

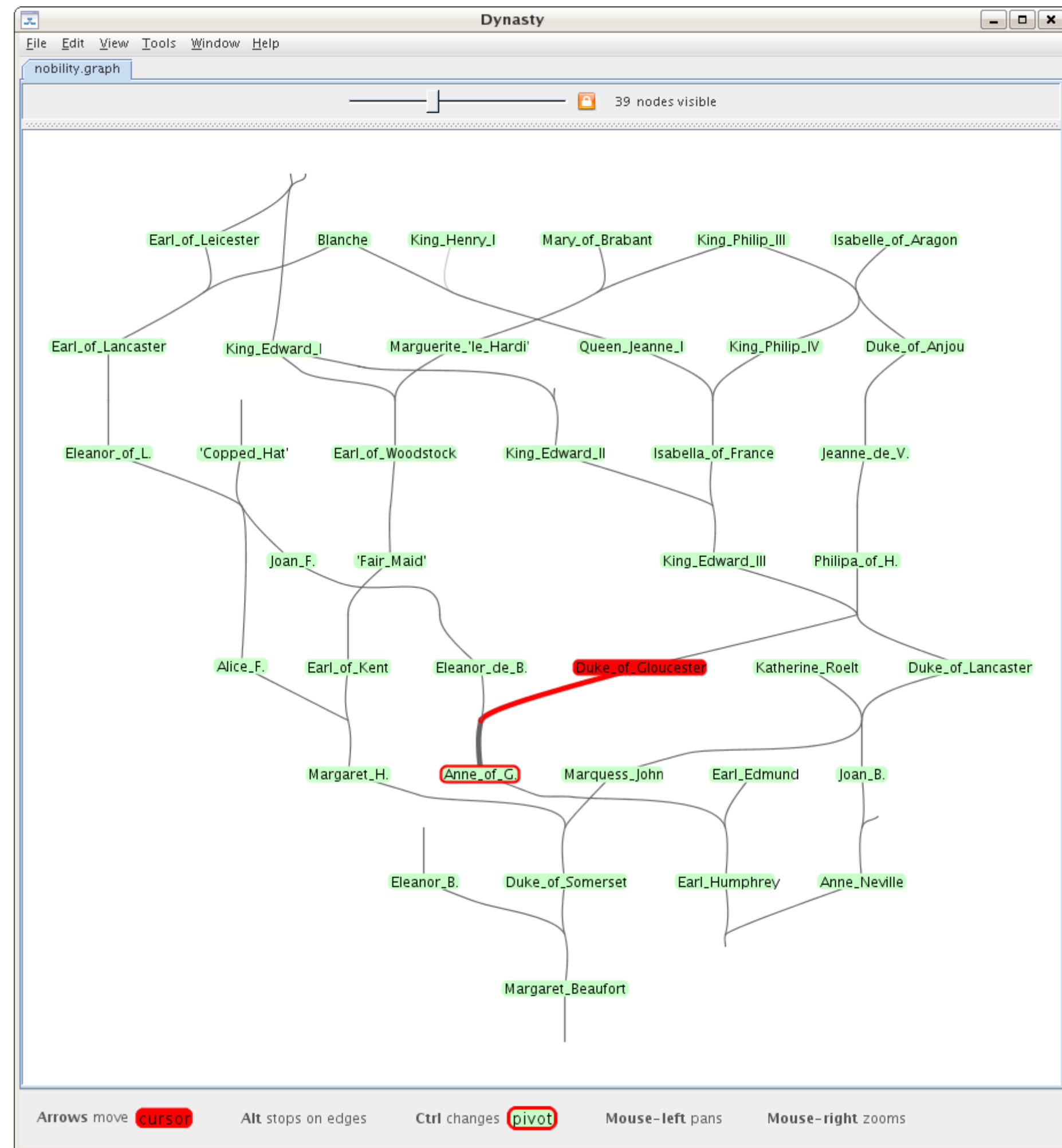
Visual Navigation Through Large Directed Graphs and Hypergraphs

Jason Eisner Michael Kornbluh Gordon Woodhull Raymond Buse Samuel Huang Constantinos Michael George Shafer

Need

Simple, intuitive utility to look around locally in large *directed* graphs.

Surprisingly, we couldn't find one. Had to write one.



Key layout requirements

Lay out only a small local subgraph.
Readable exploration, not global structure.
Change subgraph as you move into new territory.

