AST Builder Redesign for HATS

Overview

The current AST Builder demonstrates type-safe visualization composition but feels cluttered and doesn't foreground the key HATS insight: enumeration and assembly are orthogonal choices forming an N×M matrix. This plan redesigns the showcase to make that combinatorial power visible and explorable.

Current State

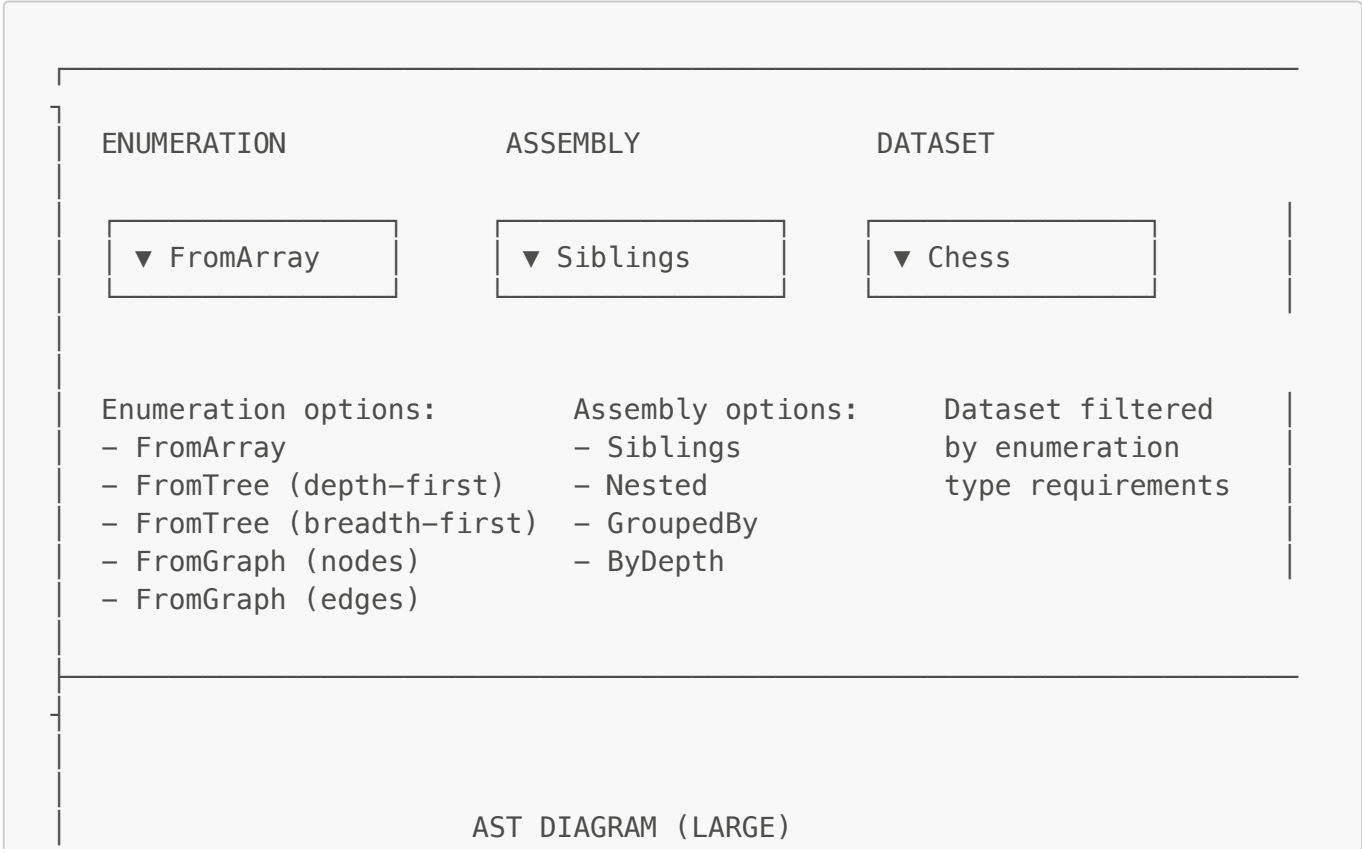
The existing UI has four zones:

