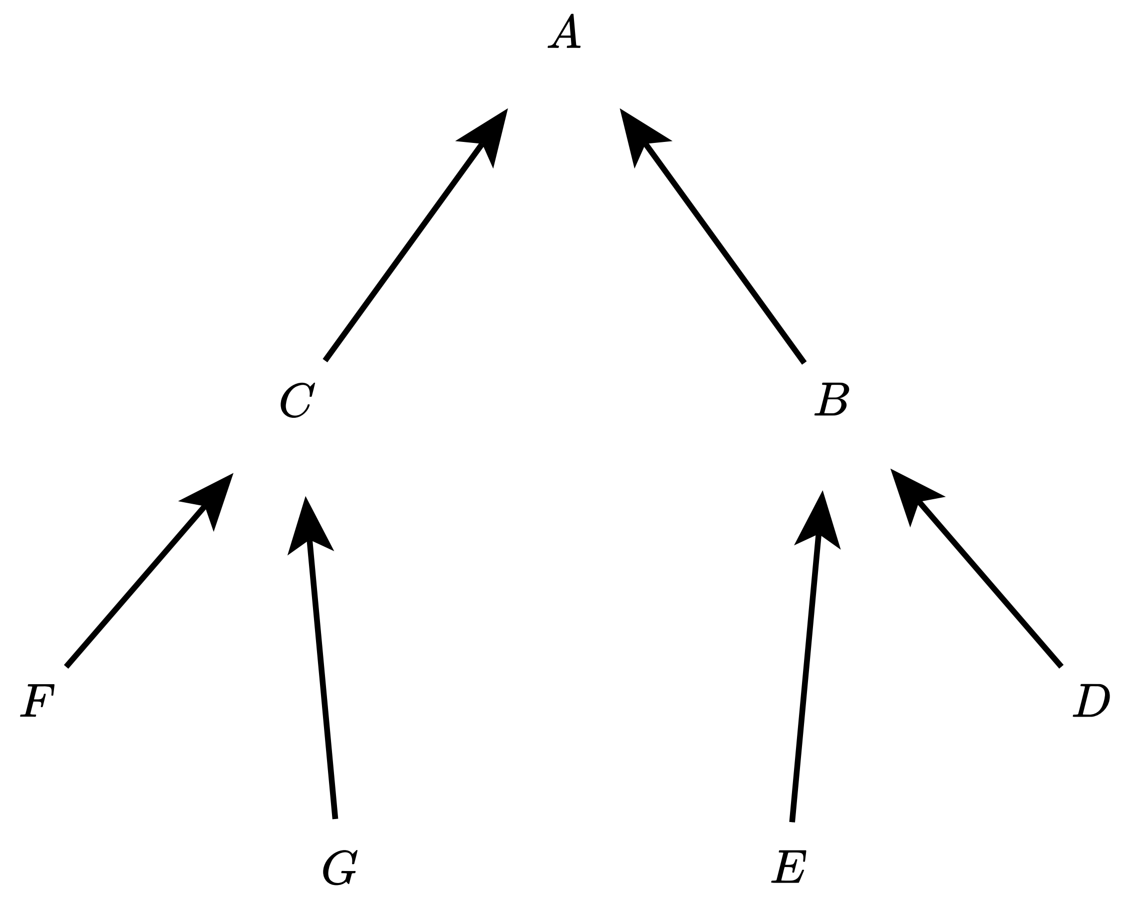




logical statements

```
Set A, B, C, D, E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```





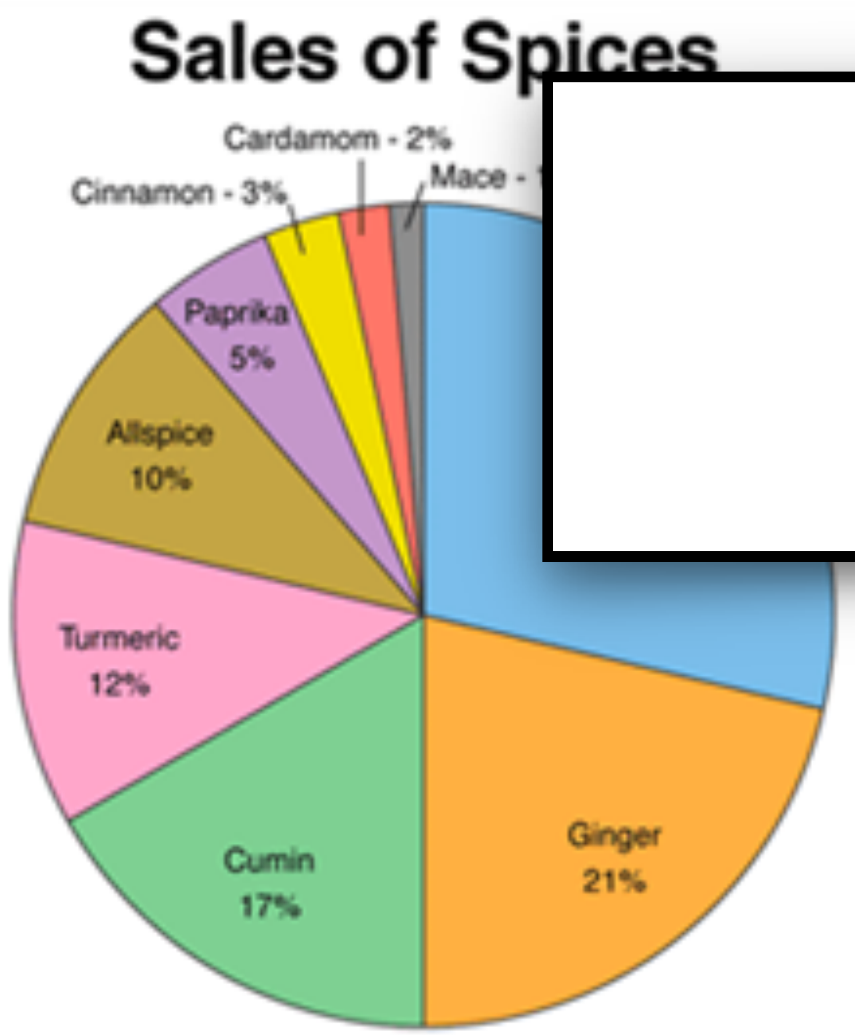
Style defines a domain's visual representation

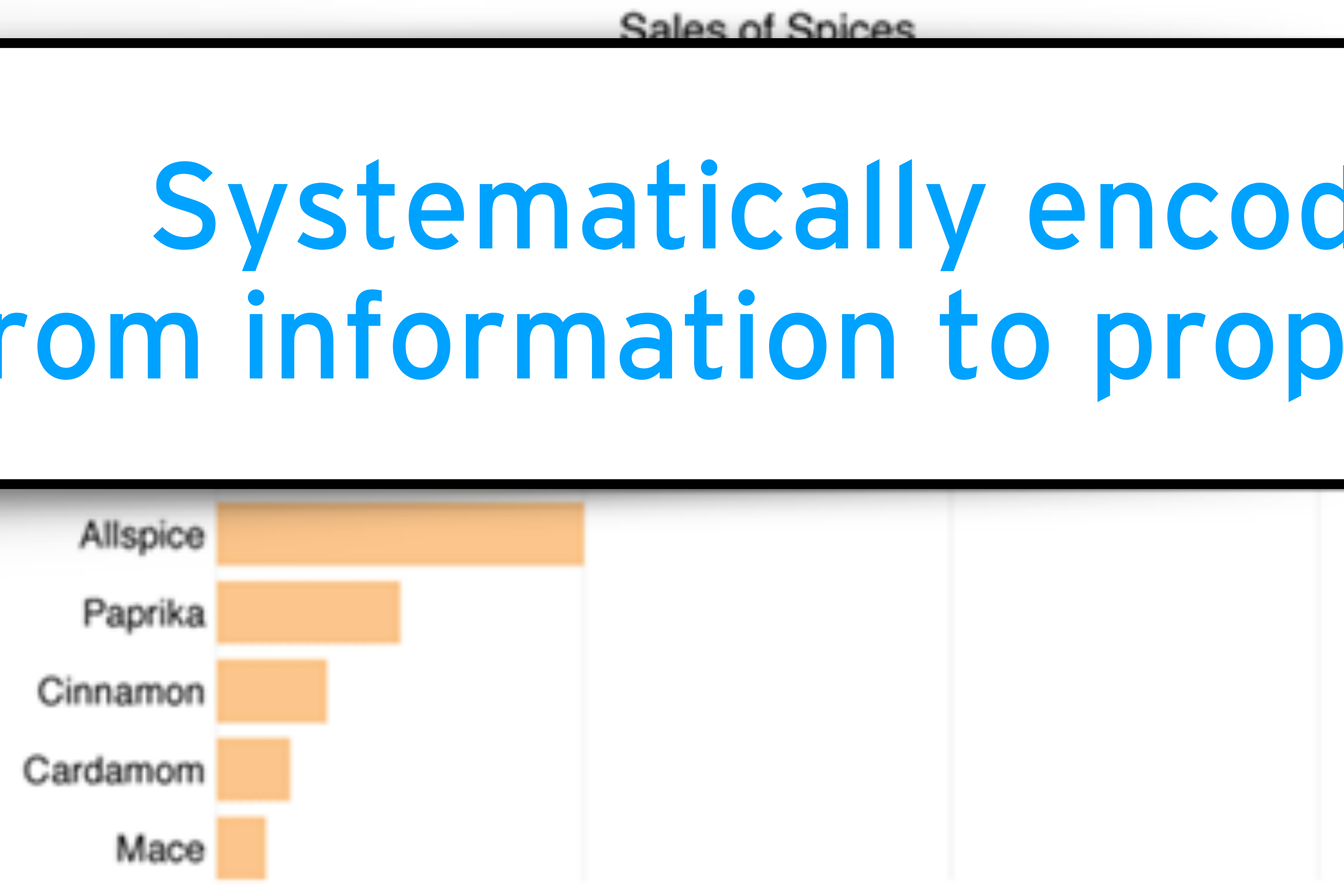
data

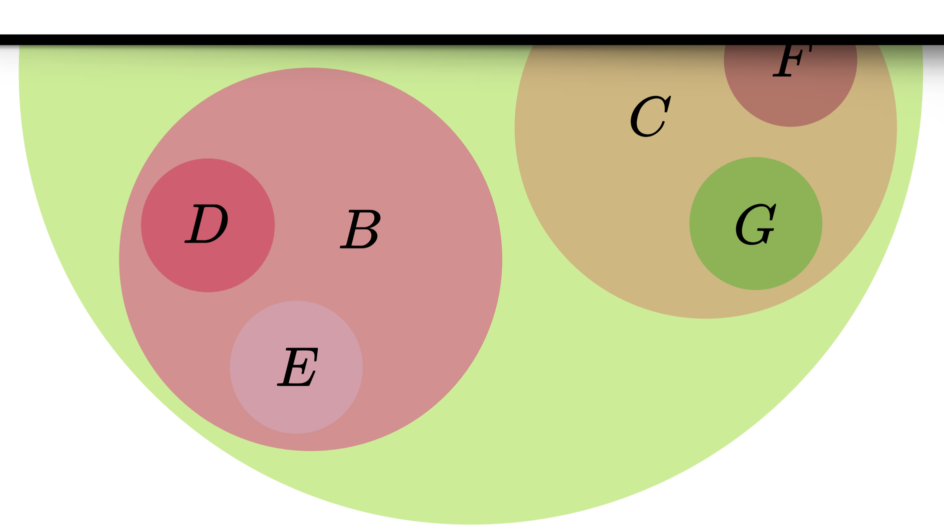
```
Saffron 58
Ginger 42
Cumin 34
Turmeric 24
Allspice 2
Paprika 1
Cinnamon 0.6
Cardamom 0.4
Mace 0.2
```

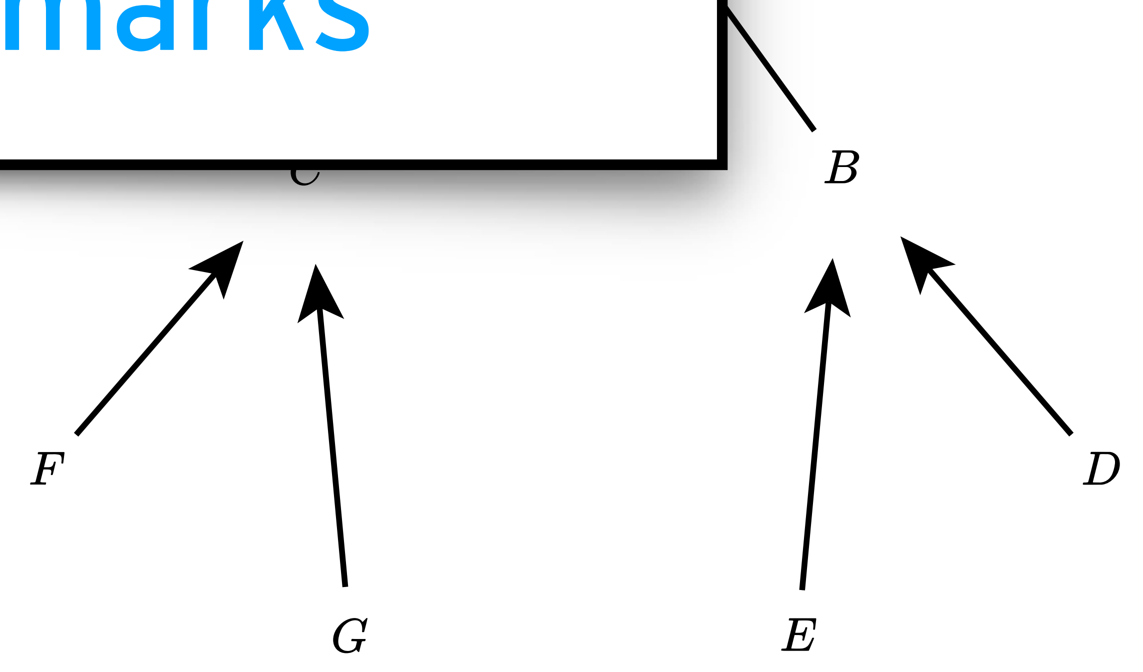
logical statements

```
Set A, B, C, D, E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```







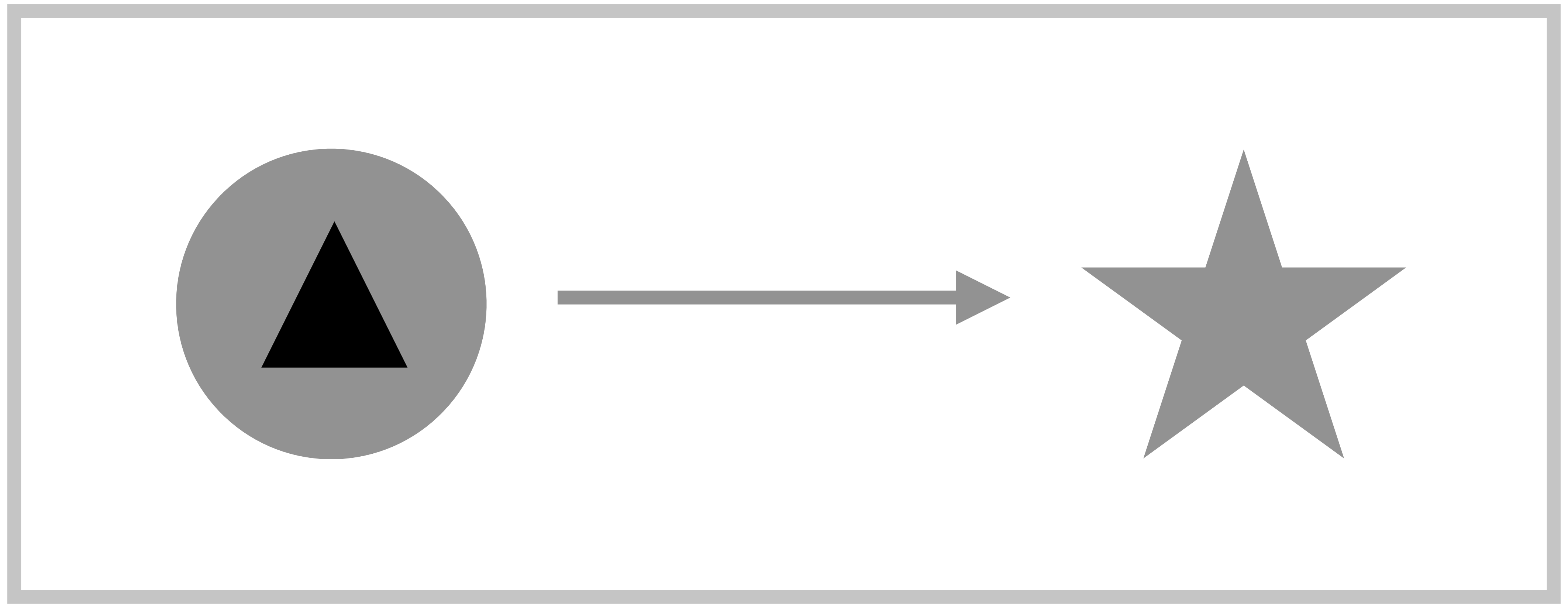


Systematically encode different mappings from information to properties of graphical marks

Style defines a family of diagrams

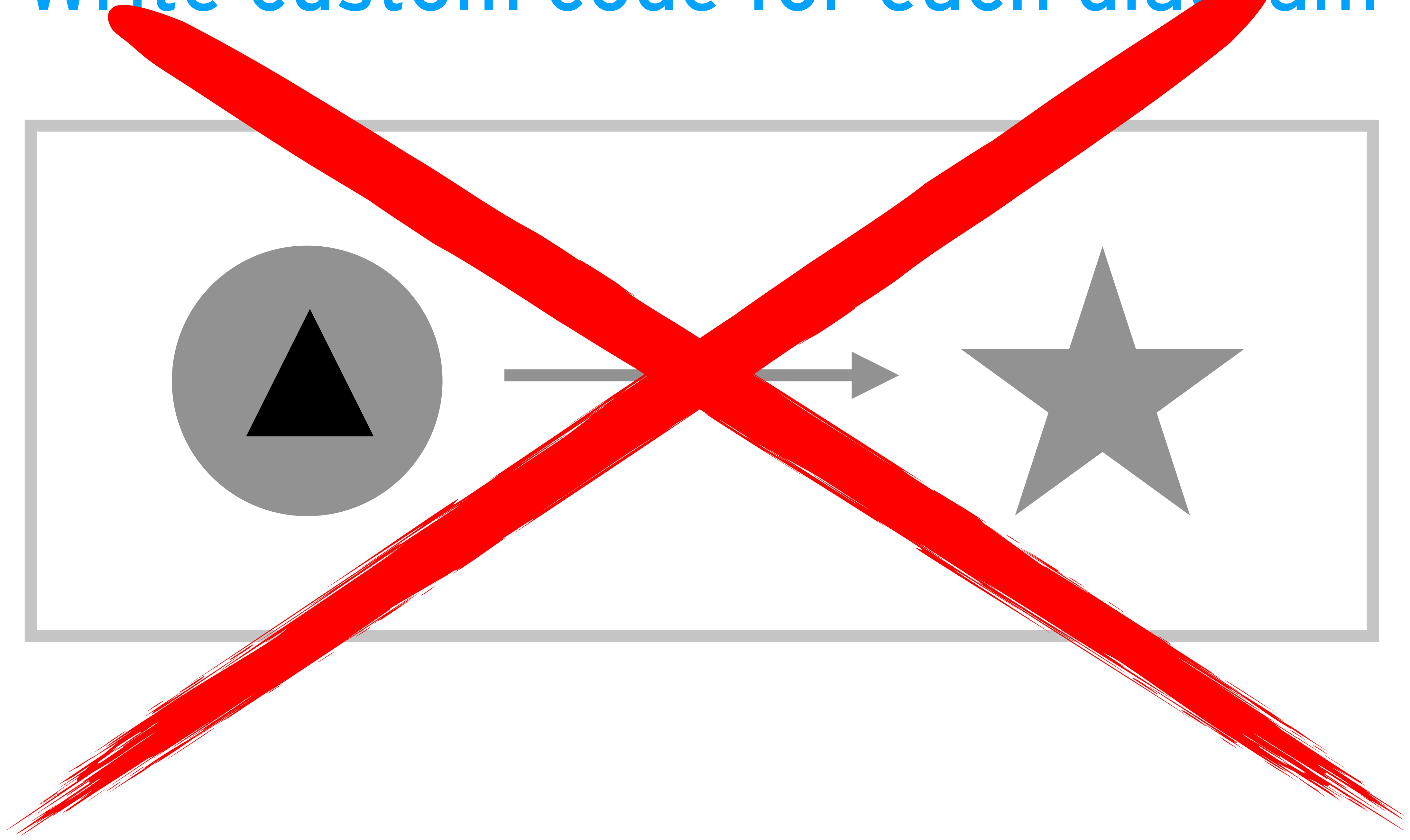
Style defines a family of diagrams

Write custom code for each diagram

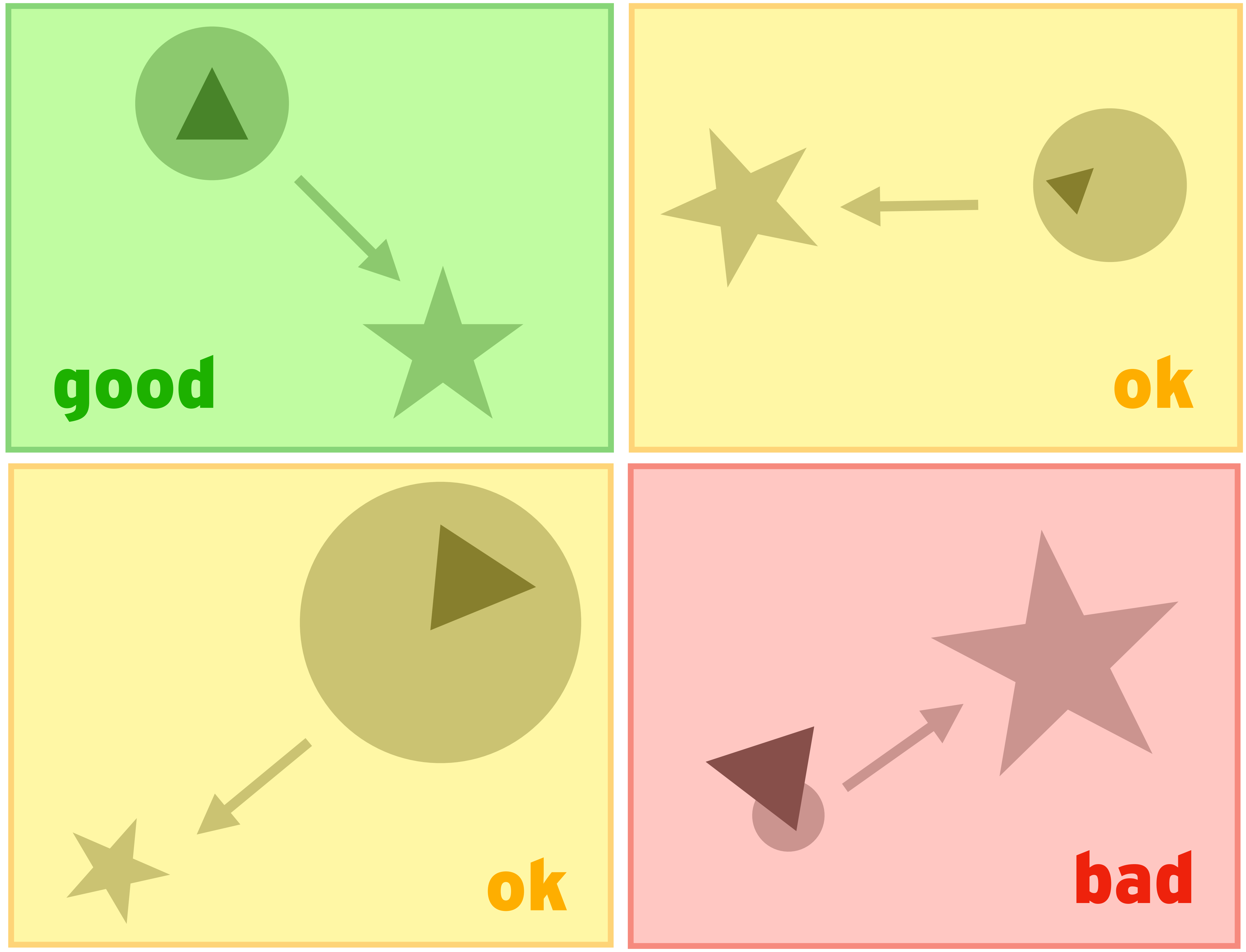


Style defines a family of diagrams

Write custom code for each diagram



Write code **once** for many diagrams



A *Style* is a list of selectors & rules

A *Style* is a list of selectors & rules

Selector

Rule (e.g. make a shape)

Rule (e.g. make a constraint)

A *Style* is a list of selectors & rules

Selector

Rule (e.g. make a shape)

Rule (e.g. make a constraint)

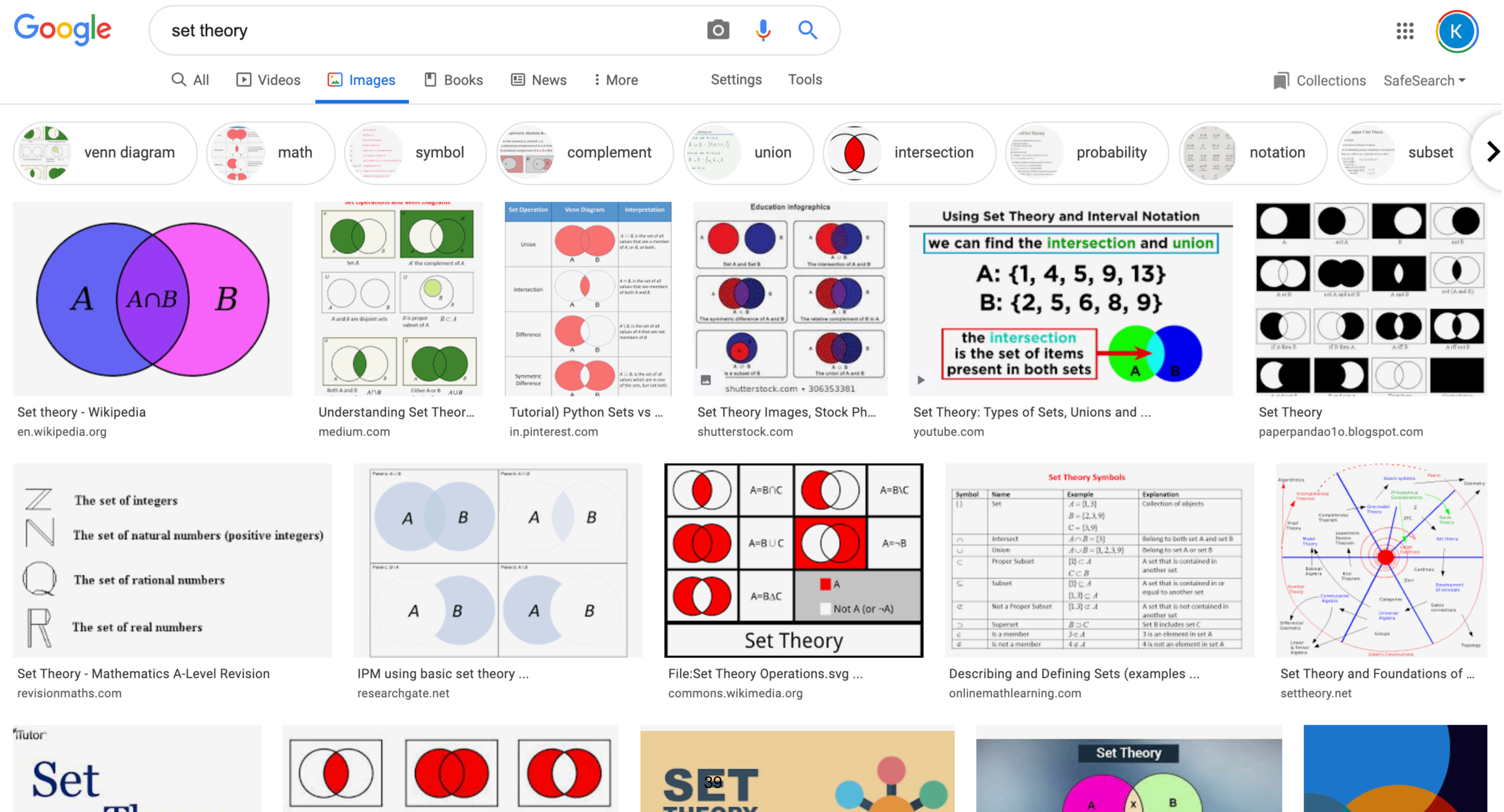
More specific selector

Rule refinement

Rule refinement

Visualizing sets & subsets

Visualizing sets & subsets



A simple *Style* for sets & subsets

A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```

A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```

A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```

A simple *Style* for sets & subsets

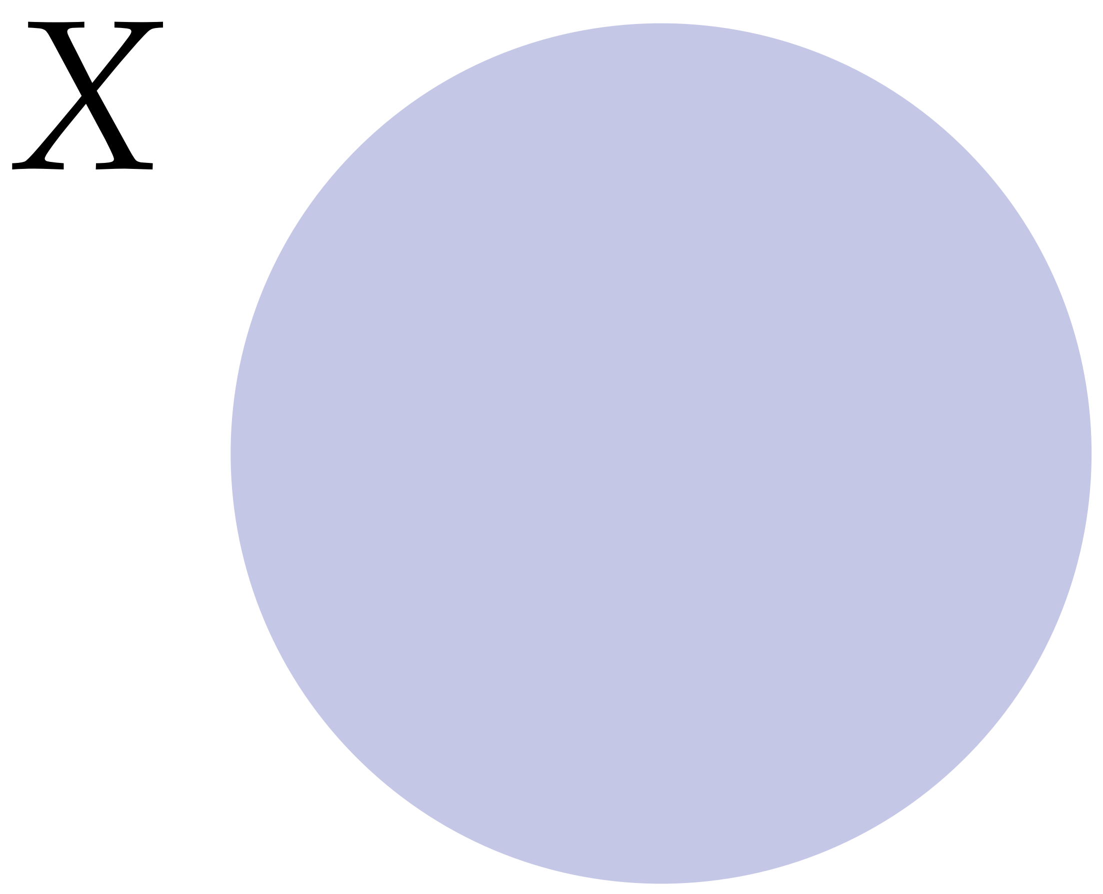
```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```

A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```

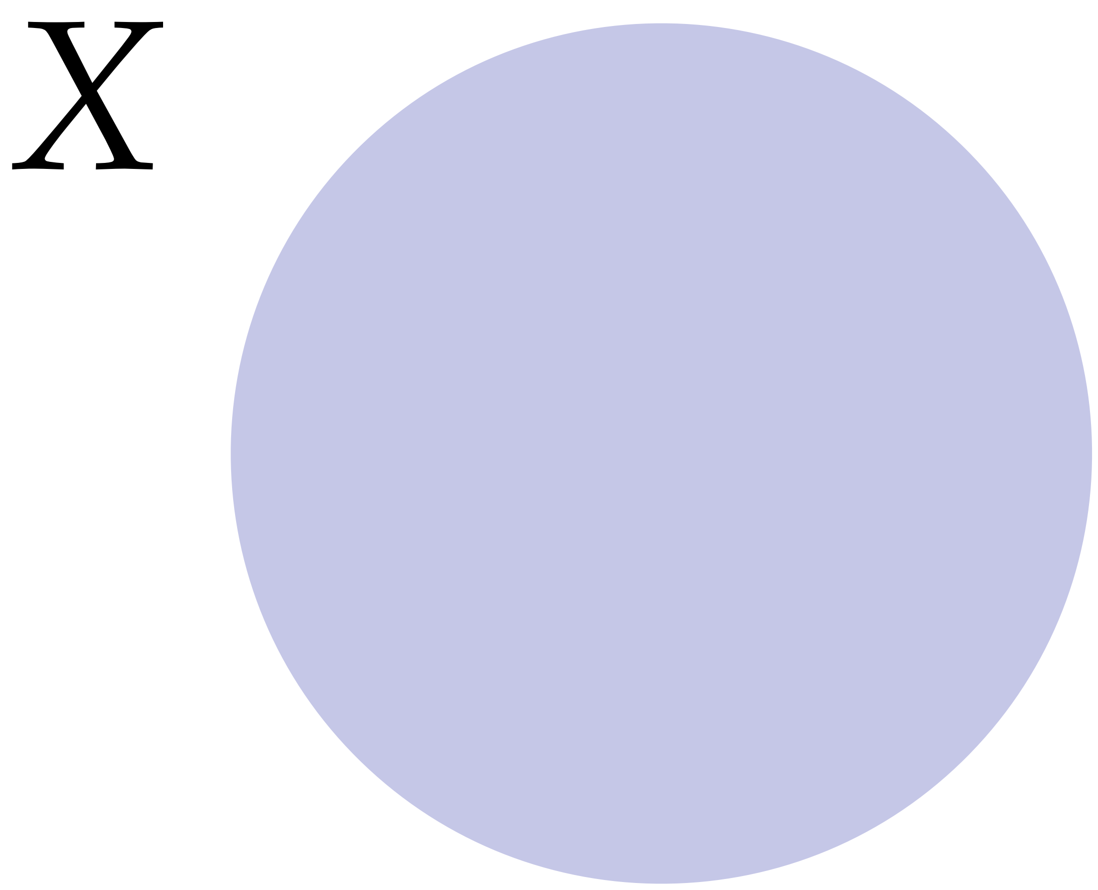
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```



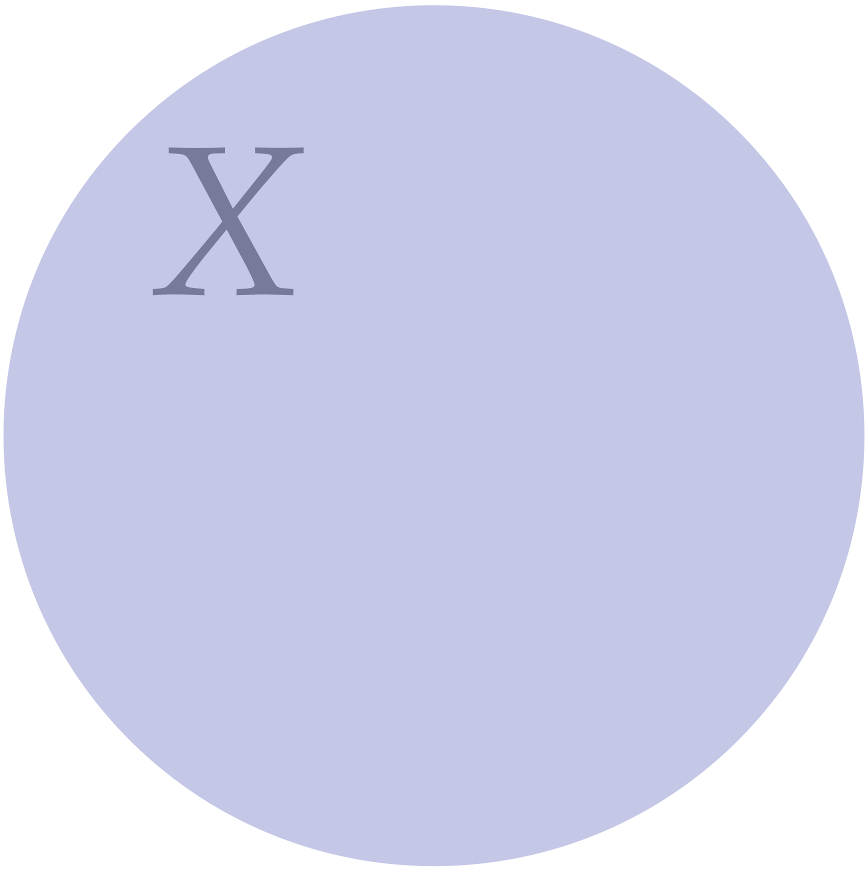
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```



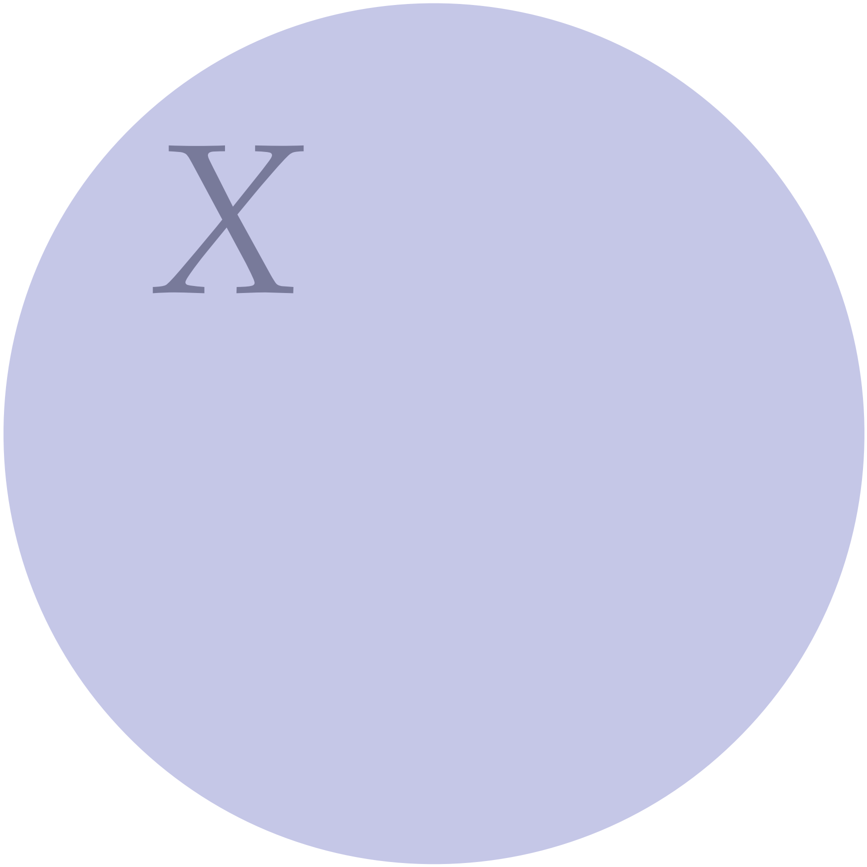
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```



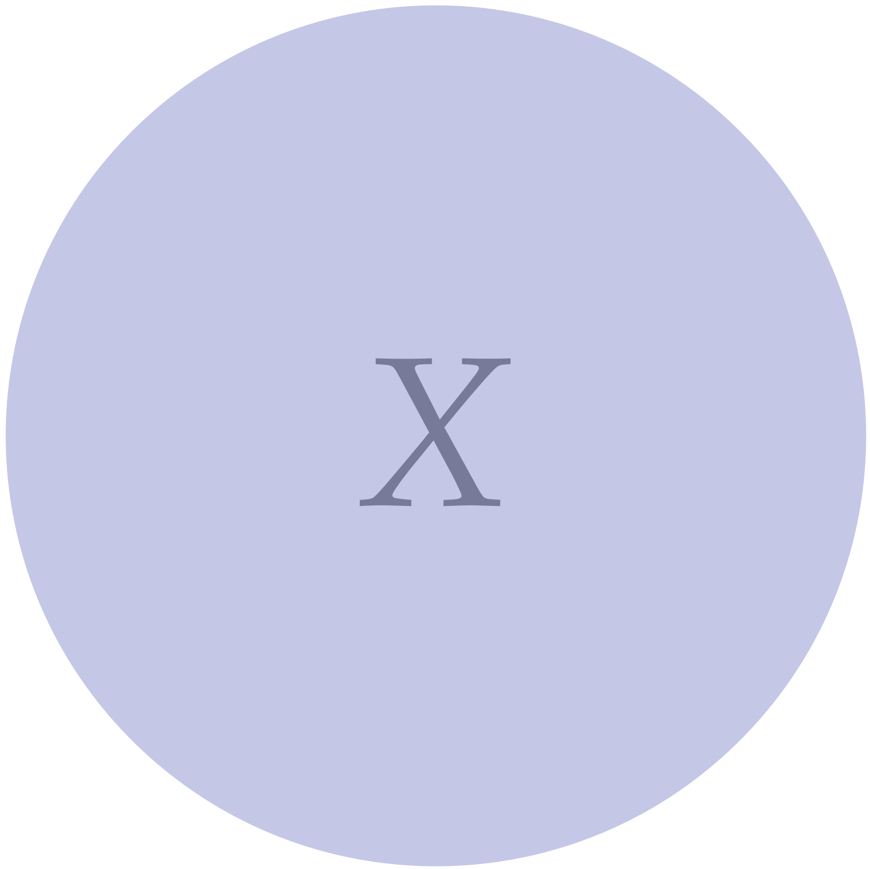
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```



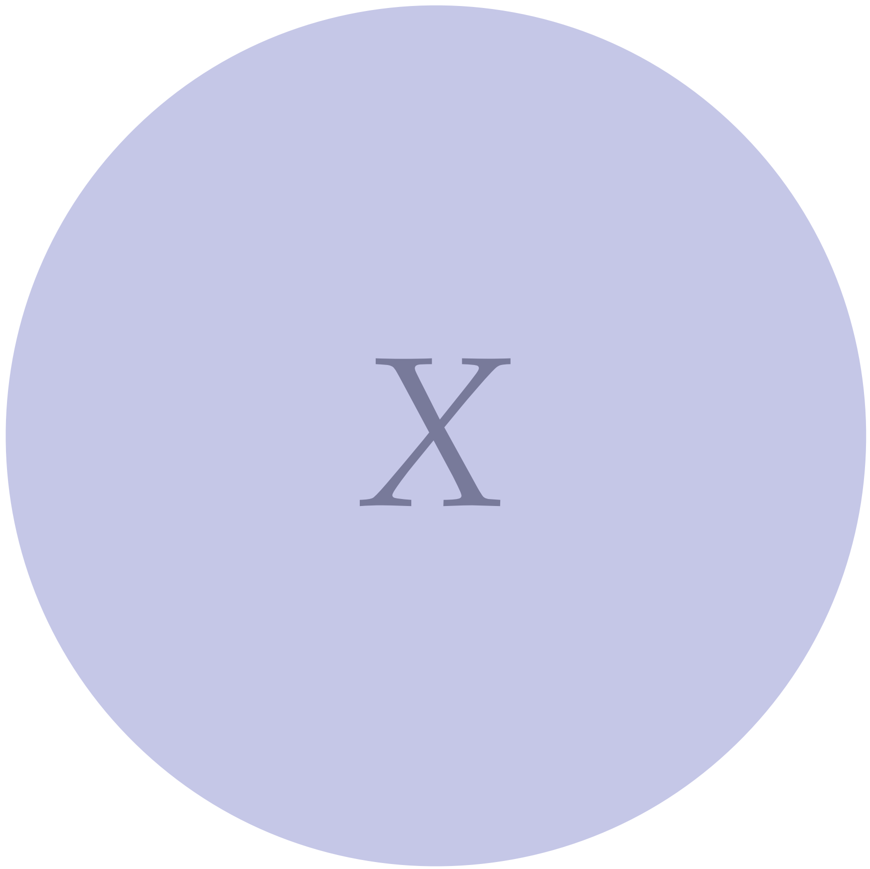
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```



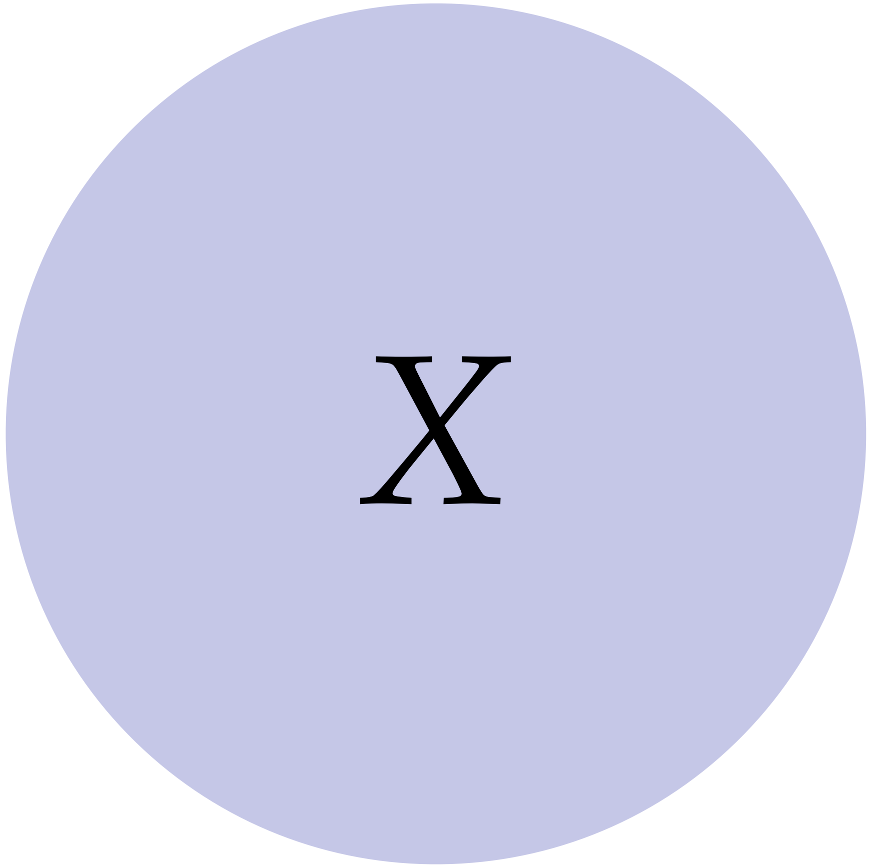
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```



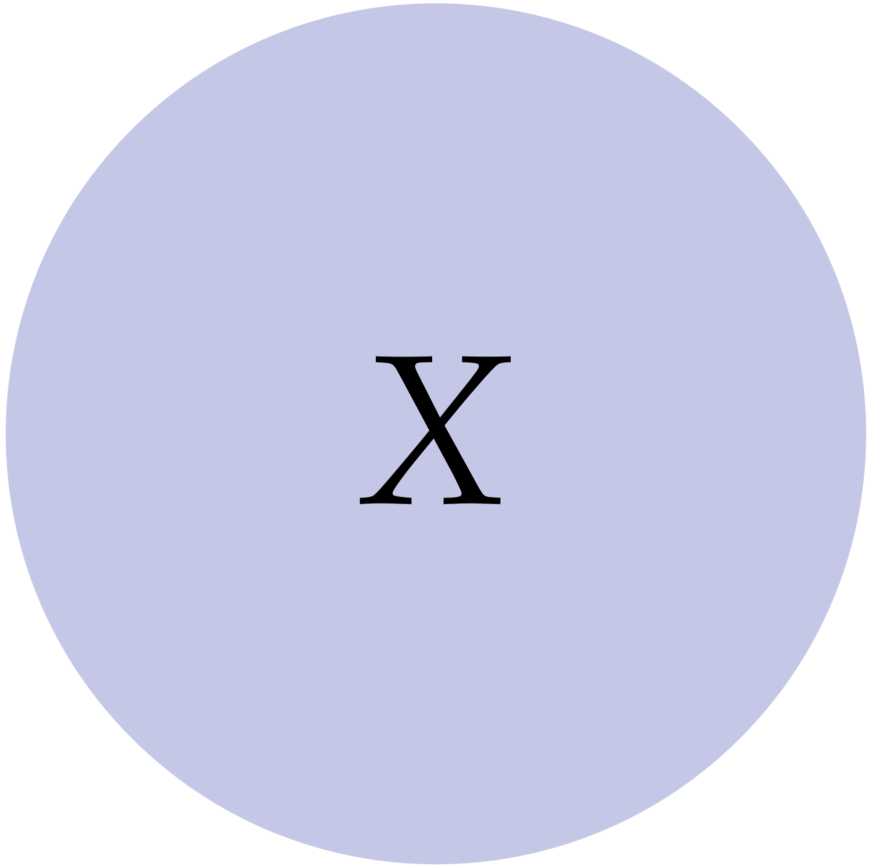
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```



A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}
```