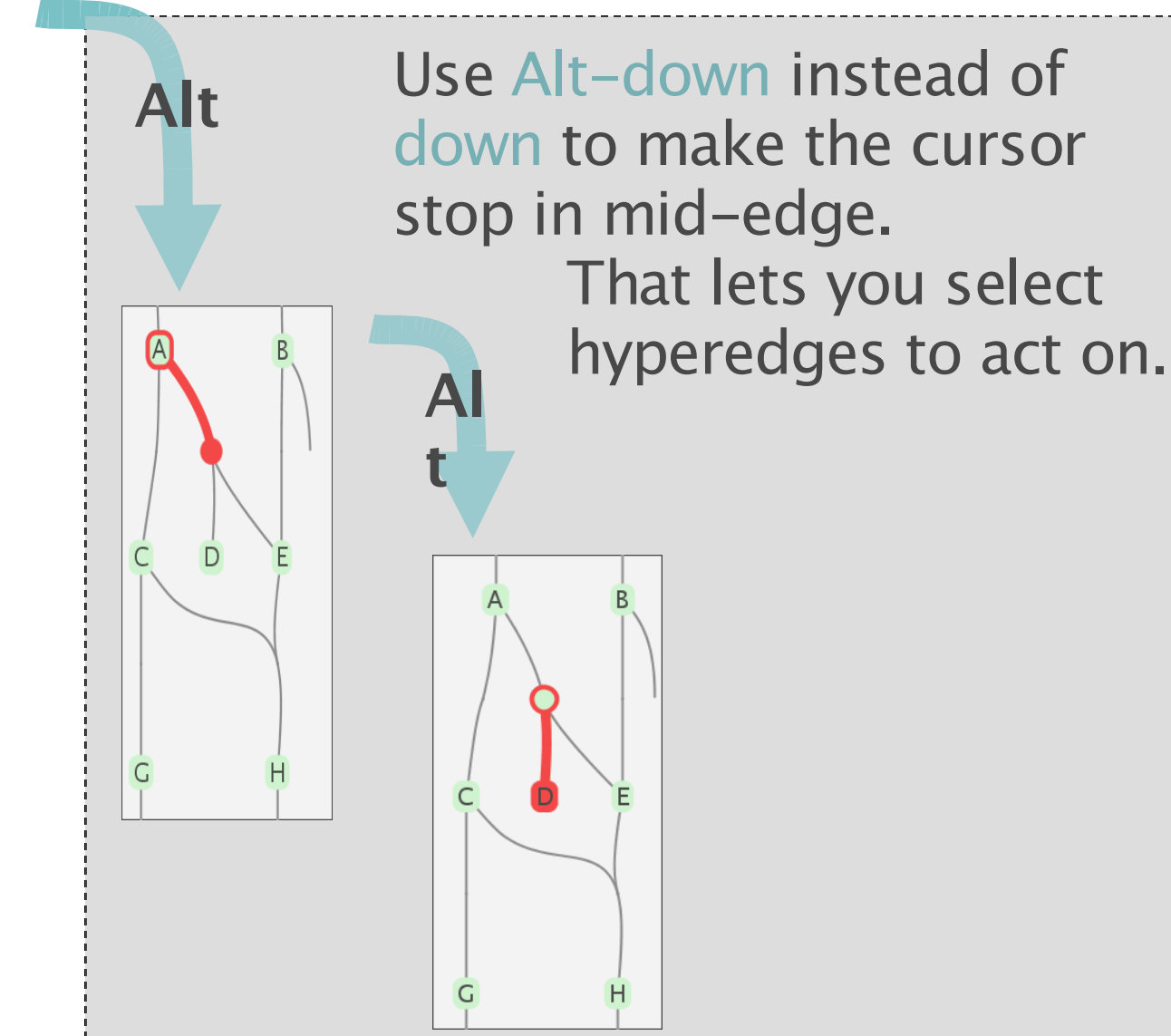
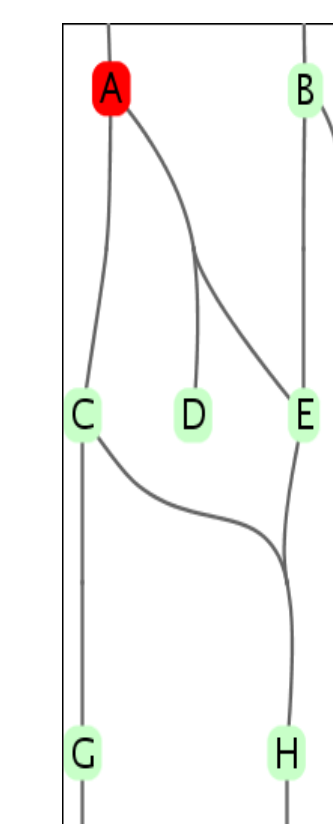
But not force-directed layout adjustment.
Dynamic version of Sugiyama-style algorithm.
Directed edges always start & end flowing down.
Hypergraph edges merge inputs & split outputs.
Preserve sibling order if semantically meaningful.

