Codifying Visual Representations

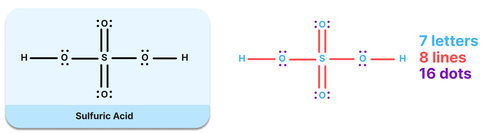
Wode Ni, Sam Estep, Hwei-Shin Harriman, Jiří Minarčík, Joshua Sunshine

Penrose Homepage | Penrose Editor | Penrose Gallery

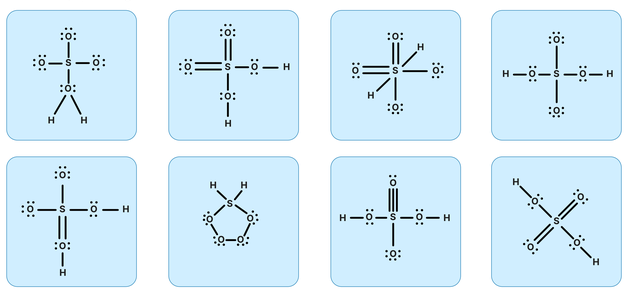
Making visually appealing and meaningful diagrams involves craftsmanship in designing the visual representation, drawing shapes, and laying them out. Can the effort spent on diagrams by an expert be reused by others, especially those without the expertise in design and drawing? In this paper, we outline our prior work on Penrose [1], a diagramming tool with first-class support for reusing visual representations. The nature of our approach to reusability necessitates a domain-agnostic method to automatically lay out a diagram. We highlight our existing approach for general diagram layout and styling, and propose a new composable approach for codifying visual representations to reuse expertise that cuts across domains.

Defining Visual Representations

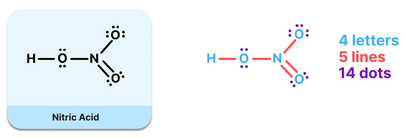
Making a diagram is a time-consuming, detail-oriented process.
Consider this Lewis structure of sulfuric acid:



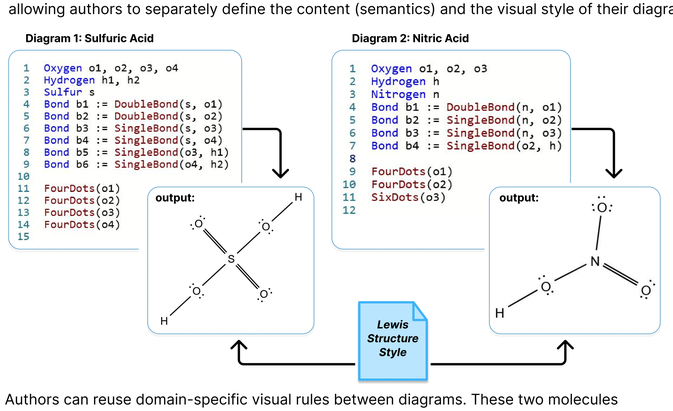
What if you need several variations? Most of your time will be taken up with copy-pasting and low-level tweaks:



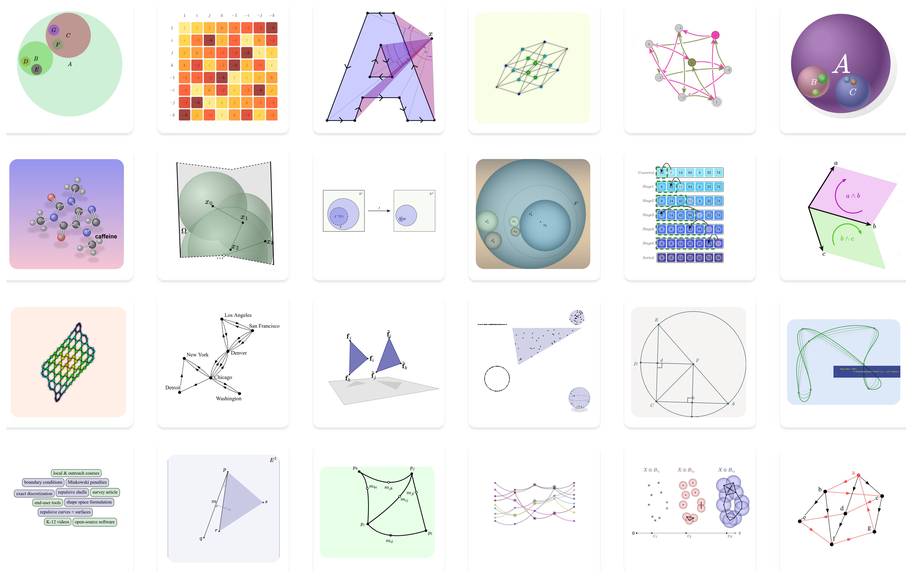
What if you need to make a Lewis structure for another molecule, such as nitric acid?
The effort that you have already put into drawing sulfuric acid barely transfers:



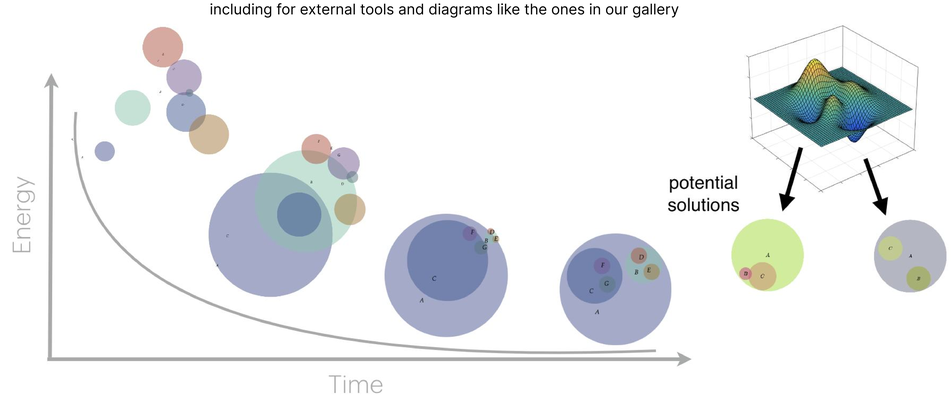
Our work on Penrose, an open-source diagramming tool, provides one solution to these issues by allowing authors to separately define the content (semantics) and the visual style of their diagram.



Authors can reuse domain-specific visual rules between diagrams. These two molecules were created using the same Lewis structure Style file in Penrose.



Penrose has been used to define visual representations across a wide range of domains, including for external tools and diagrams like the ones in our gallery



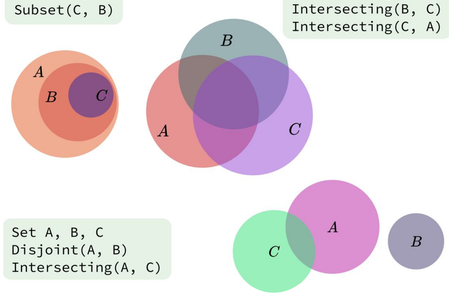
Penrose is able to find multiple diagram solutions by resampling because we internally represent diagrams as mathematical "energy landscapes" (our measure of "how good is this diagram?")

Reusing Visual Representations

Diagramming tools should let users compose diagrams in a manageable, scalable, composable way. Penrose authors build diagrams using two languages: Substance and Style:

```
Set A, B, C
Subset(B, A)
Subset(C, B)
```

```
Set A, B, C
Intersecting(A, B)
Intersecting(B, C)
Intersecting(C, A)
```



```
Set A, B, C
Disjoint(A, B)
Intersecting(A, C)
```