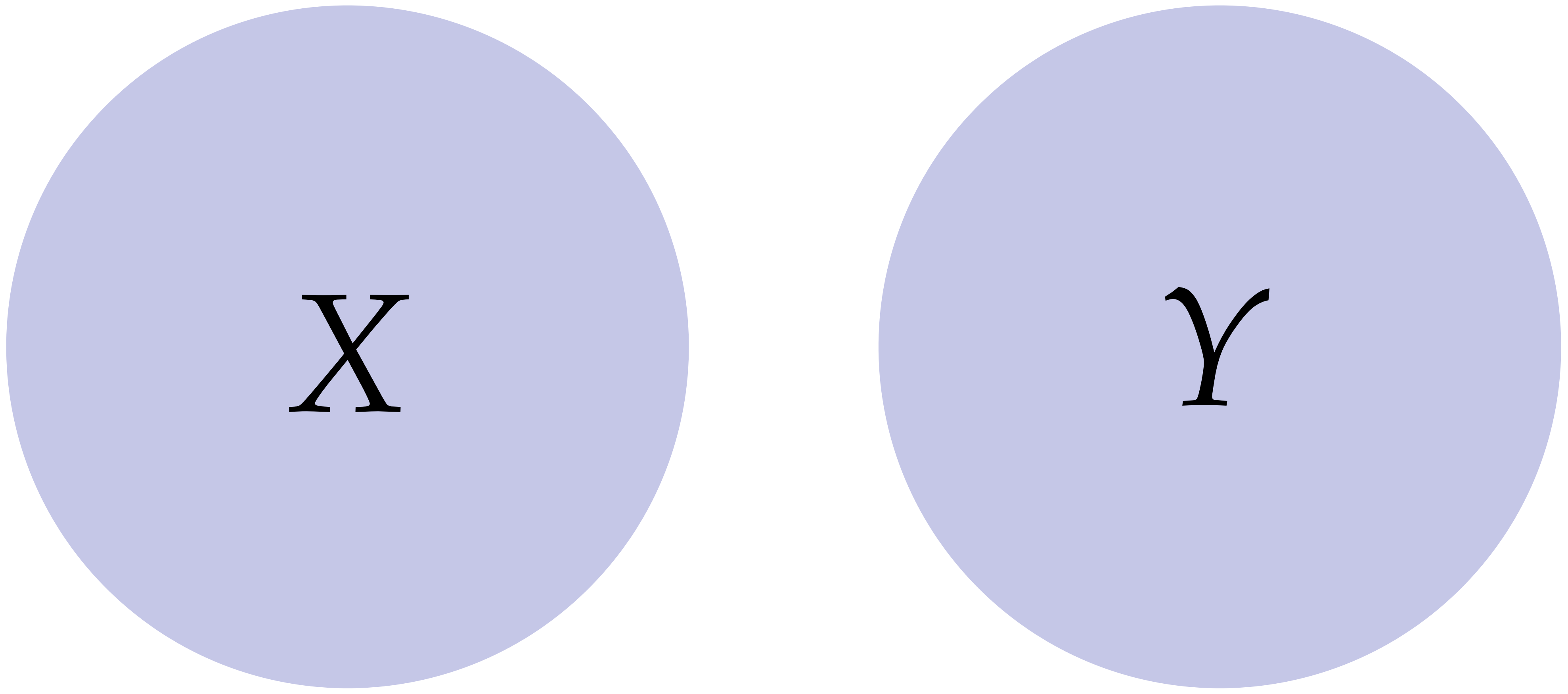


A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```

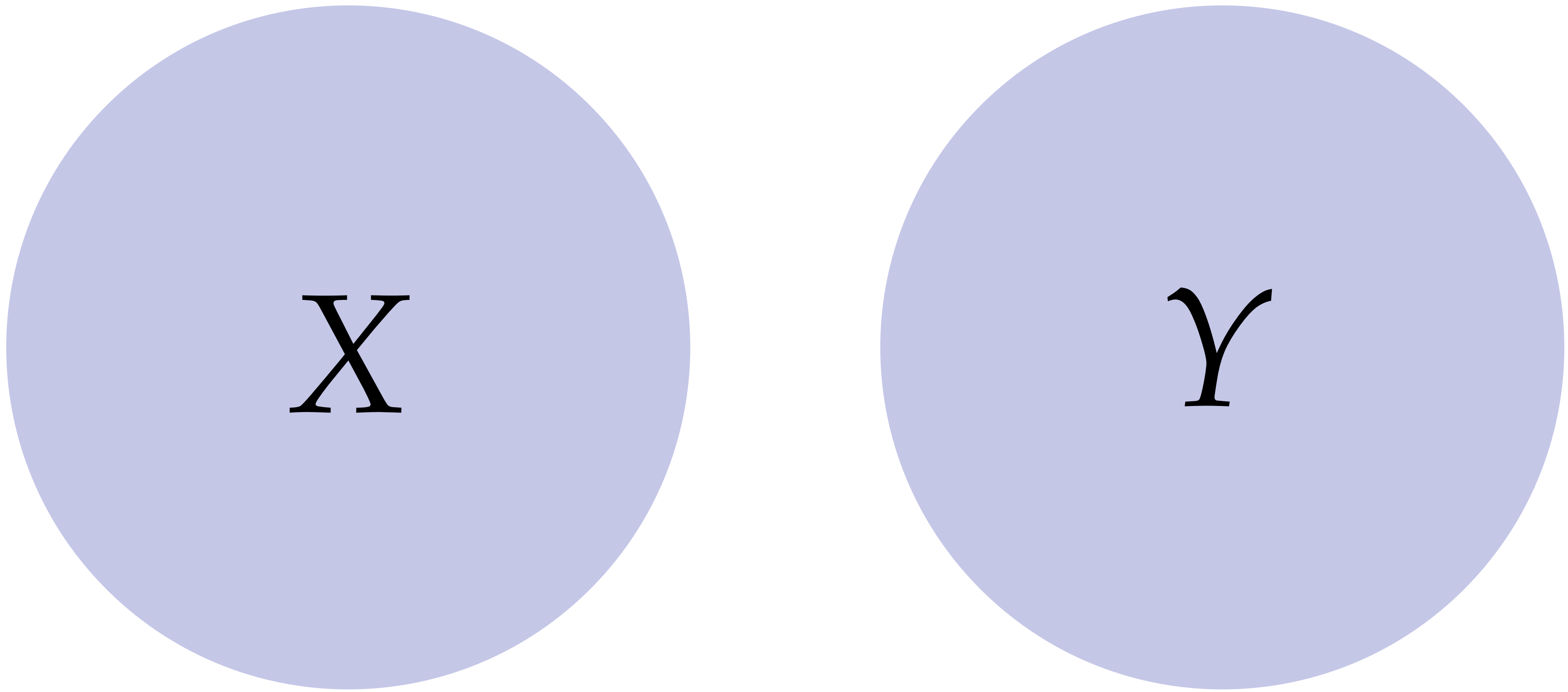
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



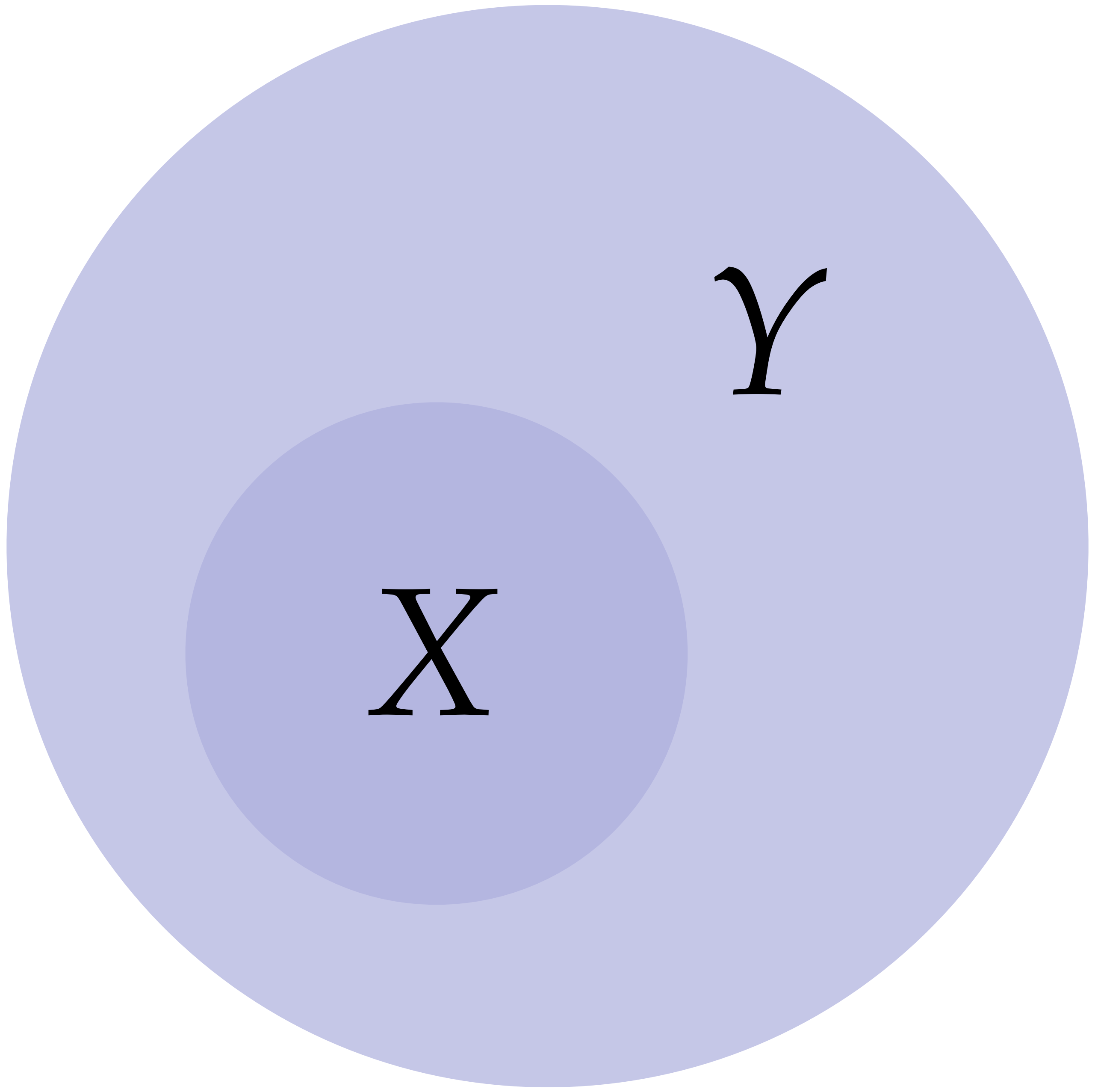
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



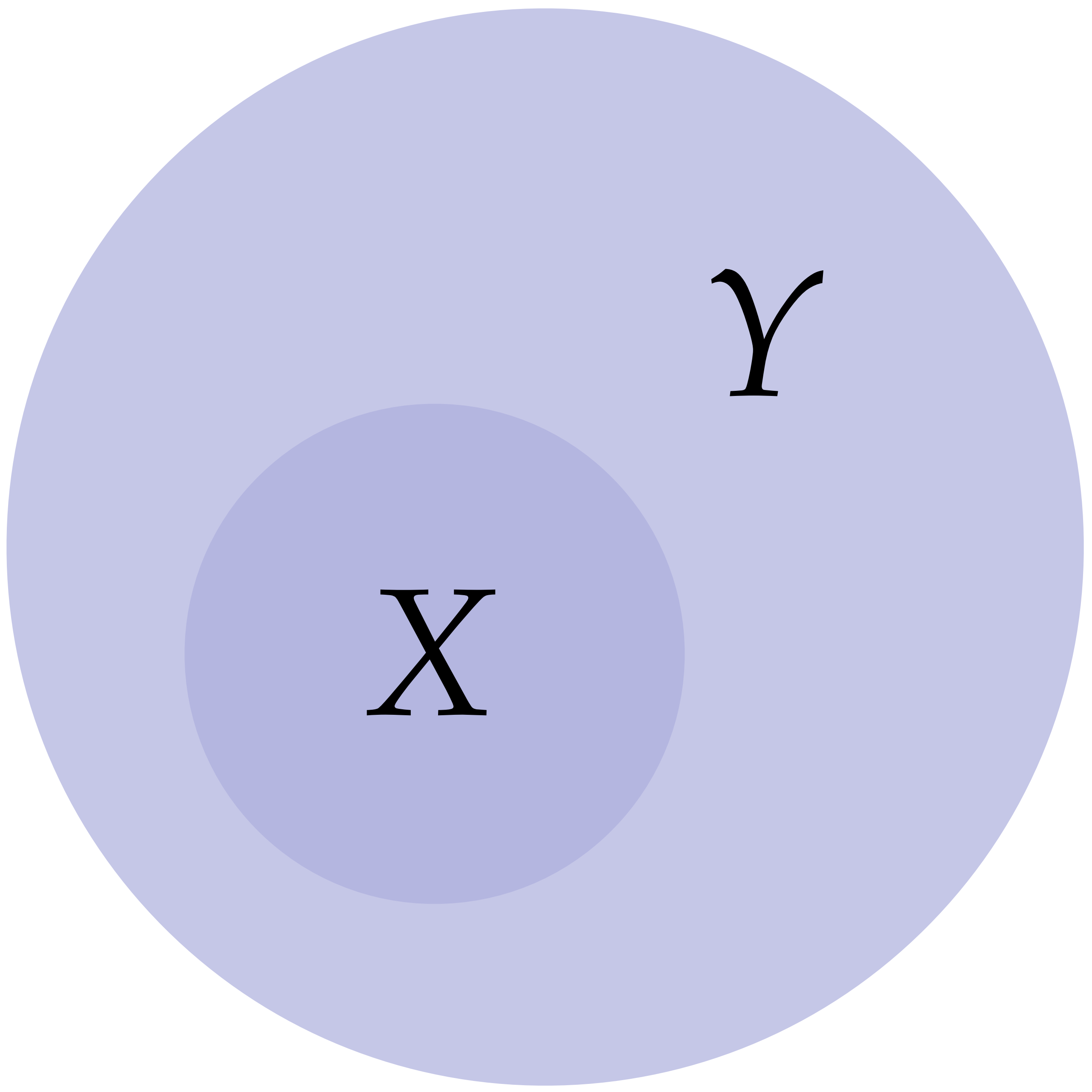
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

 forall Set x; Set y
 where IsSubset(x, y) {
     ensure contains(y.shape, x.shape)
     ensure smallerThan(x.shape, y.shape)
     ensure outsideOf(y.text, x.shape)
     layer x.shape above y.shape
     layer y.text below x.shape
 }
```

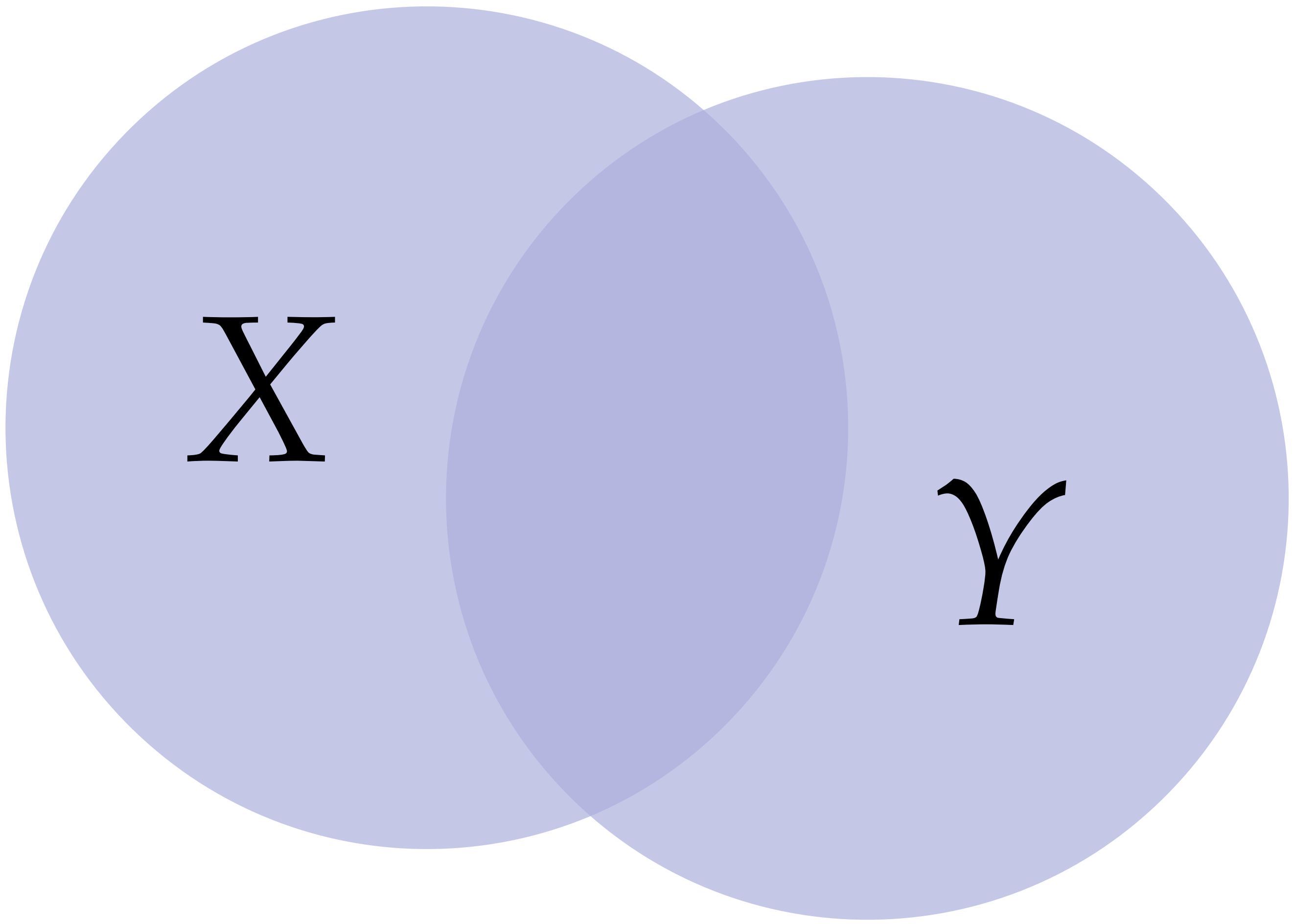


A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}

forall Set x; Set y
where Not(Intersecting(x, y)) {
    ensure disjoint(x.shape, y.shape)
}
```

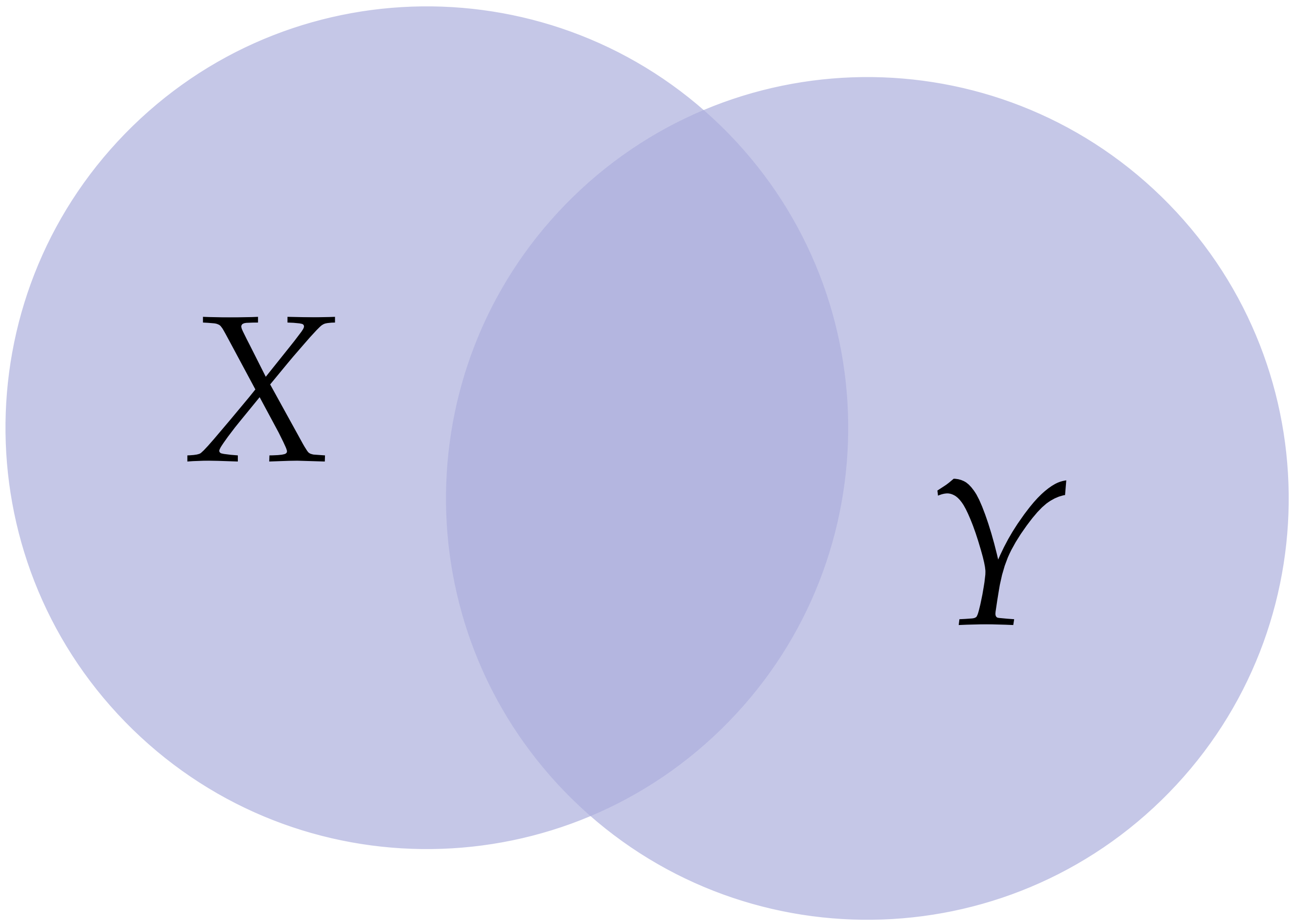
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}

forall Set x; Set y
where Not(Intersecting(x, y)) {
    ensure disjoint(x.shape, y.shape)
}
```



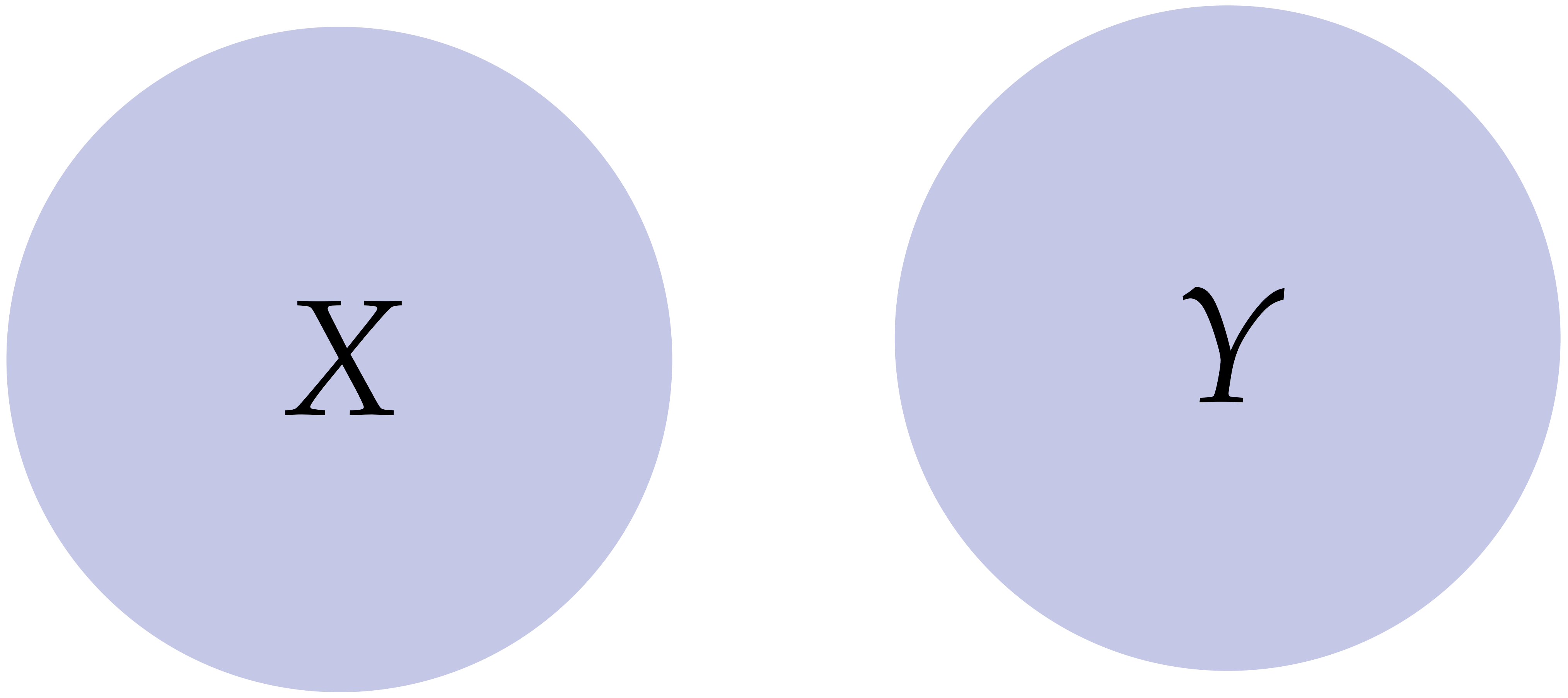
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}

forall Set x; Set y
where Not(Intersecting(x, y)) {
    ensure disjoint(x.shape, y.shape)
}
```



A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}

forall Set x; Set y
where Not(Intersecting(x, y)) {
    ensure disjoint(x.shape, y.shape)
}
```



A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}

forall Set x; Set y
where Not(Intersecting(x, y)) {
    ensure disjoint(x.shape, y.shape)
}
```

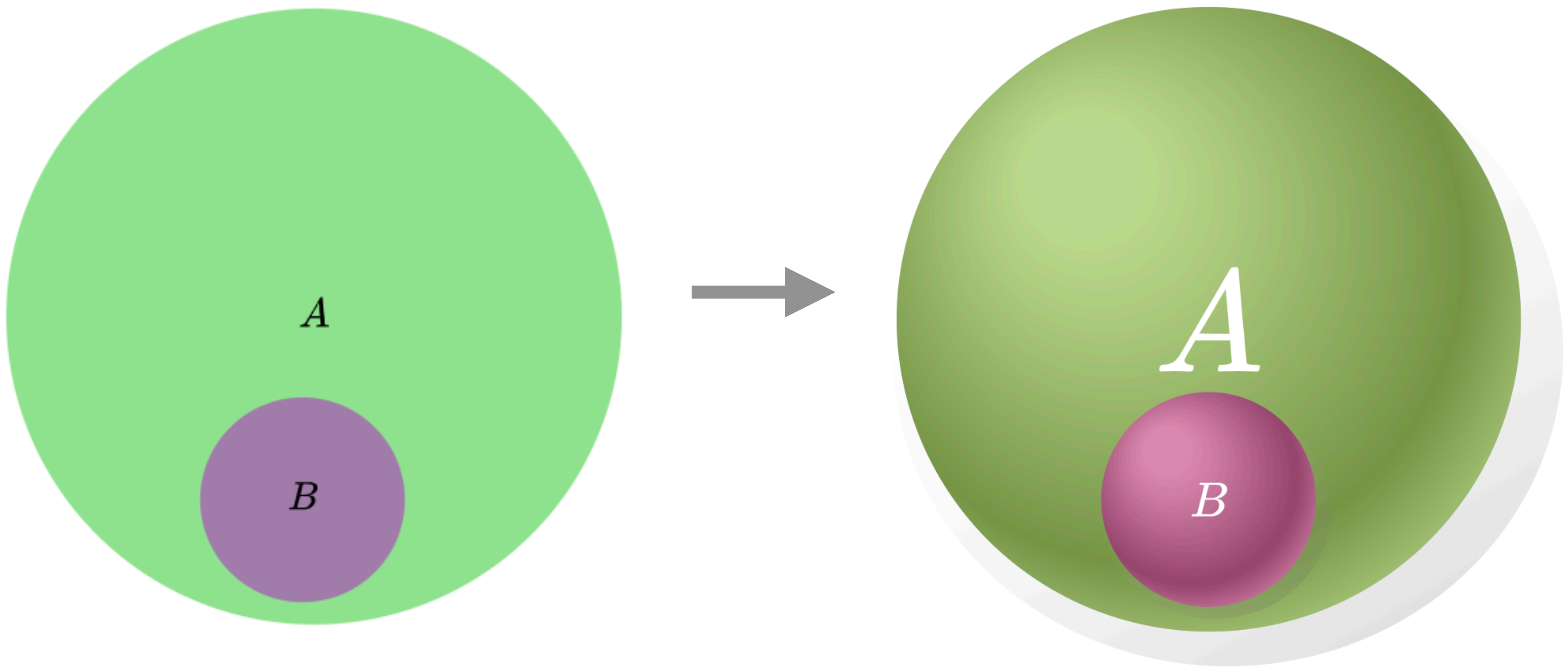
A simple *Style* for sets & subsets

```
forall Set x {
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}

forall Set x; Set y
where Not(Intersecting(x, y)) {
    ensure disjoint(x.shape, y.shape)
}
```

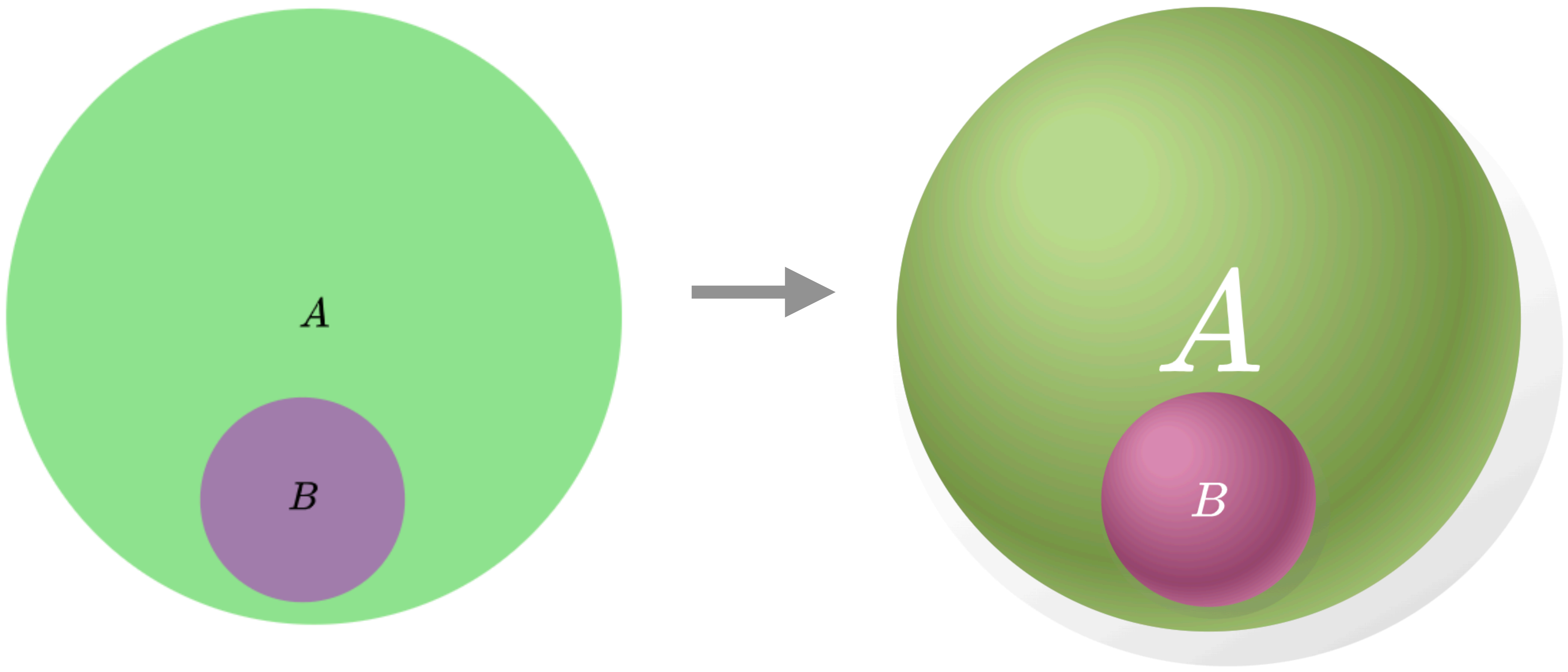
defines a family of diagrams

Style makes it easy to refine the visual style

Style makes it easy to refine the visual style



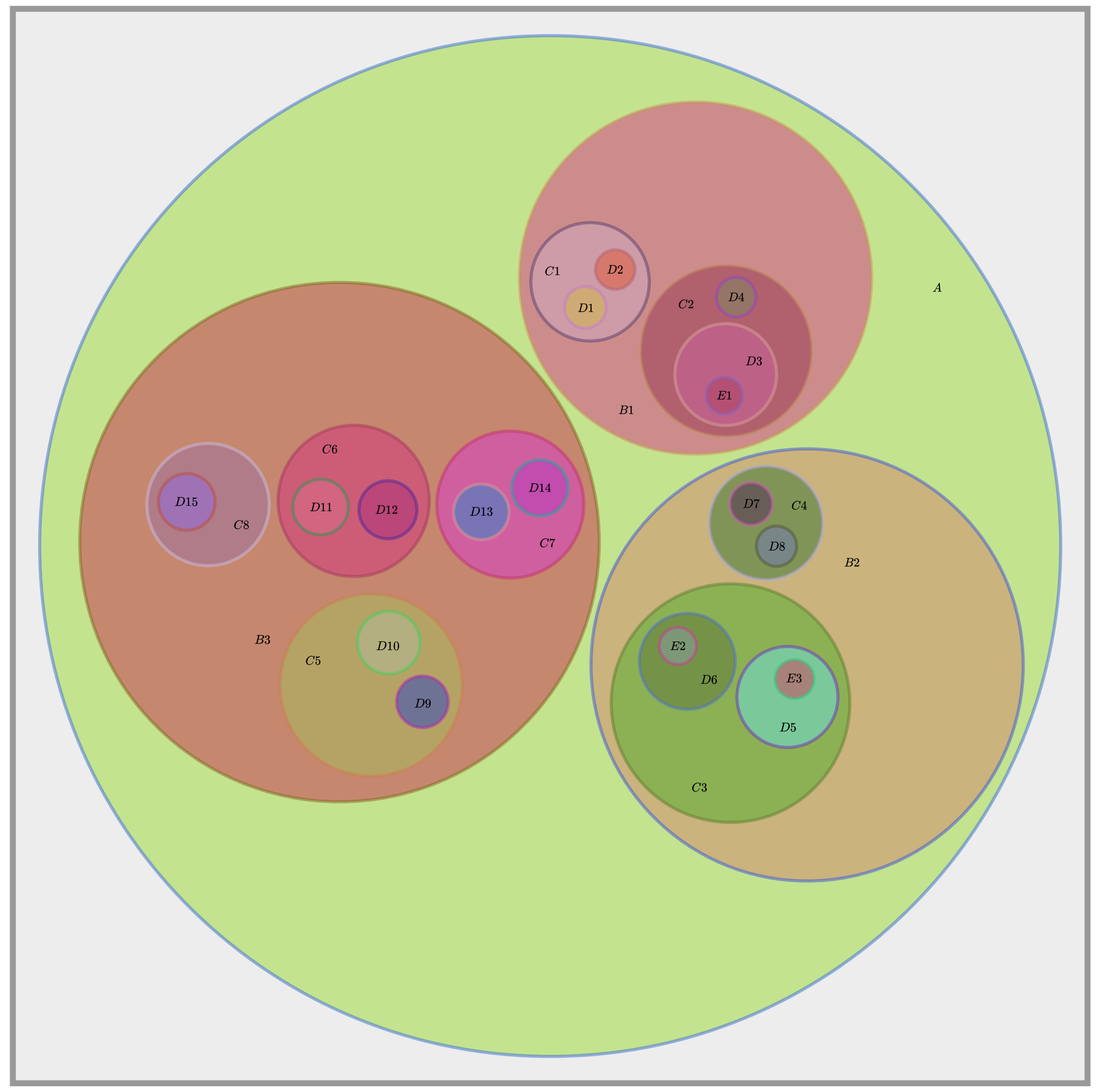
Style makes it easy to refine the visual style



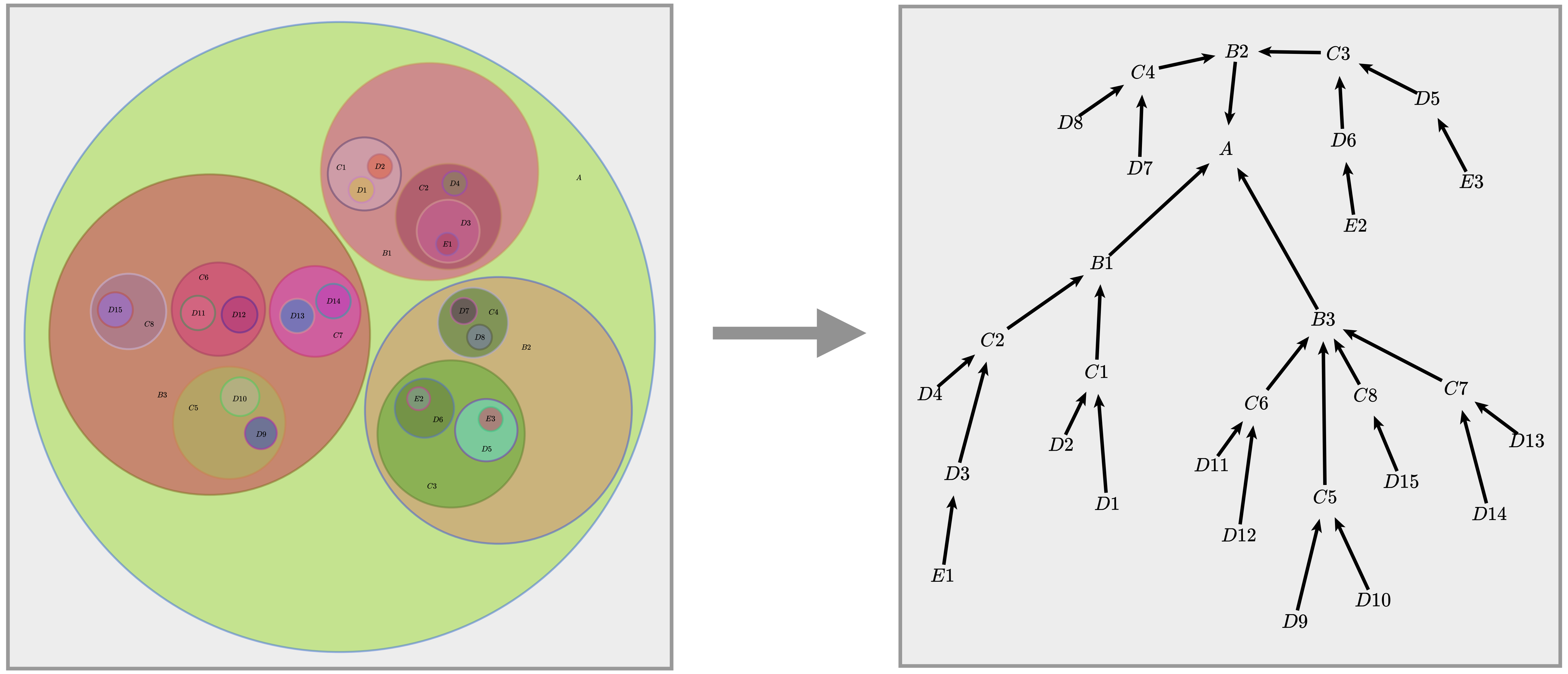

```
x.shading = Image {
    x : x.shape.x
    y : x.shape.y
    w : x.shape.r * 2.0
    h : x.shape.r * 2.0
    path : "shading.svg"
}

x.shadow = Image {
    path : "shadow.svg"
    w : x.shape.r * 2.15
    h : x.shape.r * 2.22
    x : x.shape.x + 0.03 * x.shading.w
    y : x.shape.y - 0.051 * x.shading.h
}

x.text = Text {
    string : x.label
    color: rgba(1.0, 1.0, 1.0, 1.0)
    w: 0.5 * x.shape.r
    h: 0.5 * x.shape.r
}
```

Style makes it easy to change the visual representation

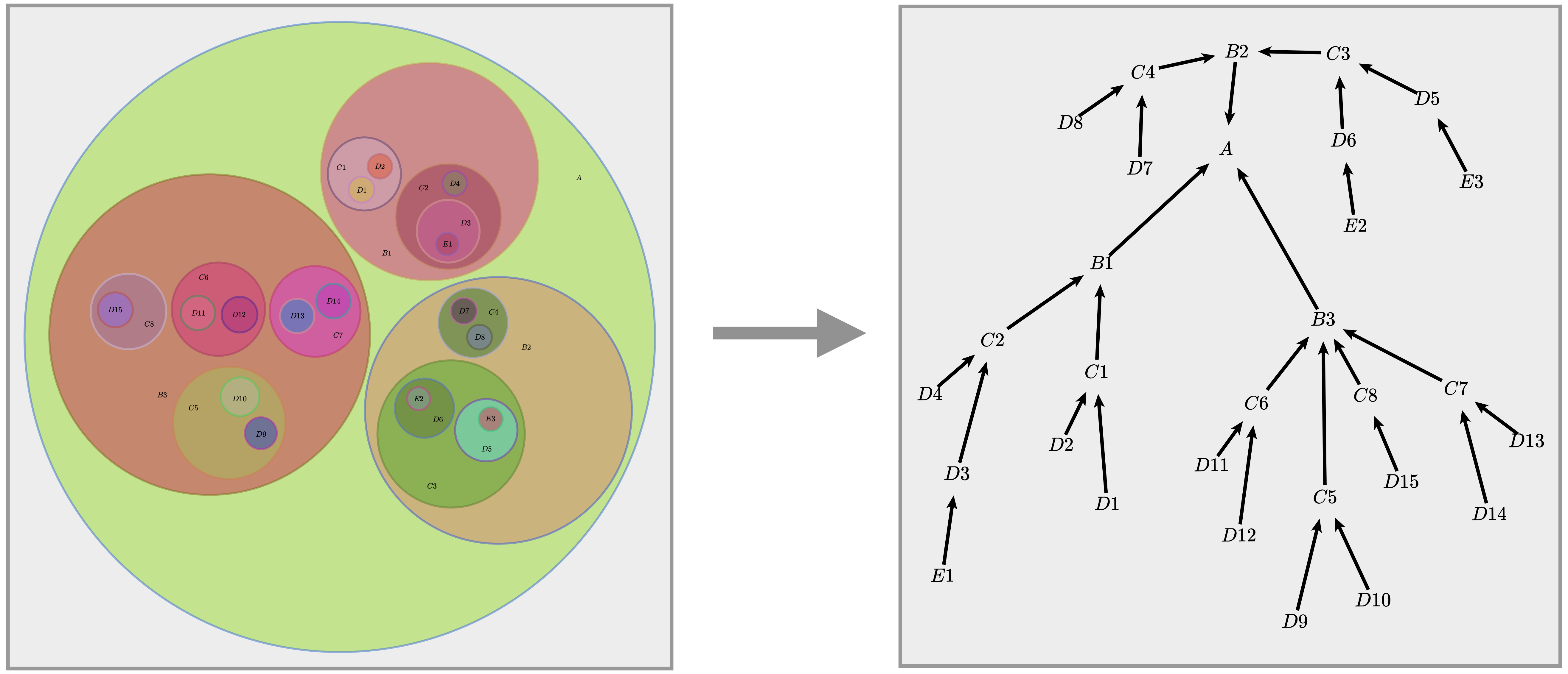
Style makes it easy to change the visual representation



Style makes it easy to change the visual representation



Style makes it easy to change the visual representation




```
Set x {
    x.shape = Text { string : x.label }
}

Set x; Set y {
   encourage repel(x.shape, y.shape, 5.0)
}

Set x; Set y
where IsSubset(x, y) {
    arrow = Arrow {
      thickness : 2.0
      color : rgba(0.0, 0.0, 0.0, 1.0)
    }

    encourage centerArrow(LOCAL.arrow,
                x.shape, y.shape)
    encourage above(y.shape, x.shape)
    encourage equal(x.shape.x, y.shape.x)
}
```

Domain schemas define the mathematical environment

Domain schemas define the mathematical environment

image credit: Martin Kuppe

Domain schemas define the mathematical environment

- There's no way to anticipate every object/idea a user might want to diagram...

image credit: Martin Kuppe

Domain schemas define the mathematical environment

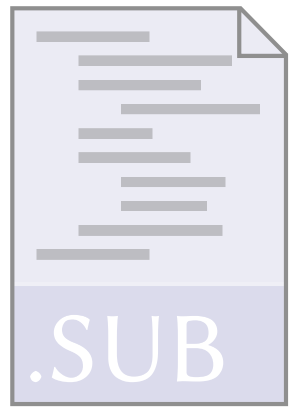
- There's no way to anticipate every object/idea a user might want to diagram...