Responsive and stable.
Animate first toward quick layout.
Change course when final layout arrives.
Don't change layout too much: favor stability.
Smooth animation and fade to help visual tracking.

Topological keyboard navigation

Use arrow keys to let user find the paths & hyperpaths through the tangles.

Cursor is on node **A**.
Where should the *down* arrow go?



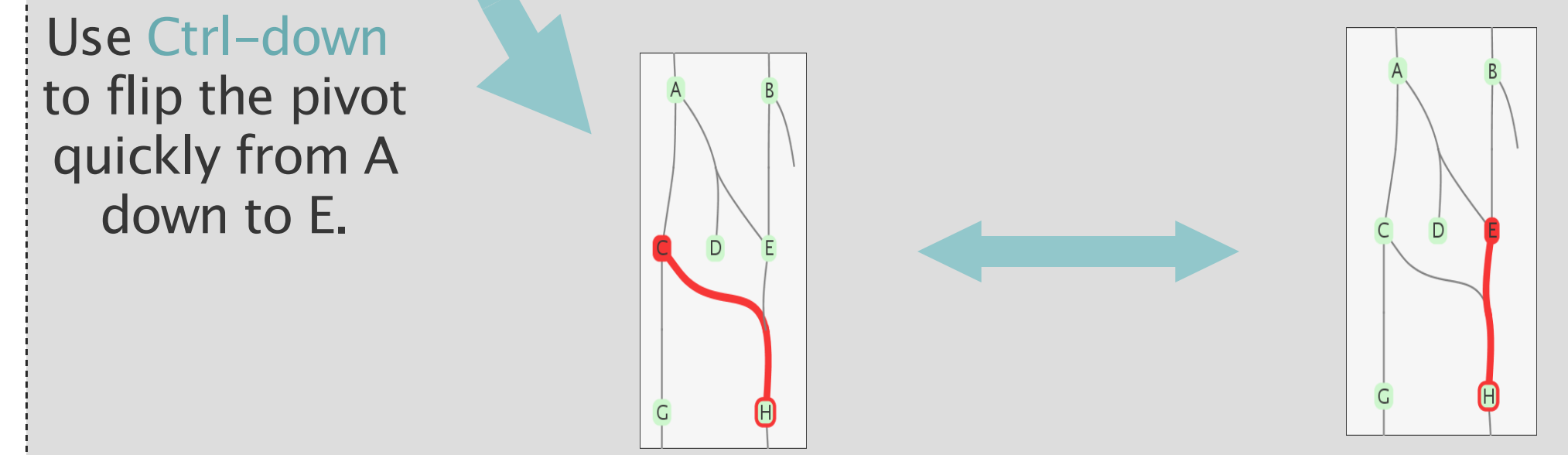
Down arrow picks any red path down from A.
(Prefers short, vertical paths).

Red trail shows where you just came from.

Stability via *trailblazing*:
This edge will be remembered for future as the currently preferred way to go down from A... or back up from C.

Ctrl Left and right step through the other paths down from the “pivot” A

Easily highlight all of A's children.



Here, left and right run through the pivot's parents instead of its children (since the pivot E is below the cursor).

Pivot A is all used up and vanishes: no more children to the right.

Dynasty guesses a new pivot B that does support right arrow.