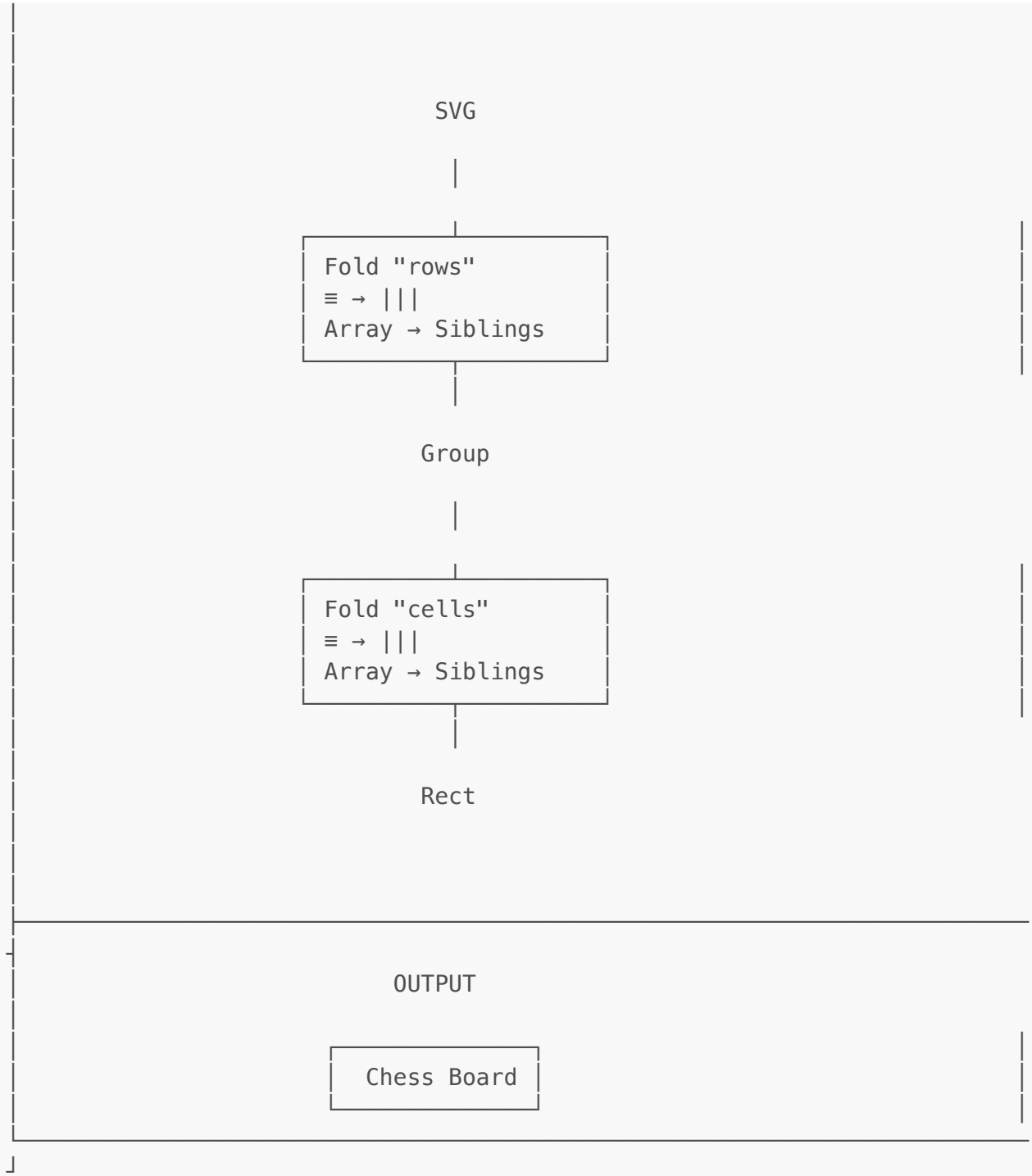
1. **PICK A RECIPE** (left): Grid, Scatter, Bubble, Tree, GUP
2. **PICK A TYPE** (center-left): Shows data types matching the recipe
3. **AST Diagram + Code** (center): Visual tree with type annotations, plus PureScript source
4. **PICK A DATASET** (right): Compatible datasets, incompatible ones grayed out
5. **VIEW OUTPUT** (bottom): Rendered visualization

Problems:

- Three-column picker flow is spread wide, feels like configuration not exploration
- "PICK A TYPE" is redundant—the AST diagram shows types
- Code block competes with the diagram for attention
- Recipe names (Grid, Scatter, etc.) are output-oriented, not input-oriented
- Doesn't show the enumeration × assembly matrix