- But users can **extend** the Substance language by writing Domain schemas that define their own domain-specific dialects

image credit: Martin Kuppe

Domain schemas define abstract types & notation

Domain schemas define abstract types & notation




```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

Domain schemas define abstract types & notation

.SUB

```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

.DSL

Domain schemas define abstract types & notation

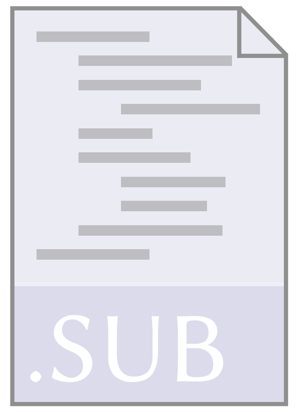
.SUB

```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

.DSL

```
type Set
```

Domain schemas define abstract types & notation




```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```


```
type Set
predicate IsSubset : Set s1 * Set s2
```

Domain schemas define abstract types & notation

.SUB

```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

.DSL

```
type Set
predicate IsSubset : Set s1 * Set s2
predicate Intersecting : Set s1 * Set s2
```

Domain schemas define abstract types & notation

.SUB

```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

.DSL

```
type Set
predicate IsSubset : Set s1 * Set s2
predicate Intersecting : Set s1 * Set s2
predicate Not : Prop p
```

Domain schemas define abstract types & notation

.SUB

```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

.DSL

```
type Set
predicate IsSubset : Set s1 * Set s2
predicate Intersecting : Set s1 * Set s2
predicate Not : Prop p
notation "A ⊂ B" ~ "IsSubset(A, B)"
```

Domain schemas define abstract types & notation

.SUB

```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

.DSL

```
type Set
predicate IsSubset : Set s1 * Set s2
predicate Intersecting : Set s1 * Set s2
predicate Not : Prop p
notation "A ⊂ B" ~ "IsSubset(A, B)"
notation "A ∩ B = ∅" ~ "Not(Intersecting(A, B))"
```

Domain schemas define abstract types & notation

.SUB

```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

.DSL

purely abstract

```
type Set
predicate IsSubset : Set s1 * Set s2
predicate Intersecting : Set s1 * Set s2
predicate Not : Prop p
notation "A ⊂ B" ~ "IsSubset(A, B)"
notation "A ∩ B = ∅" ~ "Not(Intersecting(A, B))"
```

Domain schemas define abstract types & notation

.SUB

```
Set A, B, C, D,
    E, F, G
B ⊂ A
C ⊂ A
D ⊂ B
E ⊂ B
F ⊂ C
G ⊂ C
E ∩ D = ∅
F ∩ G = ∅
B ∩ C = ∅
```

purely abstract

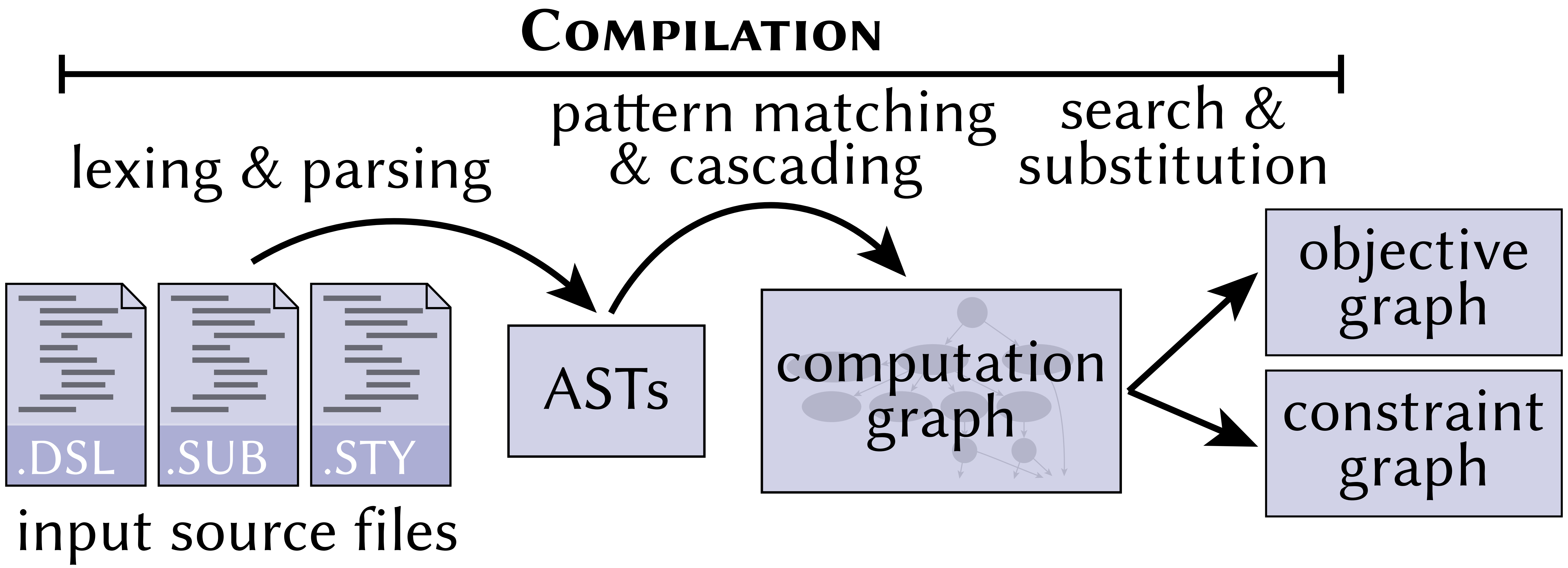
.DSL

```
type Set
predicate IsSubset : Set s1 * Set s2
predicate Intersecting : Set s1 * Set s2
predicate Not : Prop p
notation "A ⊂ B" ~ "IsSubset(A, B)"
notation "A ∩ B = ∅" ~ "Not(Intersecting(A, B))"
```

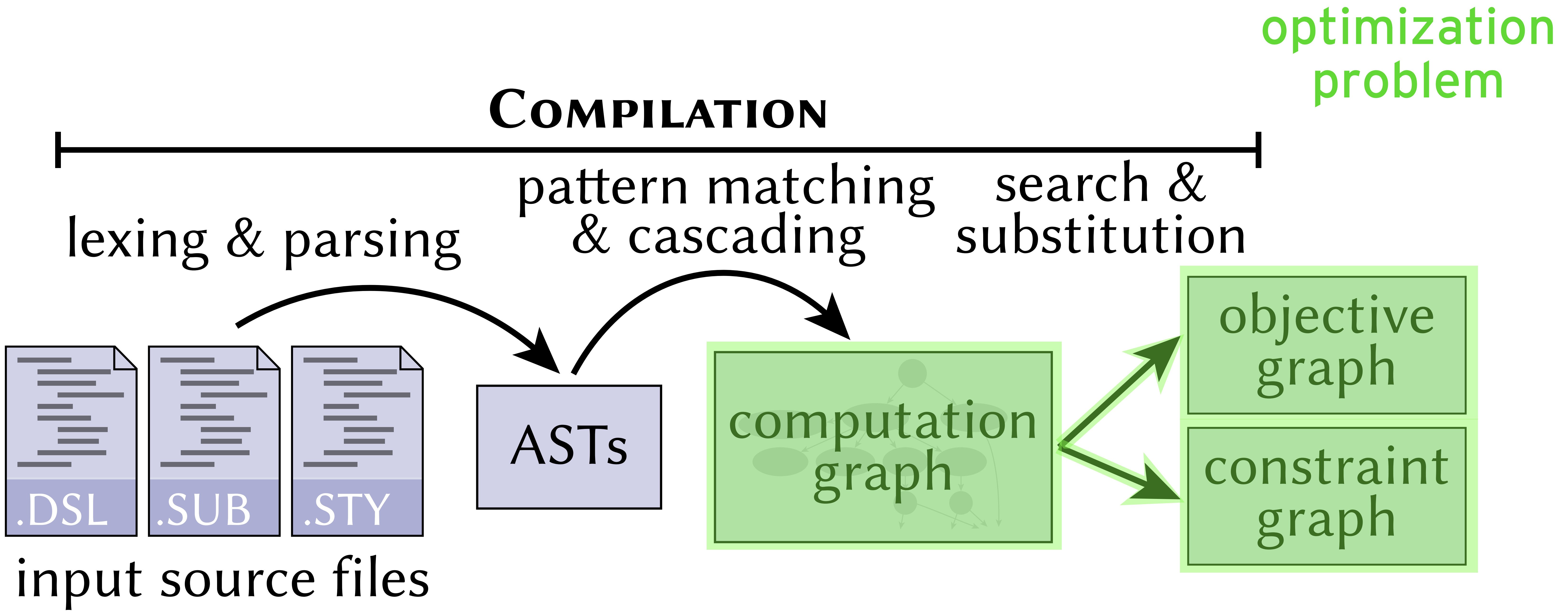
custom notation

The Penrose pipeline links specification and synthesis

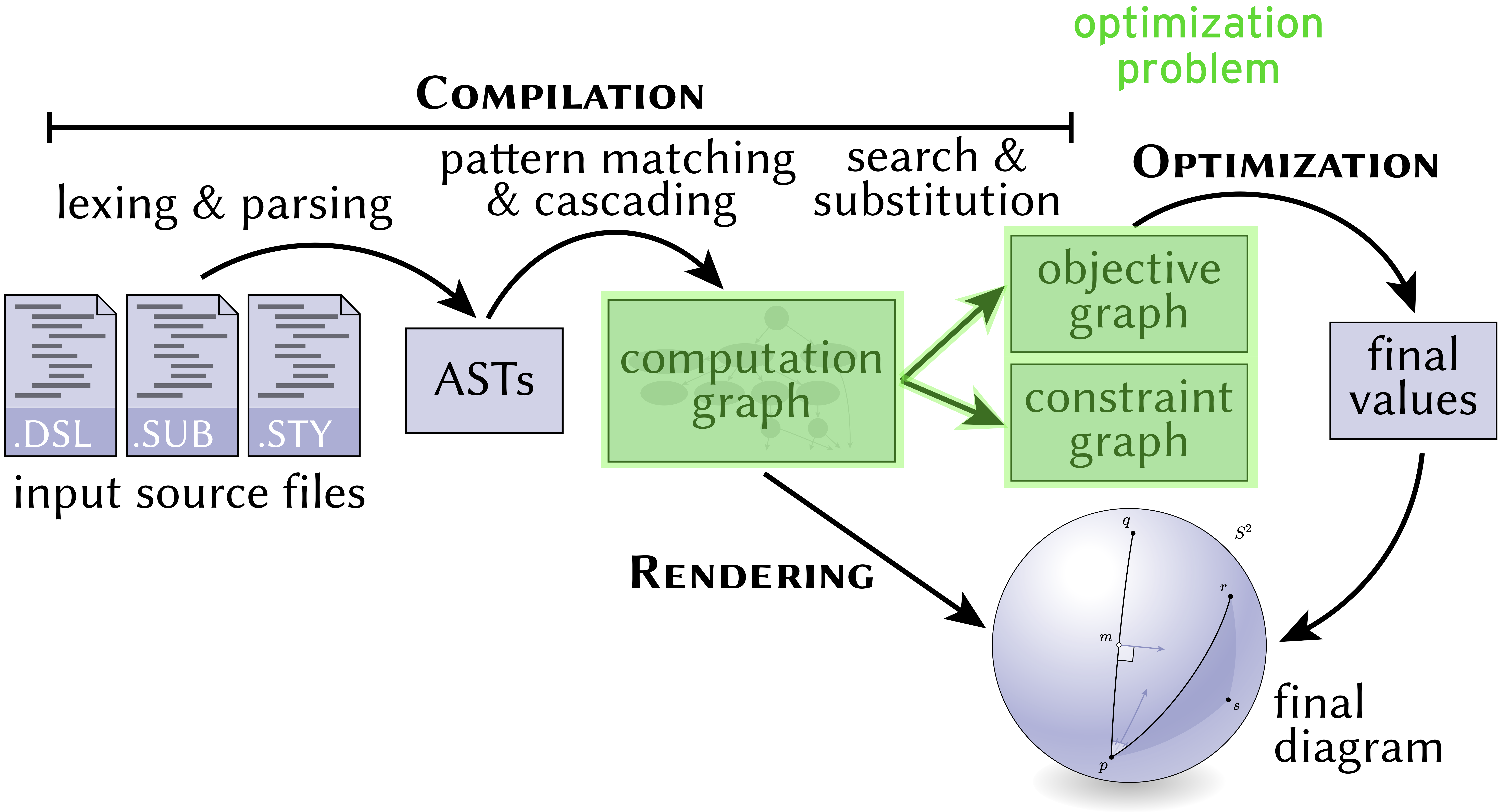
The Penrose pipeline links specification and synthesis



The Penrose pipeline links specification and synthesis



The Penrose pipeline links specification and synthesis



Building the computation graph

Building the computation graph

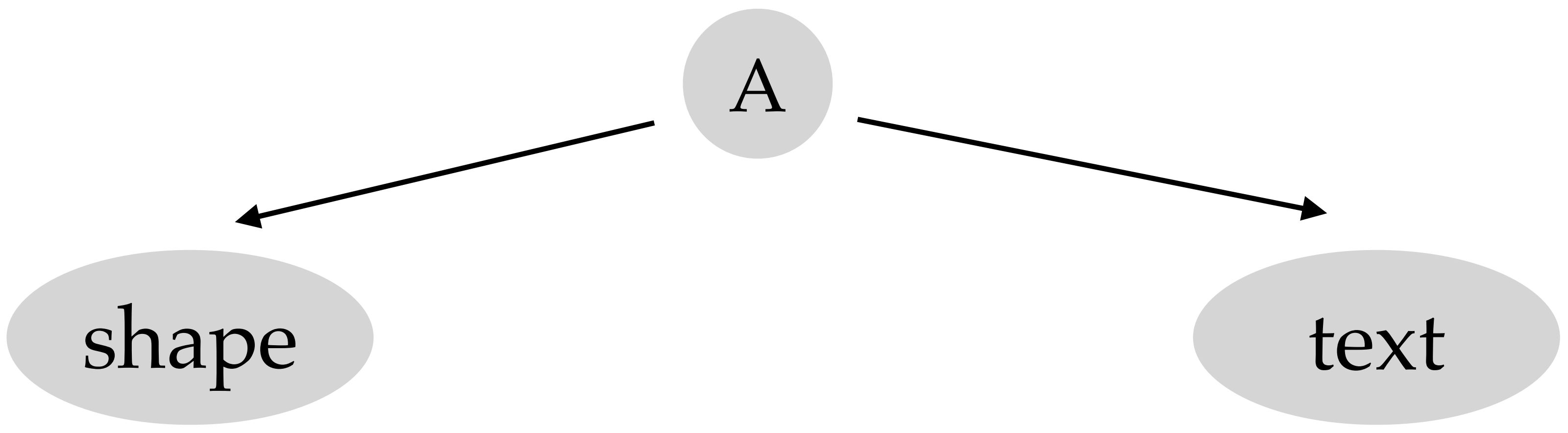
```
Set A, B
B ⊂ A
```

```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```

Building the computation graph

```
Set A, B
B ⊂ A
```

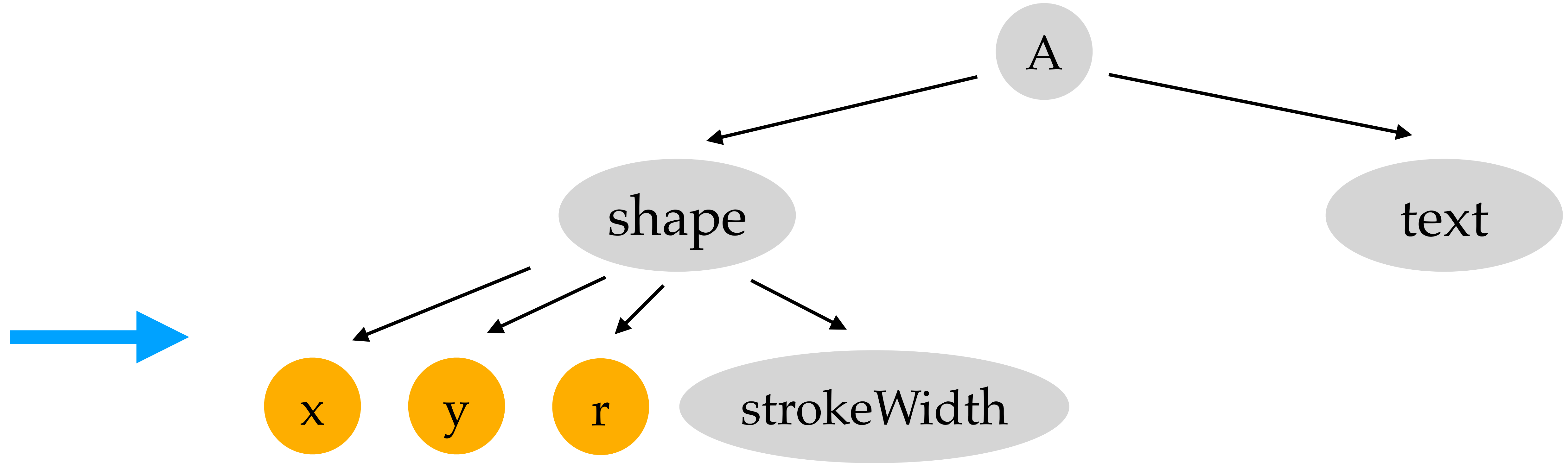
```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



Building the computation graph

```
Set A, B
B ⊂ A
```

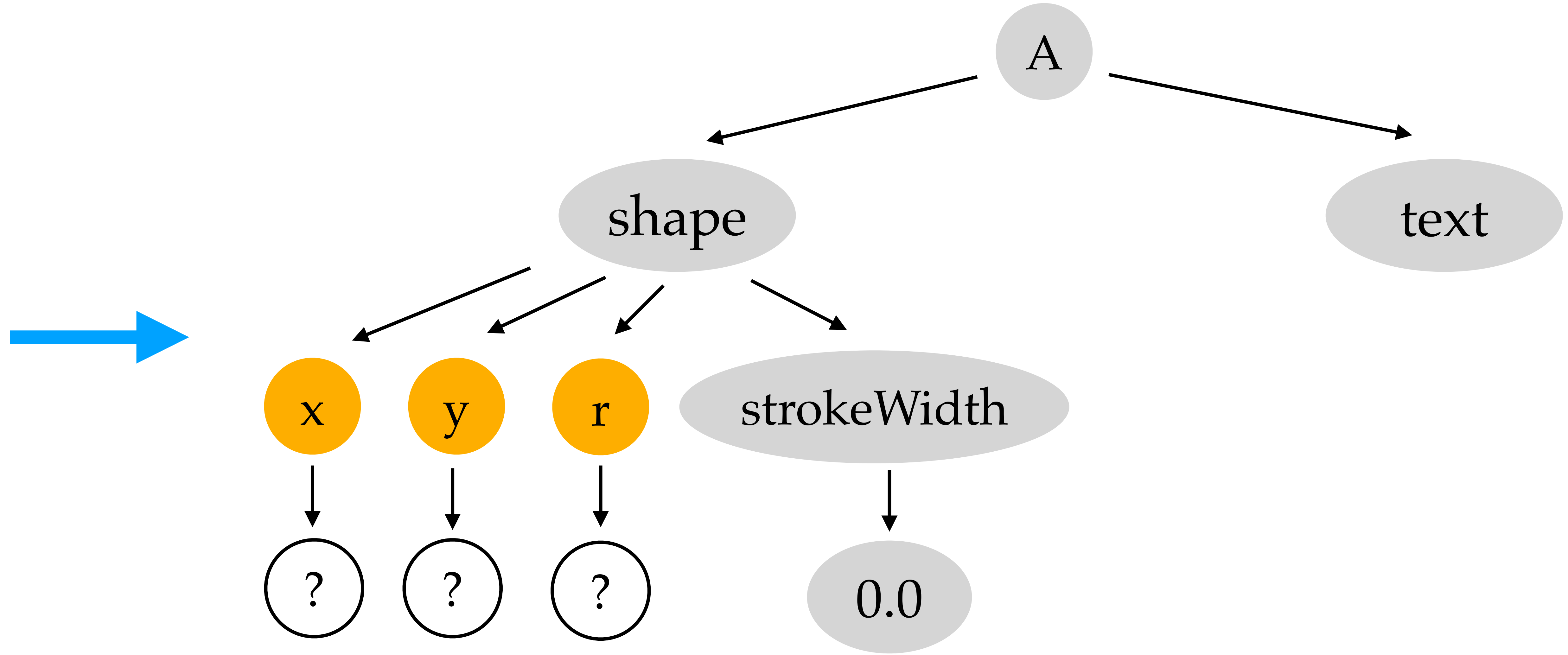
```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



Building the computation graph

```
Set A, B
B ⊂ A
```

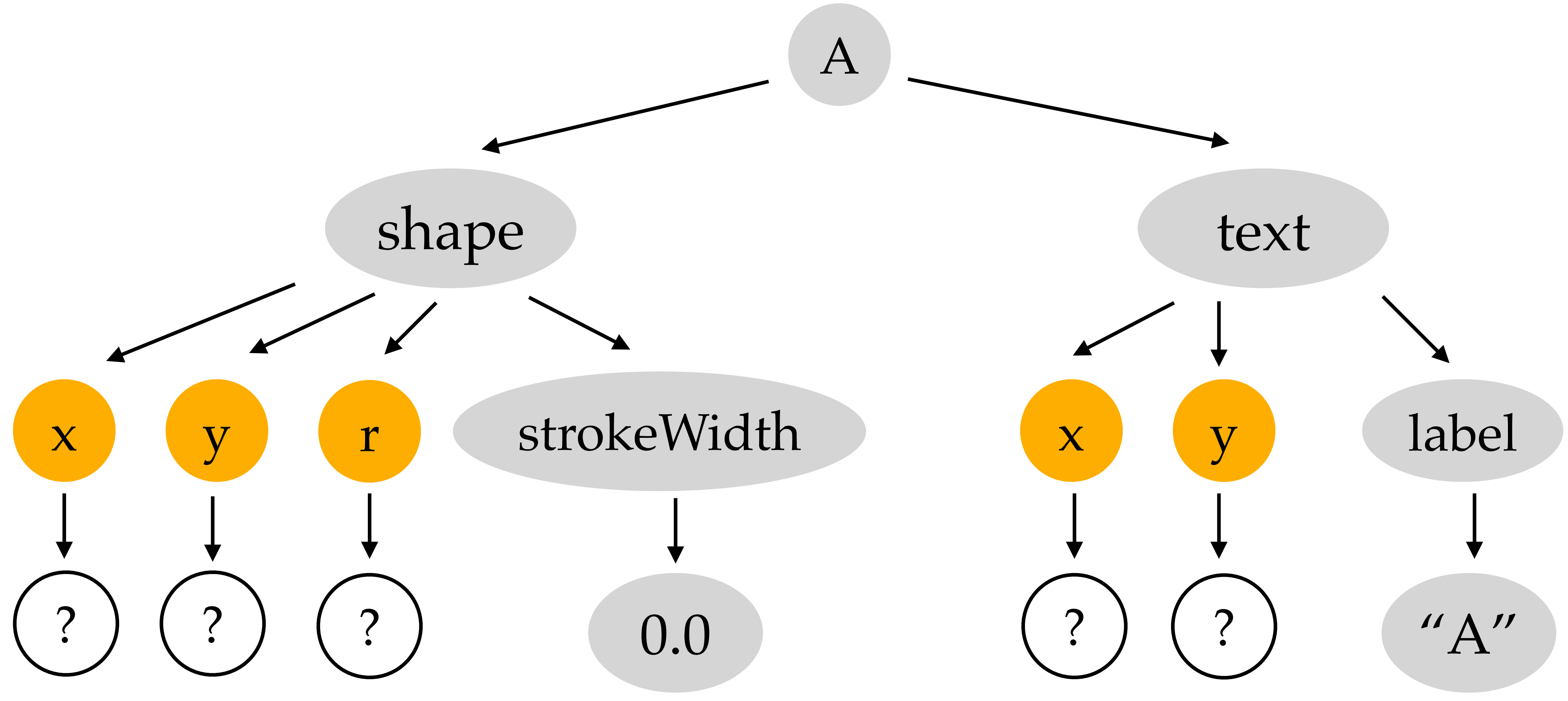
```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



Building the computation graph

```
Set A, B
B ⊂ A
```

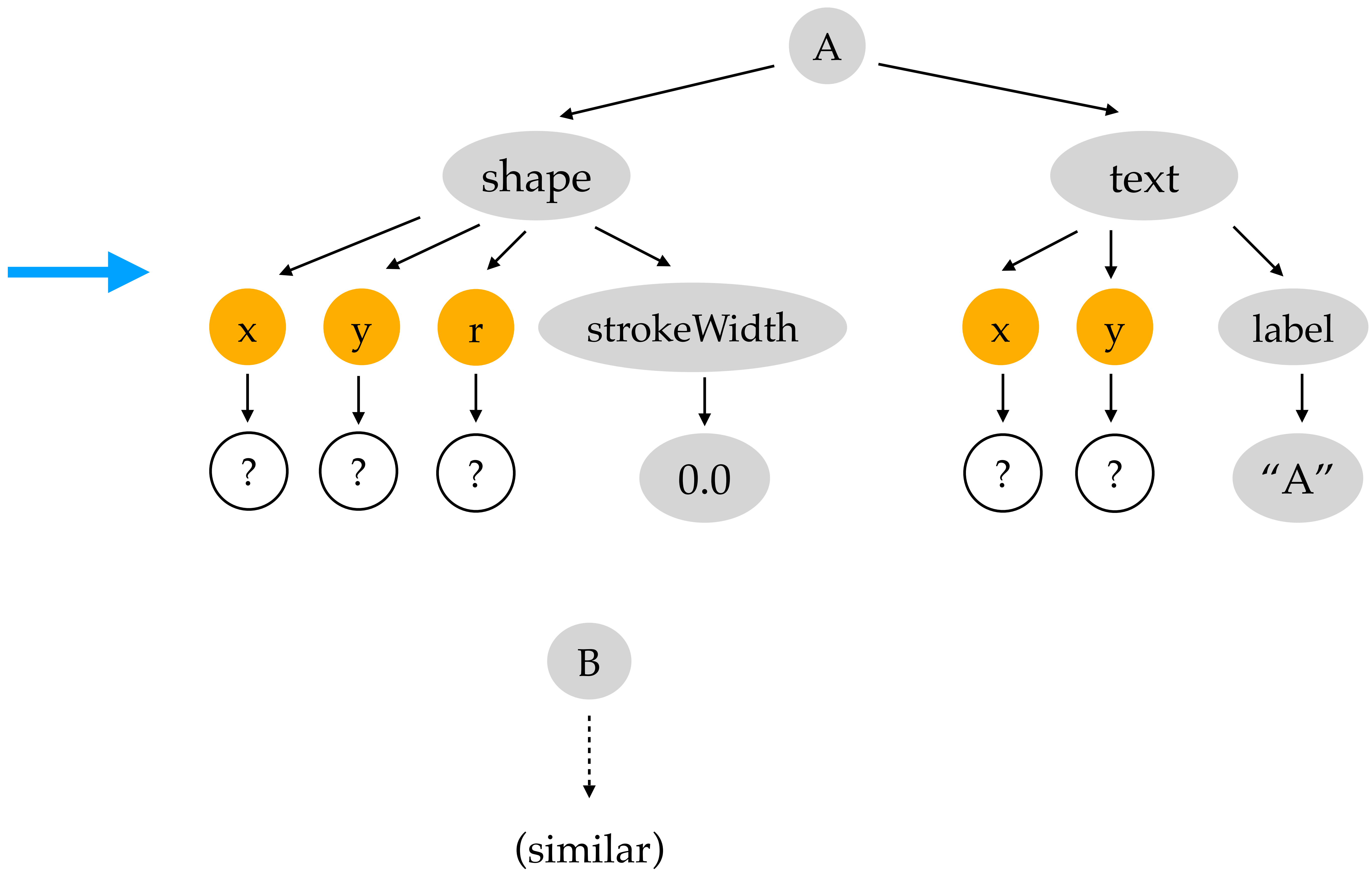
```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



Building the computation graph

```
Set A, B
B ⊂ A
```

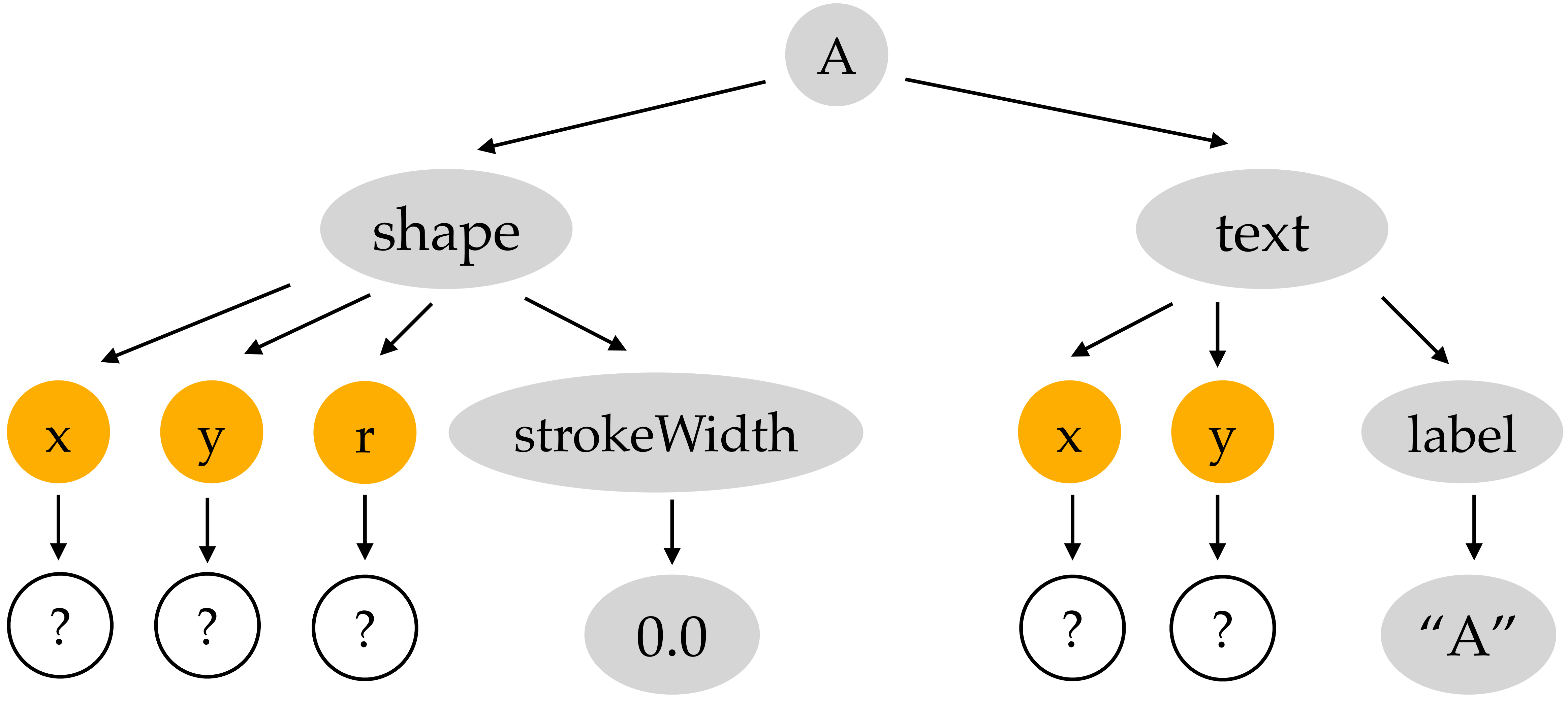
```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```

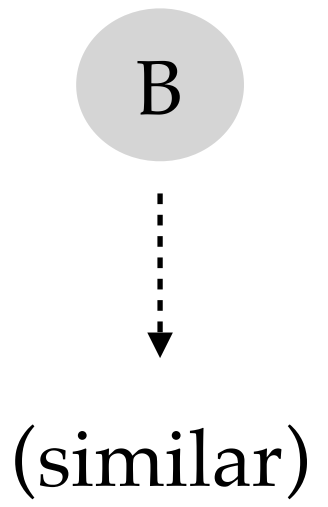


Building the computation graph

```
Set A, B
B ⊂ A
```

```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```





Degrees of freedom:

A.shape.x	B.shape.x
A.shape.y	B.shape.y
A.shape.r	B.shape.r
A.text.x	B.text.x
A.text.y	B.text.y

Building the constraint graph

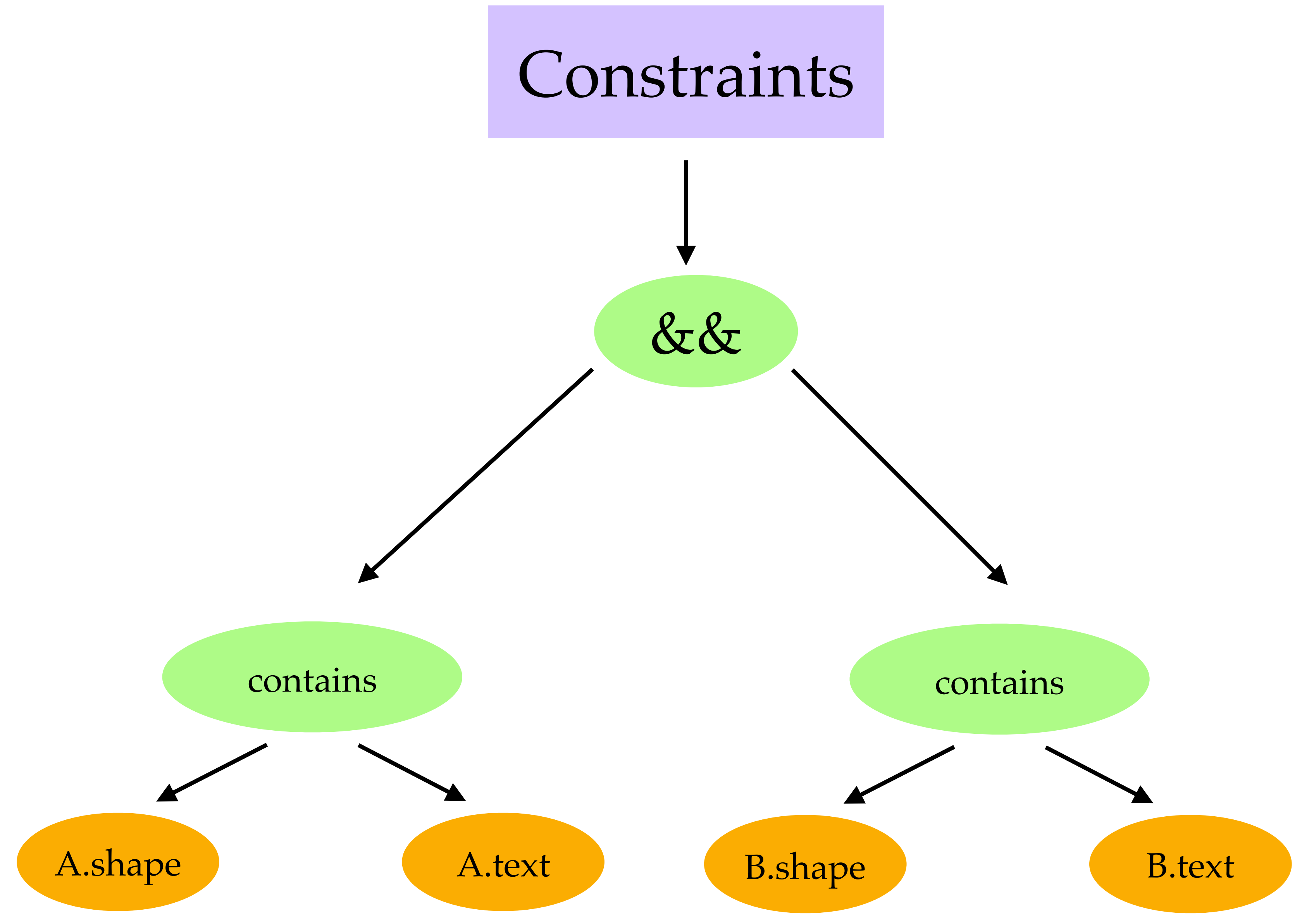
```
Set A, B
B ⊂ A
```

```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```

Building the constraint graph

```
Set A, B
B ⊂ A
```

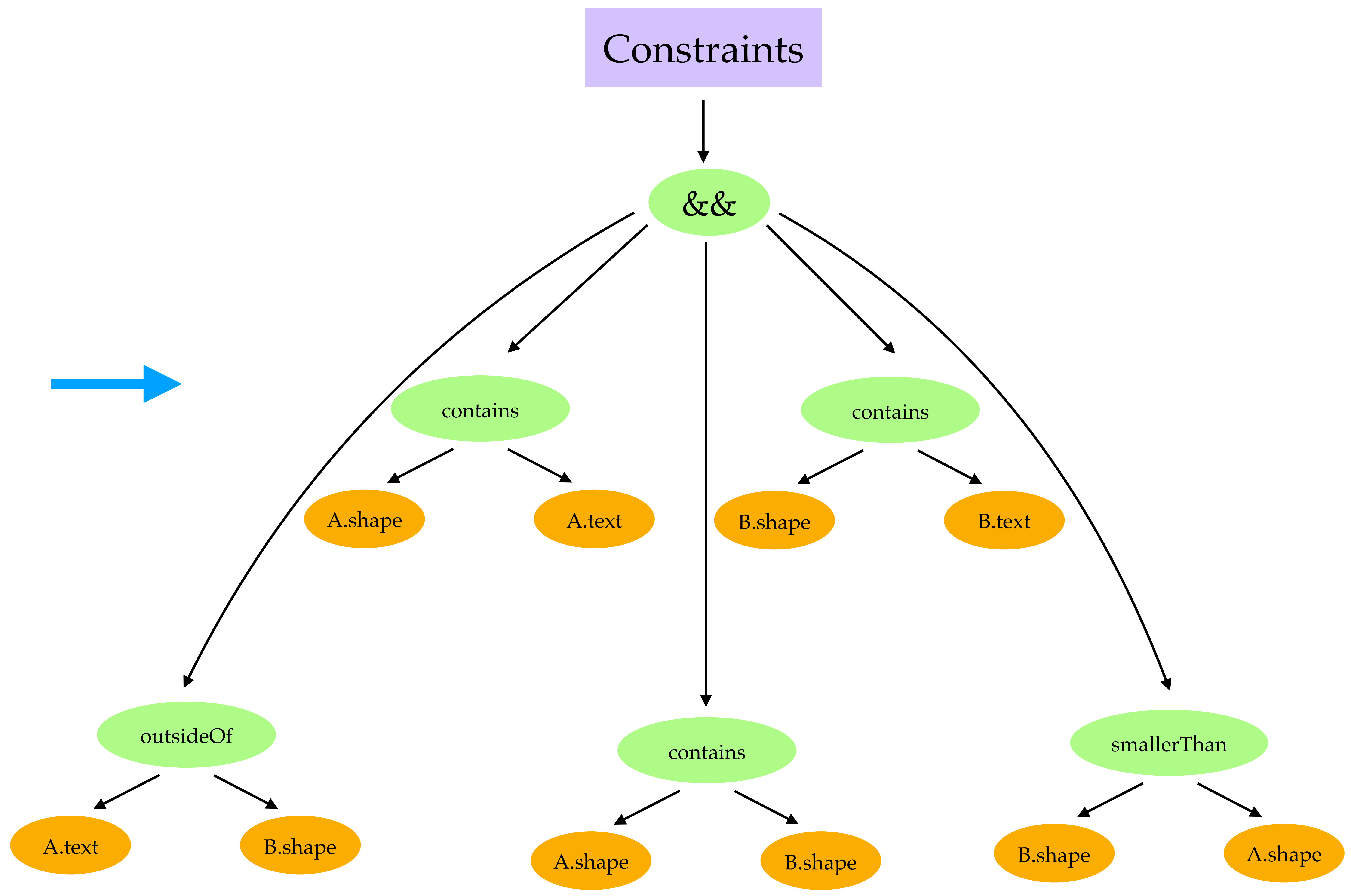
```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



Building the constraint graph

```
Set A, B
B ⊂ A
```

```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```

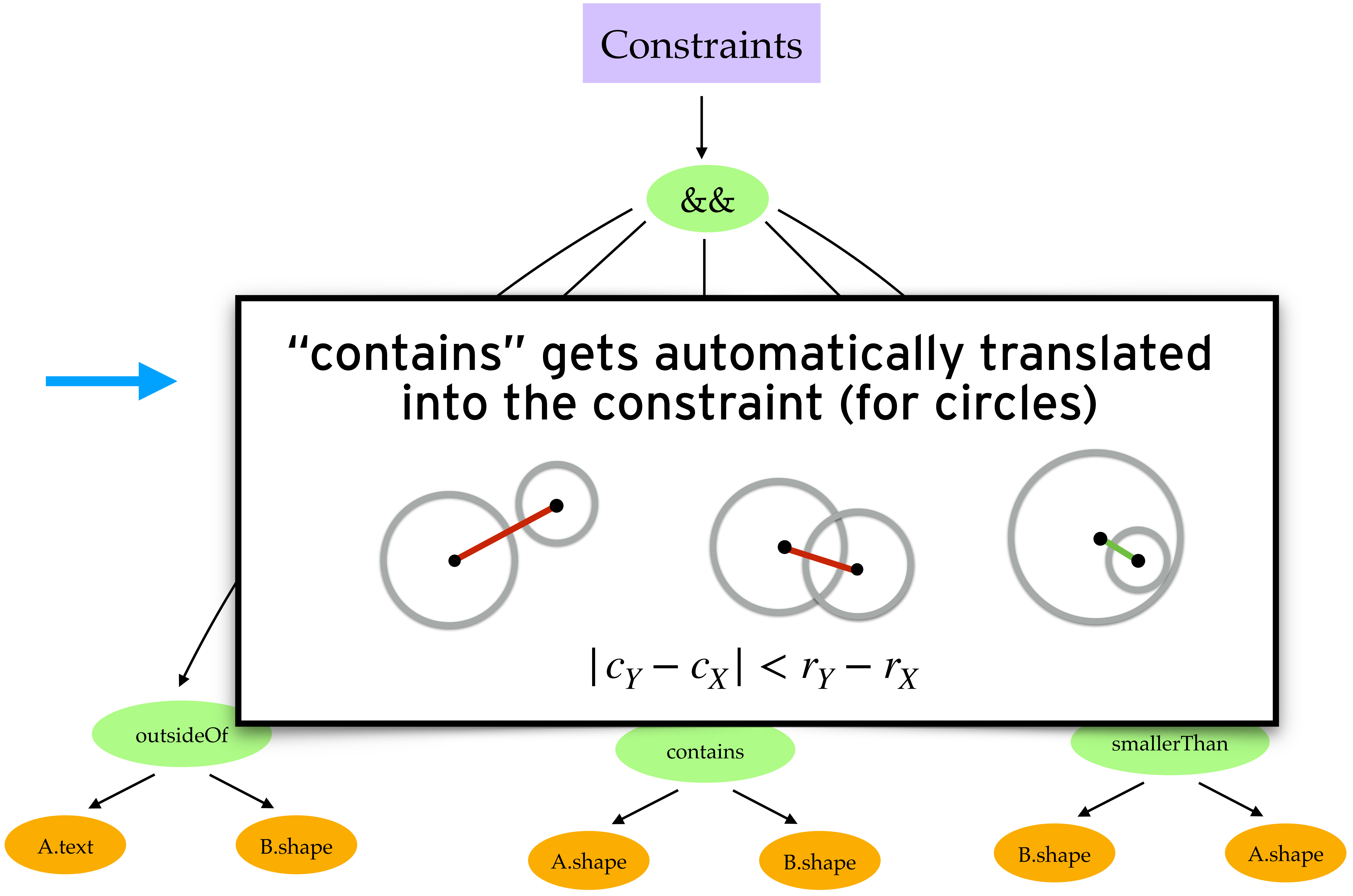


Building the constraint graph

Set A, B

B ⊂ A

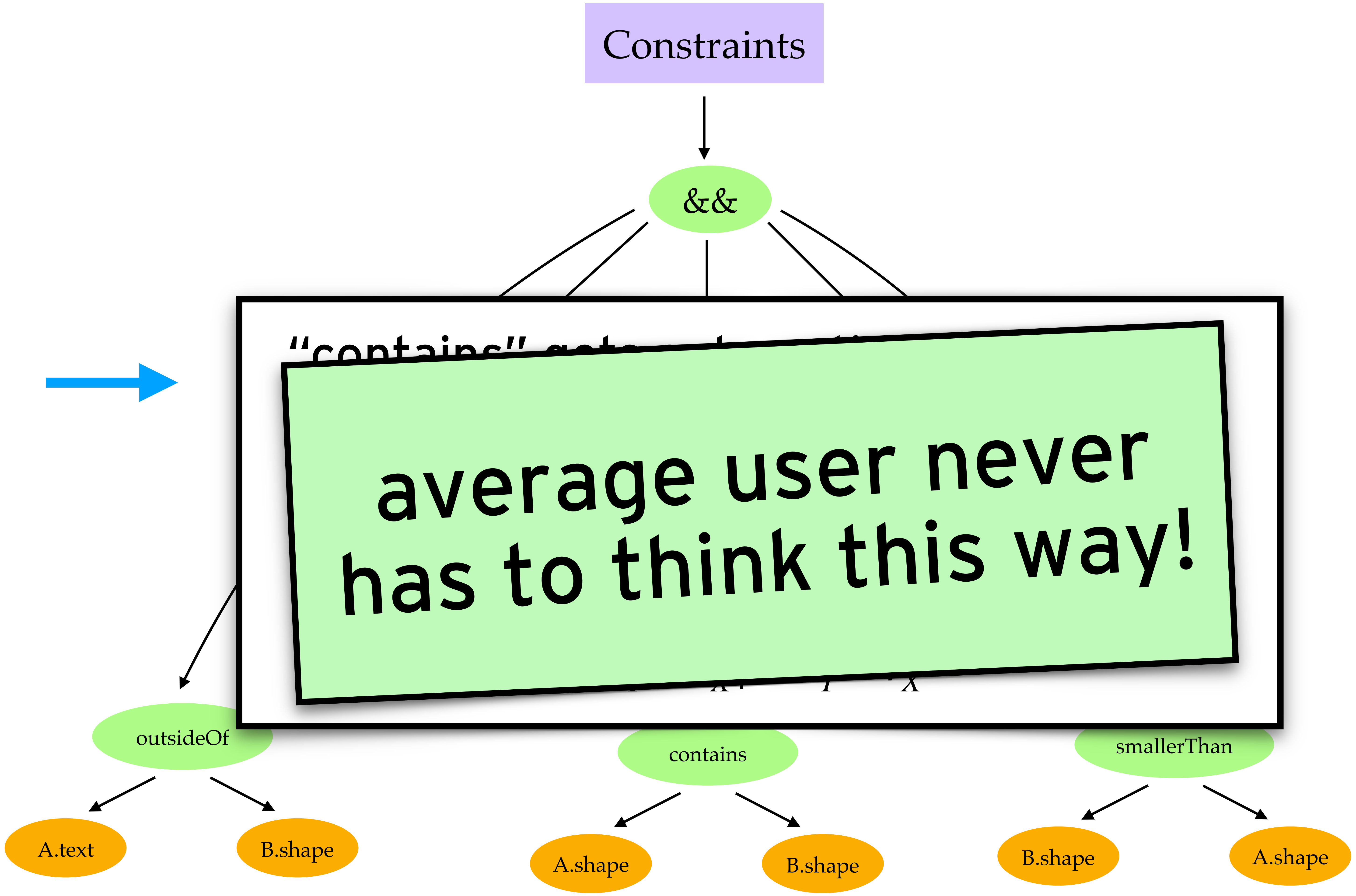
```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



Building the constraint graph

```
Set A, B
B ⊂ A
```

```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



Building the objective graph

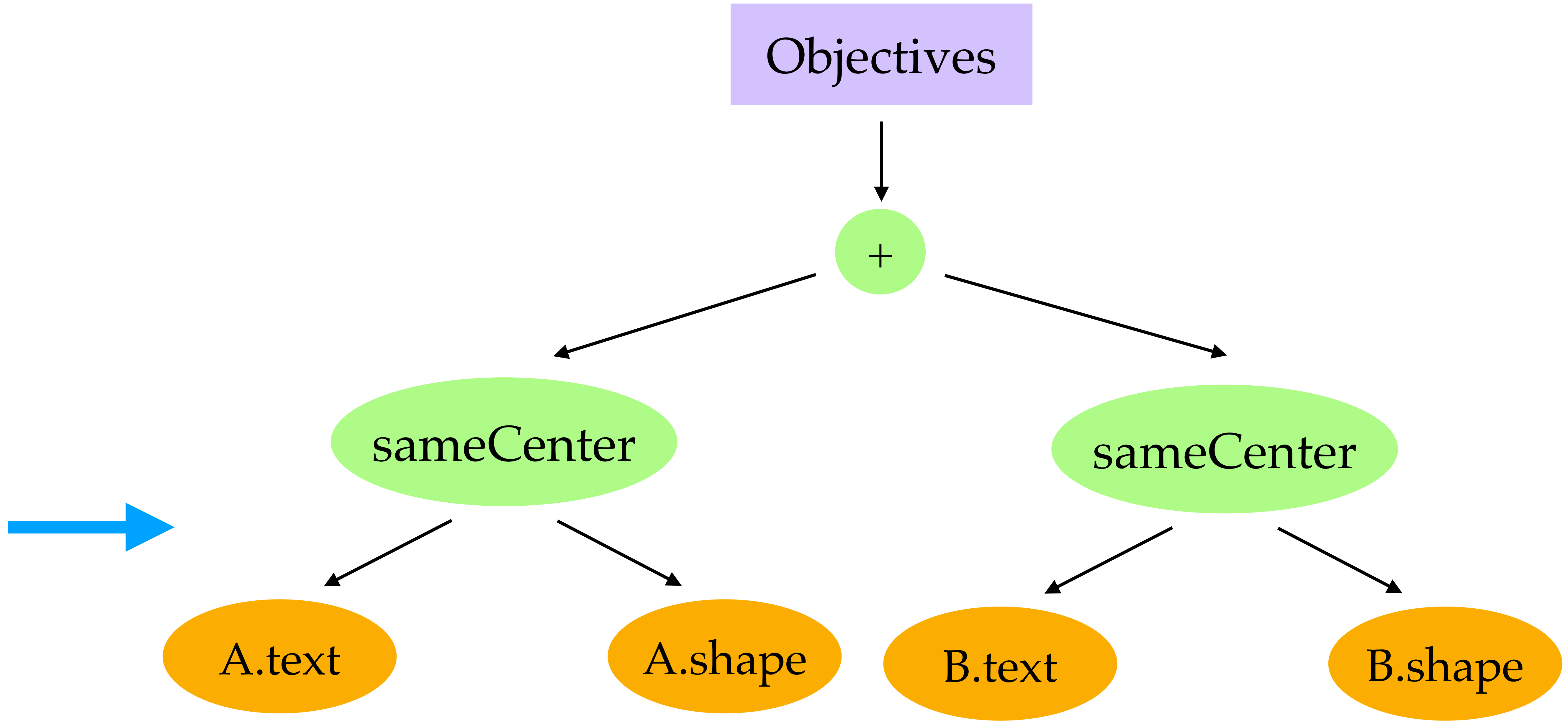
```
Set A, B
B ⊂ A
```

```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```

Building the objective graph

```
Set A, B
B ⊂ A
```

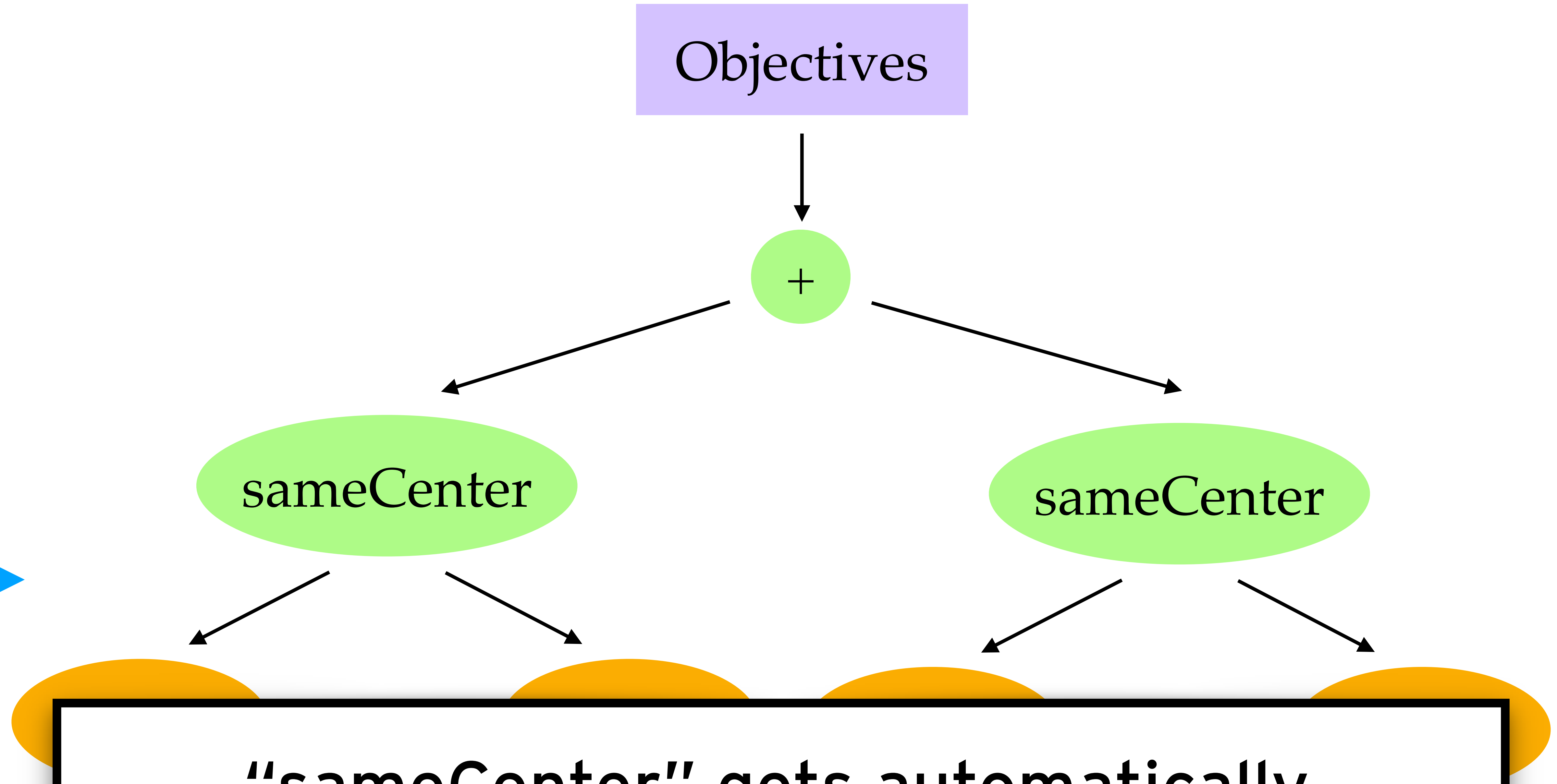
```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



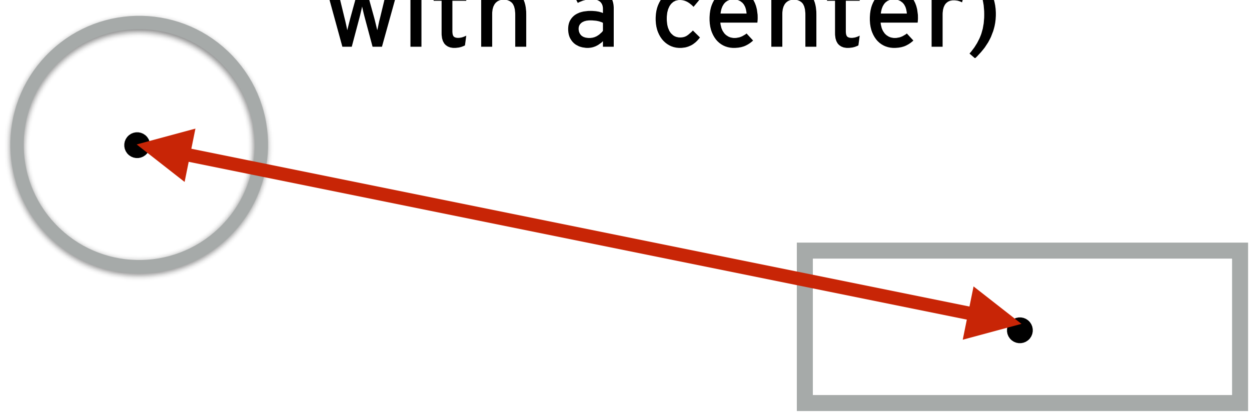
Building the objective graph