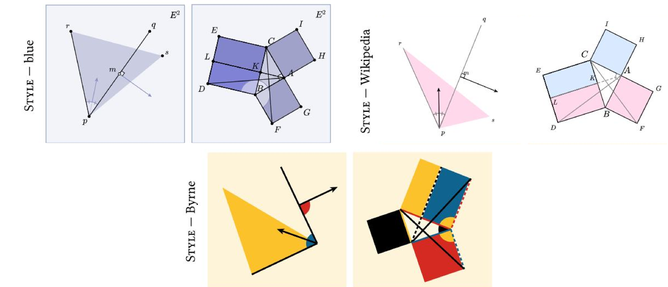
Substance: Semantic content of a diagram

```
forall Set x {
  x.shape = Circle { }
  x.text = Equation { string: x.label }
  ensure contains(x.icon, x.text)
  encourage norm(x.text.center - x.icon.center) == 0
}
forall Set x, y where Subset(x, y) {
  ensure disjoint(y.text, x.icon, 10)
  ensure contains(y.icon, x.icon, 5)
}
forall Set x, y where Disjoint(x, y) {
  ensure disjoint(x.icon, y.icon)
}
forall Set x, y where Intersecting(x, y) {
  ensure overlapping(x.icon, y.icon)
  ensure disjoint(y.text, x.icon)
  ensure disjoint(x.text, y.icon)
}
```

Style: Visual specification of a diagram domain

Penrose Style

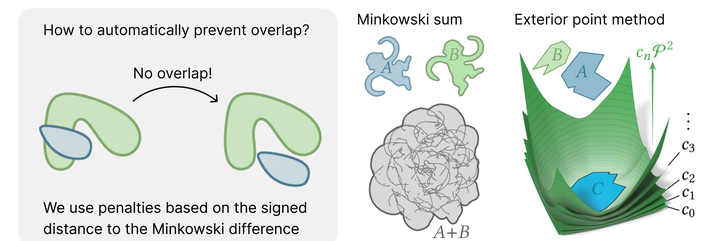
The cascading design of Style allows one to modify a diagram. All three of these figures were made by building on top of the same basic constraints and objectives.



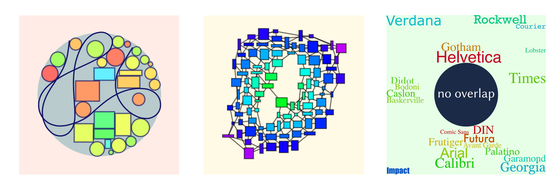
Style Objectives and Constraints - Minkowski Penalties

Style is composable in that it allows authors to build on top of existing primitives that are a part of the built-in Penrose library. One example is the disjoint function, which prevents objects from overlapping.

Ensuring that objects in a diagram (which could be irregularly shaped) do not overlap is a challenge which our recent work on Minkowski penalties [2] addresses:



Enforcing constraints via differentiable penalties enables our approach to be integrated with other energies commonly used for illustration and data visualization.

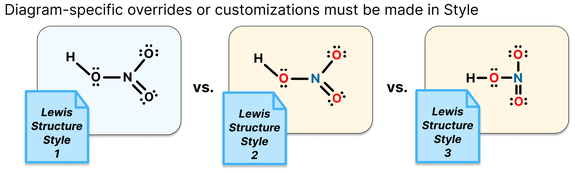


Elastic curve energy | Spectral graph energy | t-SNE embedding

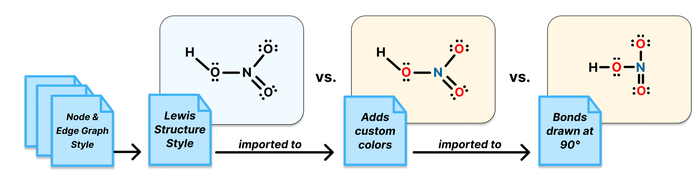
Composing Visual Representations

Limitations of the composability of Style:

- Each Style file is standalone
- Diagram-specific overrides or customizations must be made in Style



Instead, we propose that Style should become **composable** so authors can define and combine modules together to customize their diagrams and leverage existing designs without copy-pasting.



[1] Ye, K., Ni, W., Krieger, M., Ma'ayan, D., Wise, J., Aldrich, J., ... & Crane, K. (2020). Penrose: from mathematical notation to beautiful diagrams. ACM Transactions on Graphics (TOG), 39(4), 144-1.
[2] Minarčík, Jiří, Sam Estep, Wode Ni, and Keenan Crane. "Minkowski Penalties: Robust Differentiable Constraint Enforcement for Vector Graphics." In ACM SIGGRAPH 2024 Conference Papers, pp. 1-12. 2024.