Confluent Hasse Diagrams

David Eppstein and Joseph A. Simons

University of California, Irvine
Goal:

- Improve Readability of Hasse Diagrams
  - Remove Crossings
  - Replace with confluent junctions
Hasse Diagram

- Transitive reduction of DAG (Partial Order)
- All edges oriented upwards
Hasse Diagram - Planar?

- Special Case of Upward Planarity Testing
  - Restricted to transitively reduced DAGs
- U.P. Testing is NP-Complete in general
  - Efficient if single source or sink

Planar, but not \textit{Upward} Planar
DAG $\iff$ Partial Order

- Reachability in DAG defines partial order.
- $u \rightarrow v$ in DAG iff $u < v$ in partial order.
Dedekind-MacNeille Completion

Partial order $P$

Insert so that

*every pair* of nodes gets a unique LUB and GLB
Dedekind-MacNeille Completion

Partial order $P$

Smallest lattice containing $P$

Insert \( \Box \) so that

every pair of nodes gets a unique LUB and GLB
Confluent Drawing

- Draw non-planar diagrams without crossings

- Merge together groups of edges as tracks

Not \textit{Upward Planar}

\textit{Upward Confluent}
Confluent Drawing

• Tracks meet smoothly at *junctions*

• Nodes connected $\iff$ smooth path between them.
Creating Junctions

- Use *Dedekind-MacNeille completion* to find junctions

- Every *pair* of nodes gets a unique LUB and GLB
Algorithm - Input

• Input: *two-dimensional* partial order

Realized by two total orders

\[ L_1: a \ b \ c \ d \ e \ f \ g \]

\[ L_2: b \ a \ f \ d \ c \ g \ e \]
Two-Dimensional Partial Order

Inversion $\iff$ Incomparable
Algorithm - Output

- Output: Upward Confluent Drawing

Embedded in $2n + 1 \times 2n + 1$ grid

In $O(n^2)$ time
Algorithm - Completion

Insert Junctions

- Every pair gets unique LUB and GLB
Iterate over grid with simple rule:

For odd indices \( i, j \)

if for nearby nodes:

\[
\begin{align*}
    x &= i - 1 \quad \Rightarrow \quad y < j - 1 \\
    x &= i + 1 \quad \Rightarrow \quad y > j + 1 \\
    y &= j - 1 \quad \Rightarrow \quad x < i - 1 \\
    y &= j + 1 \quad \Rightarrow \quad x > i + 1
\end{align*}
\]

then insert junction

Also insert at \((1, 1)\) and \((2n + 1, 2n + 1)\)
Algorithm - Completion

Insert Junctions

Every pair gets unique LUB and GLB

<table>
<thead>
<tr>
<th>e</th>
<th>f</th>
</tr>
</thead>
<tbody>
<tr>
<td>g</td>
<td>d</td>
</tr>
<tr>
<td>c</td>
<td>a</td>
</tr>
<tr>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

$L_1$: $a$, $b$, $c$, $d$, $e$, $f$, $g$

$L_2$: $a$, $b$, $c$, $d$, $e$, $f$, $g$
Algorithm - Completion

Insert Edges
Algorithm - Finish

Draw Confluent
Worst Case: $O(n^2)$ junctions required
Traditional Diagram: $O(n^4)$ crossings
Confluent Diagram
Hasse Diagram

Random 100 node poset
Random 100 node poset
Series-Parallel

Special case of two-dimensional partial orders

Simple drawing algorithm

Recursive, linear time
Series-Parallel

Only add junctions on Series Composition and only when crossings created.
Series-Parallel

Partial Order

Decomposition Tree
Traverse Tree to construct Confluent Diagram

Grid Embedding

Decomposition Tree

![Diagram of a Decomposition Tree with nodes labeled S and P, and colored circles at the leaf nodes.](attachment:decomposition_diagram.png)
Traverse Tree to construct Confluent Diagram

Grid Embedding

Decomposition Tree
Traverse Tree to construct Confluent Diagram

Grid Embedding

Decomposition Tree
Traverse Tree to construct Confluent Diagram

Grid Embedding

Decomposition Tree
Traverse Tree to construct Confluent Diagram

Grid Embedding

Decomposition Tree
Traverse Tree to construct Confluent Diagram

Grid Embedding

Decomposition Tree
Series-Parallel

Hasse Diagram

Random
100 node
Series-Parallel poset
Series-Parallel

Confluent Diagram

Random
100 node
Series-Parallel
poset
Conclusion

Simple, worst-case optimal algorithms
Dramatically reduce visual clutter
Remove crossings, Insert Junctions

Future Work

Experimental tests:
How many crossings removed?
How much ink saved?

What if we can’t eliminate crossings?
Can we still minimize crossings?