```
Set A, B
B ⊂ A
```

```
forall Set x { -- Sets-Disks.sty
    x.shape = Circle { strokeWidth : 0.0 }
    x.text = Text { string : x.label }
    ensure contains(x.shape, x.text)
    encourage sameCenter(x.text, x.shape)
    layer x.shape below x.text
}

forall Set x; Set y
where IsSubset(x, y) {
    ensure contains(y.shape, x.shape)
    ensure smallerThan(x.shape, y.shape)
    ensure outsideOf(y.text, x.shape)
    layer x.shape above y.shape
    layer y.text below x.shape
}
```



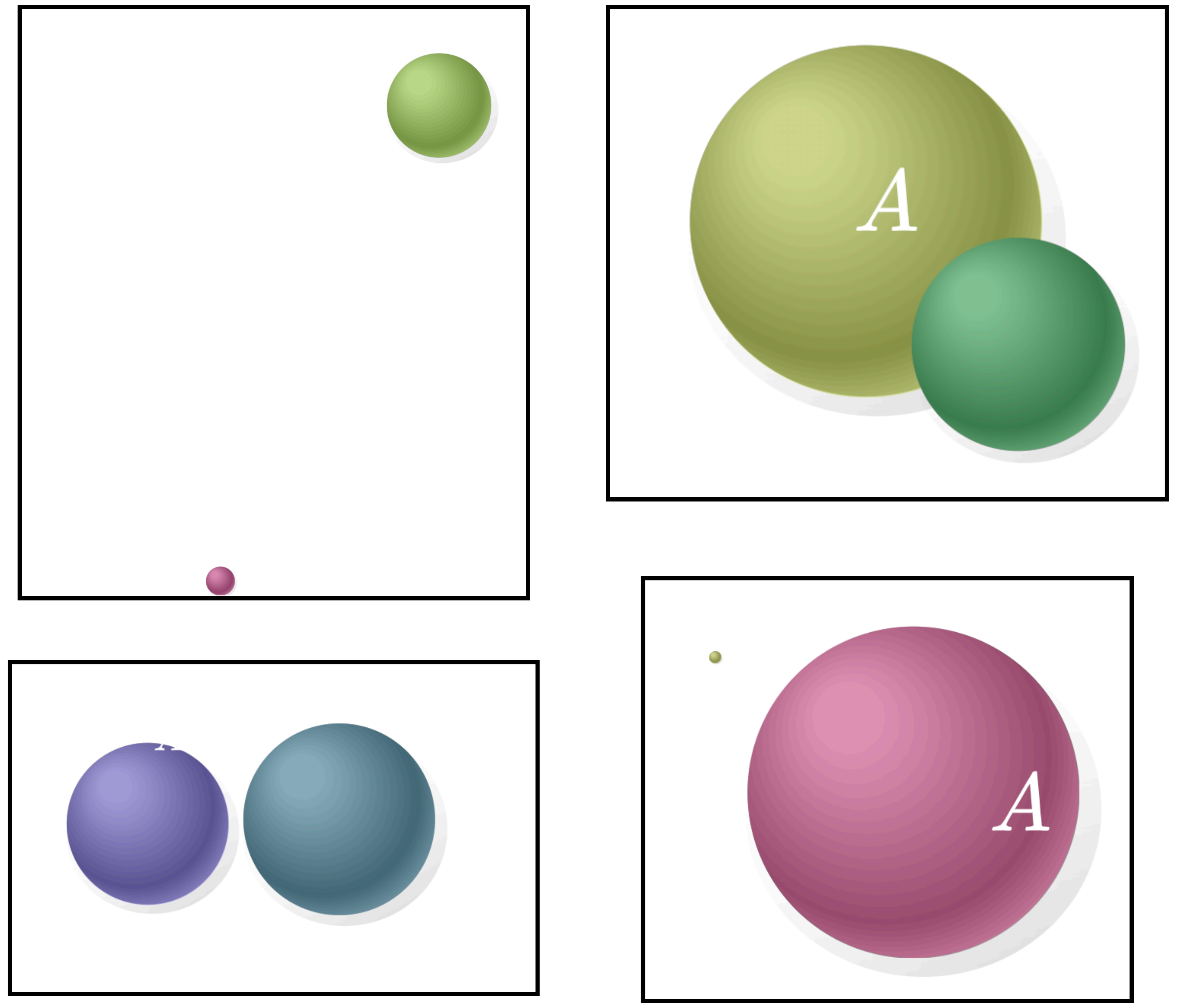
"sameCenter" gets automatically translated into an energy (on any shape with a center)



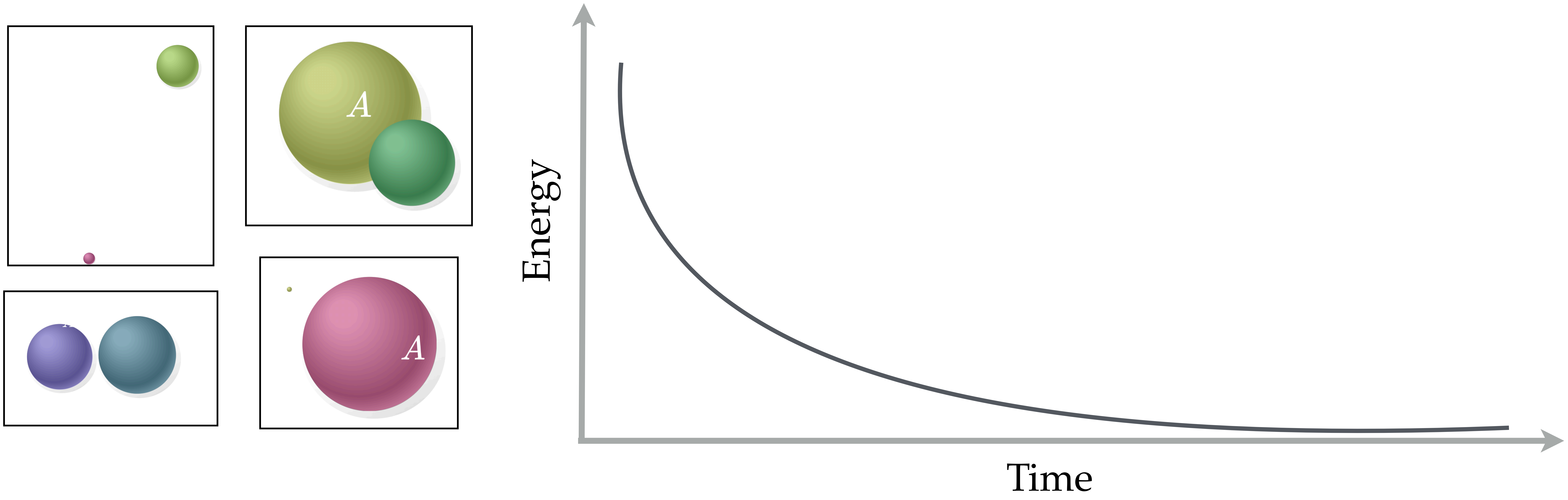
$$e(X, Y) := |c_Y - c_X|^2$$

Constrained optimization yields diagrams

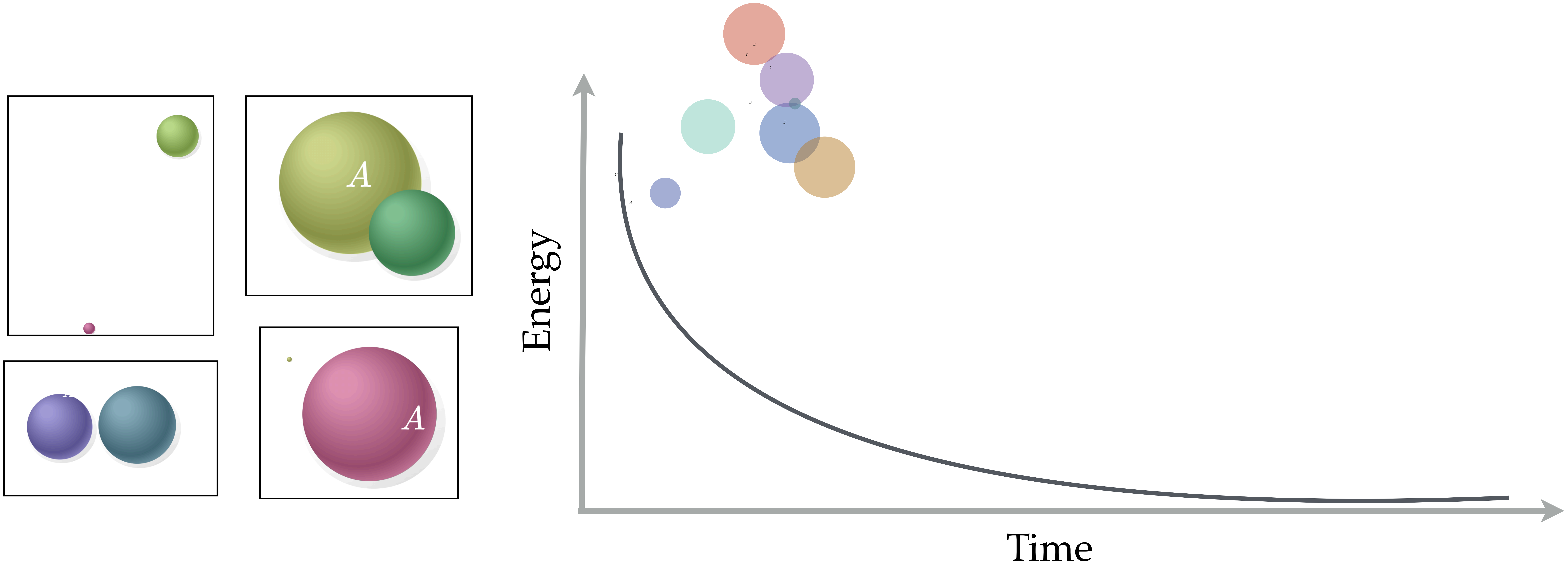
Constrained optimization yields diagrams



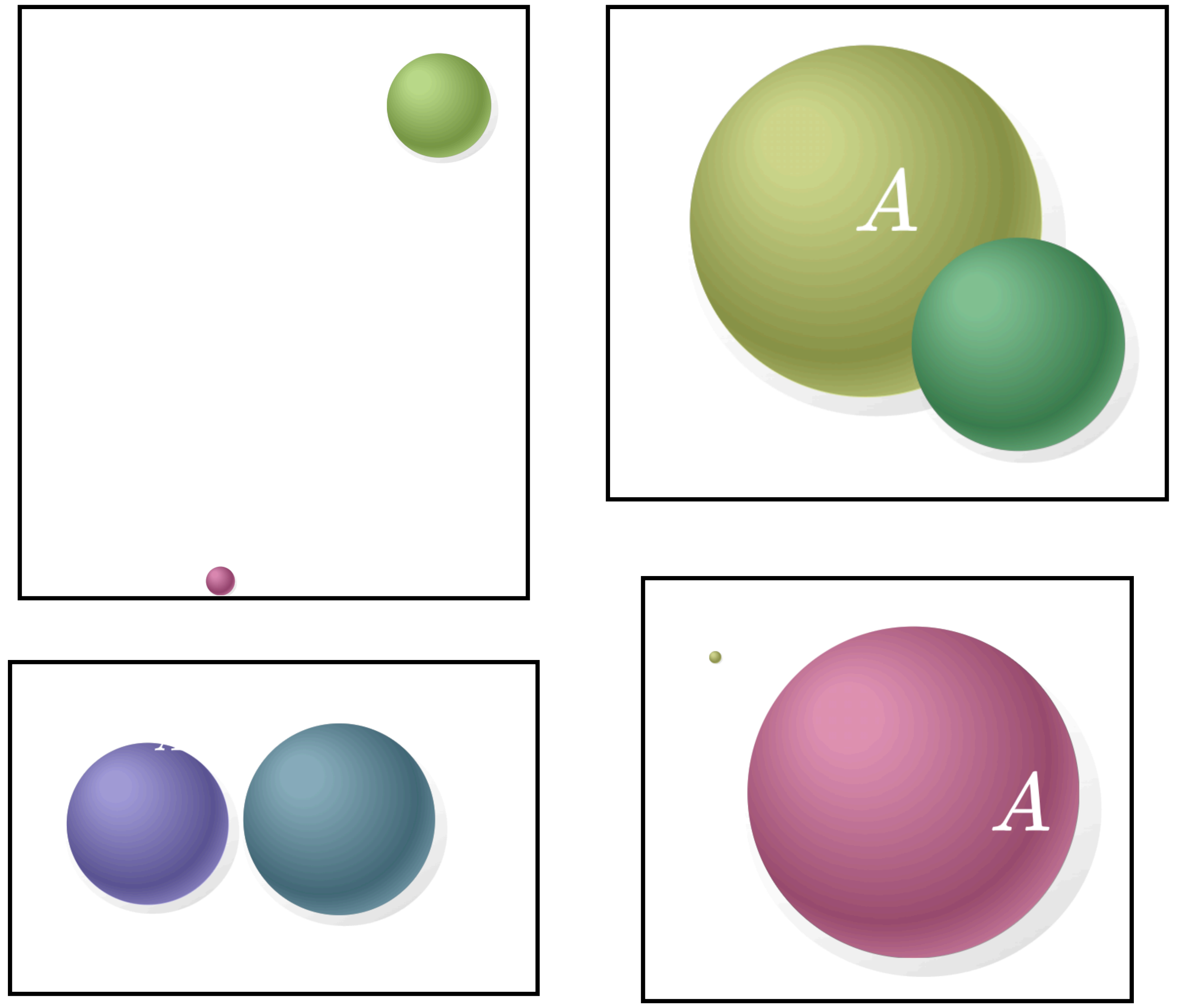
Constrained optimization yields diagrams



Constrained optimization yields diagrams



Constrained optimization yields diagrams



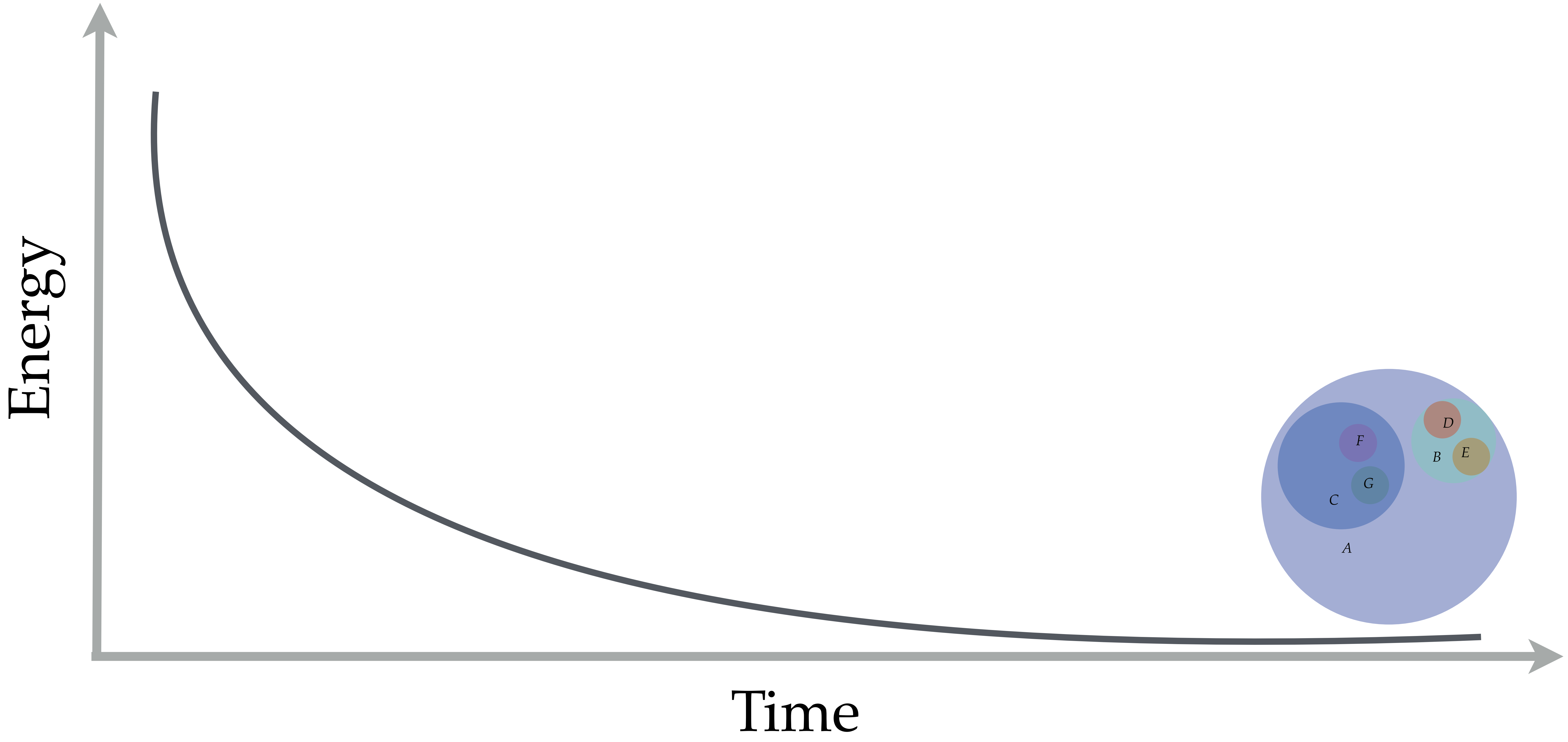


Diagram optimization is hard but tractable

Why it's hard

Why it's tractable

Diagram optimization is hard but tractable

Why it's hard

- User-defined / unpredictable

Why it's tractable

Diagram optimization is hard but tractable

Why it's hard

- User-defined / unpredictable
- Objective has many local minima (nonconvex)

Why it's tractable

Diagram optimization is hard but tractable

Why it's hard

- User-defined / unpredictable
- Objective has many local minima (nonconvex)
- Problem can be ill-conditioned, highly constrained & heterogeneous

Why it's tractable

Diagram optimization is hard but tractable

Why it's hard

- User-defined / unpredictable
- Objective has many local minima (nonconvex)
- Problem can be ill-conditioned, highly constrained & heterogeneous

Why it's tractable

- Only need local minima

Diagram optimization is hard but tractable

Why it's hard

- User-defined / unpredictable
- Objective has many local minima (nonconvex)
- Problem can be ill-conditioned, highly constrained & heterogeneous

Why it's tractable

- Only need local minima
- Few degrees of freedom (10-100)

Diagram optimization is hard but tractable

Why it's hard

- User-defined / unpredictable
- Objective has many local minima (nonconvex)
- Problem can be ill-conditioned, highly constrained & heterogeneous

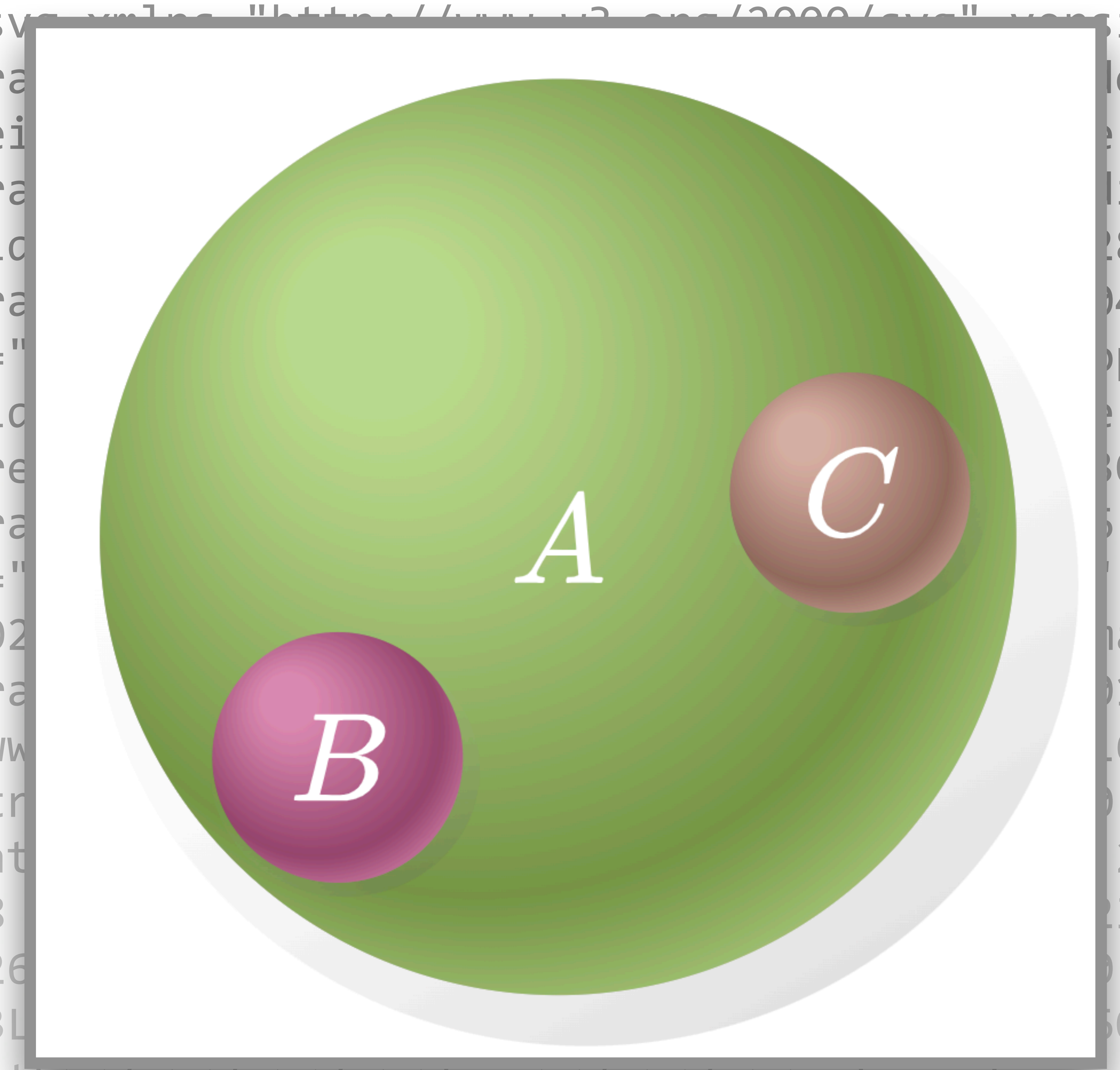
Why it's tractable

- Only need local minima
- Few degrees of freedom (10-100)
- Special problem structure can be exploited by solvers

Rendering a solution yields a 2D vector diagram


```
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" width="100%" height="100%" viewBox="0 0 800 700"><g
transform="translate(0,0)"><image href="shadow.svg" x="460.163787674463" y="160.61744059370855" opacity="1" width="258.9188447398036"
height="267.3487522017338" transform="rotate(0, 589.6232100443648, 294.29181669457546)"></image></g><g
transform="translate(0,0)"><image href="shading.svg" x="461.9702040429623" y="161.58086146056985" opacity="1"
width="240.85472860926063" height="240.85472860926063" transform="rotate(0, 582.3975683475926, 282.00822576520017)"></image></g><g
transform="translate(-726.0125168450339,21.04384106797204)"><circle cx="582.3975683475926" cy="282.00822576520017"
r="120.42736430463032" fill="#9bda32" fill-opacity="0.5" stroke="#386cd4" stroke-opacity="0.5" stroke-dasharray="" stroke-
width="0"><title>A.shape</title><desc>Circle representing A.shape</desc></circle></g><g transform="translate(0,0)"><image
href="shadow.svg" x="482.6010251852498" y="304.6750304069887" opacity="1" width="43.07014973908817" height="44.472432284558586"
transform="rotate(0, 504.13610005479387, 326.911246549268)"></image></g><g transform="translate(0,0)"><image href="shading.svg"
x="482.9015155708013" y="304.83529174694326" opacity="1" width="40.065253794072554" height="40.065253794072554" transform="rotate(0,
502.93414246783755, 324.8679186439795)"></image></g><g transform="translate(0,0)"><g pointer-events="bounding-box"
transform="translate(552.290727271435,251.90138468904257)"><svg style="vertical-align: 0px; font-size: 12pt;" xmlns="http://
www.w3.org/2000/svg" width="60.21368215231516" height="60.21368215231516" role="img" focusable="false" viewBox="0 -716 750 716"><g
stroke="none" fill="#ffffff" stroke-width="0" transform="matrix(1 0 0 -1 0 0)"><g data-mml-node="math"><g data-mml-node="mi"><path
data-c="41" d="M208 74Q208 50 254 46Q272 46 272 35Q272 34 270 22Q267 8 264 4T251 0Q249 0 239 0T205 1T141 2Q70 2 50 0H42Q35 7 35 11Q37
38 48 46H62Q132 49 164 96Q170 102 345 401T523 704Q530 716 547 716H555H572Q578 707 578 706L606 383Q634 60 636 57Q641 46 701 46Q726 46
726 36Q726 34 723 22Q720 7 718 4T704 0Q701 0 690 0T651 1T578 2Q484 2 455 0H443Q437 6 437 9T439 27Q443 40 445 43L449 46H469Q523 49 533
63L521 213H283L249 155Q208 86 208 74ZM516 260Q516 271 504 416T490 562L463 519Q447 492 400 412L310 260L413 259Q516 259 516 260Z"></
path></g></g></g></svg></g></g><g transform="translate(0,0)"><circle cx="502.93414246783755" cy="324.8679186439795"
r="20.032626897036277" fill="#d83589" fill-opacity="0.5" stroke="#c7965b" stroke-opacity="0.5" stroke-dasharray="" stroke-
width="0"><title>B.shape</title><desc>Circle representing B.shape</desc></circle></g><g transform="translate(0,0)"><g pointer-
events="bounding-box" transform="translate(497.92598574357845,319.8597619197204)"><svg style="vertical-align: 0px; font-size: 12pt;"
xmlns="http://www.w3.org/2000/svg" width="10.016313448518138" height="10.016313448518138" role="img" focusable="false" viewBox="0 -683
759 683"><g stroke="none" fill="#ffffff" stroke-width="0" transform="matrix(1 0 0 -1 0 0)"><g data-mml-node="math"><g data-mml-
node="mi"><path data-c="42" d="M231 637Q204 637 199 638T194 649Q194 676 205 682Q206 683 335 683Q594 683 608 681Q671 671 713 636T756
544Q756 480 698 429T565 360L555 357Q619 348 660 311T702 219Q702 146 630 78T453 1Q446 0 242 0Q42 0 39 2Q35 5 35 10Q35 17 37 24Q42 43 47
45Q51 46 62 46H68Q95 46 128 49Q142 52 147 61Q150 65 219 339T288 628Q288 635 231 637ZM649 544Q649 574 634 600T585 634Q578 636 493
637Q473 637 451 637T416 636H403Q388 635 384 626Q382 622 352 506Q352 503 351 500L320 374H401Q482 374 494 376Q554 386 601 434T649
544ZM595 229Q595 273 572 302T512 336Q506 337 429 337Q311 337 310 336Q310 334 293 263T258 122L240 52Q240 48 252 48T333 46Q422 46 429
```

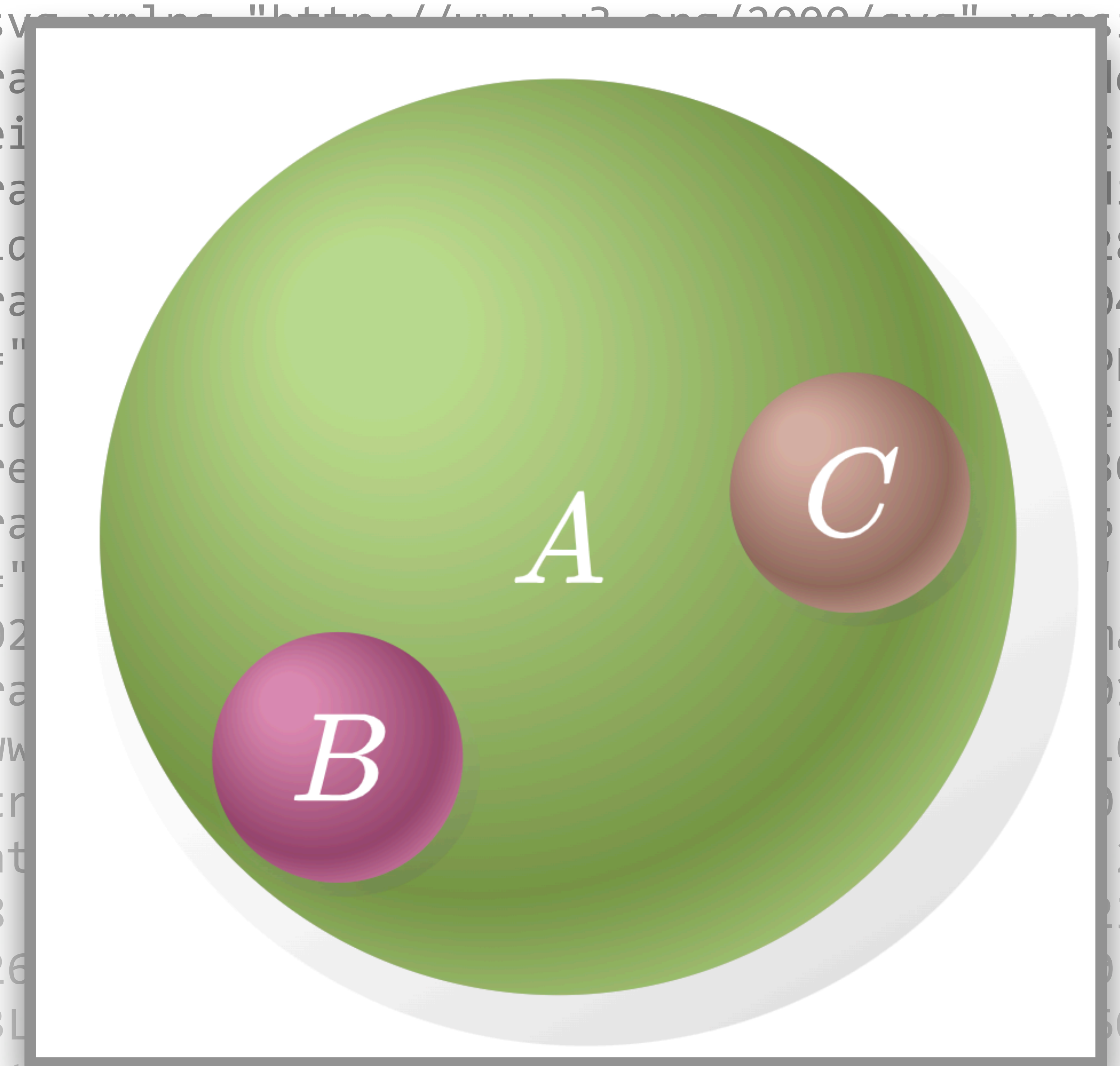
Rendering a solution yields a 2D vector diagram

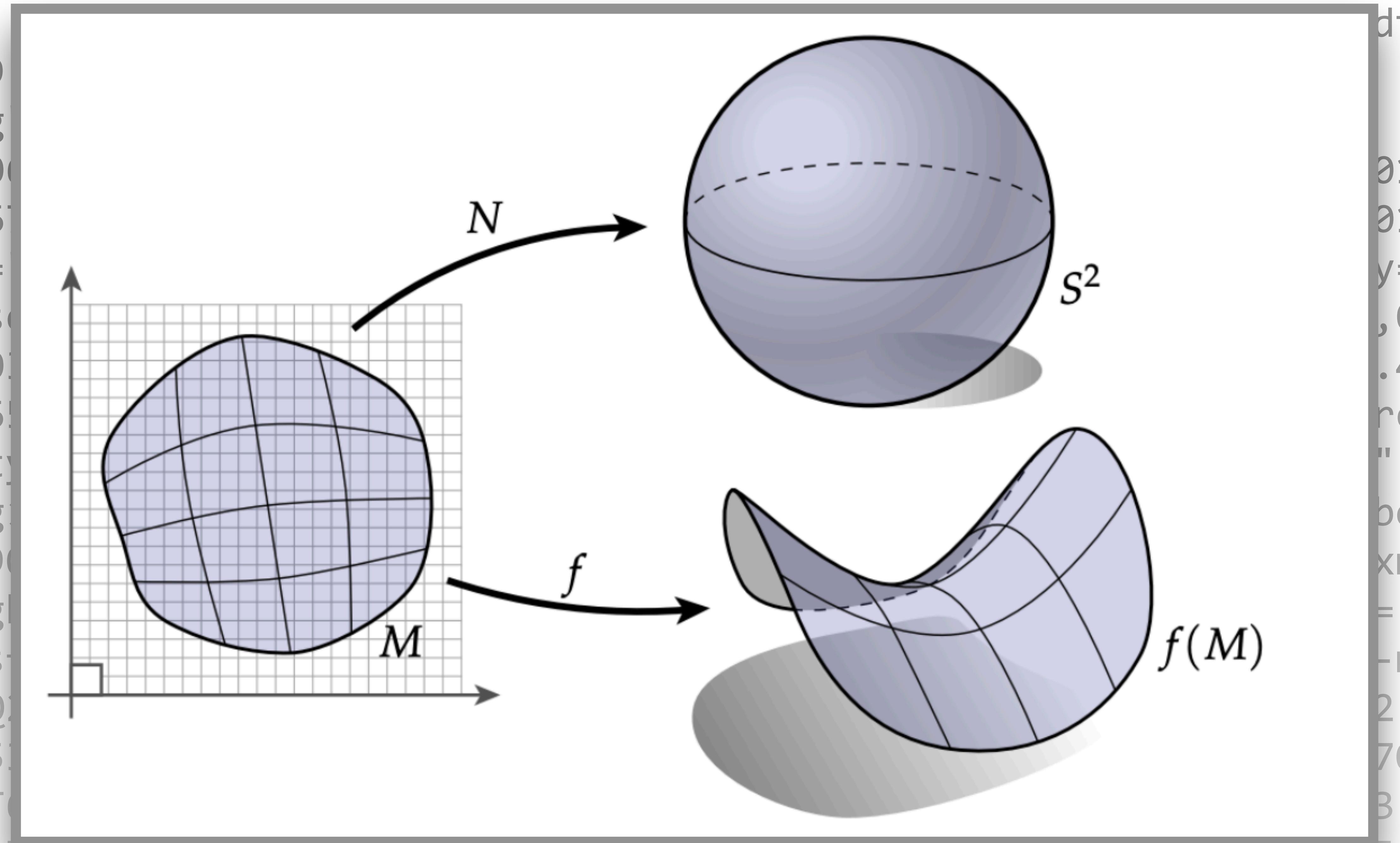



