Mathematical Explanation of the Recursive Hierarchical Reordering Algorithm

The goal of the **RecursiveHierarchicalRuleComposition** is to compute a permutation of the variable (and constraint) indices that reflects a hierarchical (block-wise) ordering of the matrix. The ordering is computed by recursively partitioning the matrix into “blocks” based on one or more block rules, and—if available—refining the order within each block using intra rules. Mathematically, the algorithm is a divide-and-conquer procedure that constructs a permutation σ (for variables) and a corresponding ordering for constraints.

1. Problem Setup

Let:

- **V** be the set of variable indices, $V = \{0, 1, \ldots, n-1\}$,
- **C** be the set of constraint (row) indices, $C = \{0, 1, \ldots, m-1\}$,
- **A** be the matrix (or problem data) representing the relationships between variables and constraints.

Our goal is to find an ordering (permutation) of the variables (and similarly for constraints) that “groups together” indices with similar structural properties.

2. Block Rules and Partitioning

A **block rule** R is a function that, given a subset of variable indices $V' \subseteq V$ and constraint indices $C' \subseteq C$, returns a partition of the block (V', C'). Formally,

$$R(V', C') = \{(V_1, C_1), (V_2, C_2), \ldots, (V_k, C_k)\}$$

where the sets V_i are disjoint (and their union is V') and similarly for the constraint sets C_i. In our implementation, block rules are realized by methods such as `score_matrix` in each rule class (e.g., in the **BoundCategoryRule**, **CardinalityRule**, or **SignPatternRule**).

Partitioning Criterion

A block rule may, for example, group variables by:

- **Cardinality:** Variables with the same number (or range) of nonzero coefficients are grouped together.
- **Sign Pattern:** Variables whose columns have the same sign pattern (all nonnegative, all nonpositive, or mixed) are grouped together.
- **Bound Category:** Variables are grouped by the "category" of their bounds (both finite and nonnegative/nonpositive, straddling zero, one bound infinite, or both infinite).

Mathematically, if the rule R uses a scoring function $f : V' \to \mathbb{Z}$ (or more generally, $f : V' \to \mathbb{R}$), then one way to partition is to define

$$V_i = \{v \in V' : f(v) = s_i\},$$

with the corresponding constraint group often being the entire C' (if the rule does not partition constraints). The partition is then the collection of pairs

$$\{(V_i, C') : s_i \text{ is a unique score}\}.$$

3. Recursive Partitioning

The algorithm defines a function

$$F(V', C'; \mathcal{R}_p, \mathcal{R}_c, \mathcal{I})$$

that returns an ordering of the indices in V' and C', where:

- $\mathcal{R}_p$ is an ordered list of **parent block rules**.
- $\mathcal{R}_c$ is an ordered list of **child block rules**.
- $\mathcal{I}$ is an ordered list of **intra rules** (for ordering within blocks).

The Recursive Definition

1. **Base Case:**
   If the recursion depth reaches a maximum L (to avoid infinite recursion) or if no rule from either $\mathcal{R}_p$ or $\mathcal{R}_c$ produces an effective partition (i.e. every rule returns only a single block), then the algorithm applies the intra rules:

- **Without intra rules:** The ordering is the identity (i.e., the indices remain in their given order).
- **With intra rules:** Each index v is assigned a score tuple

$$\text{score}(v) = \big(g_1(v), g_2(v), \ldots, g_k(v)\big)$$

where each g_i is the score produced by an intra rule. The ordering is then obtained by sorting V' lexicographically by these tuples.

2. **Recursive Case:**

To partition a block (V', C'), the algorithm proceeds as follows:

- **Parent Rule Phase:**
  The algorithm iterates over the parent's block rules $\mathcal{R}_p$ (in order) and applies each rule R until one produces an effective partition; that is,

$$R(V', C') = \{(V_1, C_1), (V_2, C_2), \ldots, (V_k, C_k)\}$$

  with $k > 1$. If such a rule is found, it is used to partition the block.
- **Child Rule Phase:**
  If none of the parent's rules yields an effective partition (i.e. all return a single block), then the algorithm switches to the child's block rules $\mathcal{R}_c$.
  In the child phase, the rules are **rotated** between recursive calls so that the same child rule is not applied twice consecutively. That is, if the child rules are initially $[R_1, R_2, \ldots, R_t]$ and a rule R_1 is applied without success, then the list is rotated to $[R_2, \ldots, R_t, R_1]$ and the process is repeated.
- **Recursion on Subblocks:**
  Once an effective partition is obtained, let the partition be

$$\{(V_1, C_1), (V_2, C_2), \ldots, (V_k, C_k)\}.$$

  Each subblock (V_i, C_i) is then ordered recursively using the same procedure. Notably, if a parent's rule produced the partition, the parent's block rule list $\mathcal{R}_p$ is restarted (i.e. the parent's rules remain available for subblocks). If the partition was produced by a child's rule, the rotated child rule list is used for the recursive calls.
- **Concatenation:**
  The final ordering for (V', C') is obtained by concatenating the orderings from each subblock. Formally, if the recursively computed orderings are $\sigma_1, \ldots, \sigma_k$ for the subblocks, then

$$\sigma(V', C') = \sigma_1 \,\|\, \sigma_2 \,\|\, \cdots \,\|\, \sigma_k,$$

where “$\|$” denotes concatenation.

4. Intra Rules and Lexicographic Ordering

If intra rules are provided, each index i in a block is assigned a tuple of scores

$$\text{score}(i) = (g_1(i), g_2(i), \ldots, g_p(i))$$

by calling methods such as `score_matrix_for_variable` (or `score_matrix_for_constraint`). The lexicographic ordering is defined by:

- i precedes j if there exists an index k such that:
  - $g_l(i) = g_l(j)$ for all $l < k$, and
  - $g_k(i) < g_k(j)$.

In the absence of intra rules, the algorithm returns the indices in the order they are passed (i.e. the identity ordering within the block).

5. Final Ordering and Scoring

Once the full ordering is computed, the final step in the algorithm assigns each variable (or constraint) a “score” equal to its position in the ordering. That is, if the computed permutation is

$$\sigma = (\sigma(0), \sigma(1), \ldots, \sigma(n-1)),$$

then the final score for variable v is defined as the index k such that $\sigma(k) = v$.

Summary

- **Partitioning:**
  The algorithm partitions the matrix using block rules R that group indices according to structural properties (e.g., cardinality, sign pattern, bounds).
- **Parent and Child Phases:**
  The algorithm first applies the parent's block rules to a block and uses the first rule that produces

a non-trivial partition. If none of the parent's rules is effective, the algorithm switches to the child's block rules, rotating among them so that the same rule is not used consecutively. If all rules (both parent's and child's) yield only a single block, the recursion stops for that block.

- **Recursion and Concatenation:**
  The algorithm recursively orders each subblock and concatenates the resulting orderings to form the final ordering.
- **Intra Rules and Lexicographic Ordering:**
  Intra rules are used to refine the order within a block when no block rule produces a partition. The indices are sorted lexicographically by the tuple of intra scores.
- **Final Score:**
  The final score for each variable (or constraint) is its position in the overall ordering.

This recursive hierarchical approach is a divide-and-conquer method that exposes and exploits the structure of the problem matrix. The underlying mathematical operations are based on partitioning (by equivalence classes defined by block rules), lexicographic ordering (via intra rules), and concatenation of ordered subblocks.