Proposed Layout





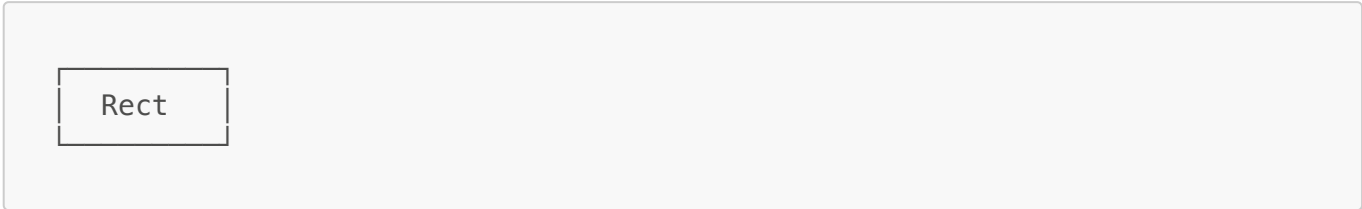
Key changes:

1. **Three dropdowns in one row**: Enumeration, Assembly, Dataset. Compact, scannable.
2. **Enumeration × Assembly as primary controls**: These are the two knobs. Everything else follows.
3. **Dataset picker is conditional**: Only shows datasets compatible with the selected enumeration. The graying-out demonstrates type safety without explanation.
4. **AST diagram dominates**: Gets most vertical space. This is the unique value.
5. **Code hidden by default**: Toggle to show. The diagram should be self-explanatory.
6. **"PICK A TYPE" column removed**: Redundant with type annotations on AST nodes.

Visual Language for HATS Nodes

Elem Nodes (DOM elements)

Simple labeled boxes, as currently:



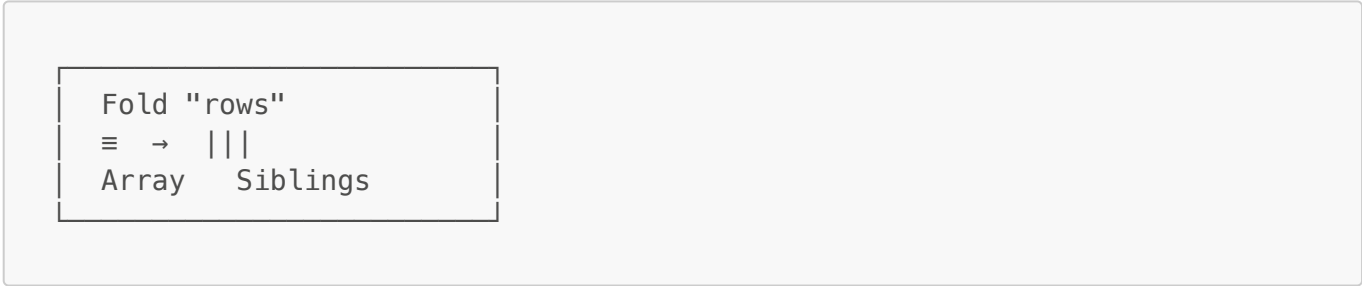
Color/style can indicate element type (SVG vs HTML, container vs leaf).

Fold Nodes (the key innovation)

Fold nodes need to convey:

- **Name**: for selection retrieval
- **Enumeration**: input structure (what shape of data)
- **Assembly**: output structure (what shape of DOM)
- **Template**: the subtree that gets repeated

Option A: Compound glyph with icons



Icon vocabulary:

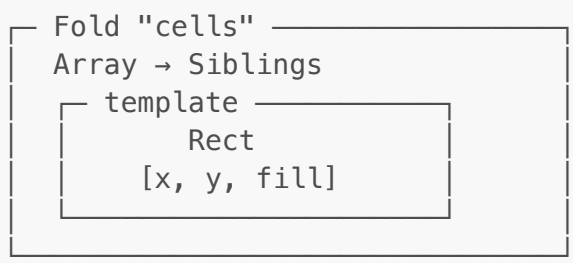
- ≡ or `[...]` = Array
- 🌳 or △ = Tree
- ◉─◉ or ⬡ = Graph
- `|||` = Siblings (flat)
- ⊏⊐ or nested squares = Nested
- ▤ = GroupedBy
- ⋮ = ByDepth

Option B: Shape reflects assembly

Fold nodes with different assemblies have different shapes:

- Siblings: wide/horizontal rectangle
- Nested: tall rectangle with inset
- GroupedBy: rectangle with partition lines
- ByDepth: stepped/tiered shape

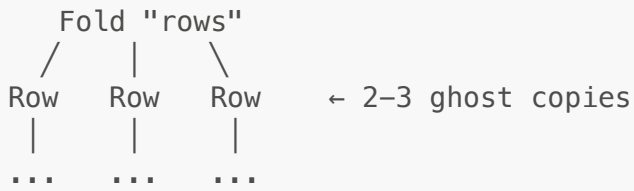
Option C: Show template as inset subtree



The template is `a -> Tree a`—a tree inside the Fold. Drawing it as contained subtree shows the recursion.