```
<svg xmlns="http://www.w3.org/2000/svg" version="1.2" width="100%" height="100%" viewBox="0 0 800 700"><g
tra                                     low.svg" x="460.163787674463" y="160.61744059370855" opacity="1" width="258.9188447398036"
hei                                     (0, 589.6232100443648, 294.29181669457546)"></image></g><g
tra                                     ling.svg" x="461.9702040429623" y="161.58086146056985" opacity="1"
wid                                     860926063" transform="rotate(0, 582.3975683475926, 282.0082257652017)"></image></g><g
tra                                     4384106797204)"><circle cx="582.3975683475926" cy="282.0082257652017"
r="                                     pacity="0.5" stroke="#386cd4" stroke-opacity="0.5" stroke-dasharray="" stroke-
wid                                     representing A.shape</desc></circle></g><g transform="translate(0,0)"><image
hre                                     04.6750304069887" opacity="1" width="43.07014973908817" height="44.472432284558586"
tra                                     .911246549268)"></image></g><g transform="translate(0,0)"><image href="shading.svg"
x="                                     opacity="1" width="40.065253794072554" height="40.065253794072554" transform="rotate(0,
502                                     age></g><g transform="translate(0,0)"><g pointer-events="bounding-box"
tra                                     138468904257)"><svg style="vertical-align: 0px; font-size: 12pt;" xmlns="http://
www                                     6" height="60.21368215231516" role="img" focusable="false" viewBox="0 -716 750 716"><g
str                                     " transform="matrix(1 0 0 -1 0 0)"><g data-mml-node="math"><g data-mml-node="mi"><path
dat                                     272 35Q272 34 270 22Q267 8 264 4T251 0Q249 0 239 0T205 1T141 2Q70 2 50 0H42Q35 7 35 11Q37
38                                      3 704Q530 716 547 716H555H572Q578 707 578 706L606 383Q634 60 636 57Q641 46 701 46Q726 46
726                                     690 0T651 1T578 2Q484 2 455 0H443Q437 6 437 9T439 27Q443 40 445 43L449 46H469Q523 49 533
63L                                     0Q516 271 504 416T490 562L463 519Q447 492 400 412L310 260L413 259Q516 259 516 260Z"></
path></g></g></g></svg></g></g><g transform="translate(0,0)"><circle cx="502.93414246783755" cy="324.8679186439795"
r="20.03262689703627" fill="#d83589" fill-opacity="0.5" stroke="#c7965b" stroke-opacity="0.5" stroke-dasharray="" stroke-
width="0"><title>B.shape</title><desc>Circle representing B.shape</desc></circle></g><g transform="translate(0,0)"><g pointer-
events="bounding-box" transform="translate(497.92598574357845,319.8597619197204)"><svg style="vertical-align: 0px; font-size: 12pt;"
xmlns="http://www.w3.org/2000/svg" width="10.016313448518138" height="10.016313448518138" role="img" focusable="false" viewBox="0 -683
759 683"><g stroke="none" fill="#ffffff" stroke-width="0" transform="matrix(1 0 0 -1 0 0)"><g data-mml-node="math"><g data-mml-
node="mi"><path data-c="42" d="M231 637Q204 637 199 638T194 649Q194 676 205 682Q206 683 335 683Q594 683 608 681Q671 671 713 636T756
544Q756 480 698 429T565 360L555 357Q619 348 660 311T702 219Q702 146 630 78T453 1Q446 0 242 0Q42 0 39 2Q35 5 35 10Q35 17 37 24Q42 43 47
45Q51 46 62 46H68Q95 46 128 49Q142 52 147 61Q150 65 219 339T288 628Q288 635 231 637ZM649 544Q649 574 634 600T585 634Q578 636 493
637Q473 637 451 637T416 636H403Q388 635 384 626Q382 622 352 506Q352 503 351 500L320 374H401Q482 374 494 376Q554 386 601 434T649
```

Rendering a solution yields a 2D vector diagram

image credit: Crane et al 2013





path></g></g></g></svg></g></g><g transform="translate(0,0)"><circle cx="502.93414246783755" cy="324.8679186439795"
r="20.032626897036277" fill="#d83589" fill-opacity="0.5" stroke="#c7965b" stroke-opacity="0.5" stroke-dasharray="" stroke-
width="0"><title>B.shape</title><desc>Circle representing B.shape</desc></circle></g><g transform="translate(0,0)"><g pointer-
events="bounding-box" transform="translate(497.92598574357845,319.8597619197204)"><svg style="vertical-align: 0px; font-size: 12pt;"
xmlns="http://www.w3.org/2000/svg" width="10.016313448518138" height="10.016313448518138" role="img" focusable="false" viewBox="0 -683
759 683"><g stroke="none" fill="#ffffff" stroke-width="0" transform="matrix(1 0 0 -1 0 0)"><g data-mml-node="math"><g data-mml-
node="mi"><path data-c="42" d="M231 637Q204 637 199 638T194 649Q194 676 205 682Q206 683 335 683Q594 683 608 681Q671 671 713 636T756
544Q756 480 698 429T565 360L555 357Q619 348 660 311T702 219Q702 146 630 78T453 1Q446 0 242 0Q42 0 39 2Q35 5 35 10Q35 17 37 24Q42 43 47
45Q51 46 62 46H68Q95 46 128 49Q142 52 147 61Q150 65 219 339T288 628Q288 635 231 637ZM649 544Q649 574 634 600T585 634Q578 636 493
637Q473 637 451 637T416 636H403Q388 635 384 626Q382 622 352 506Q352 503 351 500L320 374H401Q482 374 494 376Q554 386 601 434T649

Rendering a solution yields a 2D vector diagram

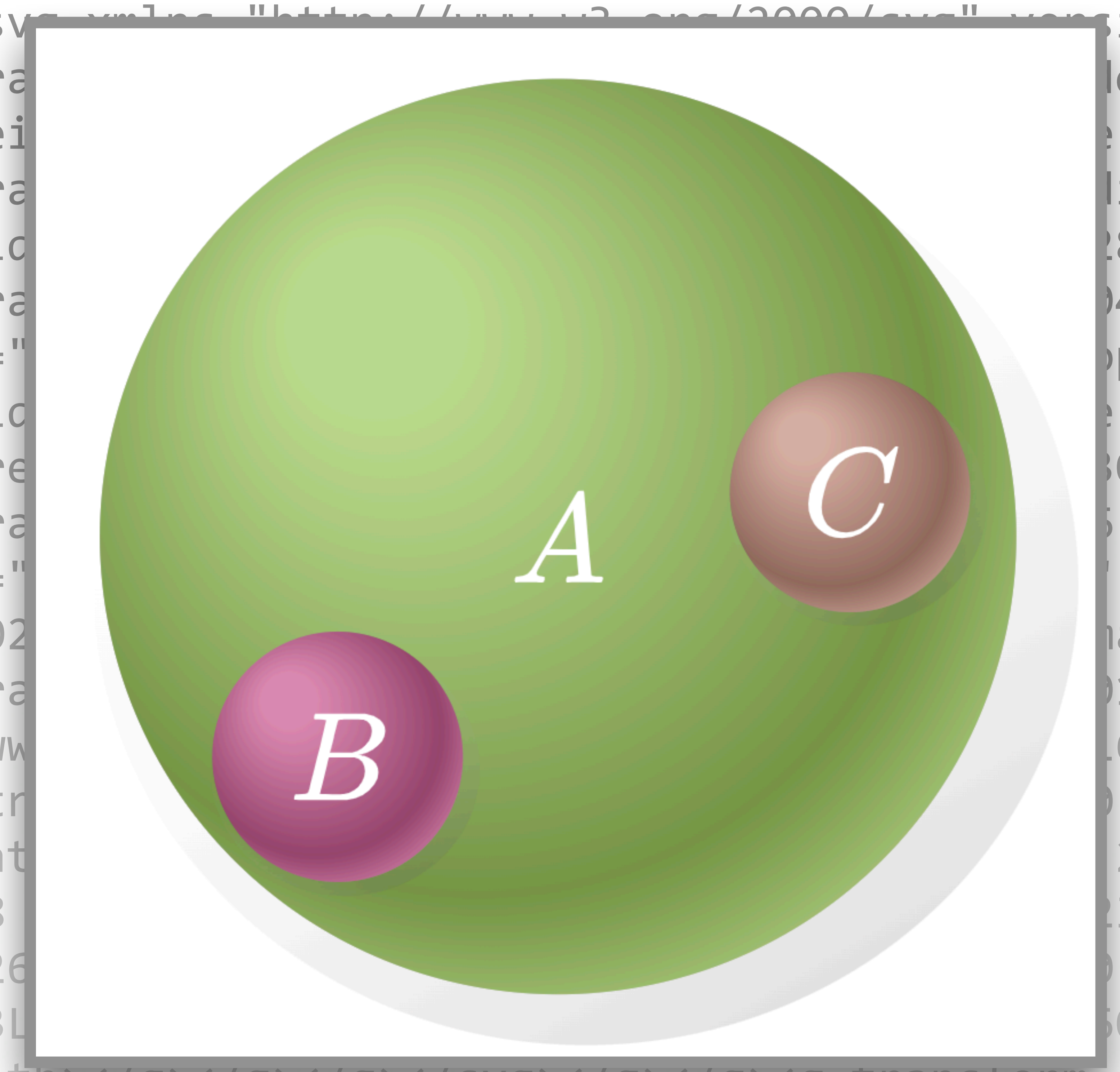
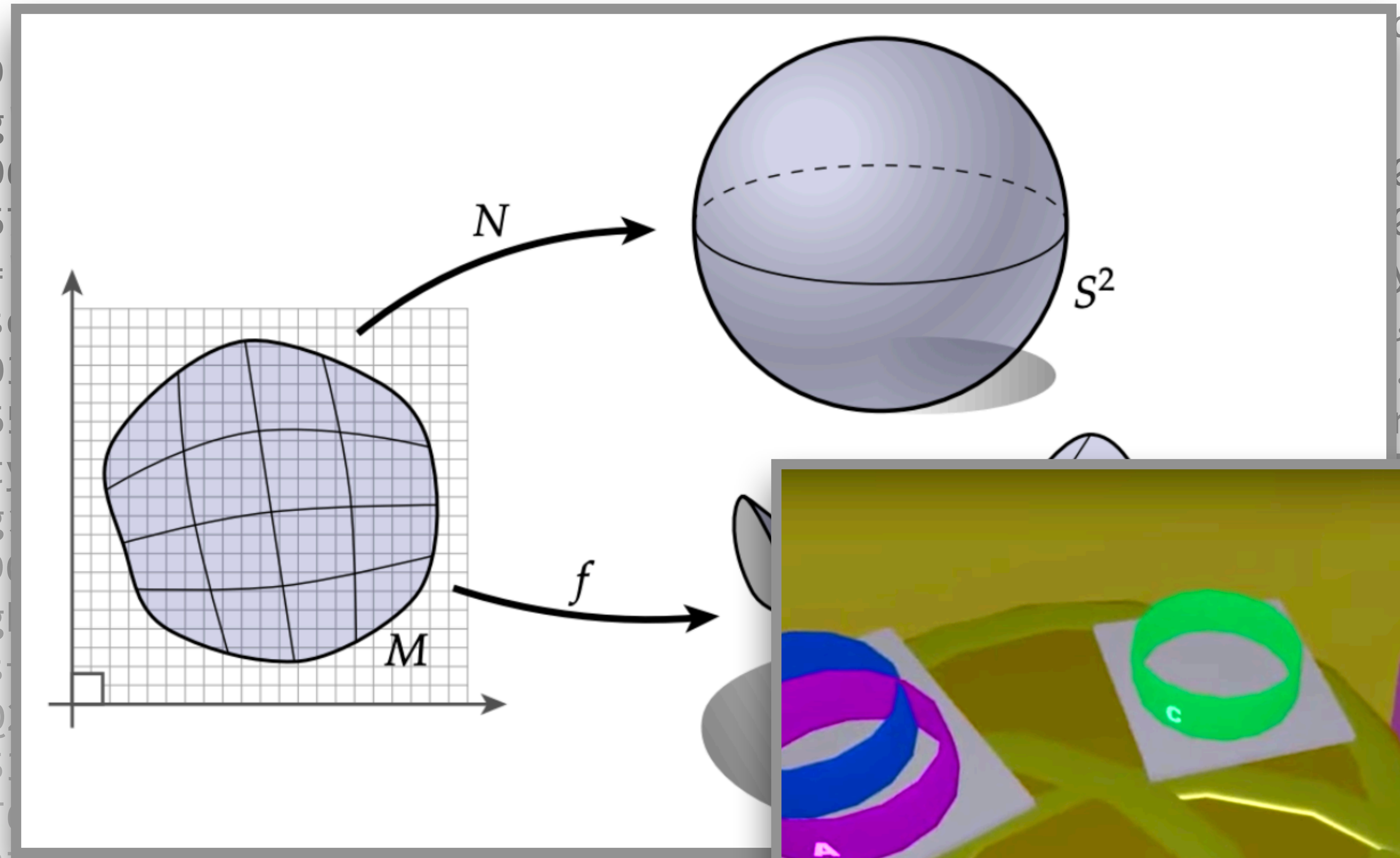


image credit: Crane et al 2013



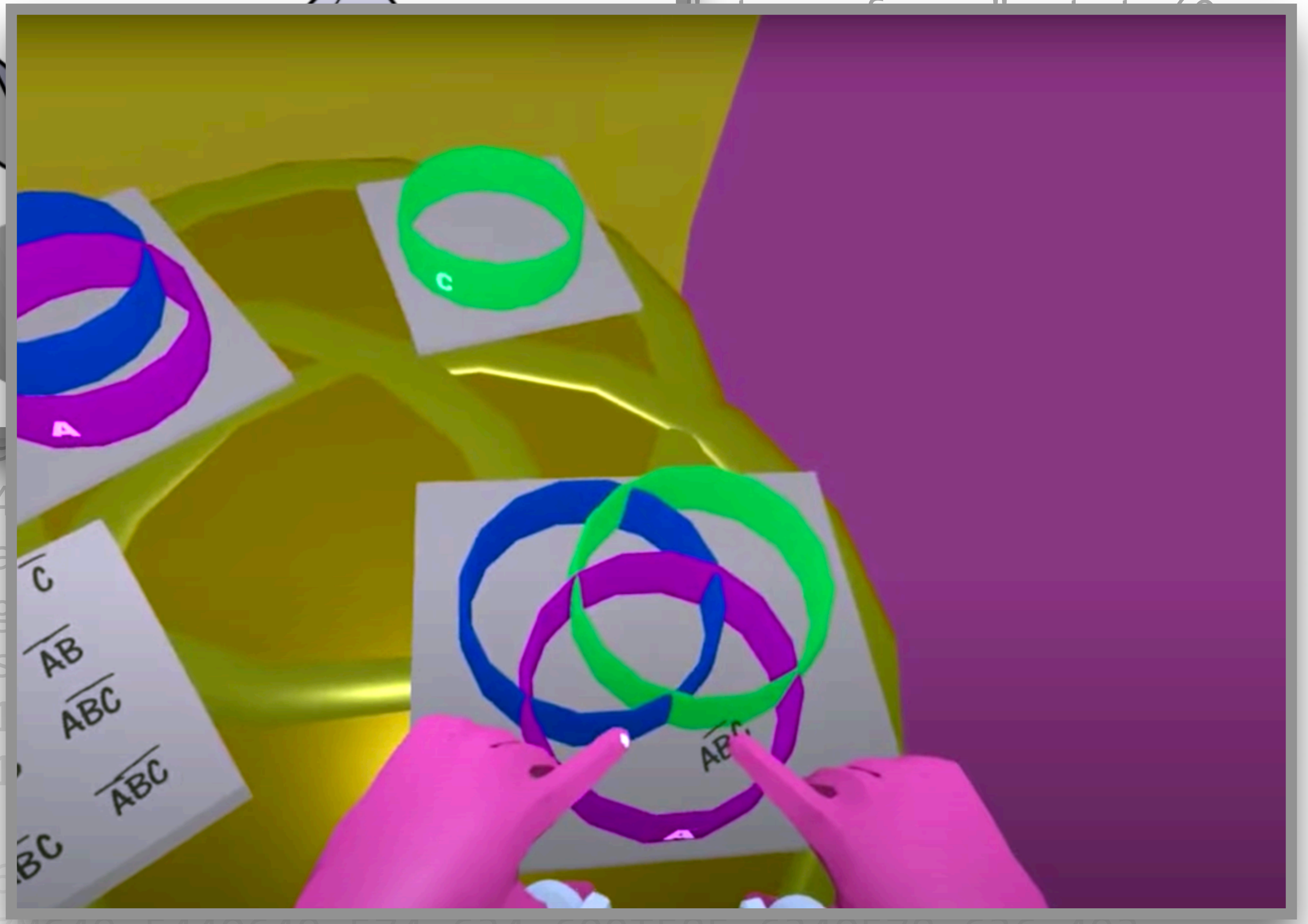
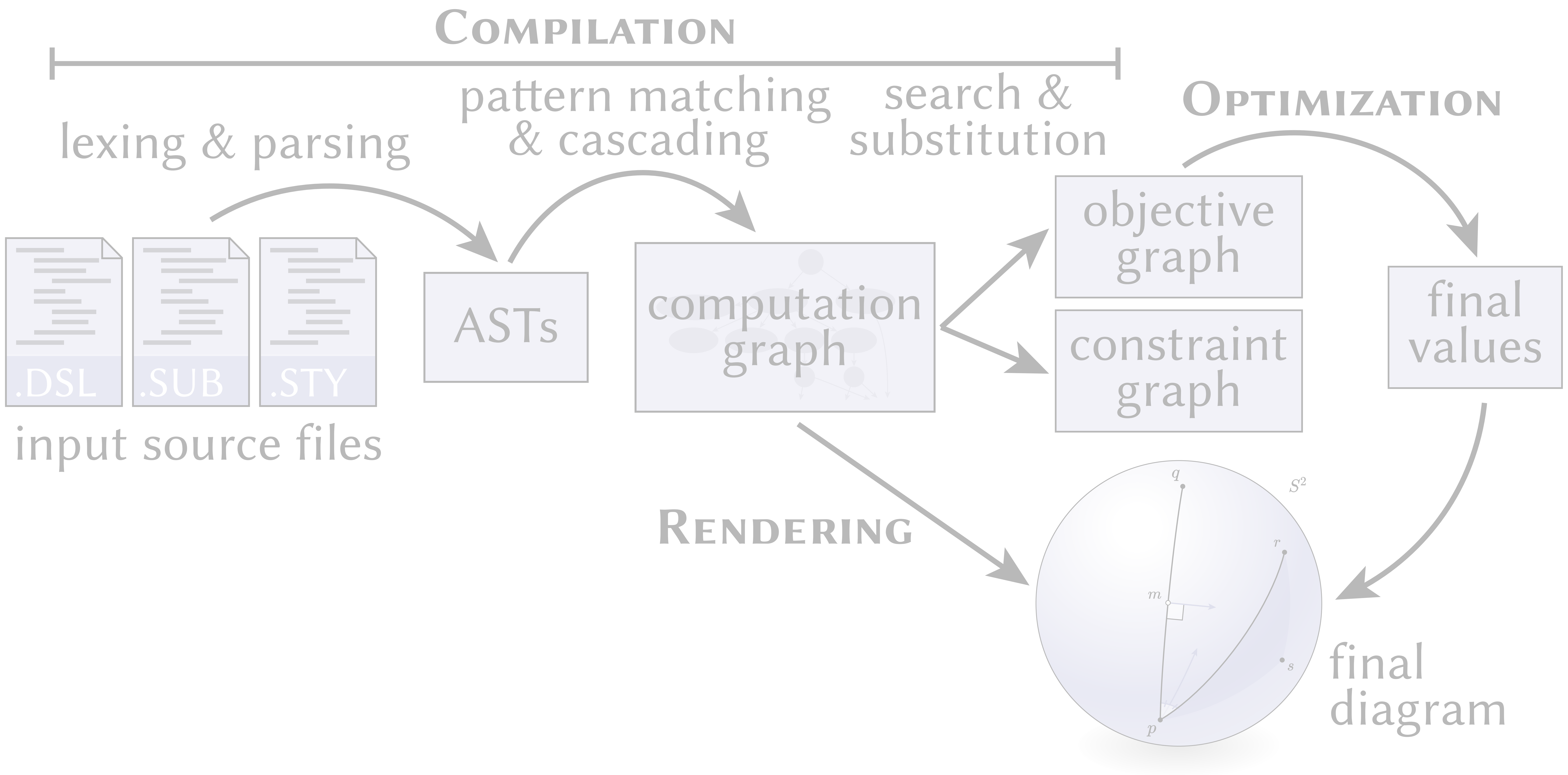


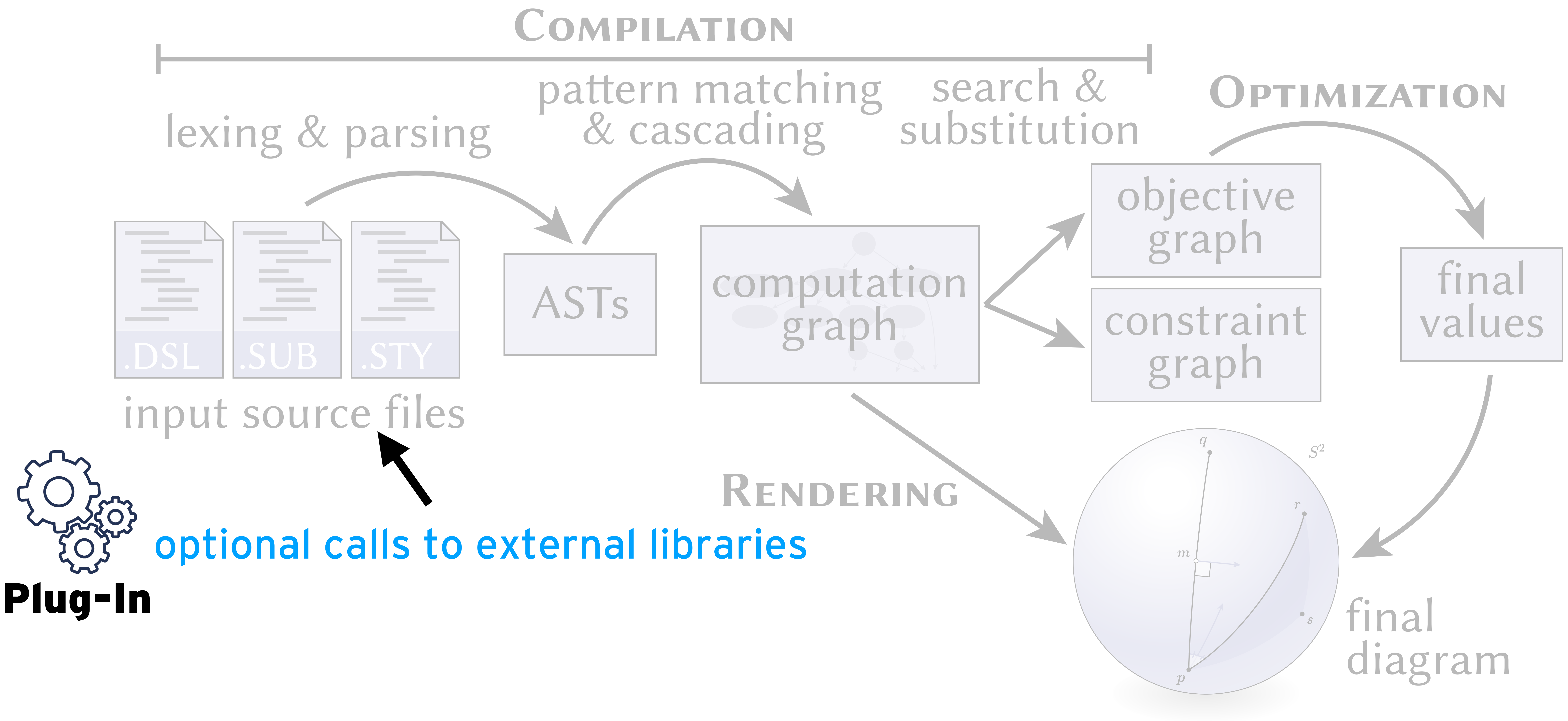
image credit: eleVR / Venn Diagram Museum

<svg xmlns="http://www.w3.org/2000/svg" version="1.2" width="100%" height="100%" viewBox="0 0 800 700"><g
path></g></g></g></svg></g></g><g transform="translate(0,0)"><circle cx="502.9341424
r="20.032626897036277" fill="#d83589" fill-opacity="0.5" stroke="#c7965b" stroke-opa
width="0"><title>B.shape</title><desc>Circle representing B.shape</desc></circle></g
events="bounding-box" transform="translate(497.92598574357845,319.8597619197204)"><
xmlns="http://www.w3.org/2000/svg" width="10.016313448518138" height="10.01631344851
759 683"><g stroke="none" fill="#ffffff" stroke-width="0" transform="matrix(1 0 0 -1
node="mi"><path data-c="42" d="M231 637Q204 637 199 638T194 649Q194 676 205 682Q206
544Q756 480 698 429T565 360L555 357Q619 348 660 311T702 219Q702 146 630 78T453 1Q44

Plug-ins enable integration with external code



Plug-ins enable integration with external code



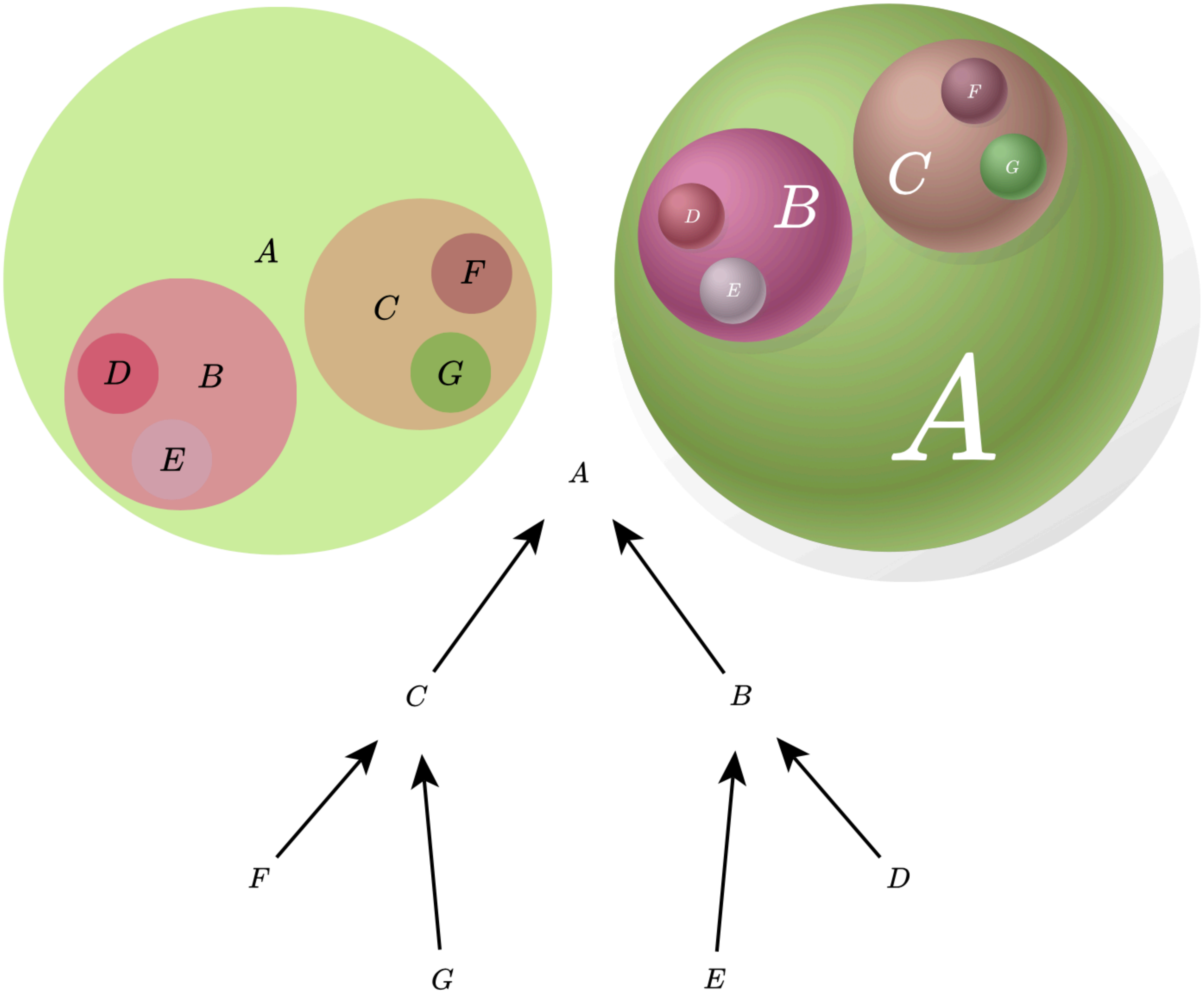
The system in action

Visualizing sets

Visualizing sets

changing style or visual representation

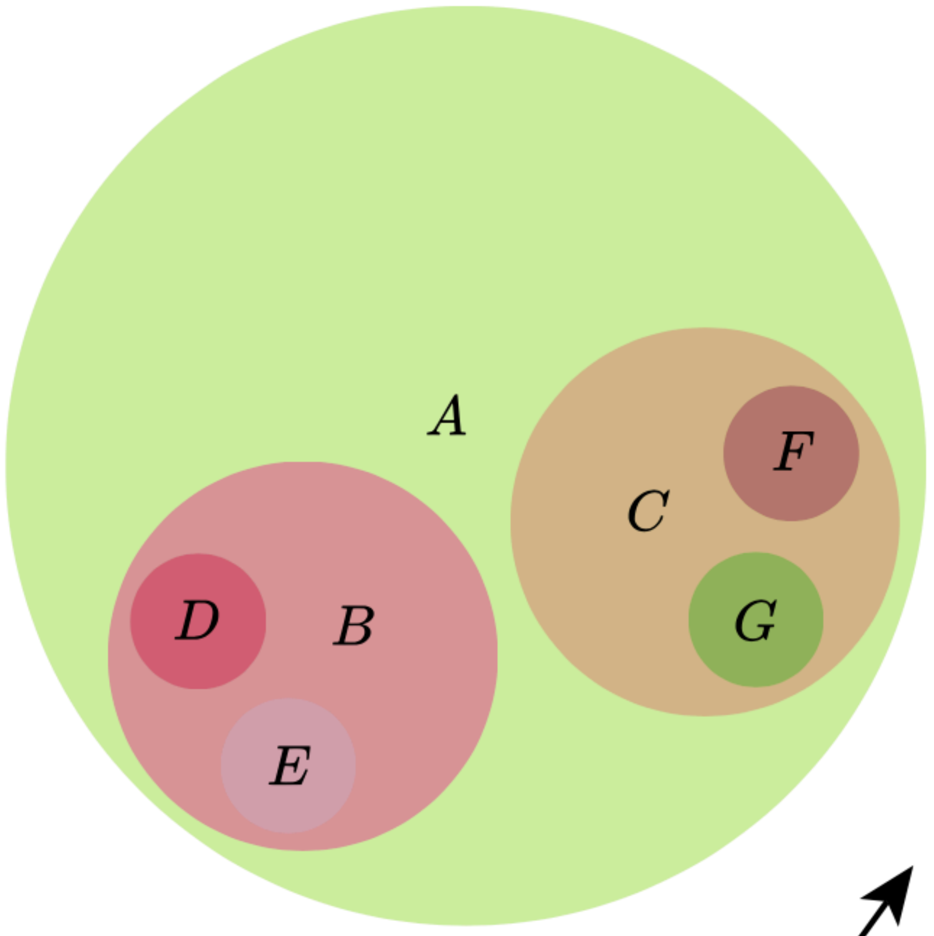
```
Set A, B, C, D, E, F, G        F ⊂ C        -- Sets.sub
B ⊂ A                          G ⊂ C
C ⊂ A                          E ∩ D = ∅
D ⊂ B                          F ∩ G = ∅
E ⊂ B                          B ∩ C = ∅
```

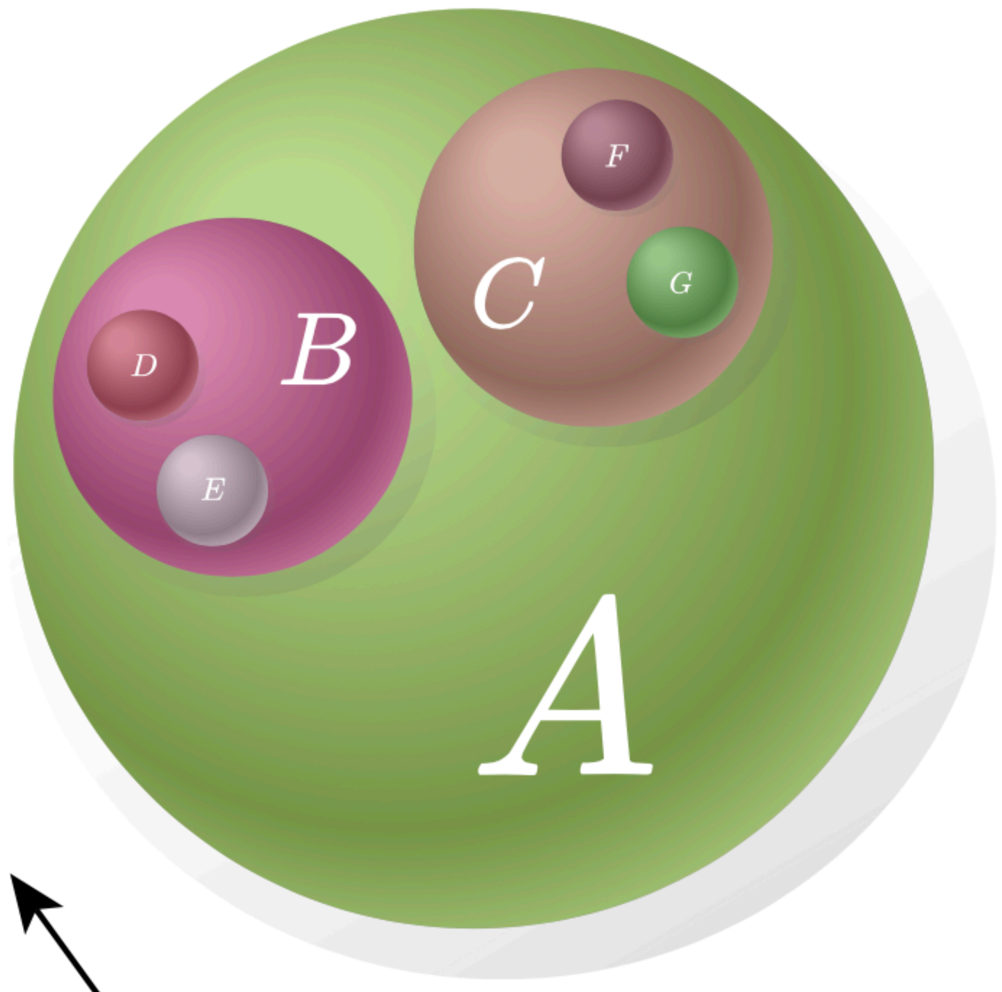


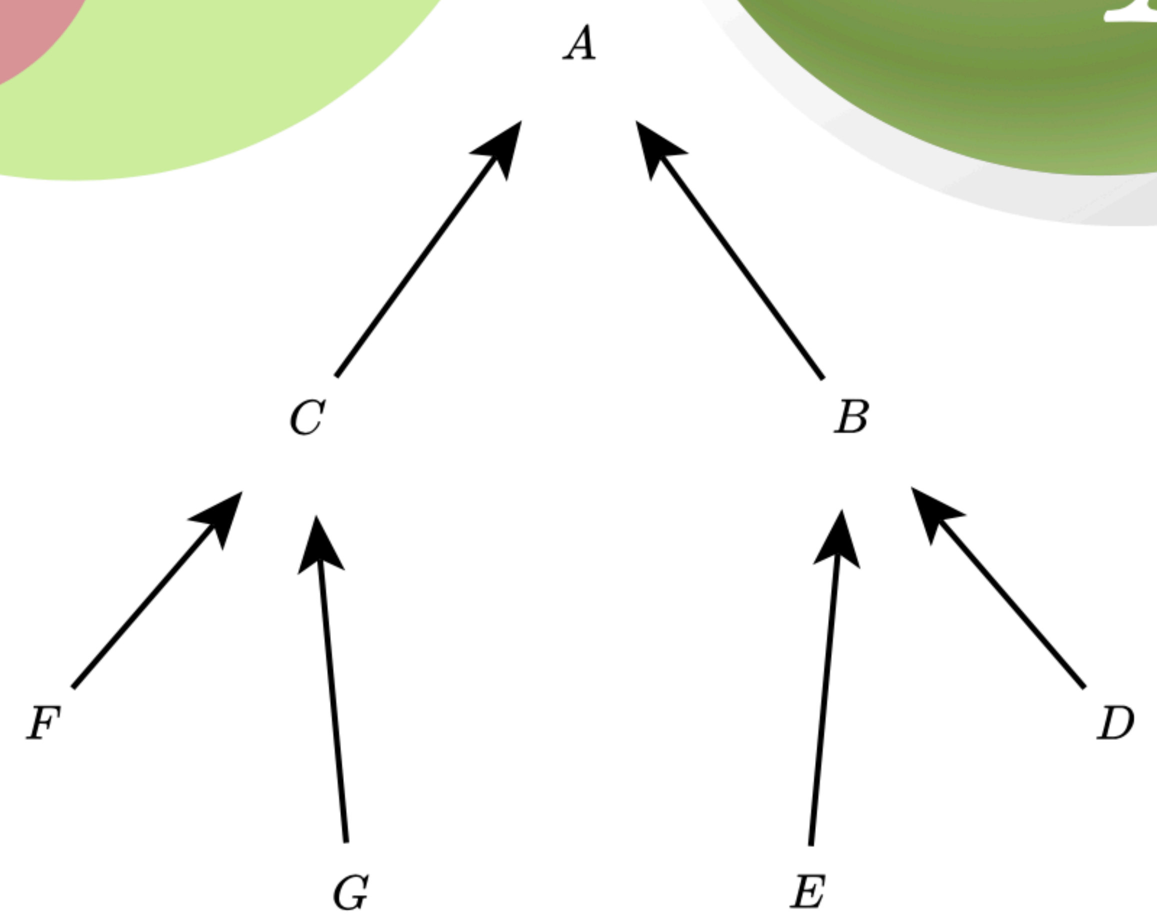
Visualizing sets

changing style or visual representation

```
Set A, B, C, D, E, F, G        F ⊂ C        -- Sets.sub
B ⊂ A                          G ⊂ C
C ⊂ A                          E ∩ D = ∅
D ⊂ B                          F ∩ G = ∅
E ⊂ B                          B ∩ C = ∅
```

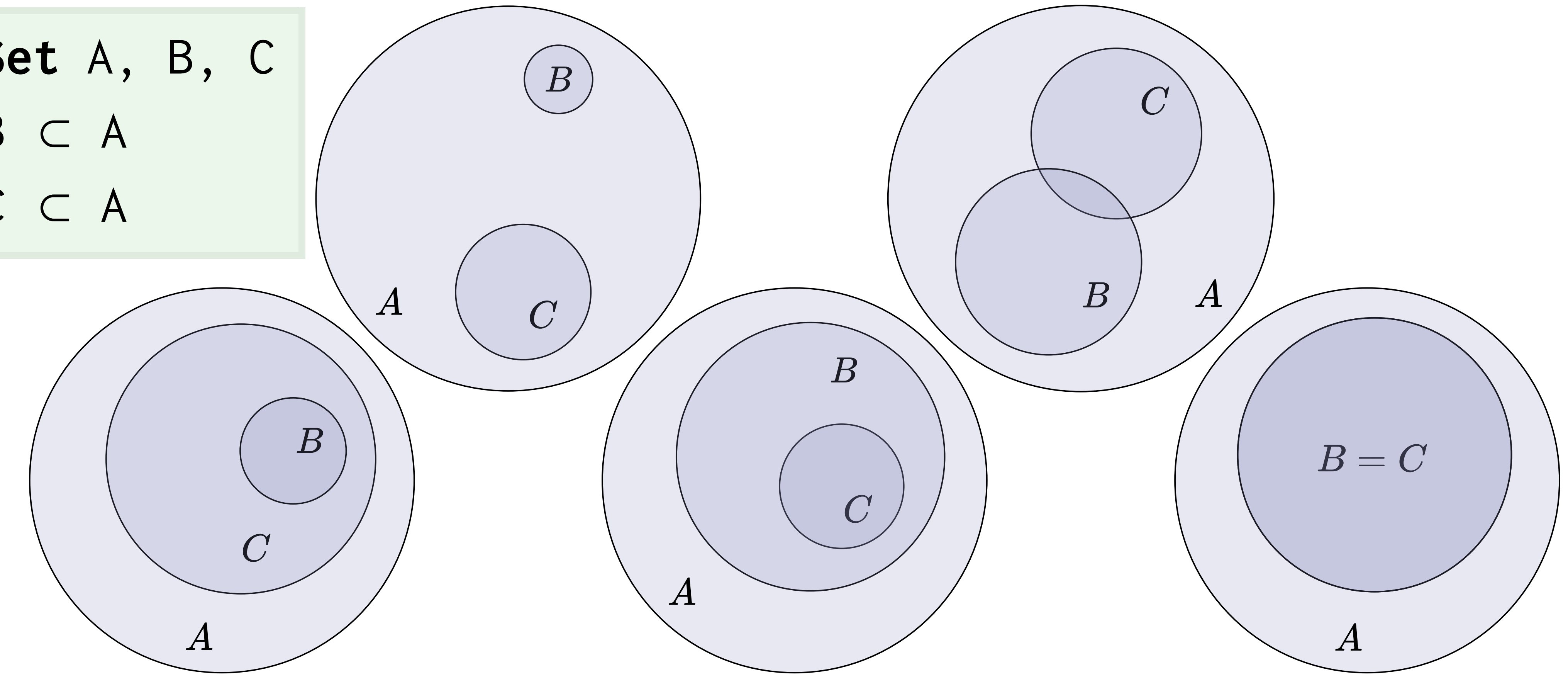






enumerating possibilities

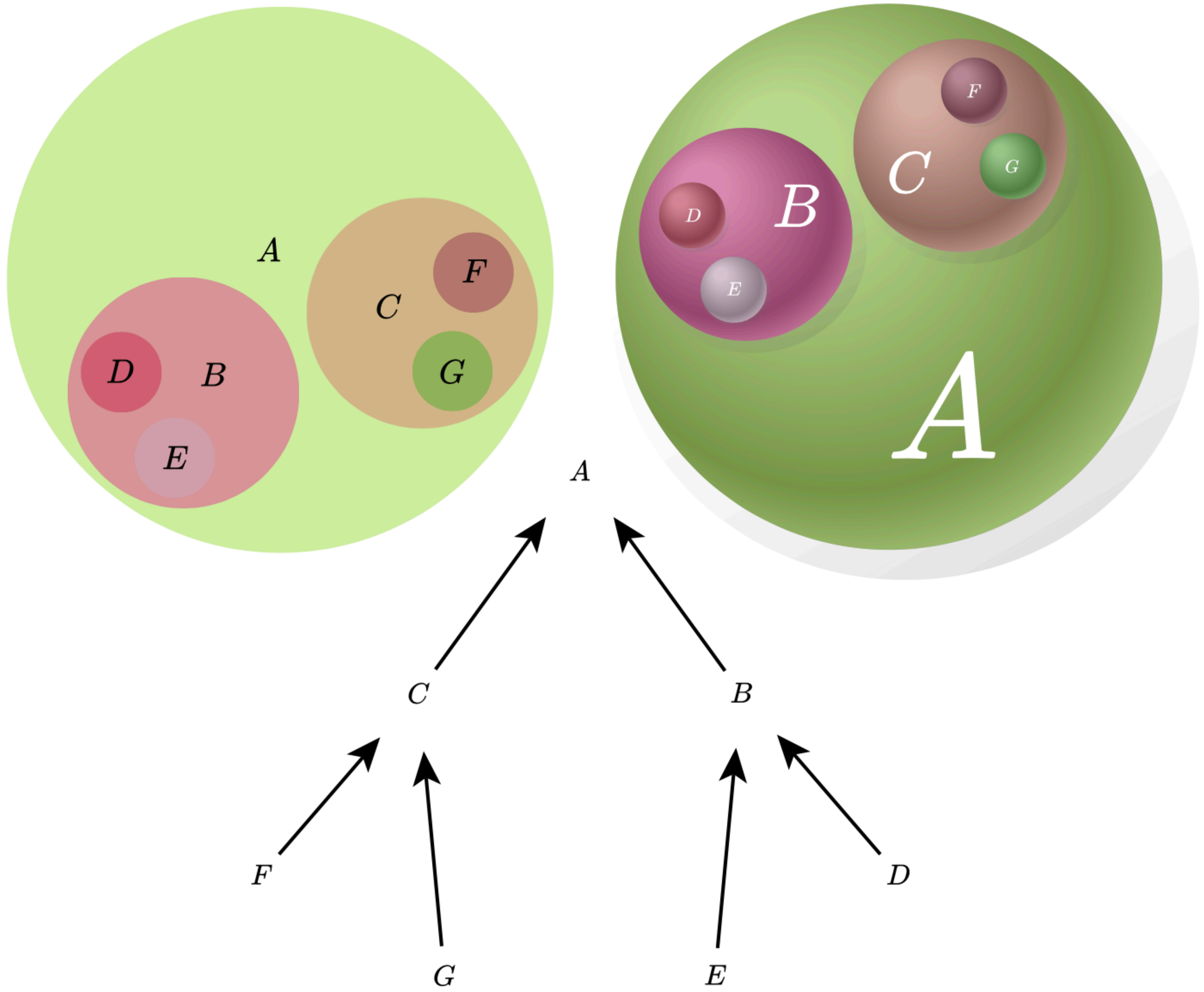
```
Set A, B, C
B ⊂ A
C ⊂ A
```



Visualizing sets

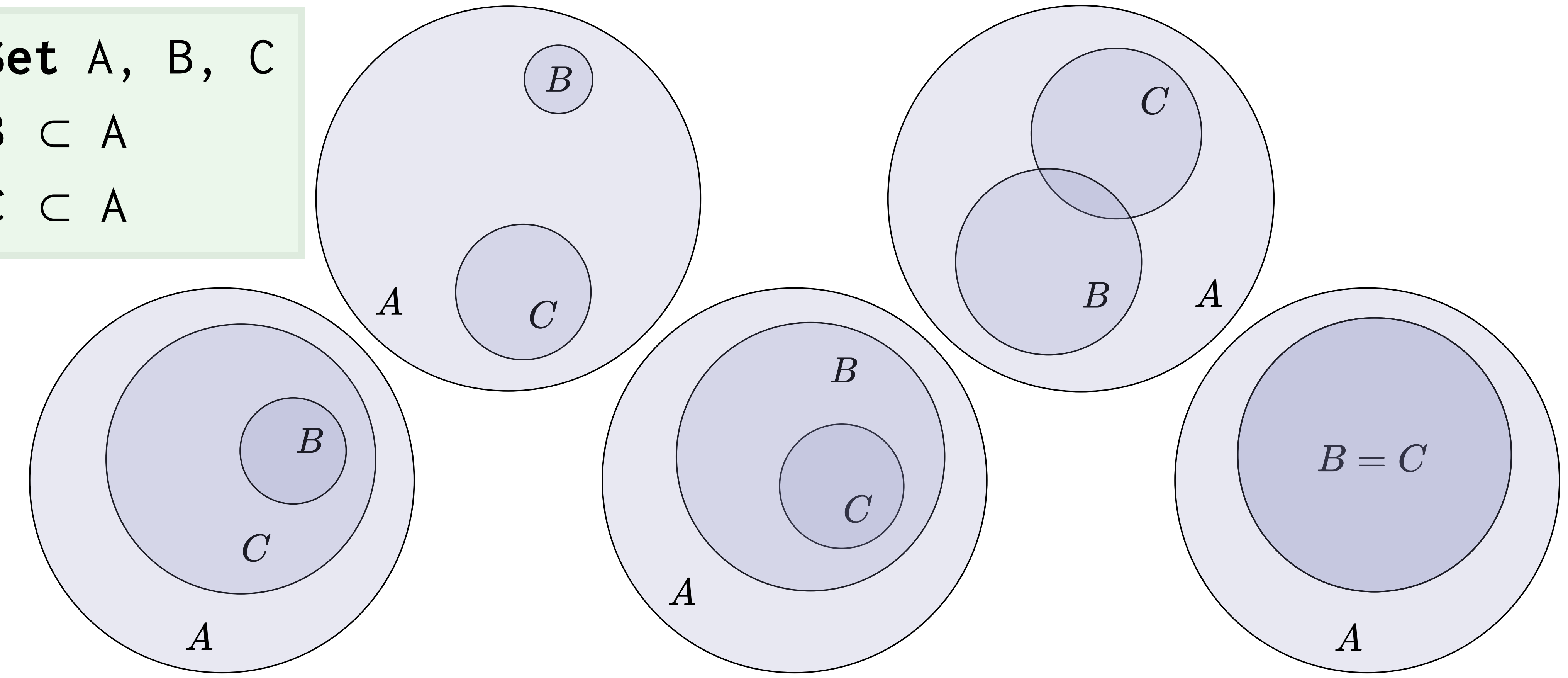
changing style or visual representation

```
Set A, B, C, D, E, F, G          F ⊂ C        -- Sets.sub
B ⊂ A                            G ⊂ C
C ⊂ A                            E ∩ D = ∅
D ⊂ B                            F ∩ G = ∅
E ⊂ B                            B ∩ C = ∅
```

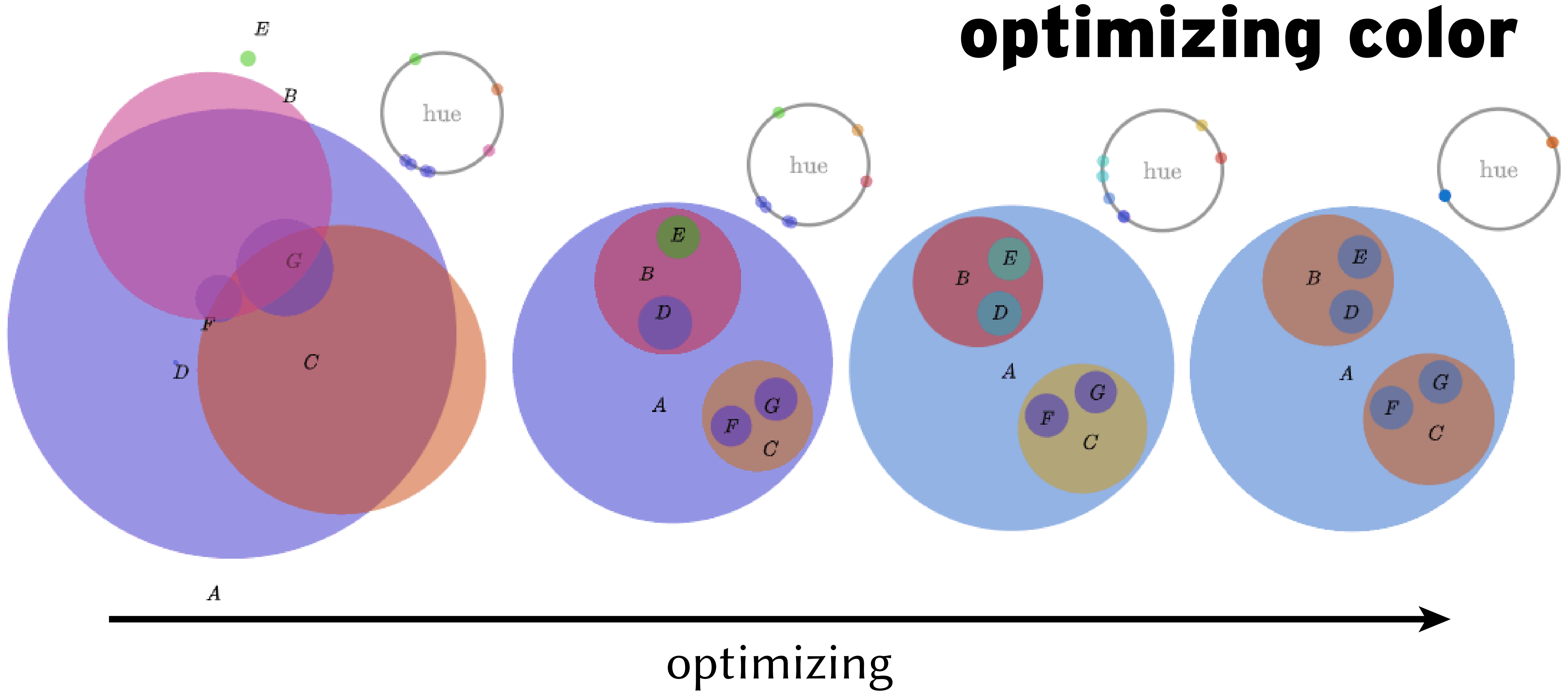


enumerating possibilities

```
Set A, B, C
B ⊂ A
C ⊂ A
```



optimizing color



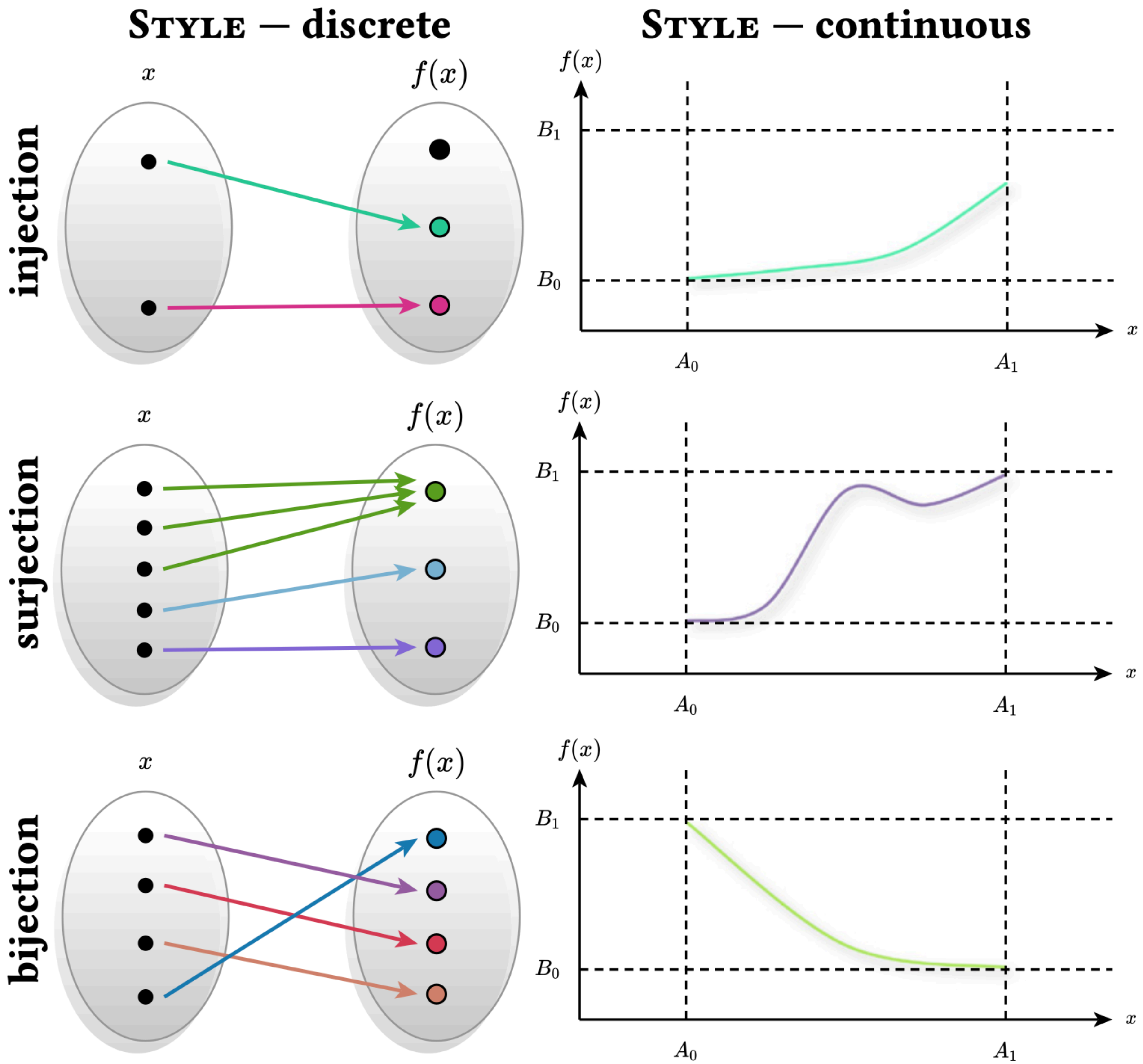
Visualizing functions

Visualizing functions

