Routing Is Not Adjudication

Flyxion
Independent Researcher

30 September 2026

Abstract

Any shared knowledge system decides two different things: what a given reader should see, and what currently stands. The first is routing and the second is adjudication. They answer different questions and fail in different ways, yet systems routinely let the output of one stand in for the output of the other, so that what is prominent is taken to be what is supported and what is hidden is taken to have been refuted. This essay separates the functions, constructs a small failure in which a display computed from a routed subset reports the wrong standing in both directions, and states a sufficient condition under which a view is faithful. The result is a design rule with a cheap marker: status is computed on the full active set, routing may use anything and may not write to the labeling, and any display whose view omits part of what bears on it says so. The formal statements are proved in the monograph *Adversaria: A Specification for a Public Claim Graph*; this essay gives them in words with one worked example.

1 Two questions

A reader arrives at a shared record with a finite amount of attention. Something must decide which of the many items will be put in front of them and in what order. That is a real and unavoidable task, and doing it well is valuable. Separately, something must decide which contributions stand: which claims are supported at present, which have been defeated, and which are unresolved. That is also a real task, and it has its own standards.

The first question is *what should reach this person?* The second is *what stands?* A good answer to the first is judged by whether material reaches those to whom it is useful. A good answer to the second is judged by whether standing follows the arguments. One can be excellent while the other is poor. A system can route exactly the right material to exactly the right readers and have no defensible account of what is supported. A system can compute standing impeccably and have no way to get any of it in front of anyone.

The error this essay is about is not bad routing and not bad adjudication. It is letting one function's output serve as the other's. When routing output is read as adjudication output, prominence becomes a proxy for standing and absence becomes a proxy for rejection. Both readings are natural, both are usually unintended by whoever built the system, and both are wrong in ways that can be shown exactly.

2 Four functions

A shared knowledge system performs four functions, which any implementation may merge or separate.

function	question it answers	may depend on
storage	what has been recorded, and is it intact	content, references
routing	what should reach this reader, in what order	reader context, the record, anything about relevance, including labels
admission	does this record take part in evaluation on this surface	the record, the records it references, the policy, who is authorized
evaluation	what stands, and where	the set of admitted records and the policy, nothing else

The last column carries the argument. Evaluation is allowed to depend on very little: the admitted records and the declared policy. Routing is allowed to depend on anything, including the output of evaluation. That asymmetry is deliberate. Relevance is a matter of context and taste, and a router that could not use context would be poor. Standing is a matter of what has been argued and under what rules, and an evaluator that could use context, such as who is looking or what is popular, would no longer be computing standing.

Call the set of admitted records under a given policy the *active set*. Routing takes a reader and produces a *view*: a subset of the active set together with an order. The view is what the reader sees. The active set is what bears on the question.

3 Why the error is tempting

Three pressures push toward letting the view stand in for the active set.

The first is economy. A reader who sees a view has no cheap way to know what was left out, and a system that shows a status next to an item has the quick option of computing that status from whatever is at hand, which is the view. The second is proxy drift. Routing is often tuned on signals such as engagement, time spent or agreement, and these correlate with the thing the reader wants, namely material that is good and relevant, well enough that the correlation is mistaken for identity. The third is presentation: a ranked list looks like a verdict. Position one reads as "best" whether or not the list was built to say so.

None of these requires bad faith, and that is why the error is durable. A system can be built by careful people and still compute a label from a view simply because the view was what was loaded.

4 A constructed failure

The following example is invented, and no real system is described. A shared record holds a claim c: *a reflective roof coating lowers indoor temperature in summer in a dry climate.* Three further items bear on it.

- a is an argument for c: a field comparison of coated and uncoated roofs, with the warrant that such a comparison supports a claim of this kind over the conditions sampled.
- u is an objection to a: a contributor argues that the comparison did not control for roof insulation, so the warrant does not apply to the evidence. This is an undercutter: it attacks the inference and not the data.
- v is an objection to u: another contributor shows that the insulation records u relies on come from the wrong building set, so u's grounds are unsound. This undermines u.

The defeat chain is v against u, and u against a. On the full active set, v has no defeater and stands; hence u does not stand; hence a is undefeated and stands. The argument for c is in good shape, and the reason is three steps deep.

Now consider four views a router might produce, and what a display would report if it computed status from the view alone.

view	status shown for a	true status of a	what has happened
$\{a, u, v\}$	IN	IN	faithful: the whole chain is present
$\{a\}$	IN	IN	correct, but by accident: nothing attacks a in this view
$\{a, u\}$	OUT	IN	wrong: u looks undisputed, so a looks defeated
$\{u, v\}$	(no claim shown)	n/a	u shown as defeated with no indication of what it was against

The third row is the instructive one. It is easy to imagine how it arises. The objection u is lively and attracts replies, so a router tuned to engagement puts it in front of readers. The reply v is technical and quiet, so it is ranked below the fold. A reader with the view $\{a, u\}$ sees an argument with an unanswered objection. A status computed on that view says the argument has been defeated, and it has not.