Option D: Ghost repetition

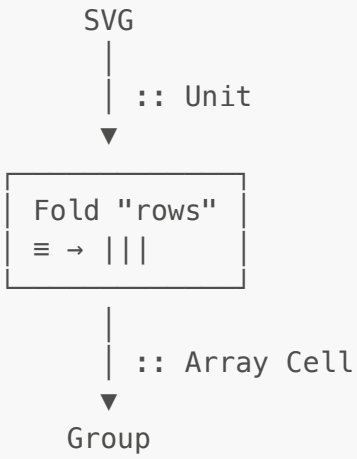
Show faded copies indicating "this repeats N times":



Recommendation: Combine A and C. Use compound glyph for quick scanning, show template as inset for detail. Ghost repetition as optional animation/hover state.

Type Annotations

Show on connecting edges or as node subtitles:



The type flows through the tree. Clicking a node could highlight what type it receives and what it passes down.

Empty Nodes

Minimal or invisible unless explicitly shown:

```
┌ ─ ─ ─ ─ ─ ┐
  Empty        (or just dashed box)
└ ─ ─ ─ ─ ─ ┘
```

Interaction Patterns

1. Change Enumeration

- Dropdown changes from `FromArray` to `FromTree`
- AST diagram animates: Fold nodes update their left glyph
- Dataset picker updates: incompatible datasets gray out
- Output re-renders (if dataset still compatible) or clears (if not)

2. Change Assembly

- Dropdown changes from `Siblings` to `Nested`
- AST diagram animates: Fold nodes update their right glyph
- Fold node shape may change (if using Option B)
- Output re-renders showing different DOM structure

3. Hover on Fold Node

- Expand to show template subtree (if collapsed)
- Show ghost repetition animation
- Highlight corresponding region in output

4. Click on Fold Node

- Select it for detail view
- Show the full Fold specification (name, enumerate, assemble, template, gup)
- Optional: show generated code for just that node

5. Matrix Preview (optional feature)

Thumbnail grid showing all valid enumeration × assembly combinations:

```
              Siblings  Nested  GroupedBy  ByDepth
FromArray        ■        ■        ■         □
FromTree         ■        ■        ■         ■
FromGraph        ■        ■        □         □
```

Click a cell to jump to that combination. □ = invalid/not implemented.

What Stays

- AST diagram as central visualization
- Type annotations on nodes
- Dataset picker with graying for incompatible types
- Output panel showing rendered result
- The chess board and other good example datasets

What Goes

- "PICK A RECIPE" column (replaced by Enumeration dropdown)
- "PICK A TYPE" column (redundant)
- Code block in main view (moved to toggle/detail)
- "Join" / "NestedJoin" terminology (replaced by Fold with enum×assembly)
- Explanatory text about "not a SQL JOIN" (let the UI demonstrate)

Implementation Notes

AST Changes

The visualization code needs to use HATS `Fold` instead of `Join`/`NestedJoin`. The MetaAST interpreter needs to:

- Extract enumeration and assembly from Fold nodes
- Render them with the new visual language
- Support the icon vocabulary

Datasets

Existing datasets should map to enumeration types:

- Chess, Sudoku, Scrabble → Array (Array a) — nested arrays
- Force Graph → Graph
- Tree datasets → Tree
- Flat arrays → Array

May need to add more tree/graph datasets to show those enumeration paths.

Incremental Path

1. Replace terminology: Join → Fold in the UI
2. Add Assembly dropdown (currently implicit)
3. Redesign Fold node rendering with compound glyph
4. Collapse layout to single row of dropdowns
5. Remove redundant columns
6. Add matrix preview (optional, later)

Success Criteria

- User can change enumeration and assembly independently
- The N×M matrix is evident from the UI
- Type safety is demonstrated by dataset graying, not explained by text

- AST diagram clearly shows both dimensions of each Fold
- Feels like exploration, not configuration

Open Questions

1. Should the template subtree be shown inline or on hover/click?
2. How prominent should type annotations be? Always visible or on hover?
3. Should we show the "dataflow strip" (input → unfold → template → fold → output)?
4. Naming: "AST Builder", "Tree Builder", "Fold Explorer", "Hylograph Playground"?

Status / Next Steps

- [ ] Sketch UI mockups for the new layout
- [ ] Design icon vocabulary for enumeration and assembly types
- [ ] Update MetaAST to use HATS Fold
- [ ] Implement new Fold node rendering
- [ ] Migrate existing datasets to enumeration type annotations
- [ ] User testing to validate "exploration not configuration" feel