```
-- Injection.sub
Set A, B
f: A -> B
Injection(f)
Not(Surjection(f))
```

```
-- Surjection.sub
Set A, B
f: A -> B
Surjection(f)
Not(Injection(f))
```

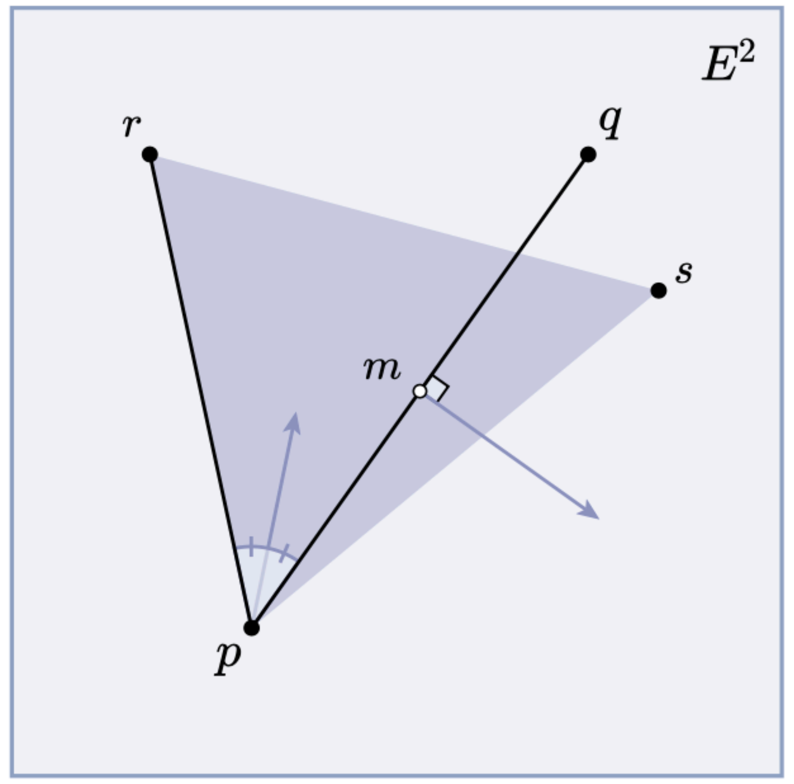
```
-- Bijection.sub
Set A, B
f: A -> B
Surjection(f)
Injection(f)
```

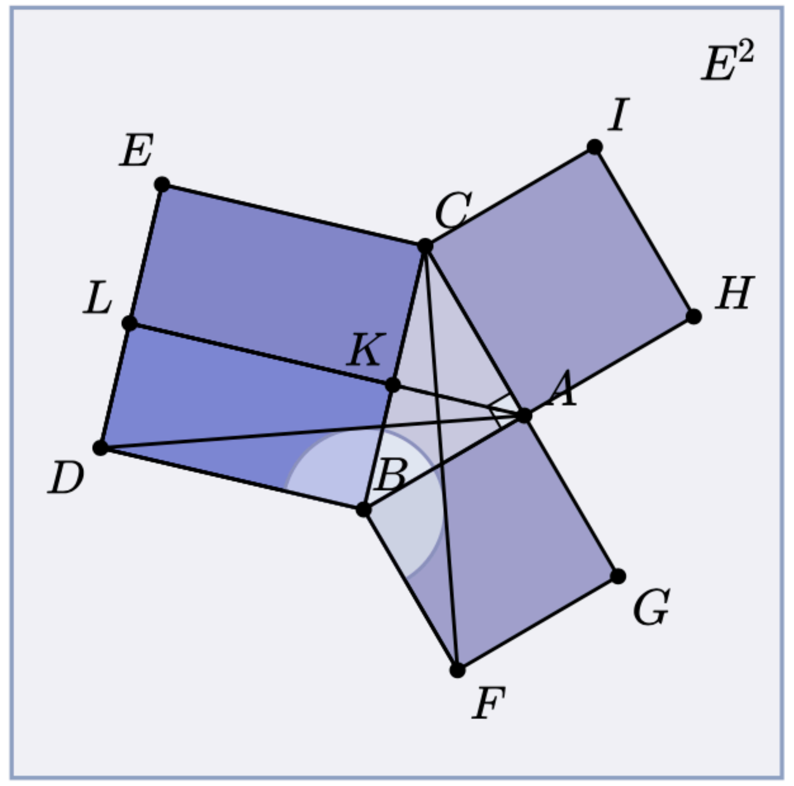


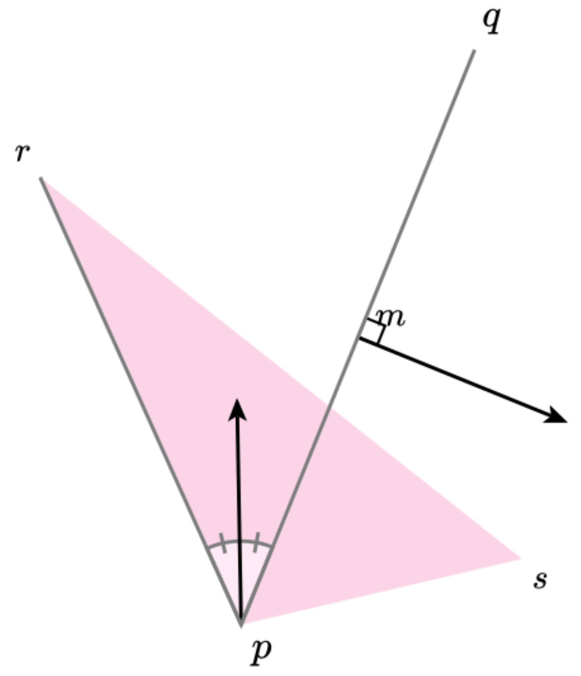
Visualizing Euclidean geometry

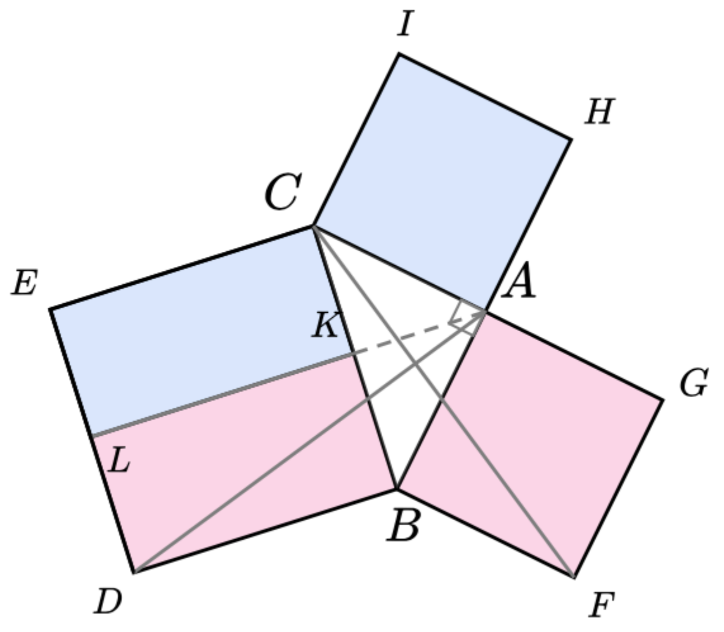
Visualizing Euclidean geometry

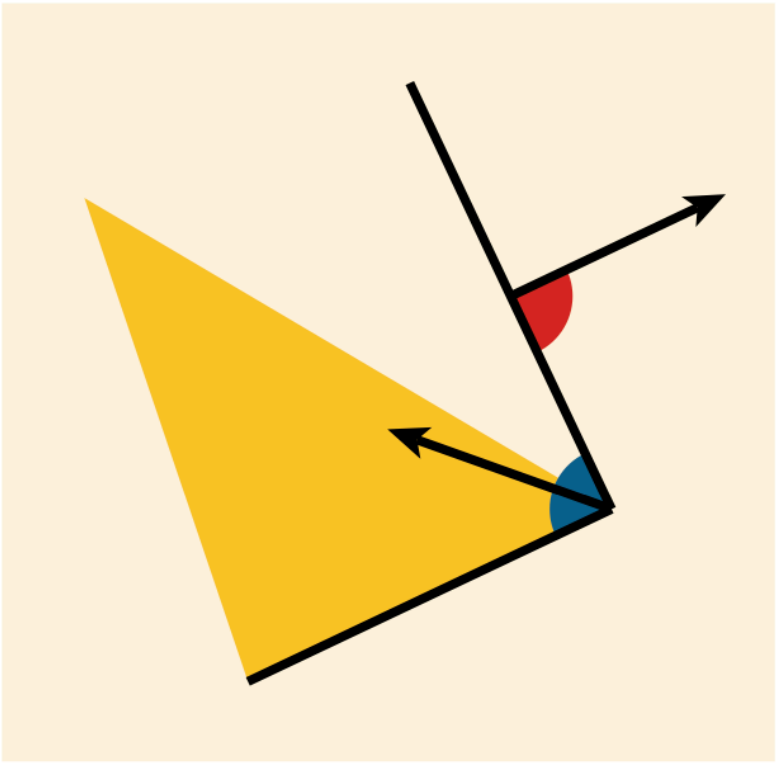
changing the styling

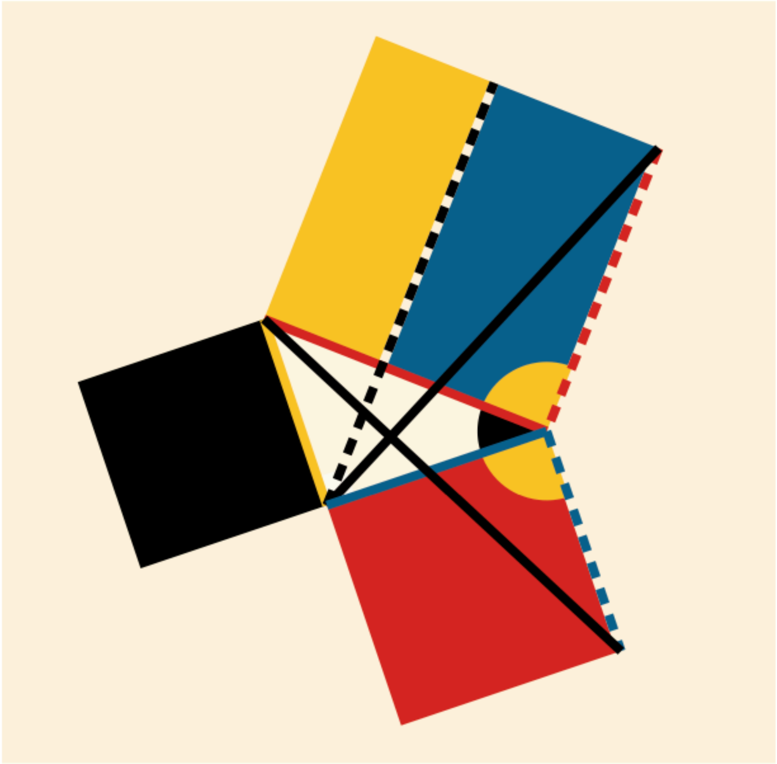






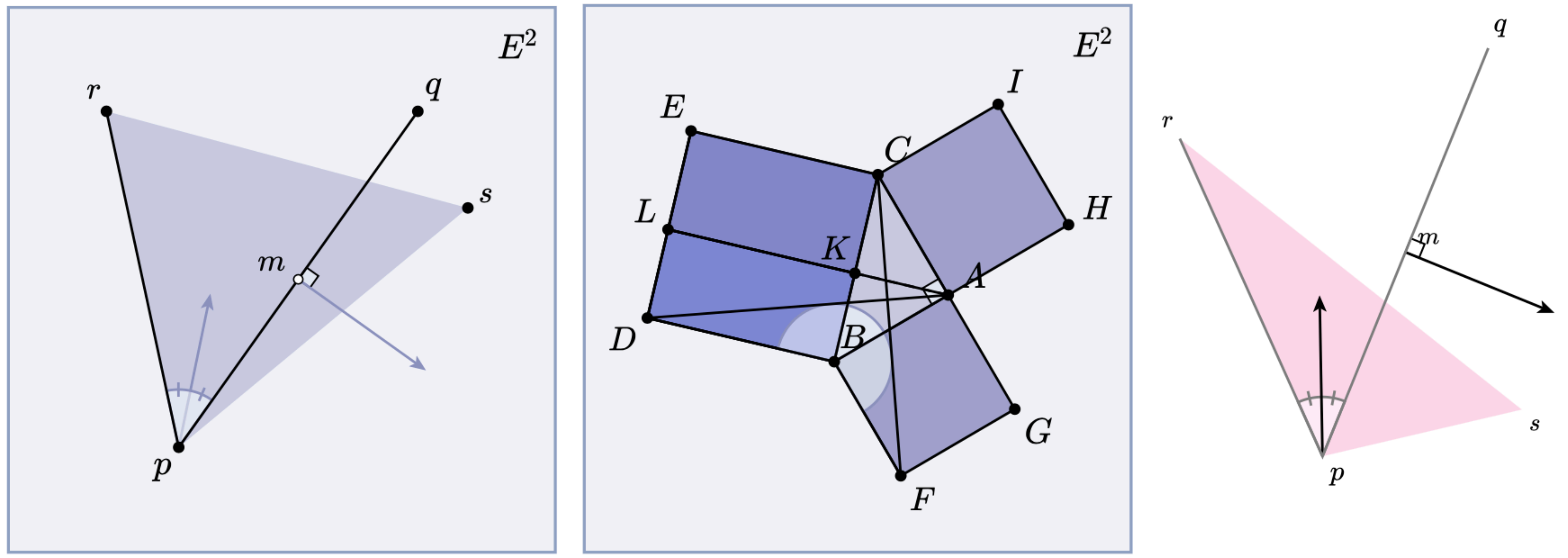




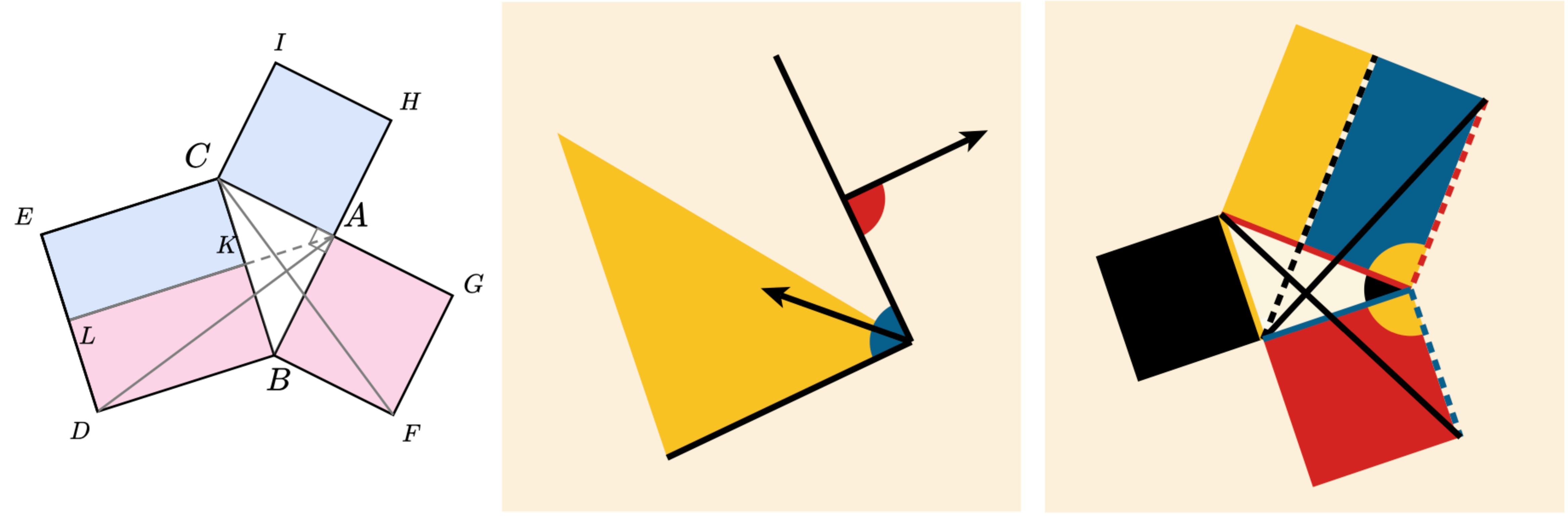


```
Point A, B, C                    -- PythagoreanTheorem.sub
-- define a right triangle       -- split hypotenuse area
Triangle ABC := {A,B,C}          Segment AK := Altitude(ABC,θ)
Angle θ := ∠(C,A,B)              Point K := Endpoint(AK)
Right(θ)                         Segment DE := {D,E}
-- square each side              Point L
Point D, E, F, G, H, I           On(L, DE)
Square CBDE := [C,B,D,E]         Segment KL := {K,L}
Disjoint(CBDE, ABC)              Perpendicular(KL, DE)
Square BAGF := [B,A,G,F]         Rectangle BDLK := {B,D,L,K}
Disjoint(BAGF, ABC)              Rectangle CKLE := {C,K,L,E}
Square ACIH := [A,C,I,H]         -- (plus additional objects
Disjoint(ACIH, ABC)              -- from Byrne's diagram)
```

Visualizing Euclidean geometry

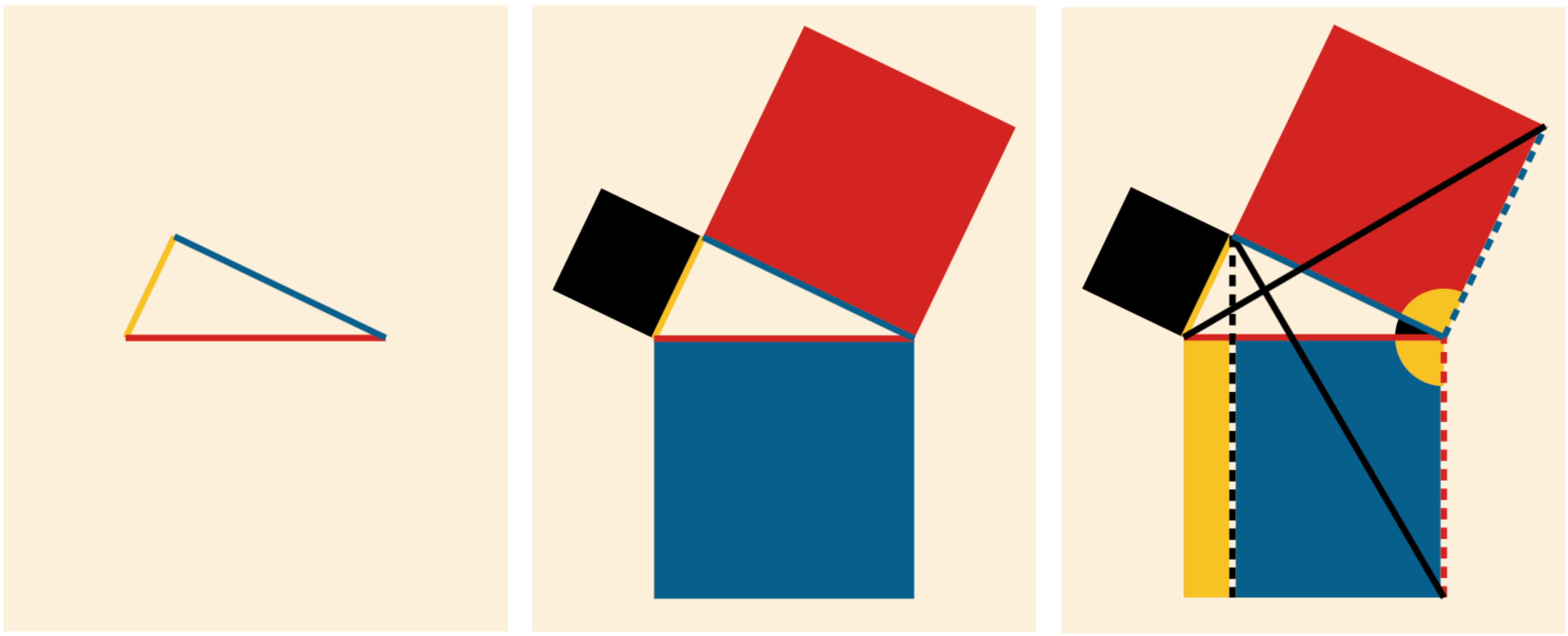


changing the styling




```
Point A, B, C
-- define a right triangle
Triangle ABC := {A,B,C}
Angle θ := ∠(C,A,B)
Right(θ)
-- square each side
Point D, E, F, G, H, I
Square CBDE := [C,B,D,E]
Disjoint(CBDE, ABC)
Square BAGF := [B,A,G,F]
Disjoint(BAGF, ABC)
Square ACIH := [A,C,I,H]
Disjoint(ACIH, ABC)
-- PythagoreanTheorem.sub
-- split hypotenuse area
Segment AK := Altitude(ABC,θ)
Point K := Endpoint(AK)
Segment DE := {D,E}
Point L
On(L, DE)
Segment KL := {K,L}
Perpendicular(KL, DE)
Rectangle BDLK := {B,D,L,K}
Rectangle CKLE := {C,K,L,E}
-- (plus additional objects
-- from Byrne's diagram)
```

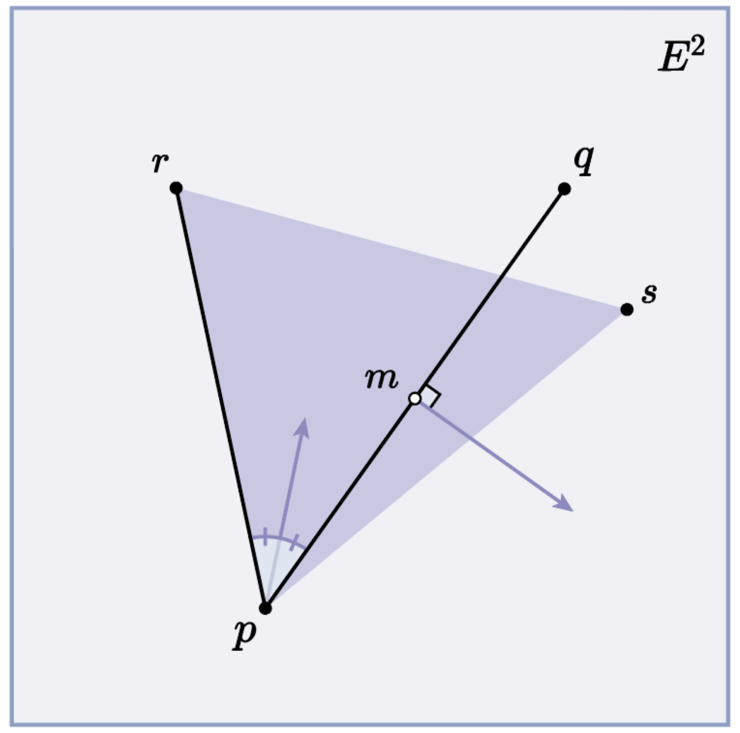
sequences of diagrams



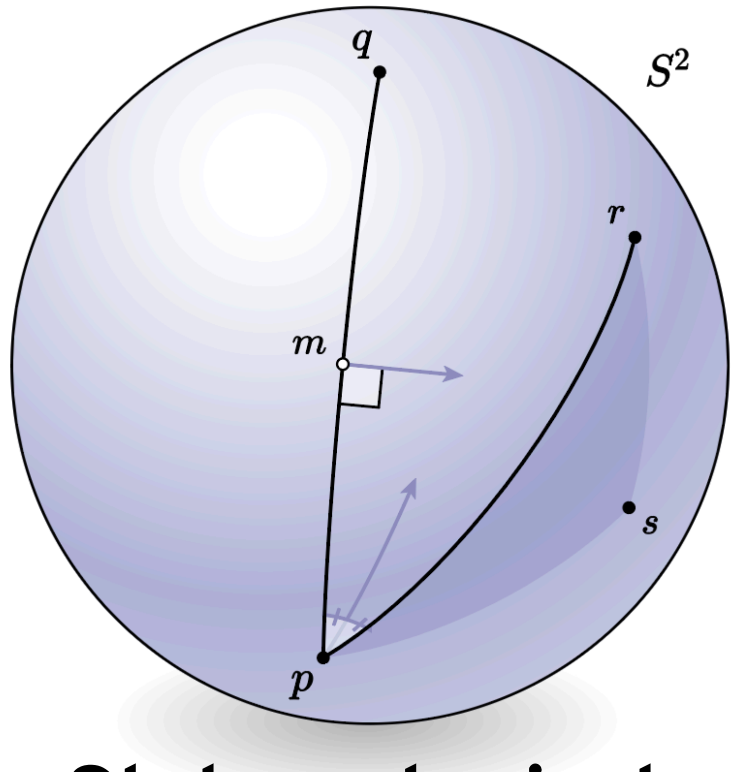
Visualizing neutral geometry

Visualizing neutral geometry

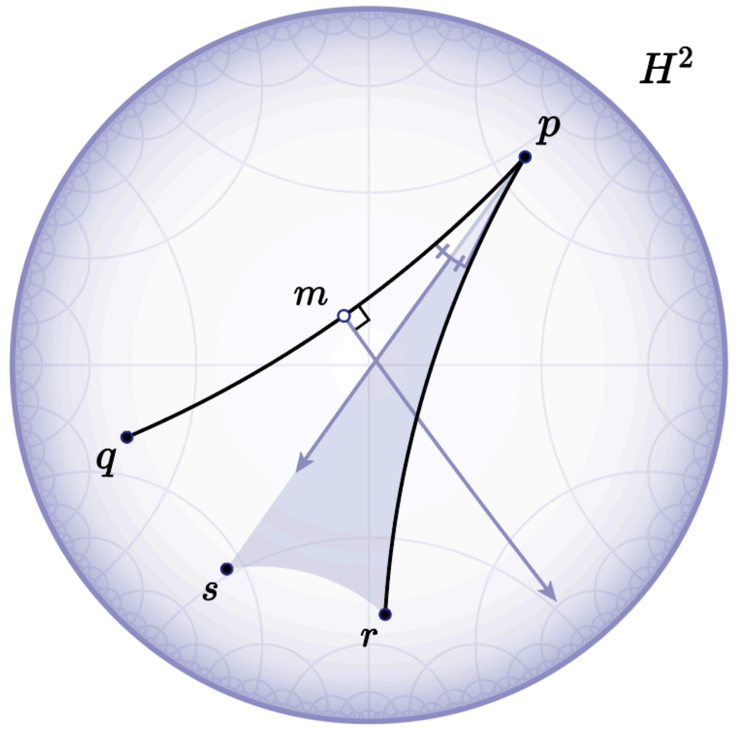
changing the geometric interpretation



Style–Euclidean



Style–spherical

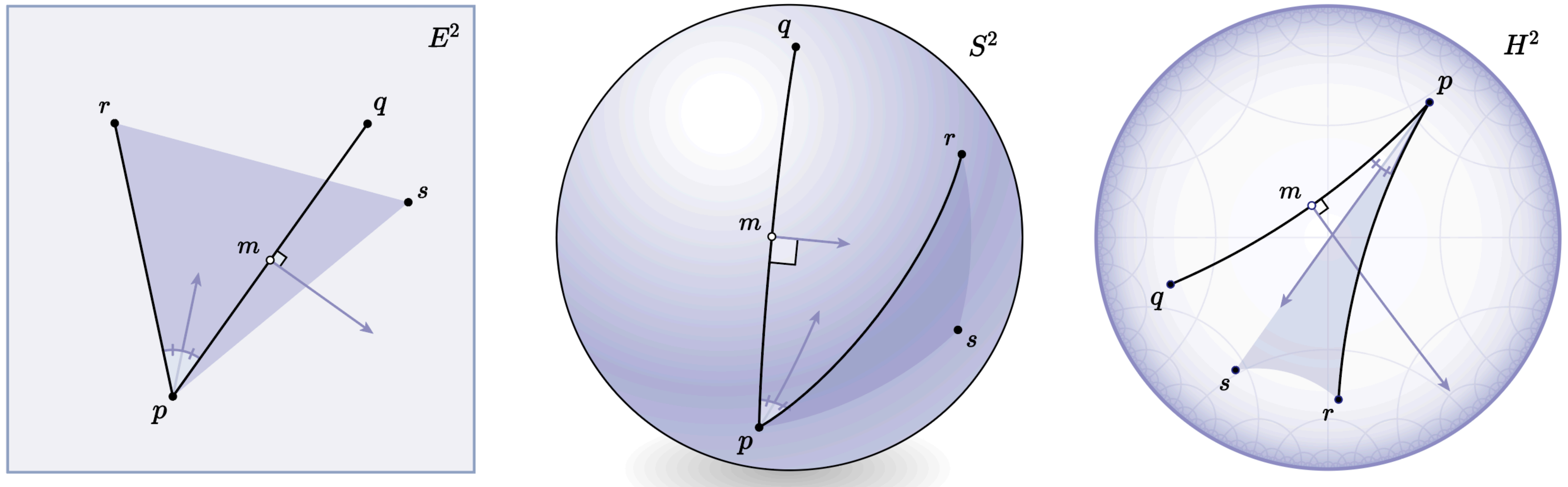


Style–hyperbolic

```
Point p, q, r, s
Segment a := p, q
Segment b := p, r
Point m := Midpoint(a)
Angle theta := ∠(q, p, r)
Triangle t := p, r, s
Ray w := Bisector(theta)
Ray h := PerpendicularBisector(a)
```

Visualizing neutral geometry

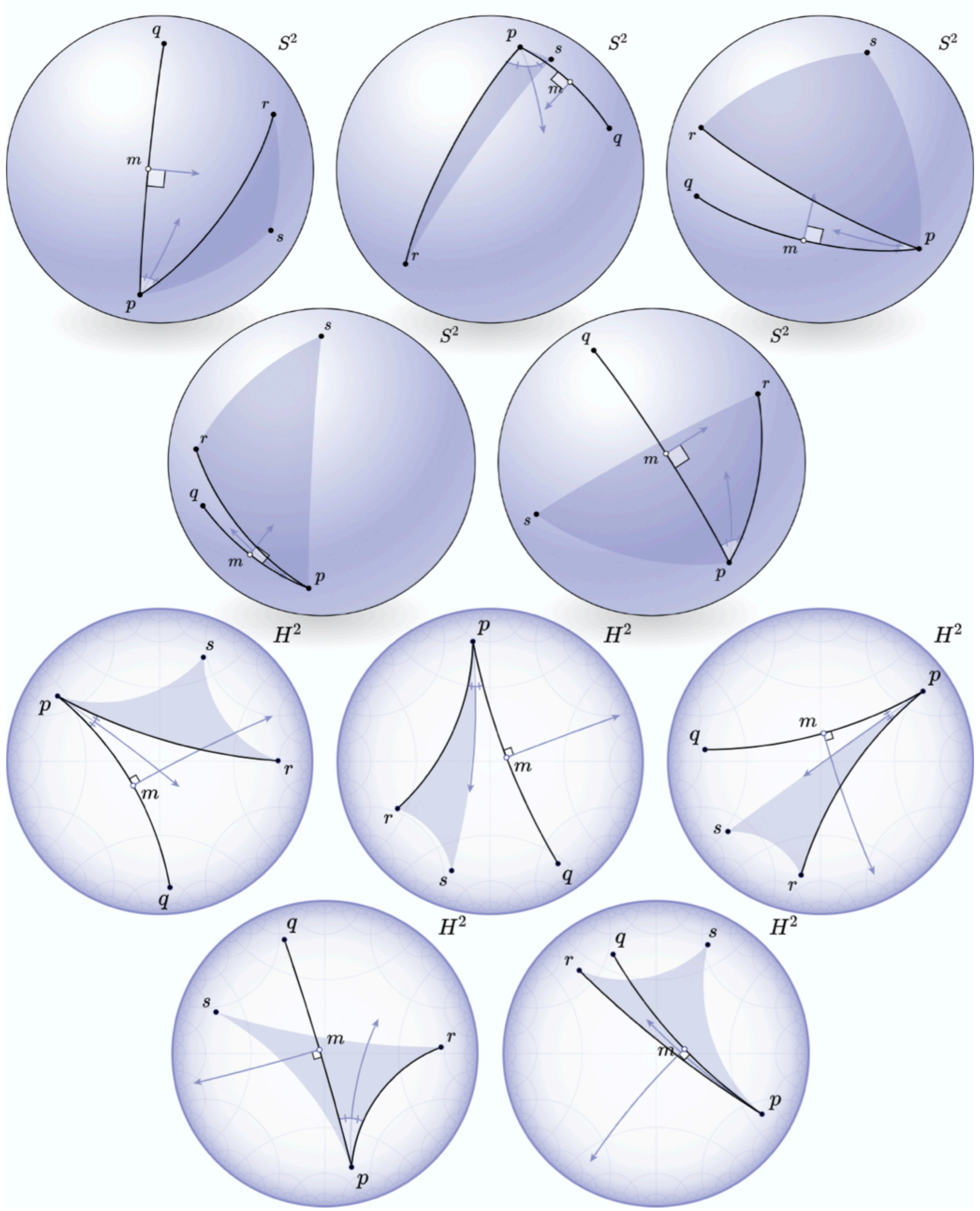
changing the geometric interpretation



Style–Euclidean Style–spherical Style–hyperbolic

```
Point p, q, r, s
Segment a := p, q
Segment b := p, r
Point m := Midpoint(a)
Angle theta := ∠(q, p, r)
Triangle t := p, r, s
Ray w := Bisector(theta)
Ray h := PerpendicularBisector(a)
```

generating a gallery of alternatives

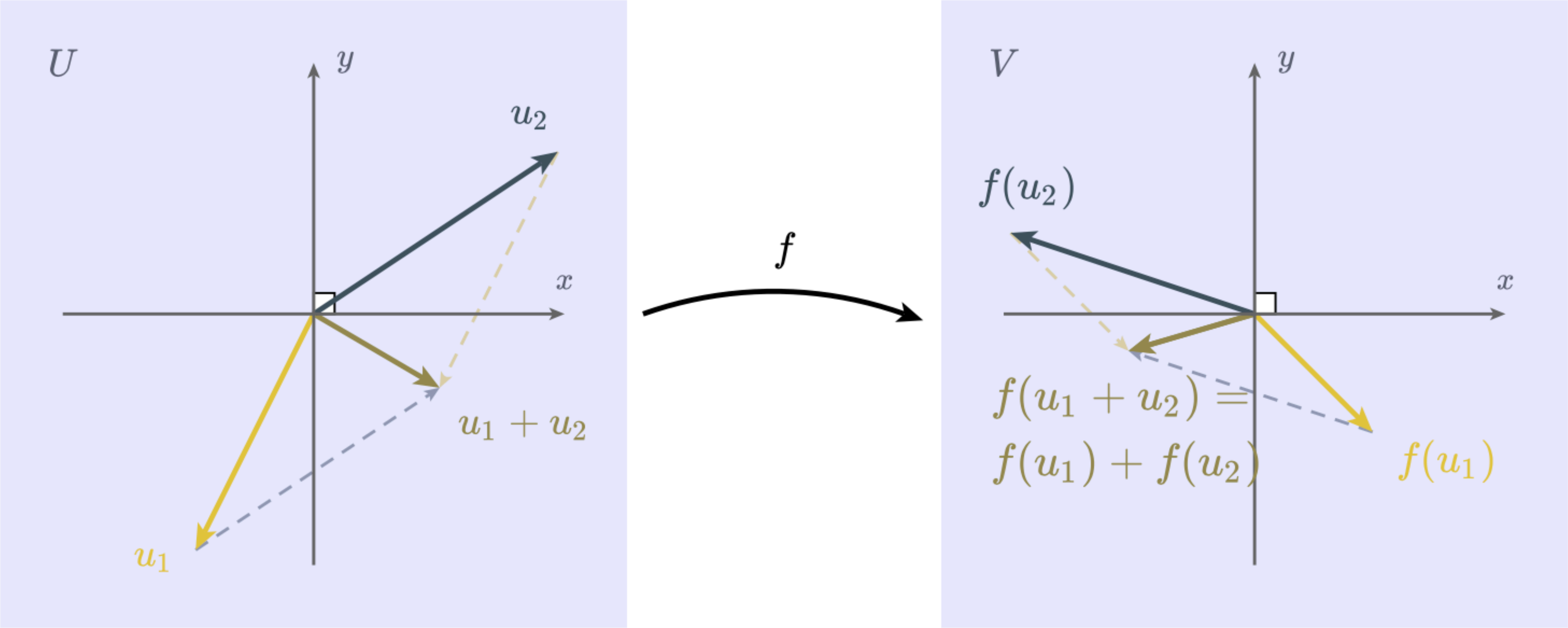


Visualizing vectors

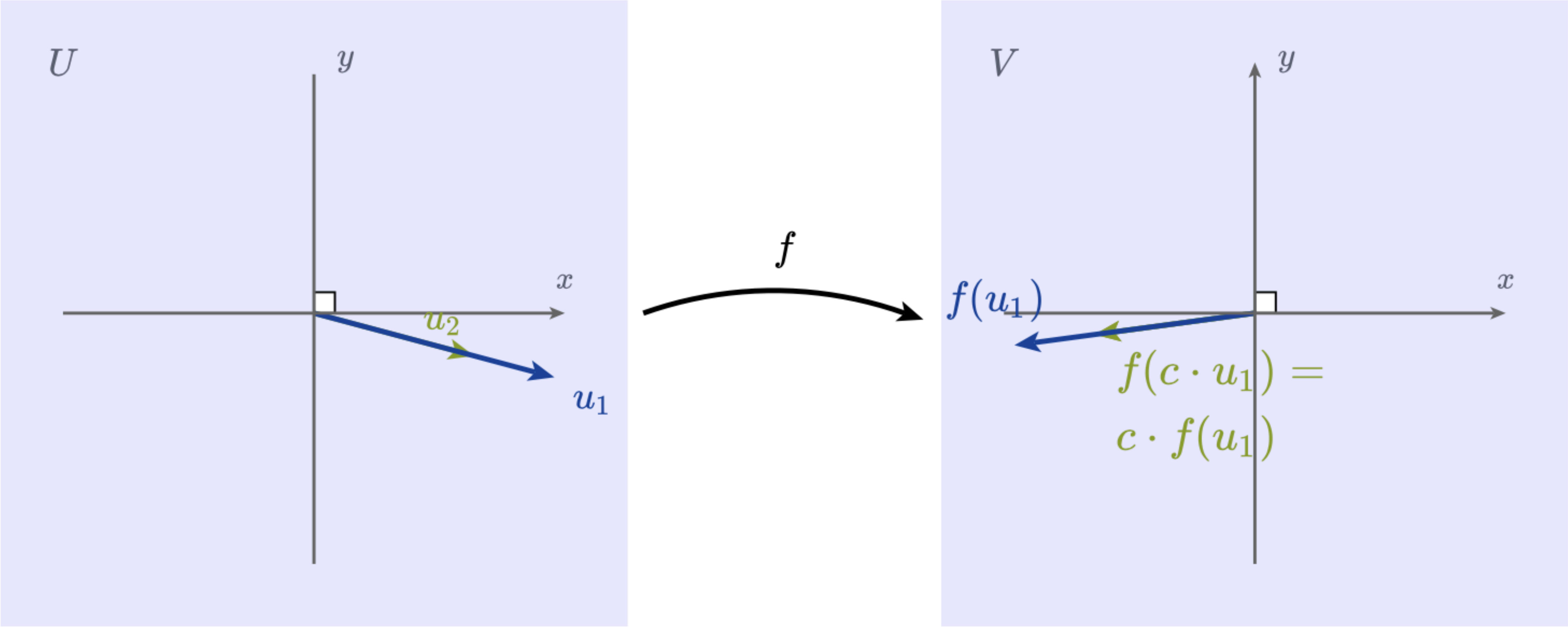
Visualizing vectors

composition of math translates to composition of visualization

```
VectorSpace U, V
LinearMap f : U → V
Vector u1, u2 ∈ U
Vector v1, v2, v3 ∈ V
Scalar c
u2 := c * u1
v1 := f(u1)
v2 := f(u2)
v3 := c * v1
```



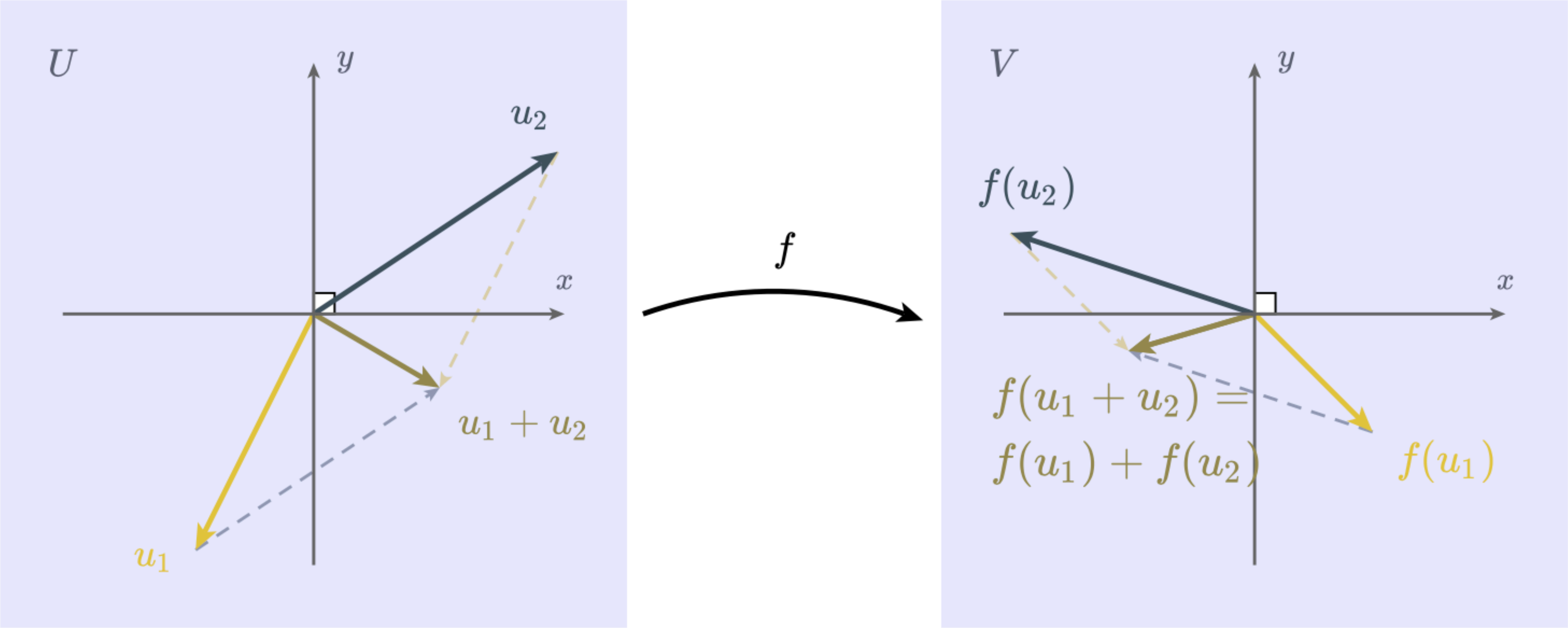

```
VectorSpace U, V
LinearMap f : U → V
Vector u1, u2, u3 ∈ U
Vector v1, v2, v3, v4 ∈ V
u3 := u1 + u2
v1 := f(u1)
v2 := f(u2)
v3 := f(u3)
v4 := v1 + v2
```



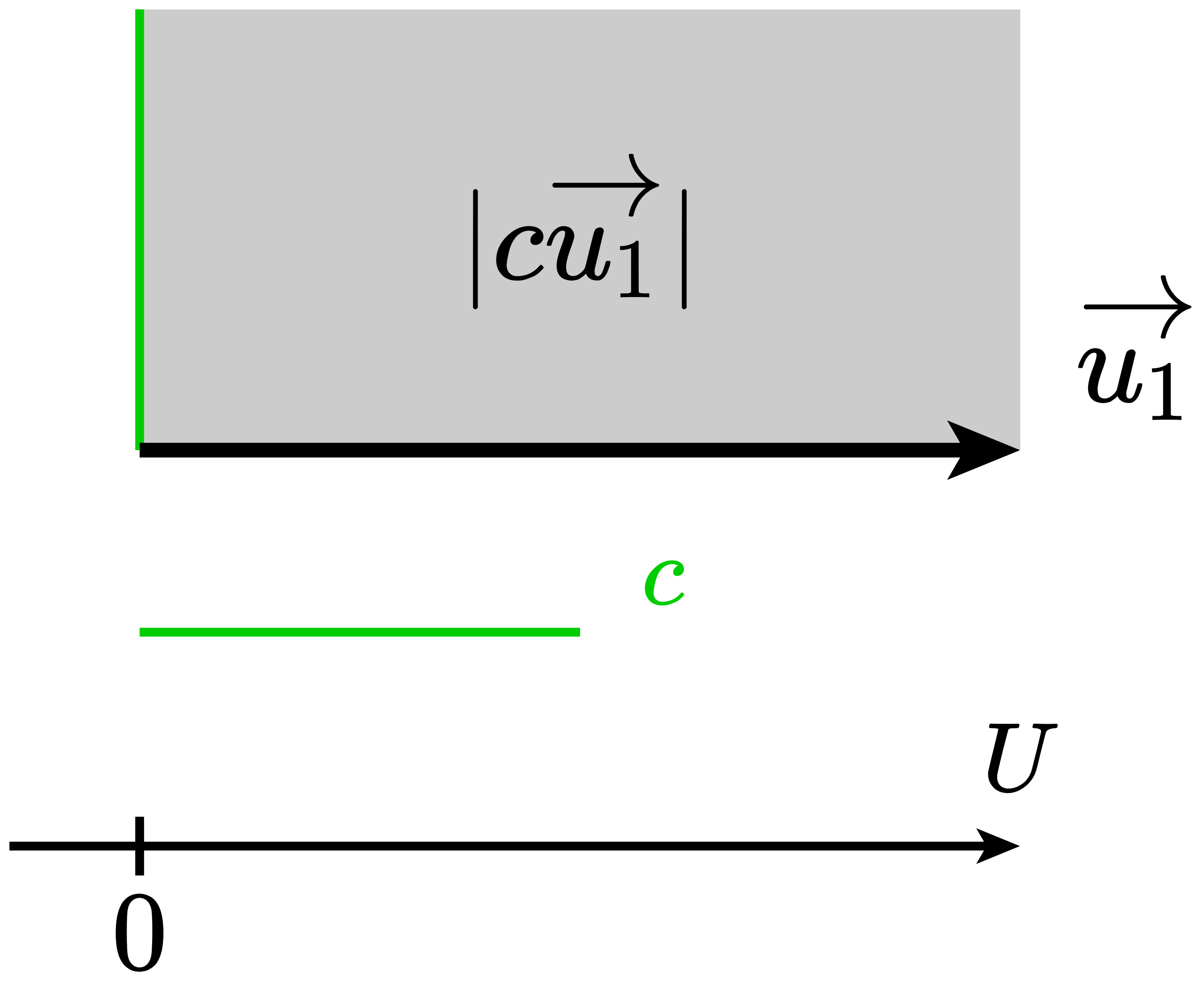
Visualizing vectors

composition of math translates to composition of visualization

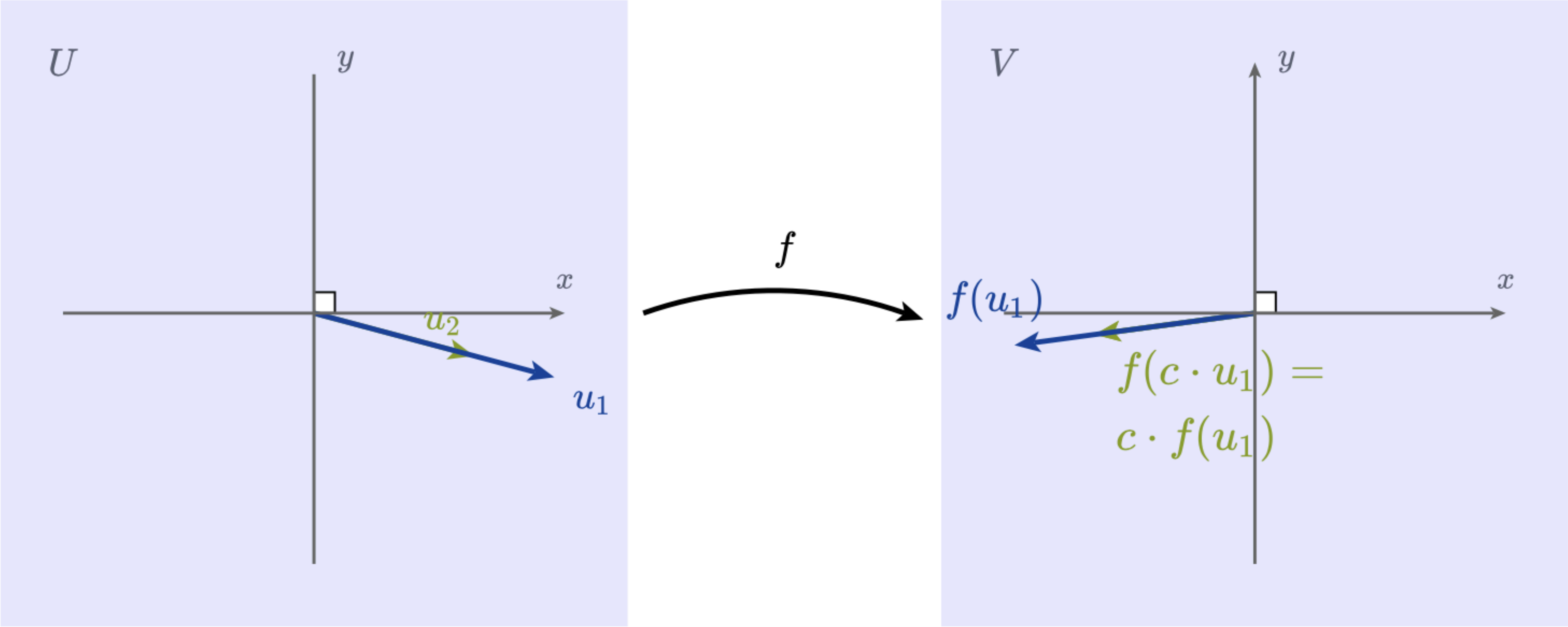
```
VectorSpace U, V
LinearMap f : U → V
Vector u1, u2 ∈ U
Vector v1, v2, v3 ∈ V
Scalar c
u2 := c * u1
v1 := f(u1)
v2 := f(u2)
v3 := c * v1
```



adapting the representation to the dimension




```
VectorSpace U, V
LinearMap f : U → V
Vector u1, u2, u3 ∈ U
Vector v1, v2, v3, v4 ∈ V
u3 := u1 + u2
v1 := f(u1)
v2 := f(u2)
v3 := f(u3)
v4 := v1 + v2
```



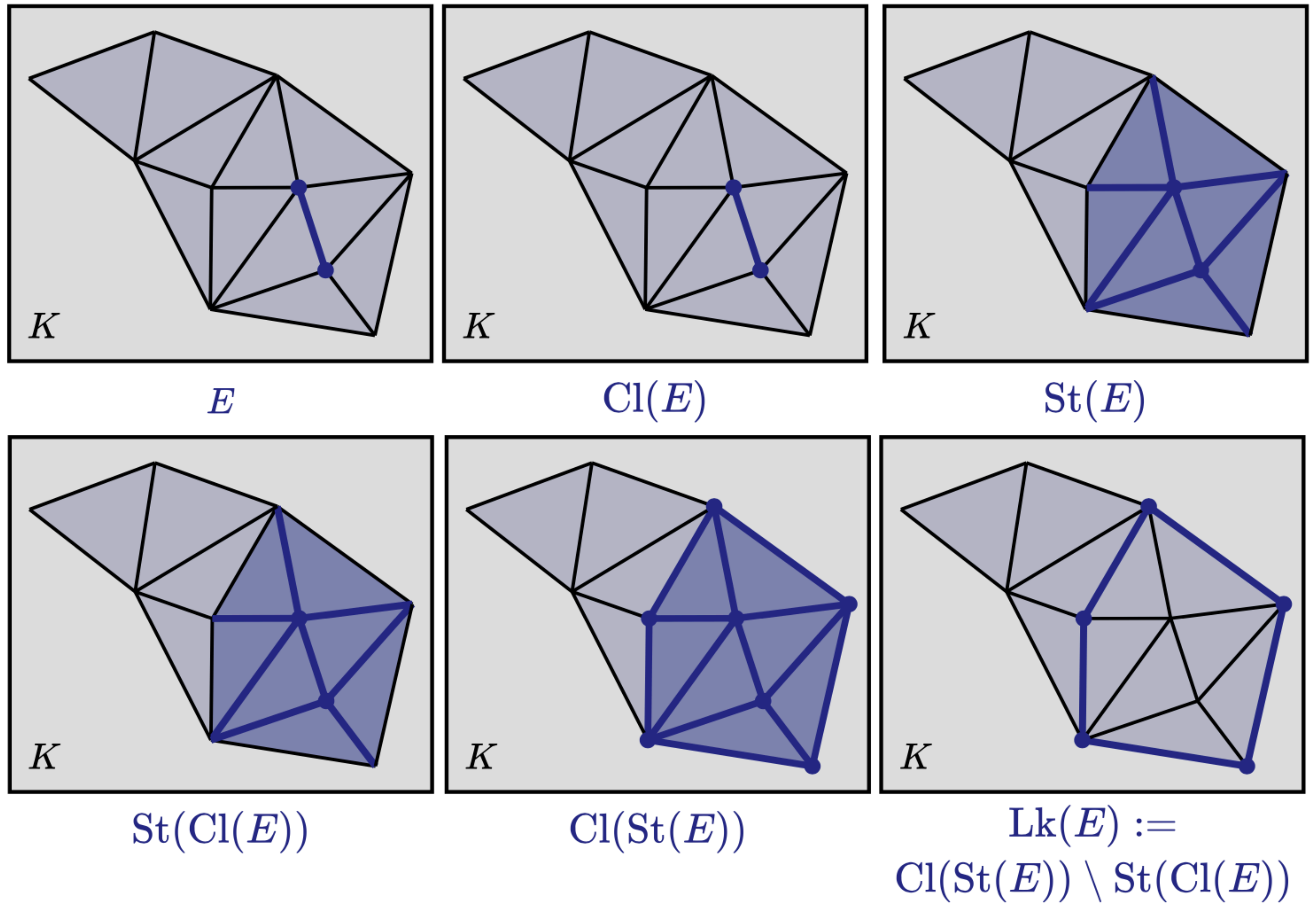
Visualizing meshes

Visualizing meshes

illustrating steps of a complex process

```
SimplicialComplex K

Edge e ∈ K
Subcomplex E ⊆ K
E := Closure(e)