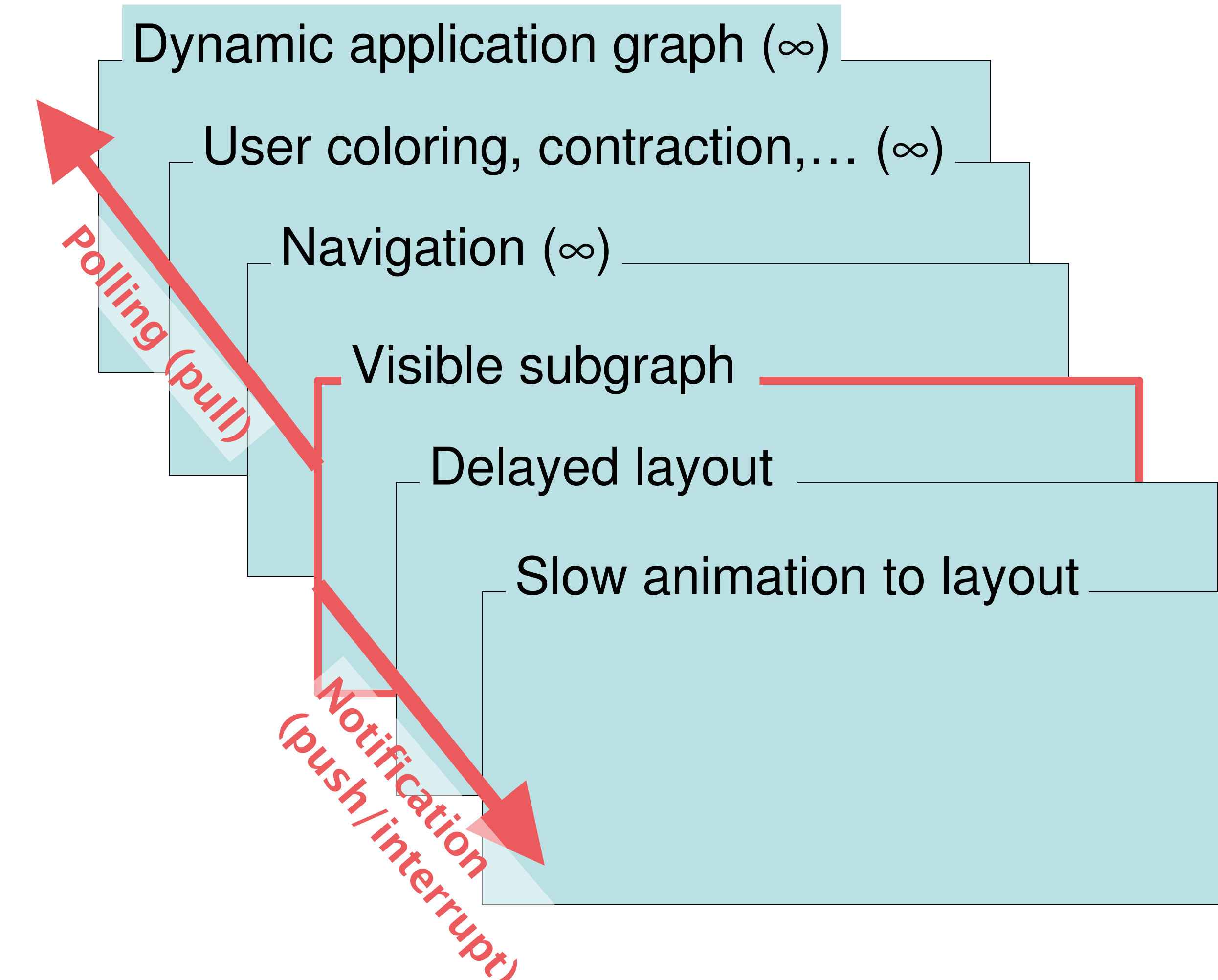
Right once more brings the cursor to unseen node F, which is revealed.

We also reveal F's neighbors (K) and their connections to existing nodes (H).

As they all fade in, faraway nodes (G) fade out to keep the graph small.

Any relayout is guaranteed to preserve the order of the trail and other edges where they fan out from the pivot B, so that left/right traverse a stable order.

Asynchronous dynamic layers



Coming soon ...

Improved selection and search
More actions on selected nodes/edges
Merge into supernode
Contract hypergraph across
Coloration, fonts, thresholding, etc.
Show relationship paths, etc.

Better support high- or ∞-degree nodes (already handles infinite graphs)
Let layout pick which nodes to prune

Integration as Dyna debugger

Why browse large hypergraphs?

Declarative or omniscient debugging

We built a declarative programming language (Dyna). Tracing execution would be confusing & irrelevant – the order of computations is up to the compiler. Instead, we wanted debugging to let you explore **where a value came from and how it was used.**

Producer-consumer networks

Family trees
Digital circuits
Chemical reactions
Flow of manufacturing materials
Proof forests (from theorem provers)
Parse forests (from natural-language parsers)
Multiple inputs combine into reusable output(s).

Ordinary large graphs

Ever wanted to use dot on a large or dense graph? Trees or near-trees, social and physical networks, data structure layout, finite-state automata, call graphs ...
Dynasty is “browseable dot.”

IEEE Symposium on Information Visualization, October 2006.
This work has been supported in part by NSF ITR grant IIS-0313193 (“Weighted Dynamic Programming for Statistical Natural Language Processing”) to the first author and by Joseph C. Pisirito Research Fellowships to the second and fifth authors. The views expressed are the authors' only. We also thank Jeffrey Heer for the preface lookit.