SimplicialSet StE ⊆ K
StE := Star(E)
```

```
Subcomplex ClStE ⊆ K
ClStE := Closure(StE)
Subcomplex ClE ⊆ K
ClE := Closure(E)
SimplicialSet StClE ⊆ K
StClE := Star(ClE)
SimplicialSet LkE ⊆ K
LkE := SetMinus(ClStE, StClE)
```

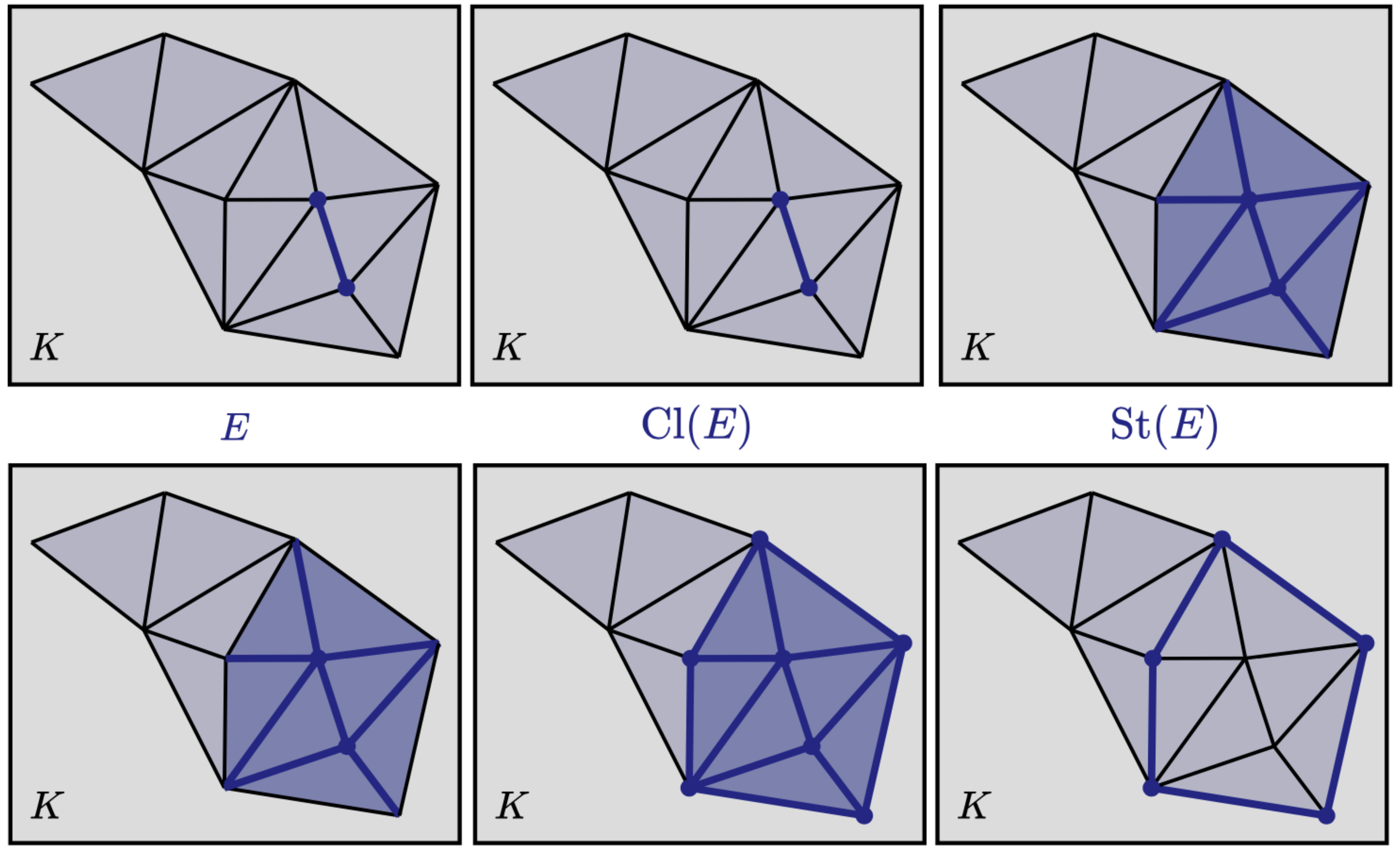


Visualizing meshes

illustrating steps of a complex process

```
SimplicialComplex K

Edge e ∈ K
Subcomplex E ⊆ K
E := Closure(e)

SimplicialSet StE ⊆ K
StE := Star(E)
```

```
Subcomplex ClStE ⊆ K
ClStE := Closure(StE)
Subcomplex ClE ⊆ K
ClE := Closure(E)
SimplicialSet StClE ⊆ K
StClE := Star(ClE)
SimplicialSet LkE ⊆ K
LkE := SetMinus(ClStE, StClE)
```



E $\quad$ $\mathrm{Cl}(E)$ $\quad$ $\mathrm{St}(E)$

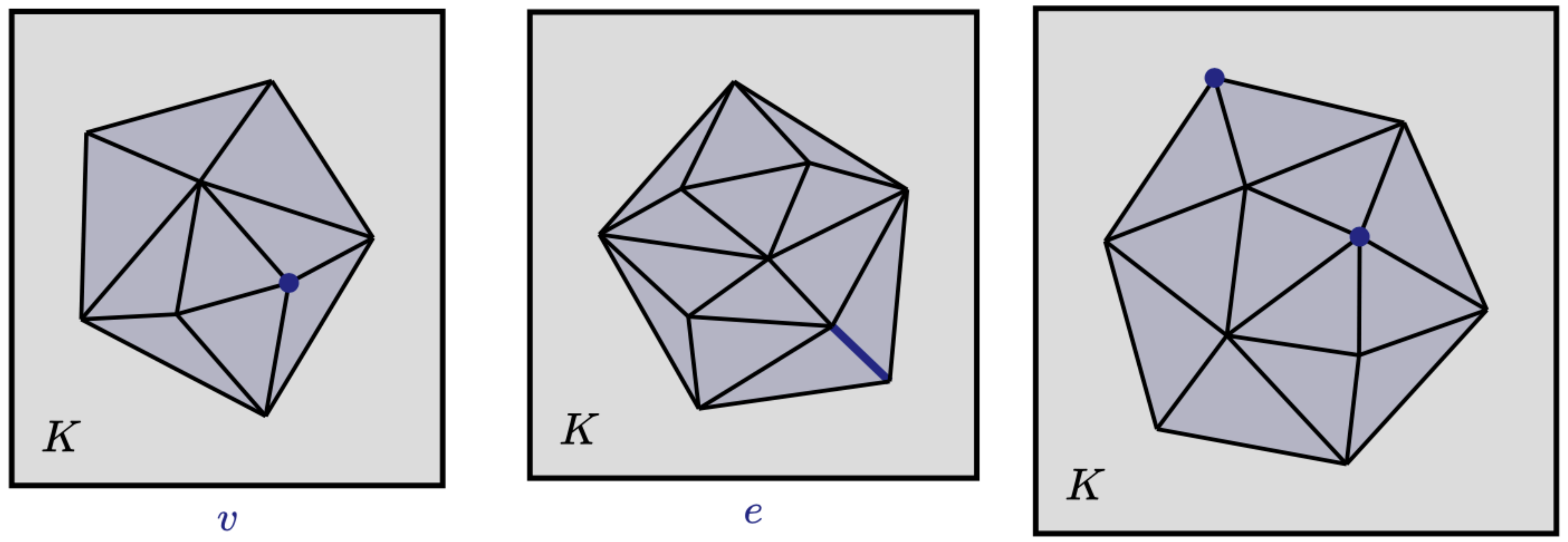
$\mathrm{St}(\mathrm{Cl}(E))$ $\quad$ $\mathrm{Cl}(\mathrm{St}(E))$ $\quad$ $\mathrm{Lk}(E) := \mathrm{Cl}(\mathrm{St}(E)) \setminus \mathrm{St}(\mathrm{Cl}(E))$

using many examples to build intuition

```
Vertex S ∈ K
```

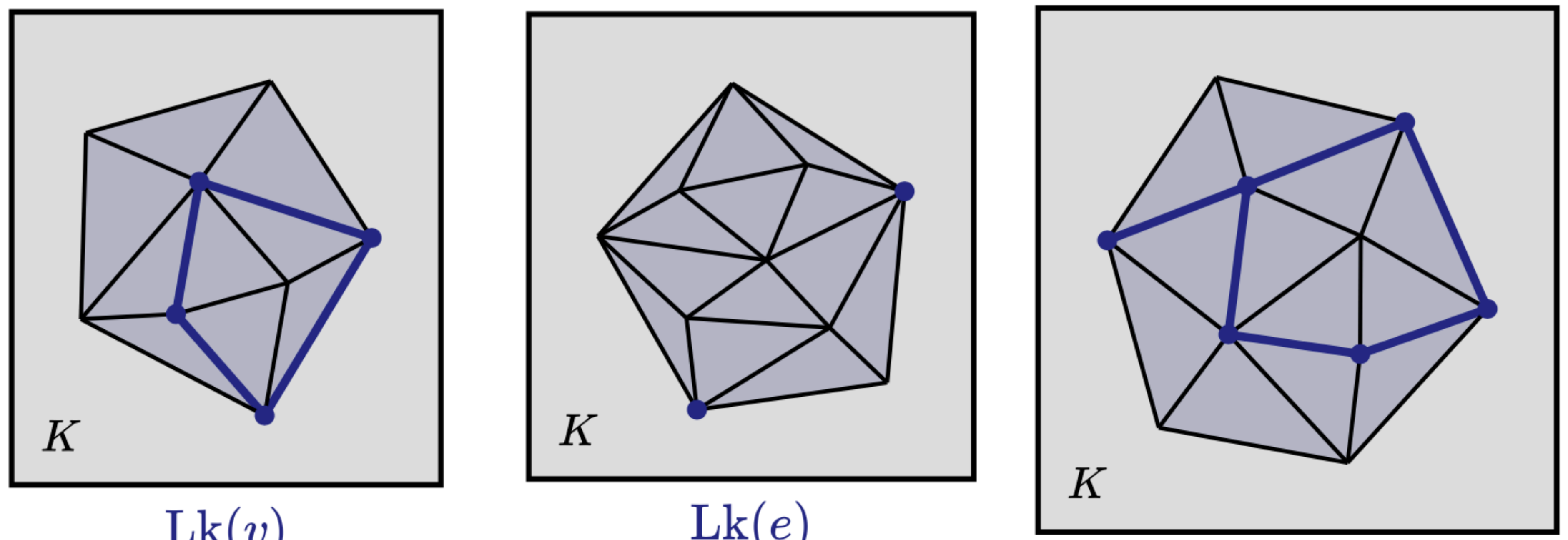
```
Edge S ∈ K
```

```
Vertex v1 ∈ K
Vertex v2 ∈ K
SimplicialSet S ⊆ K
S := {{v1},{v2}}
```



v $\quad$ e $\quad$ S

```
SimplicialComplex K
SimplicialSet LkS := Link(S)
```

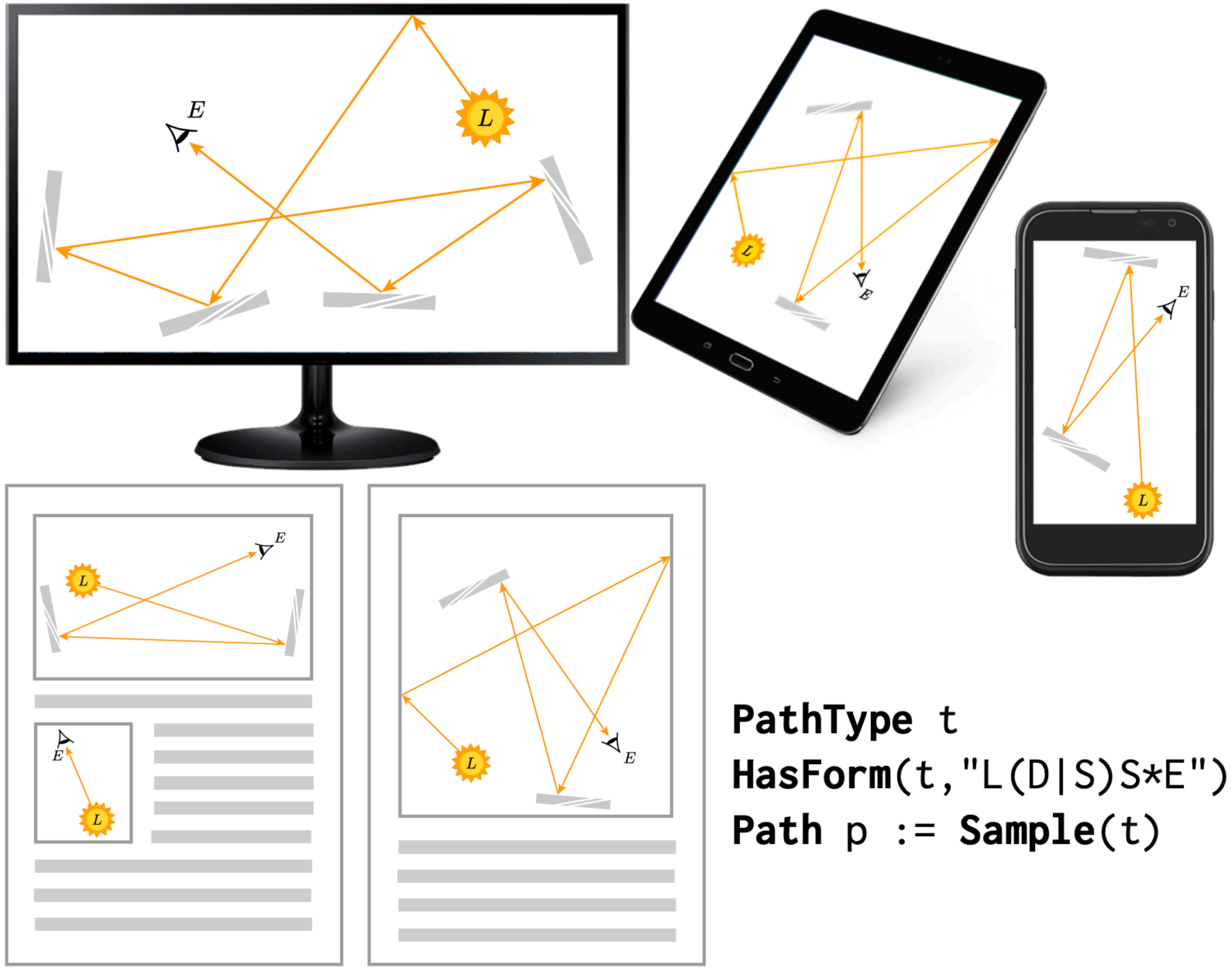


$\mathrm{Lk}(v)$ $\quad$ $\mathrm{Lk}(e)$ $\quad$ $\mathrm{Lk}(S)$

Visualizing rays

Visualizing rays

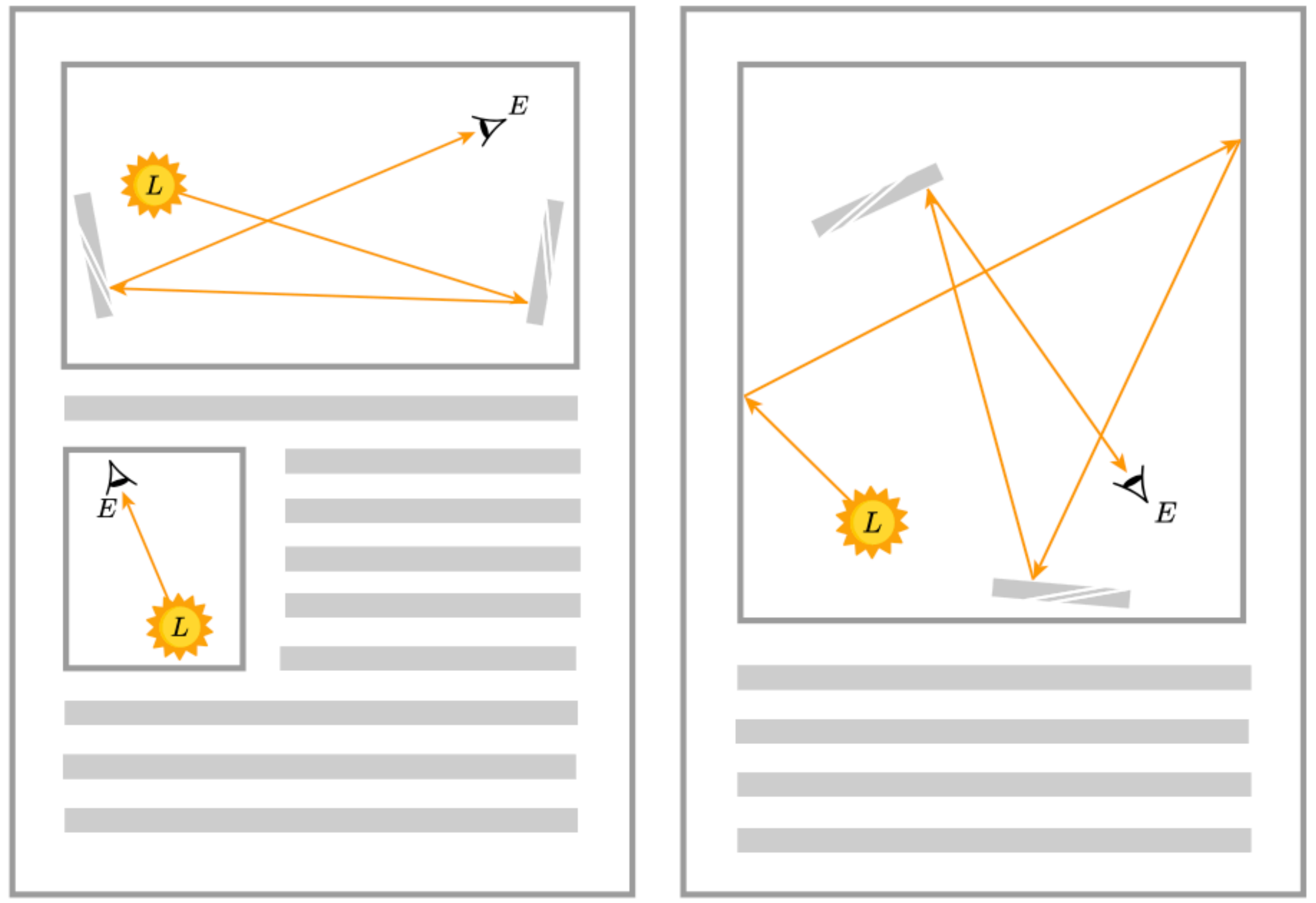
re-generating diagrams for a target platform




```
PathType t
HasForm(t,"L(D|S)S*E")
Path p := Sample(t)
```

Visualizing rays

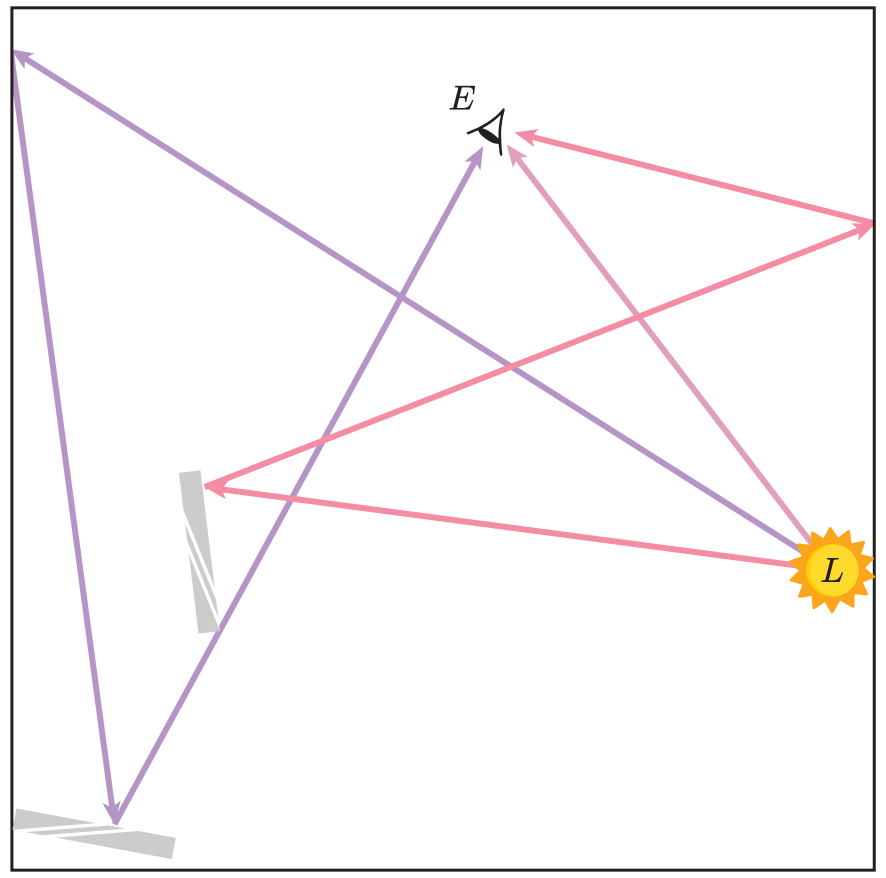
re-generating diagrams for a target platform

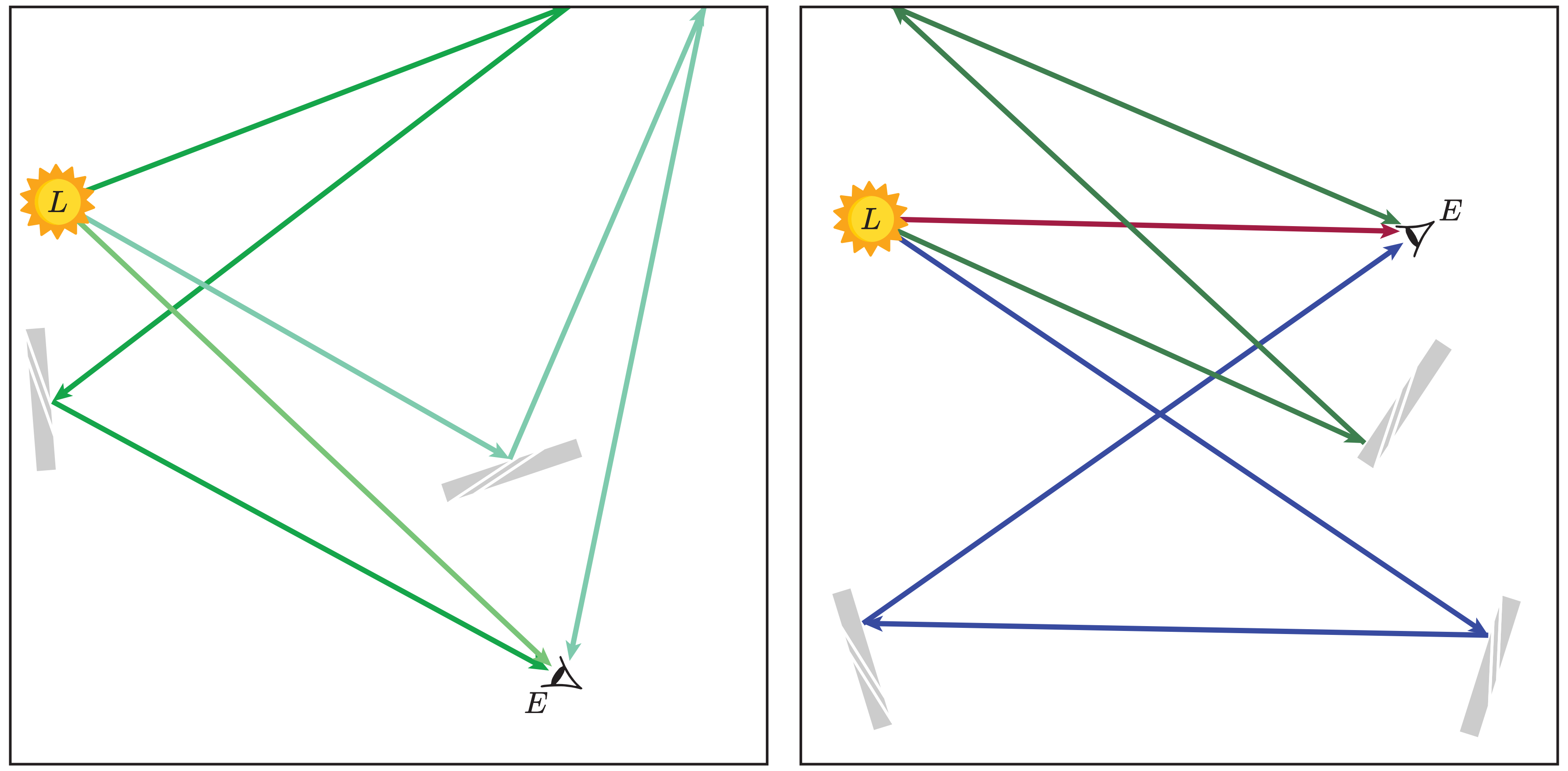



```
PathType t
HasForm(t,"L(D|S)S*E")
Path p := Sample(t)
```

handling a tricky specification

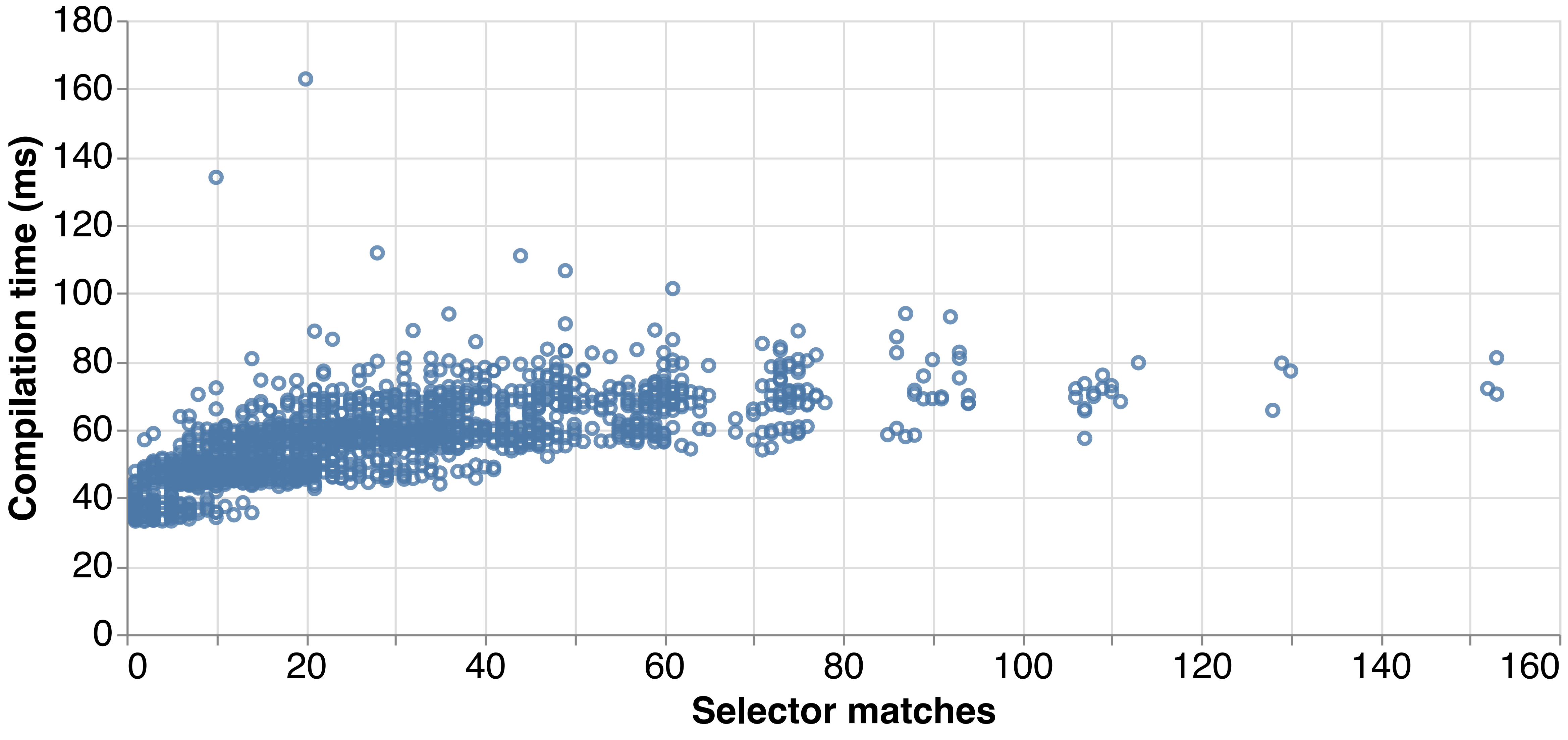
```
PathType t1, t2, t3
HasForm(t1, "L(D|S)SE")
HasForm(t2, "LSDE")
HasForm(t3, "LE")

Path p1, p2, p3
p1 := Sample(t1)
p2 := Sample(t2)
p3 := Sample(t3)
```



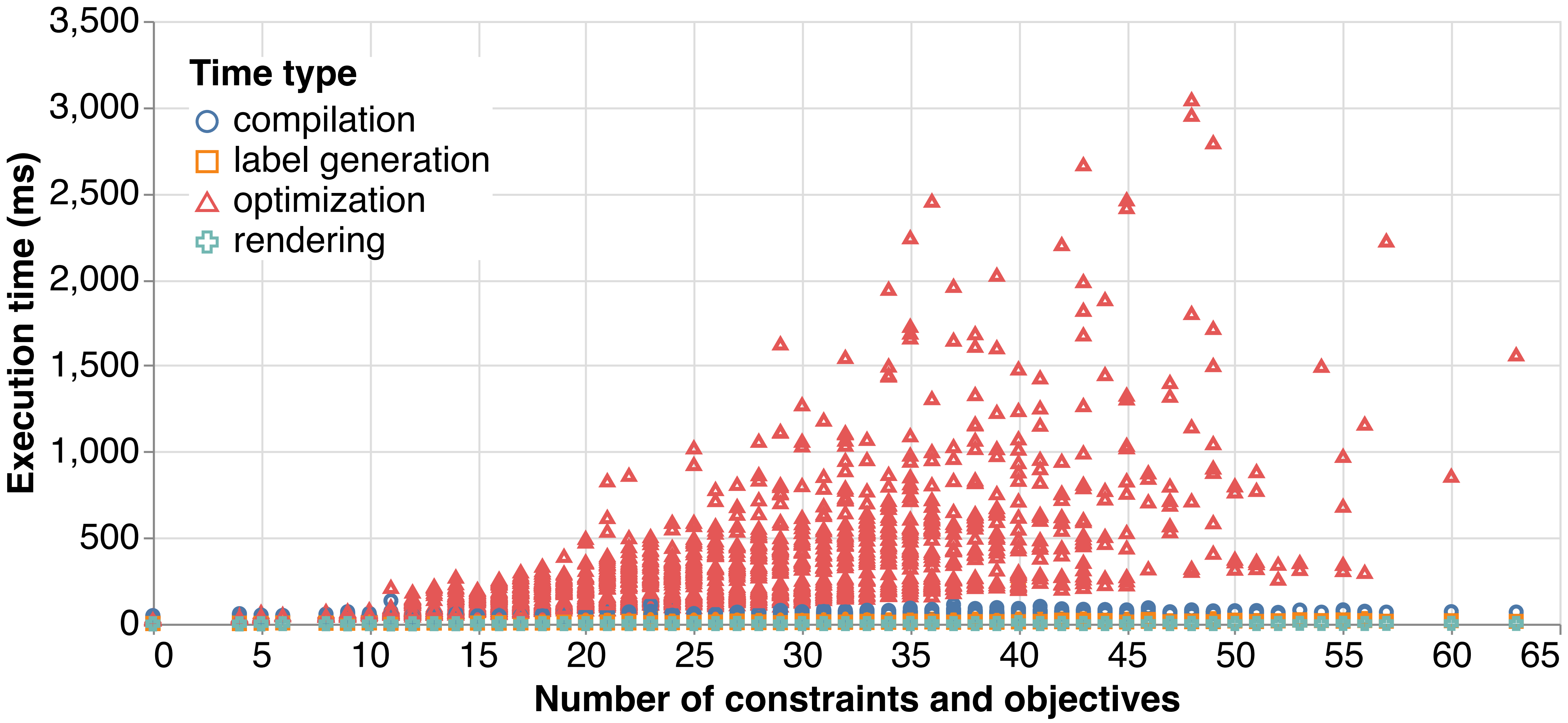


Performance scaling

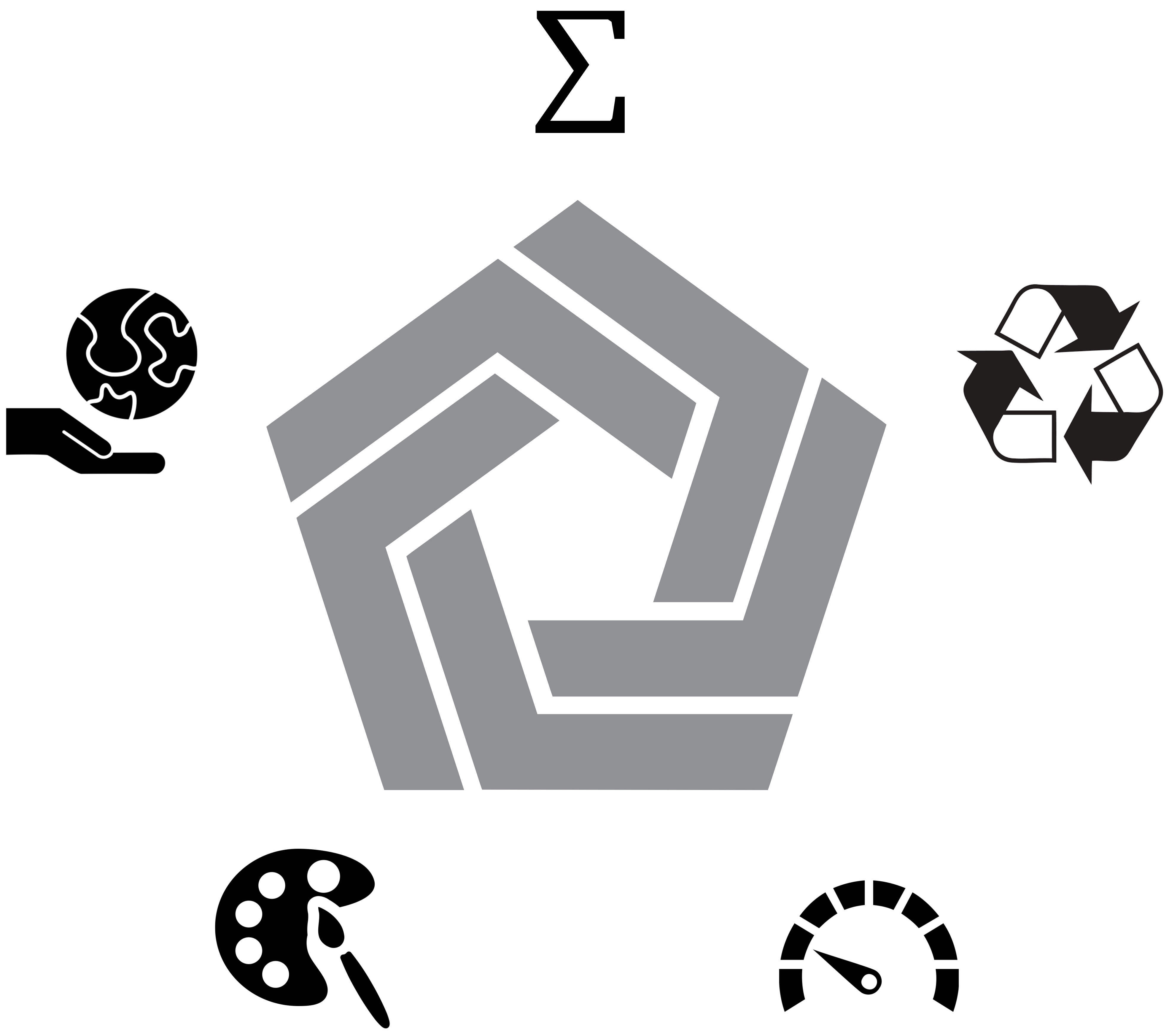
compilation time vs. program size



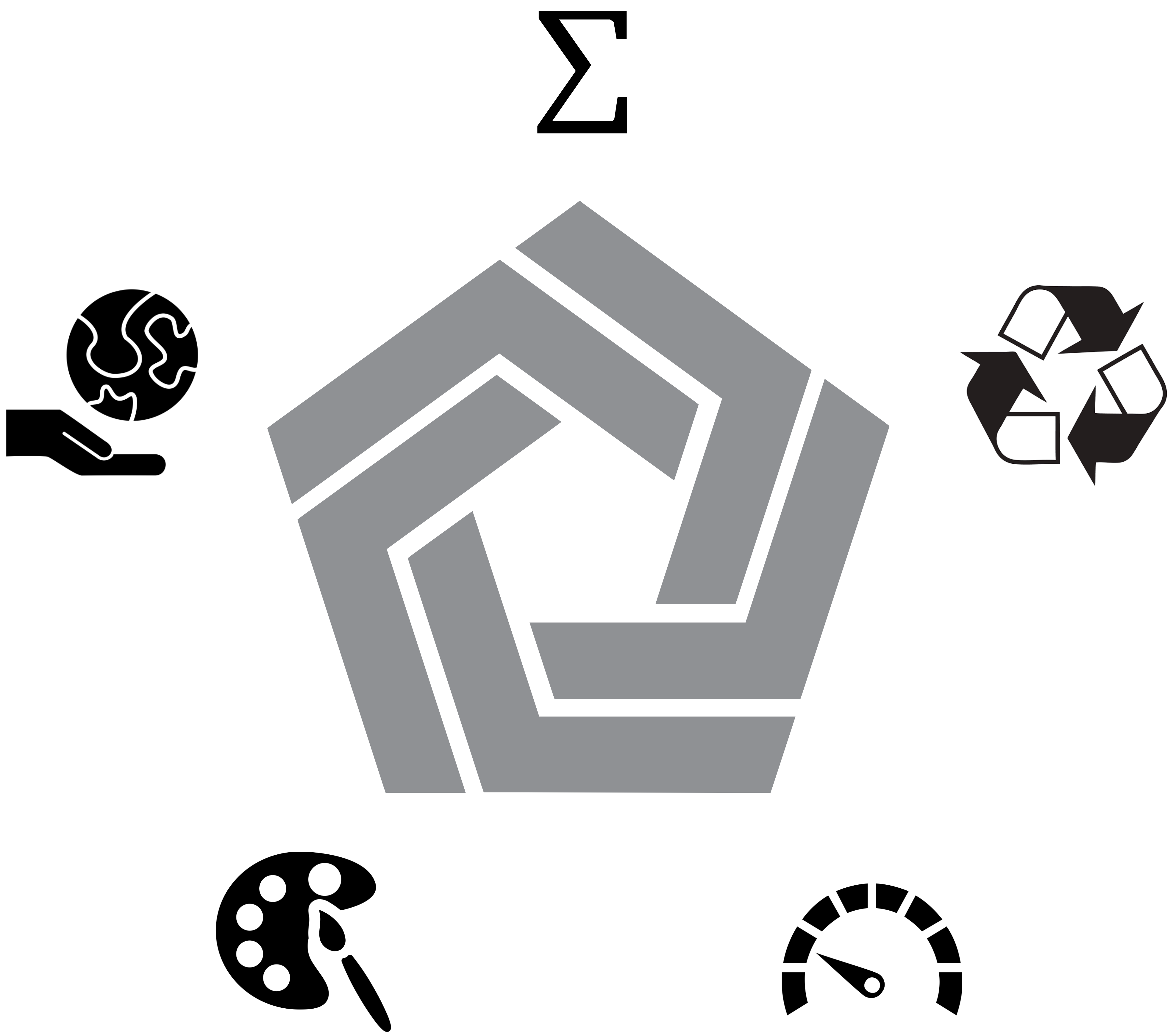
optimization time vs. problem size



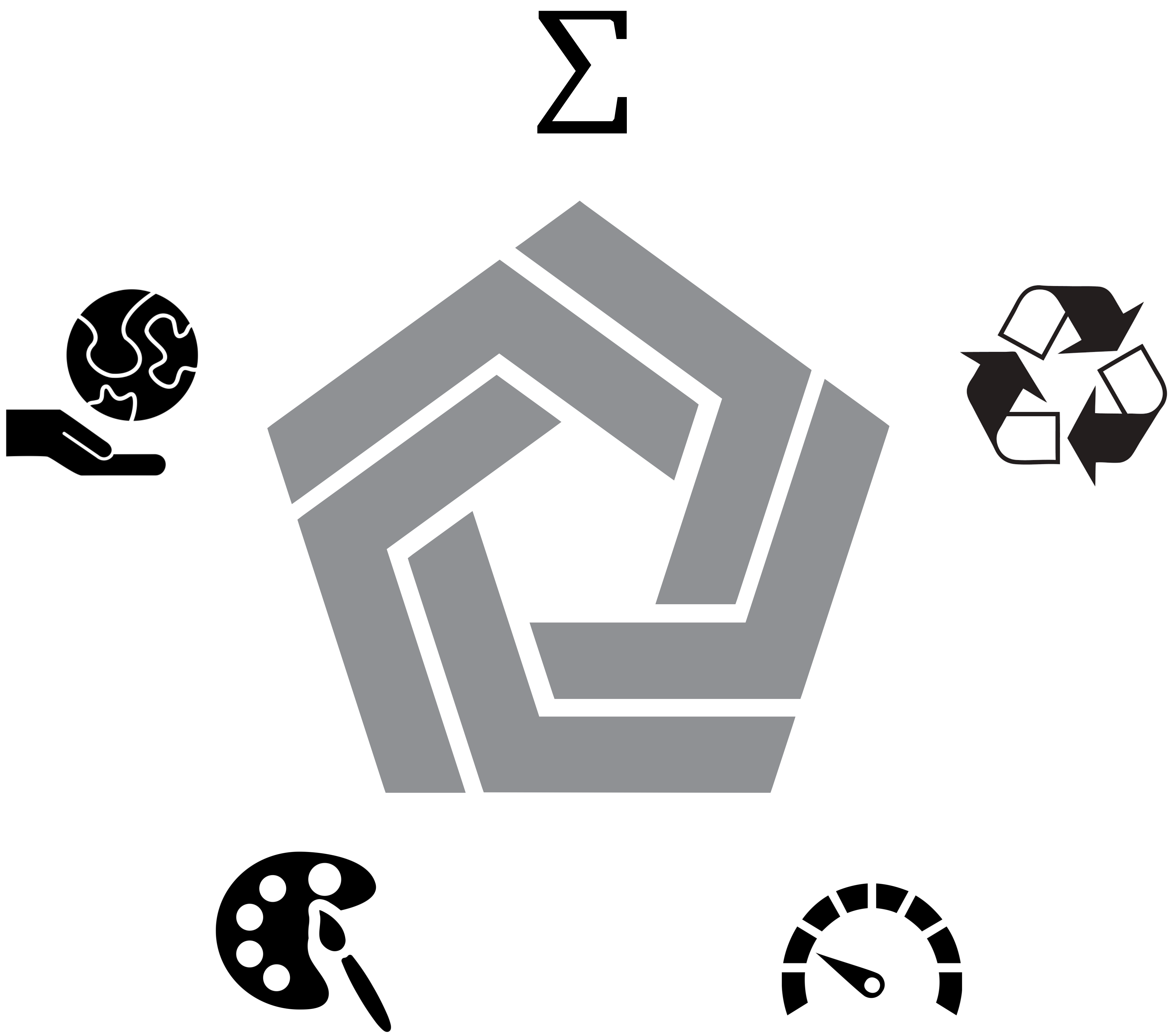
Insights



Insights

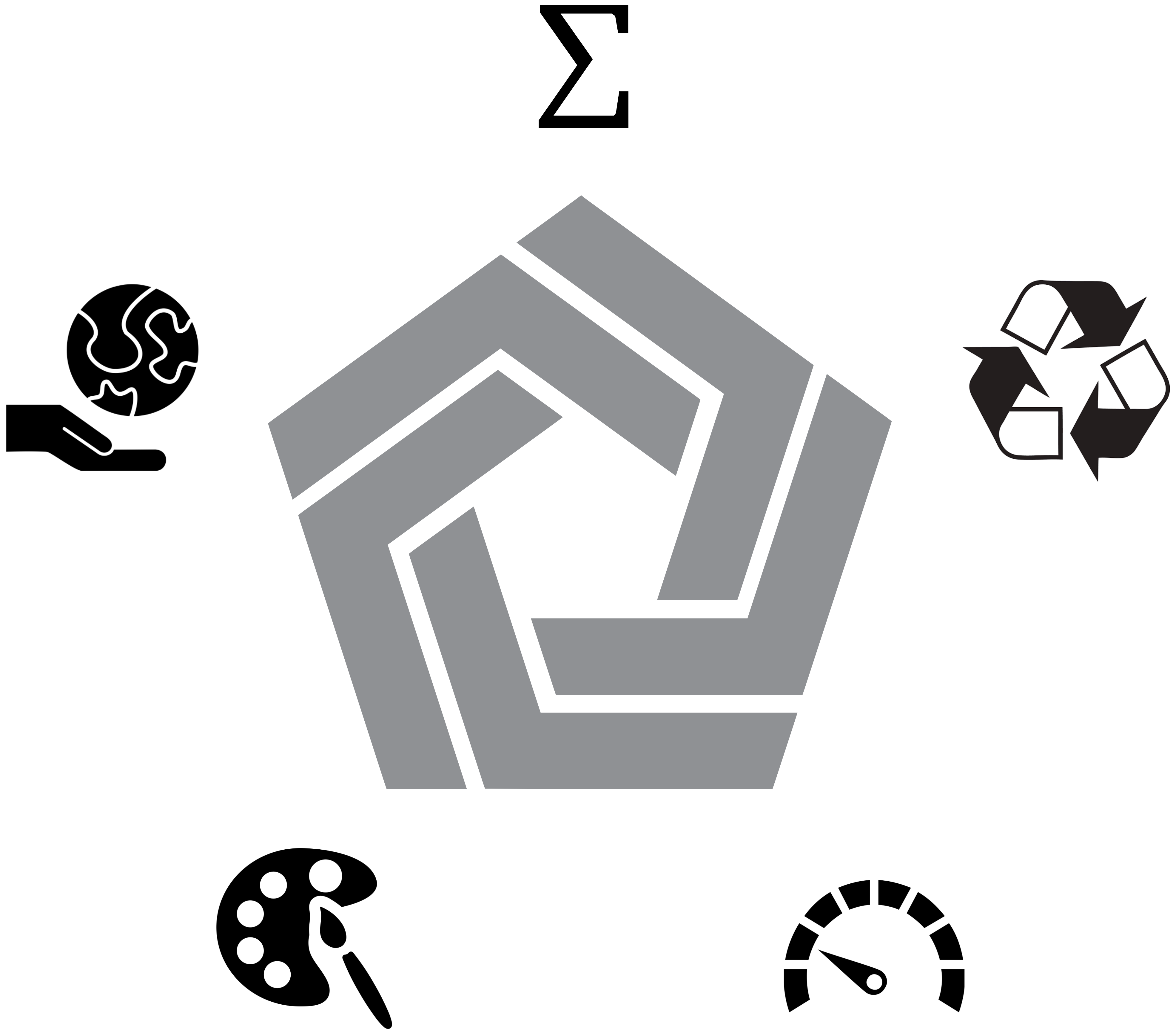


Insights



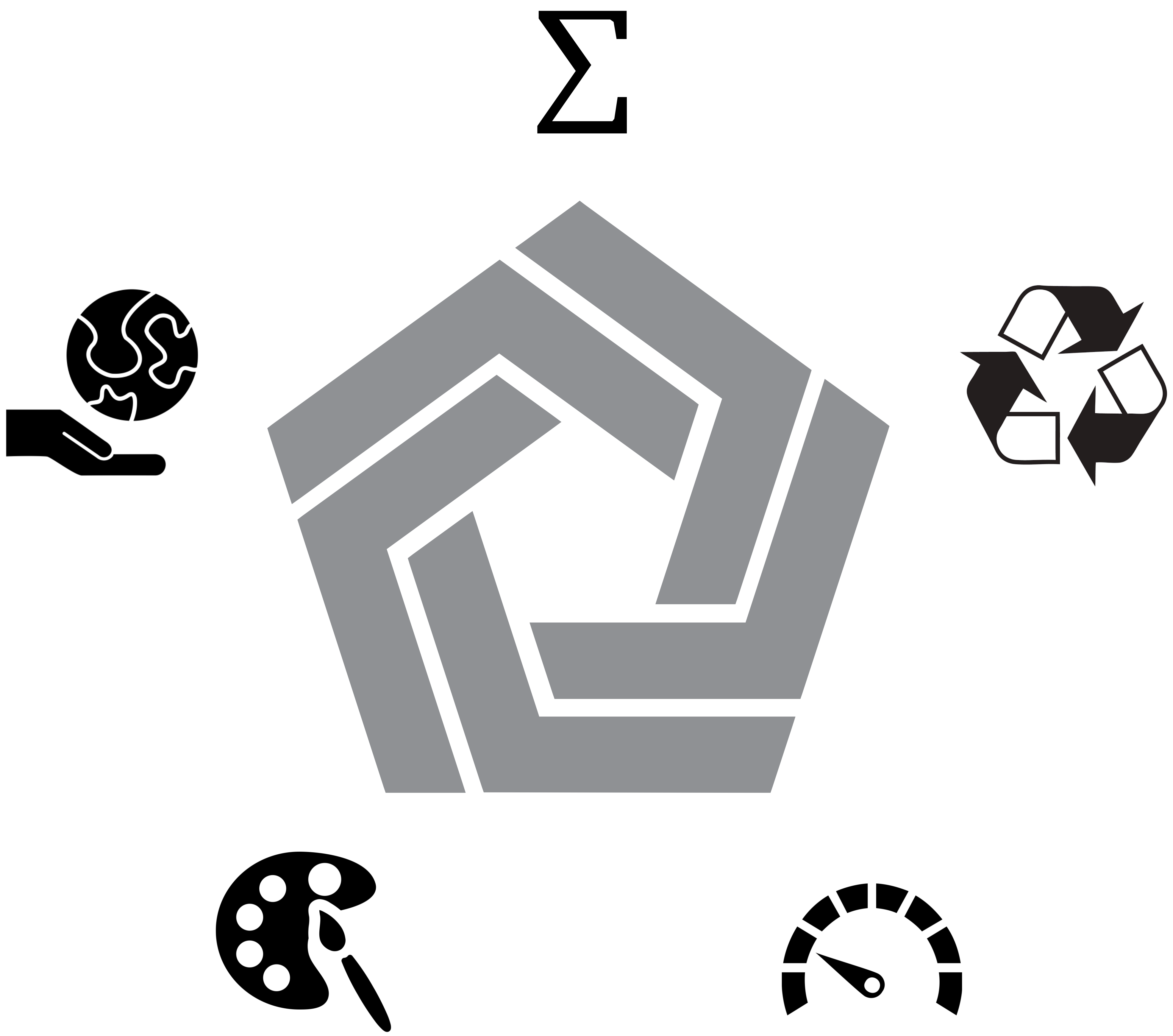
Insights

- Separating content and visual representation provides new capabilities.



Insights

- Separating content and visual representation provides new capabilities.
- The mapping from abstract objects to a visual representation provides rich semantics.

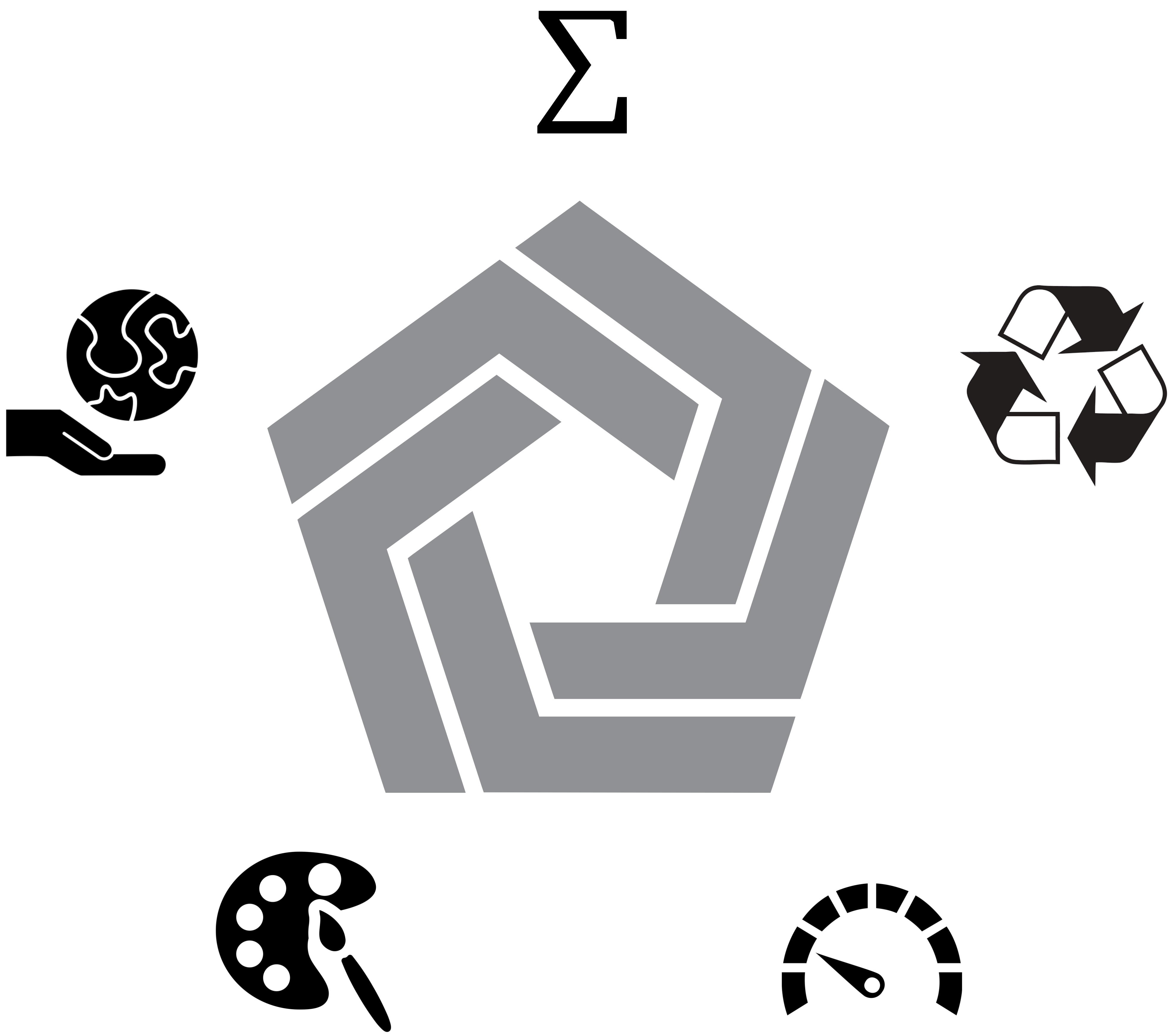


Insights

- Separating content and visual representation provides new capabilities.

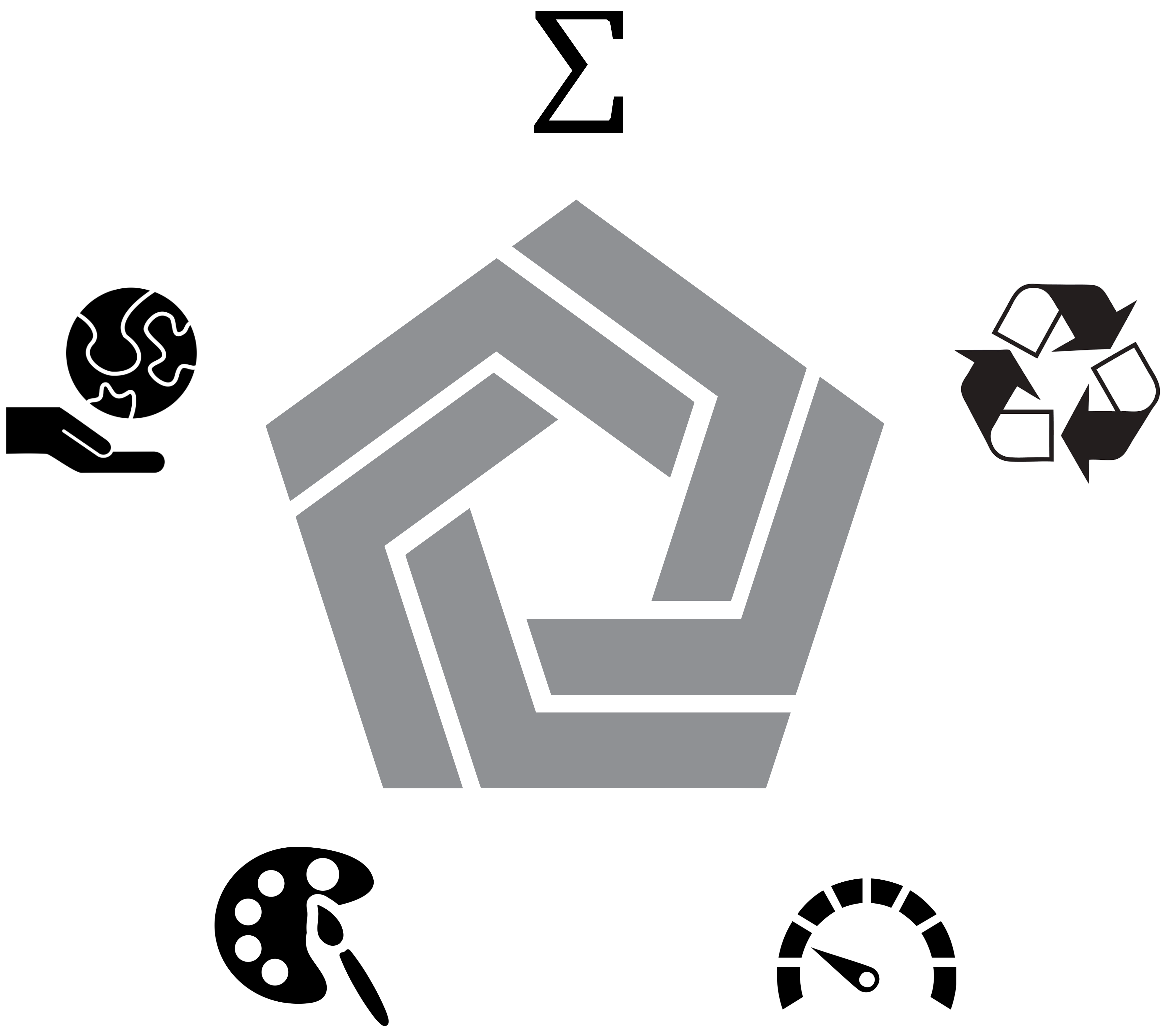
- The mapping from abstract objects to a visual representation provides rich semantics.

- Constrained optimization can be an effective general-purpose solution for diagram layout.



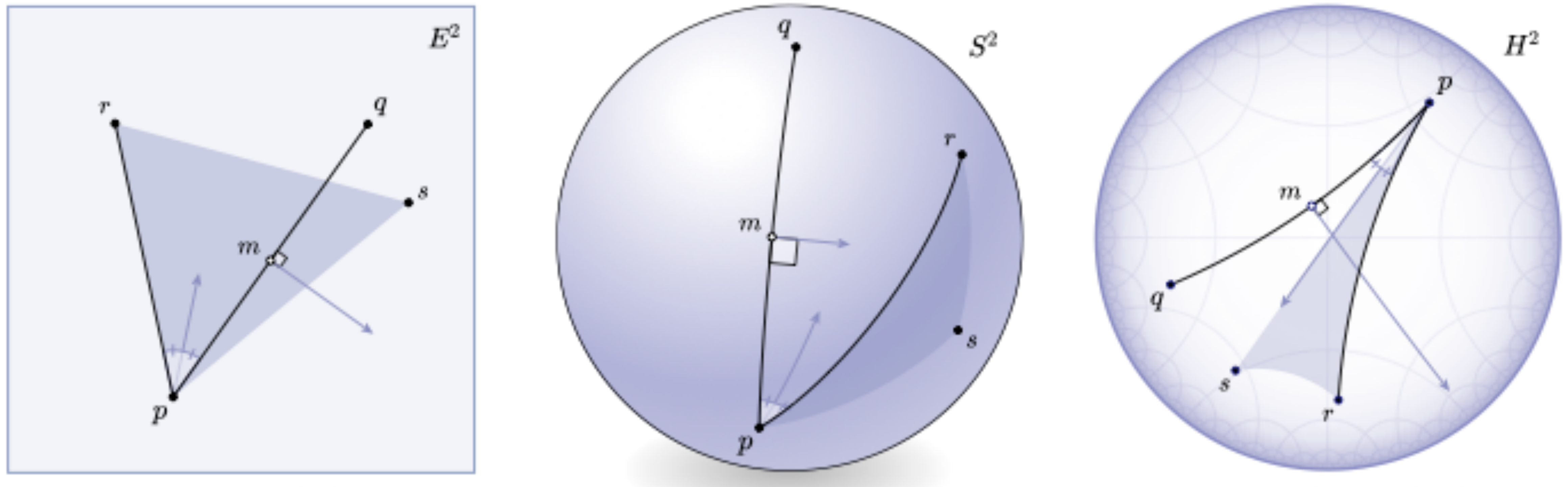
Insights

- Separating content and visual representation provides new capabilities.
- The mapping from abstract objects to a visual representation provides rich semantics.
- Constrained optimization can be an effective general-purpose solution for diagram layout.
- Penrose acts as a nexus for diagram generation, connecting **specification** and **synthesis**.

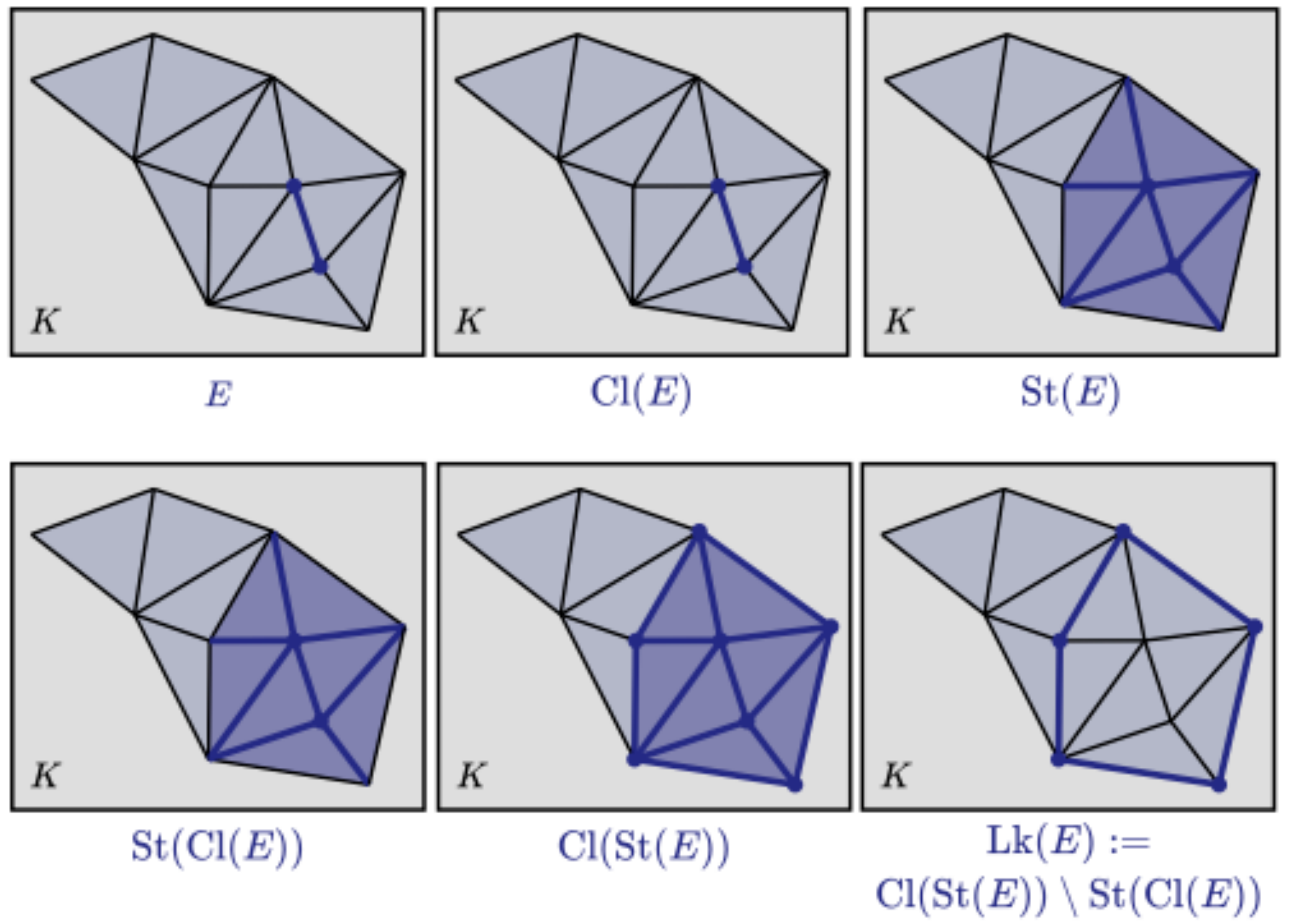


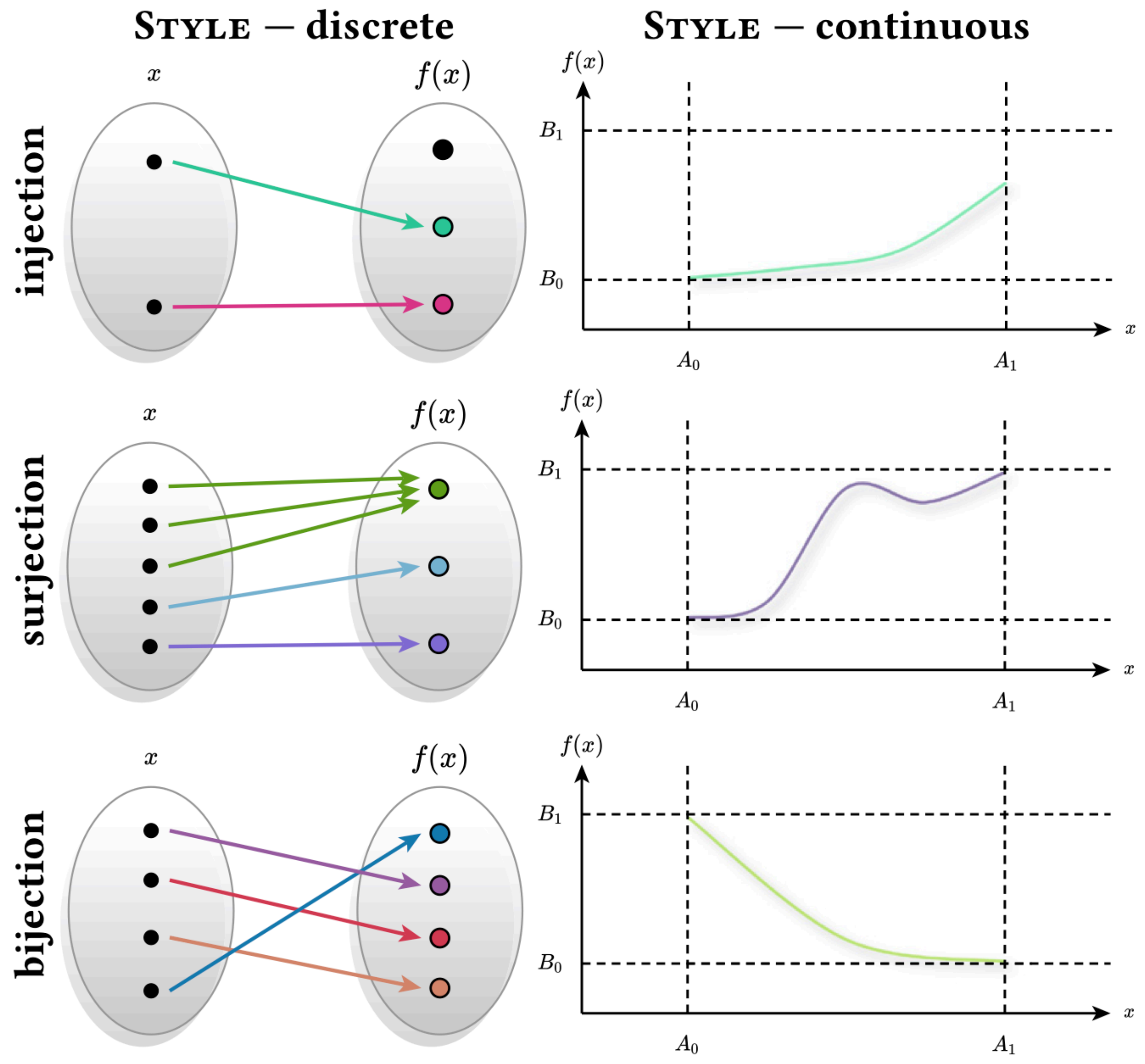
We want to make diagrams like these the norm, not the exception!

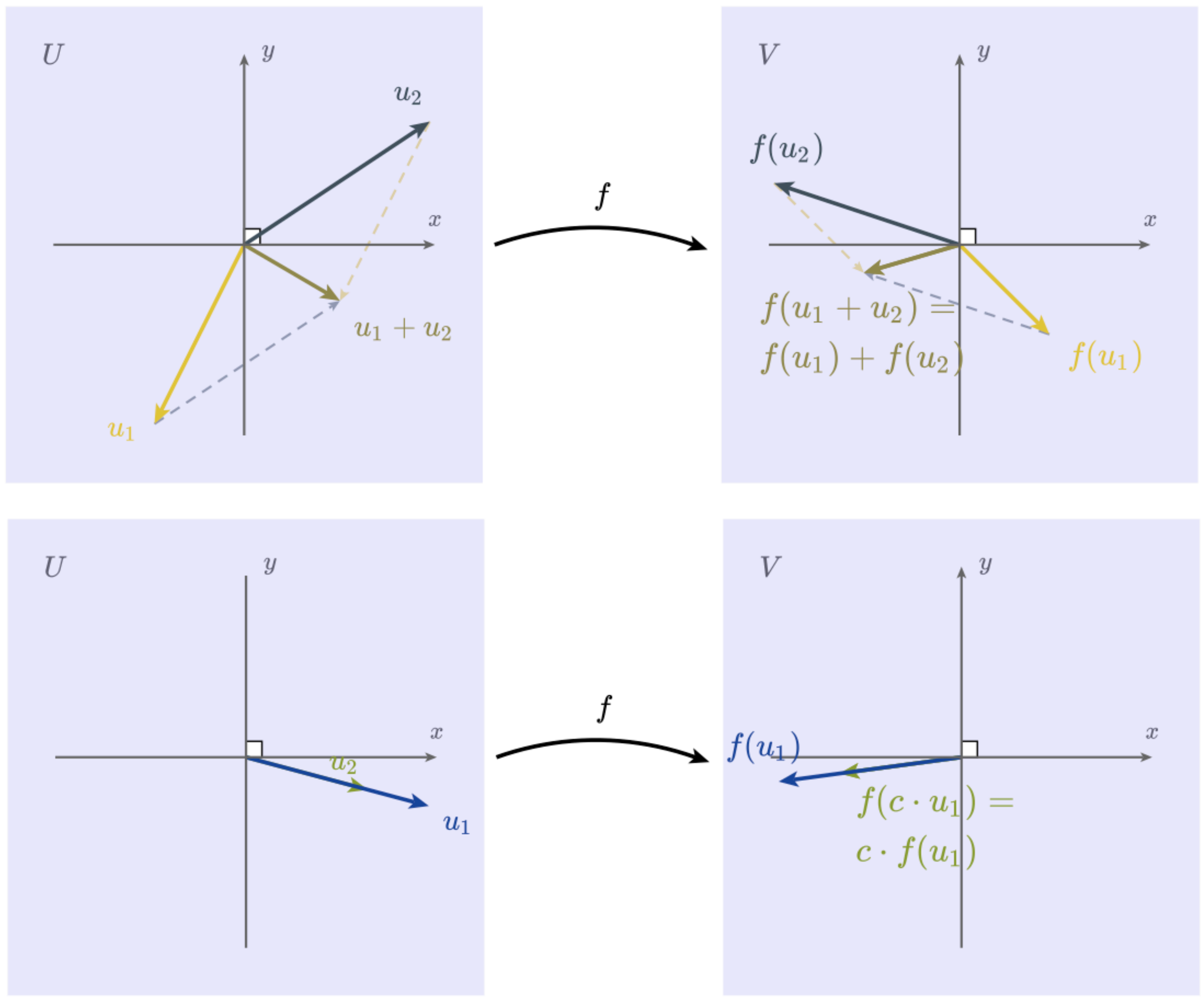
We want to make diagrams like these the norm, not the exception!



Style — Euclidean Style — spherical Style — hyperbolic







What's next?

What's next?

- Making the languages more expressive

What's next?

- Making the languages more expressive
- Making the system's built-ins more extensible (solver, renderer, functions, shapes)

What's next?

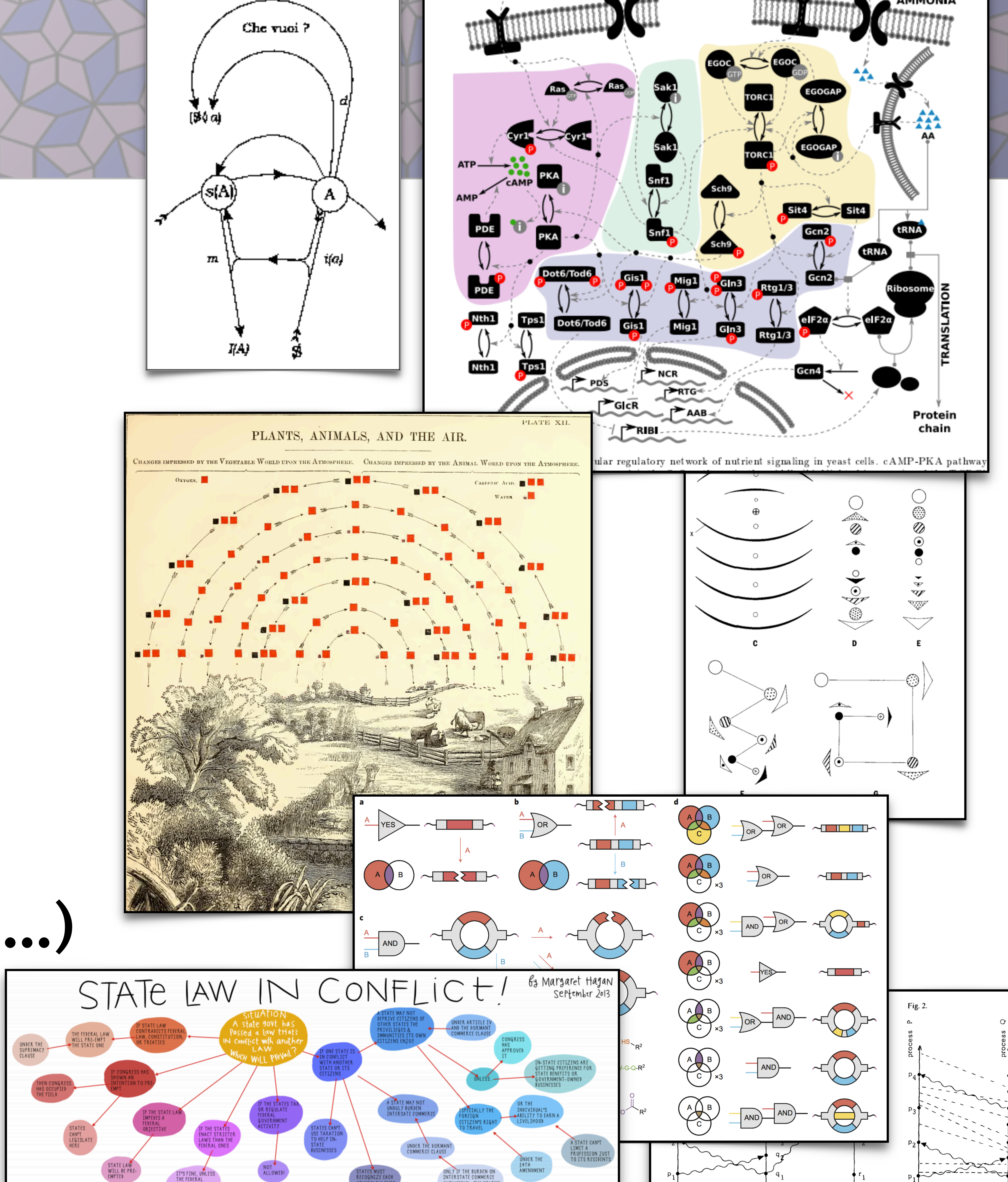
- Making the languages more expressive
- Making the system's built-ins more extensible (solver, renderer, functions, shapes)
- Improving optimization speed and strategy

What's next?

- Making the languages more expressive
- Making the system's built-ins more extensible (solver, renderer, functions, shapes)
- Improving optimization speed and strategy
- Visualizing concepts in other fields (law, philosophy, biology…)

What's next?

- Making the languages more expressive
- Making the system's built-ins more extensible (solver, renderer, functions, shapes)
- Improving optimization speed and strategy
- **Visualizing concepts in other fields (law, philosophy, biology...)**



Possibilities for a visualization platform

(mockups)

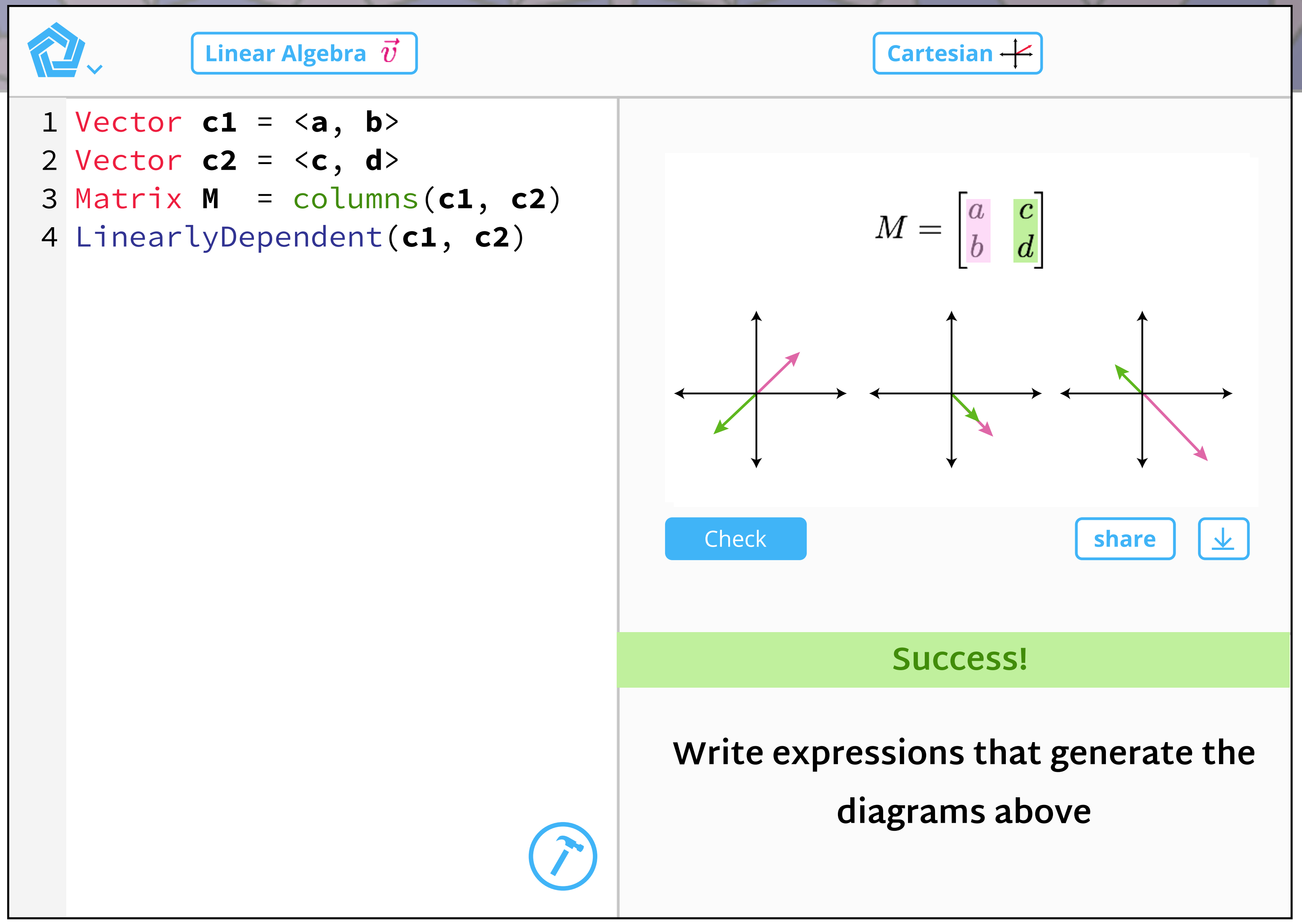
Possibilities for a visualization platform

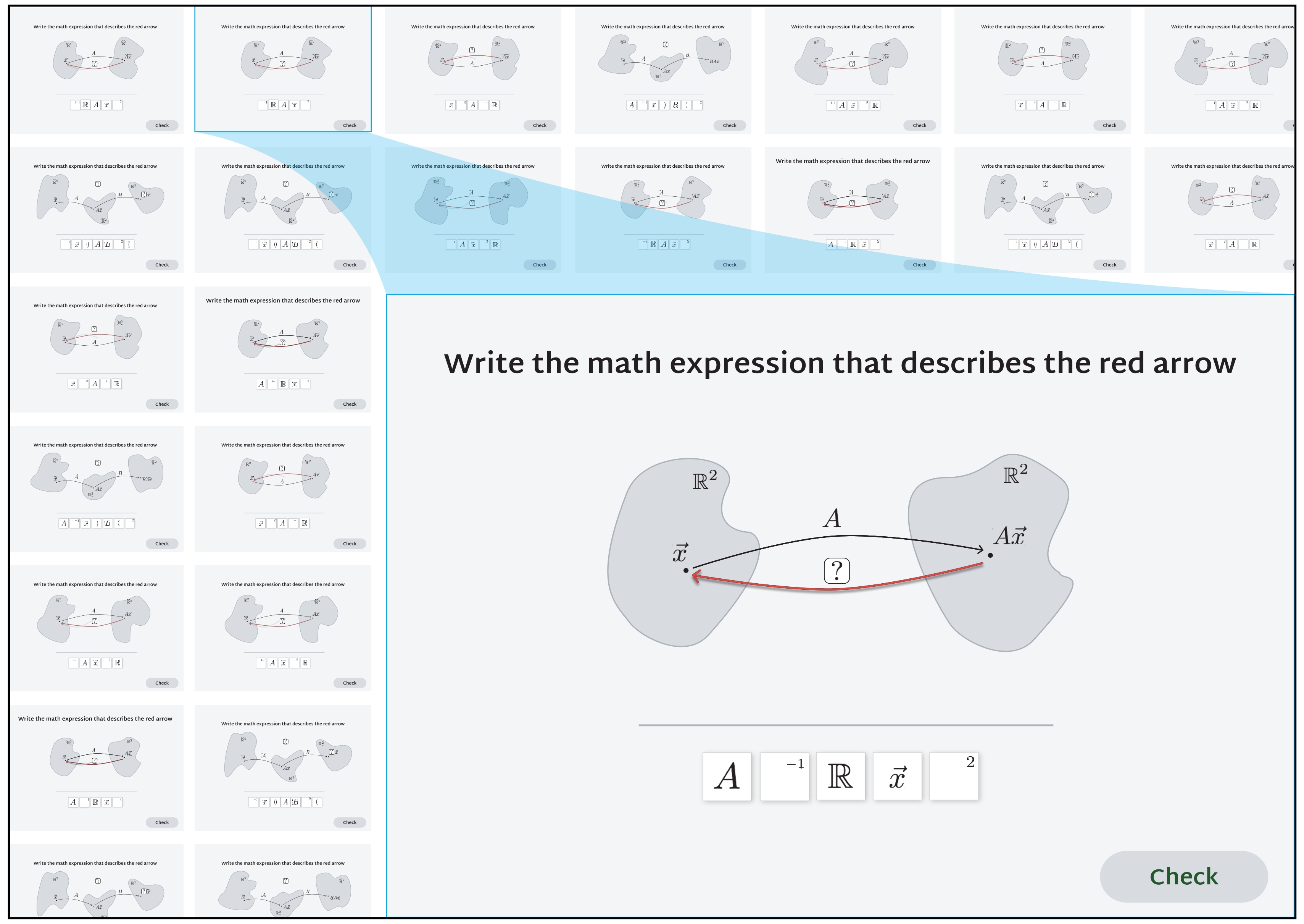
- **Visual tutoring**: auto-create custom visual problem sets at scale

(mockups)

Possibilities for a visualization platform

- **Visual tutoring:** auto-create custom visual problem sets at scale

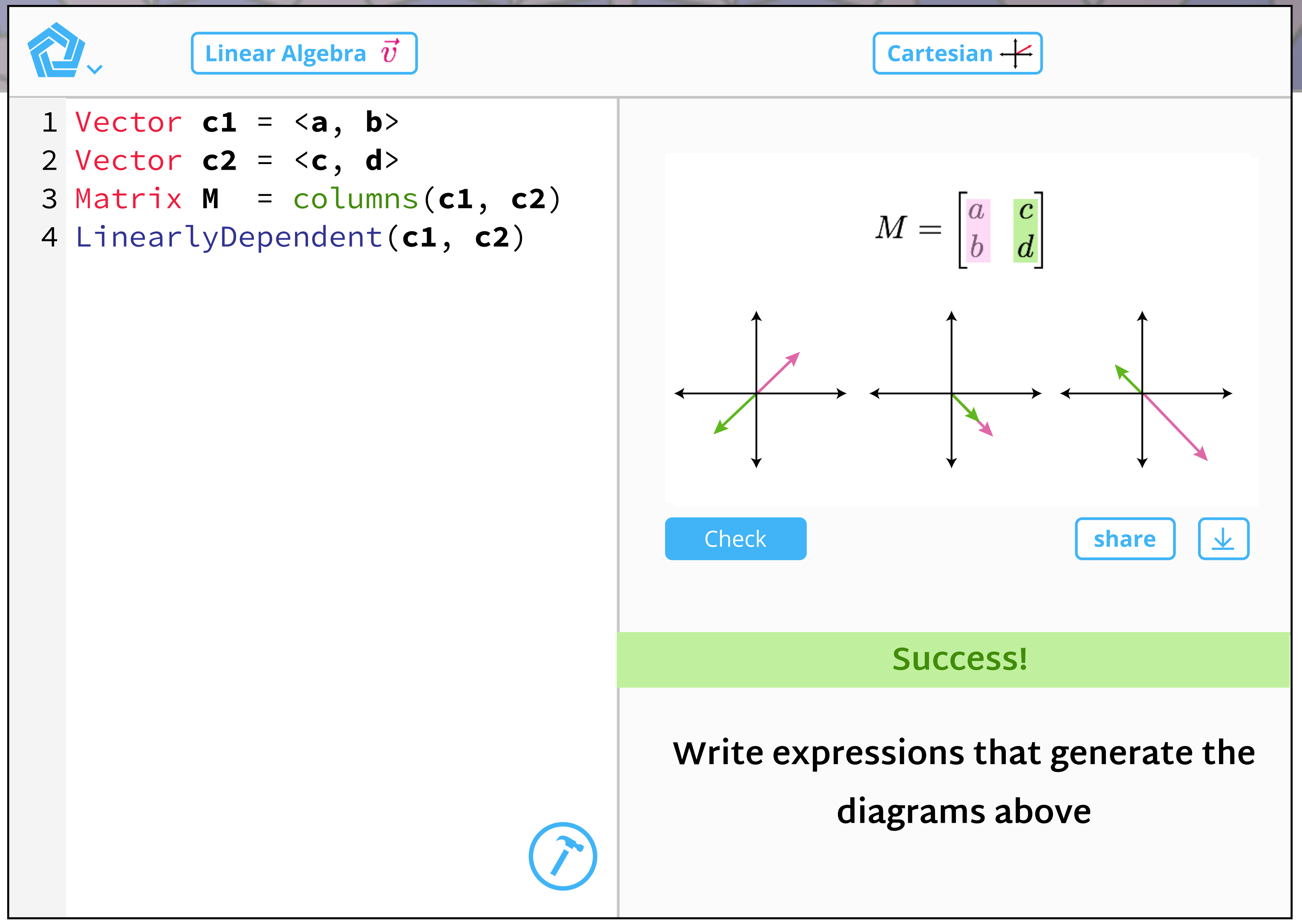


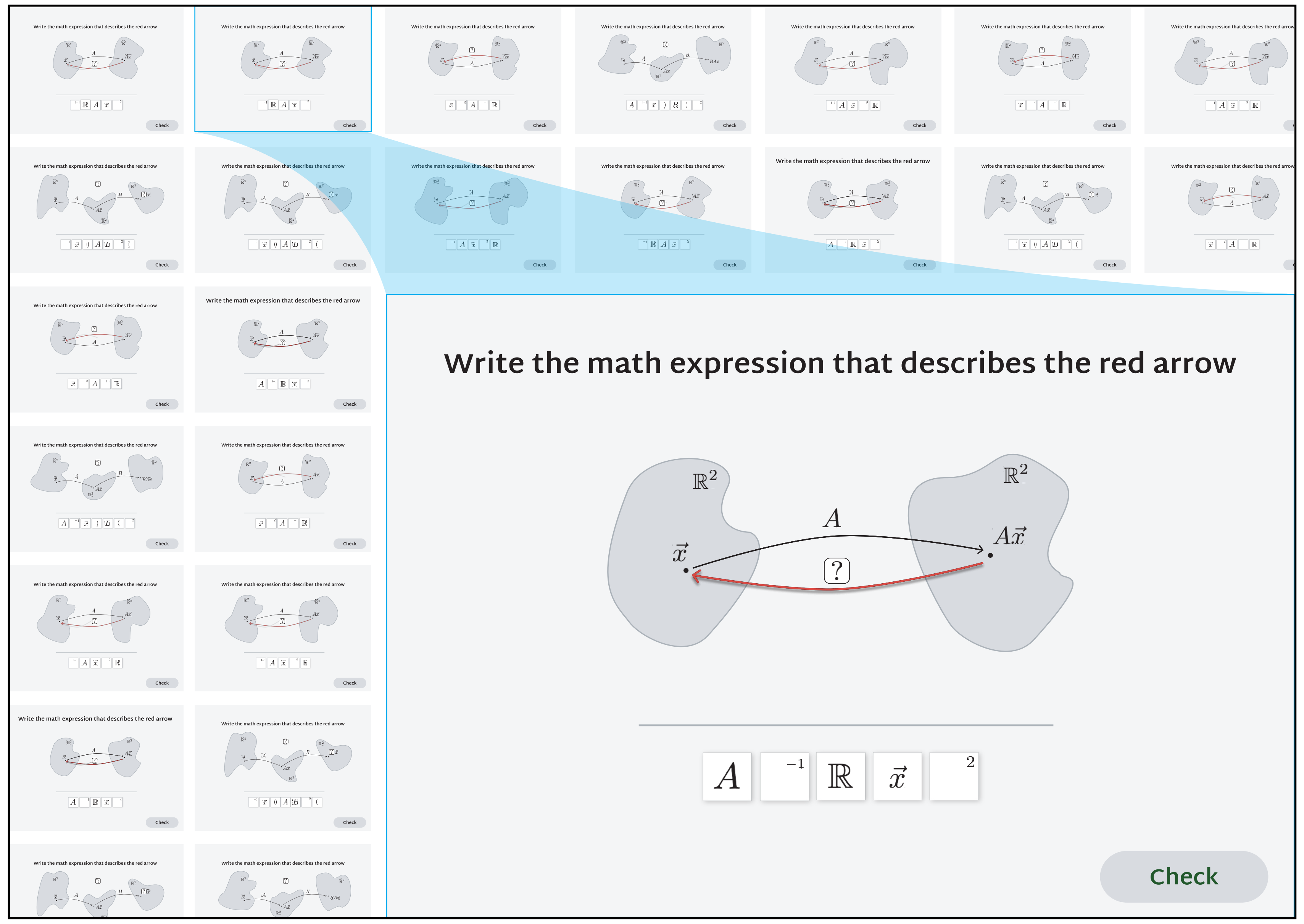


(mockups)

Possibilities for a visualization platform

- **Visual tutoring**: auto-create custom visual problem sets at scale
- **Rich visuals**: auto-create animated, interactive, staged diagrams

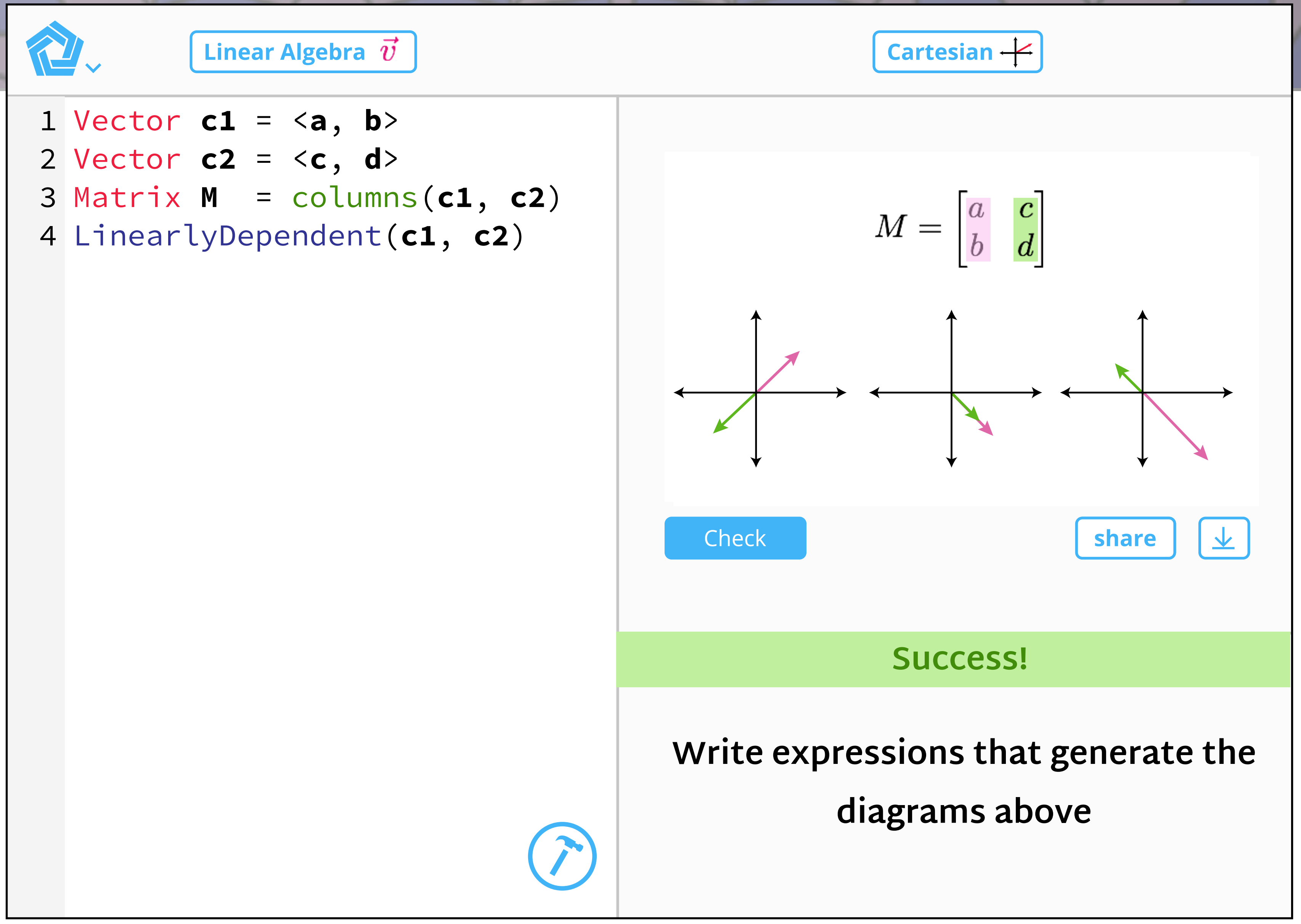


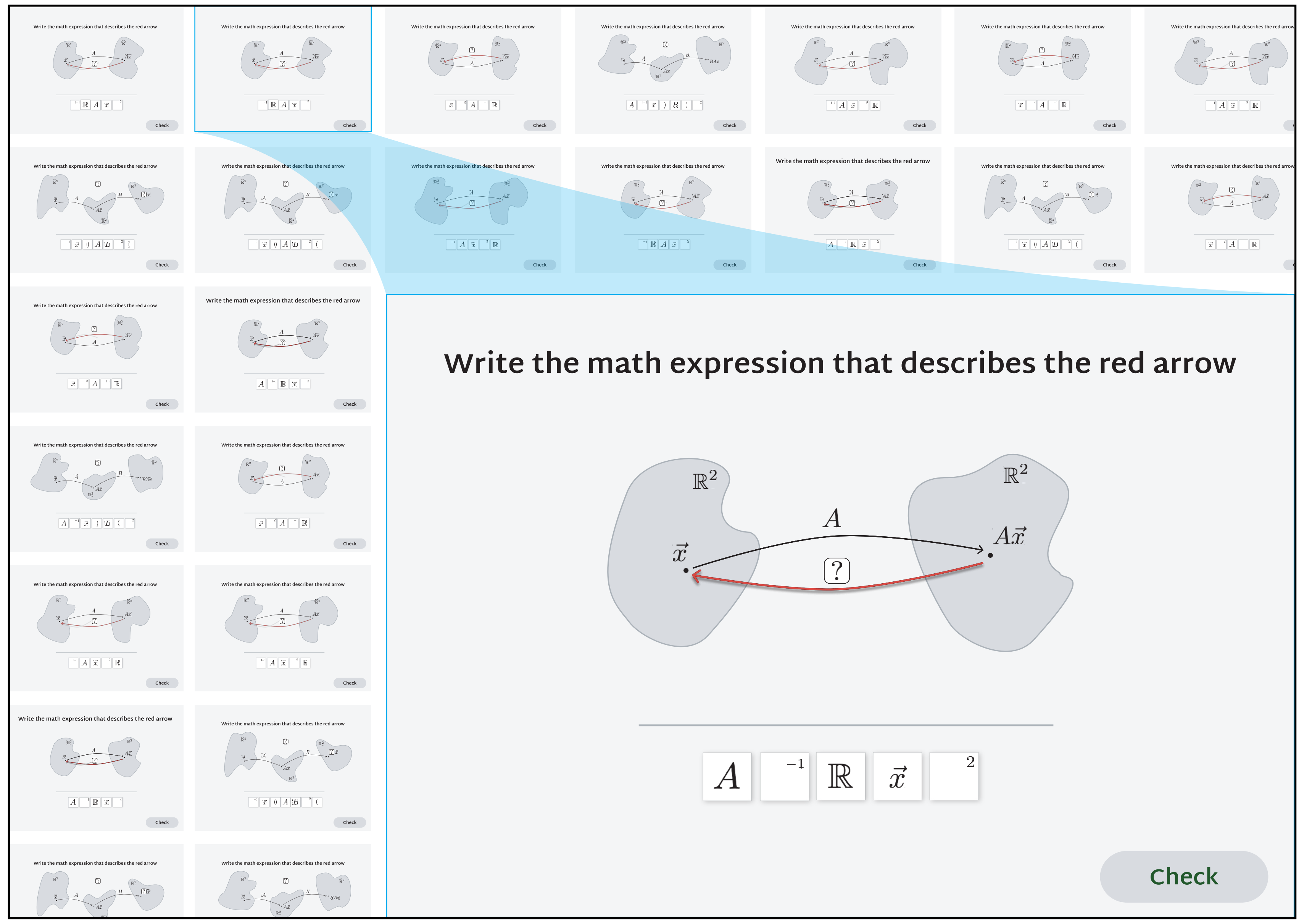


(mockups)

Possibilities for a visualization platform

- **Visual tutoring**: auto-create custom visual problem sets at scale
- **Rich visuals**: auto-create animated, interactive, staged diagrams
- **Meaningful interfaces**: auto-create a direct manipulation editor that is aware of diagrams' underlying meaning





(mockups)

Thanks!

https://penrose.ink