Now reverse the situation. Suppose instead that the objection to a is sound and unanswered, call it b, and that b is a rebuttal which the policy prefers. The true status of a is then OUT. A view containing only a, because b was routed to a different audience, shows a as IN. The display is wrong in the other direction. One constructed world thus produces a wrongly defeated argument and a wrongly supported one, depending only on which items a router happened to include.

5 What the model proves

The monograph states the exact content of these observations.

Faithfulness condition. For an argument a at a unit x of its scope, let its *backward closure* be the set of arguments that can affect its label at x: its defeaters, their defeaters, and so on. If a view contains the whole backward closure of a, then the label computed from the view equals the label computed on the full active set. This is a sufficient structural guarantee and not a necessary condition. A view can omit part of the closure and still agree with the full labeling: the view $\{a\}$ of Section 4 gives the correct label for a although it omits u and v, because the omitted attack and counterattack happen to cancel. Closure completeness guarantees agreement without relying on such cancellation. It is a short result, since labels depend only on defeaters and their defeaters, but it is the one the design rule rests on.

Under stated assumptions, every non-standing argument can be made to look standing. If a does not stand at x on the full active set, then there is a view containing a on which it does stand, provided a does not attack itself and the policy imposes no other eligibility condition that would prevent its acceptance. Under those assumptions the view $\{a\}$ works, because an argument with no attackers in the view is undefeated there. If self-attacks are permitted, $\{a\}$ need not make a stand. This result says nothing about any particular router. It says that the set of possible views is large enough to contain, for each failing argument, one in which the failure is invisible. A router is therefore not constrained by anything in the model to avoid such views. Whether it does so is a matter of how it was built, and the worked example suggests that an engagement-tuned router may do so without anyone intending it.

Dependence runs one way. Define a function to *depend on* another if its output can change when the other's output changes while its remaining inputs are held fixed. Then evaluation depends on admission and on nothing else. Admission depends on storage and on authorization. Routing may depend on all three. Storage, admission and evaluation do not depend on routing. The dependence relation among the four functions has no cycle.

The consequence is stated in the monograph in one sentence: routing, which is not an input to the labeling, cannot change any label. A system that appears to have routing change labels must have let a display compute labels from the view, which is precisely the error above.

6 The rule and the marker

The design rule that follows has two halves.

Status is computed on the active set. The status shown for an argument or claim at a unit is the label computed on the full active set under the surface's policy. Routing selects and orders what is shown and does not enter the computation.

A view says when it is partial. When some member of the backward closure of a displayed item is not in the reader's view, the display marks the status as *not closure-complete* and states how many members of the closure are hidden.

The first half guarantees that no routing choice can change a status. The second half handles the residual honesty problem: the reader still sees a subset, and the subset may omit something that matters. The marker does not force the reader to look at what is hidden and does not say what it is. It says that something bearing on this status is not on screen, and how much.

Once the relevant closure is available, the marker is cheap. The backward closure is part of the evidence a system must retain to justify a label at all, so, where closures are retained, checking whether the closure is a subset of the view is an inclusion test and needs no new computation. A display that cannot afford the test can always err toward marking.

Applied to the worked example, the rule changes the third row. On the view $\{a, u\}$, the display shows a as IN, its true status, and marks it as not closure-complete with one hidden member, v. The reader sees an objection u and sees that the argument still stands, with a flag that part of the story is off screen. A reader who cares can follow it. A reader who does not has still not been told something false.

7 What each architecture is for

The separation is not a judgment that routing is second-rate. Attention is scarce, material is plentiful, and a record that delivered everything to everyone in arbitrary order would be useless. A router can reasonably use prerequisites, reader interest, novelty and load. It can also use labels: showing first the arguments that stand, or surfacing the unresolved defeaters of a claim that a reader is following, is good routing and depends on standing without altering it.

The table below states the division.

routing architecture	adjudication architecture
allocates attention among readers and workspaces	assigns standing to contributions
succeeds when material reaches those to whom it is useful	succeeds when standing follows the arguments
fails by sending the wrong thing, or nothing	fails by counting what should not count, or ignoring what should
output: a subset and an order	output: a labeling and a status field
legitimately uses prerequisites, interest, novelty, load	legitimately uses records, warrants, objections, a declared policy

Two inferences are tempting and neither is licensed. Relevance does not establish support: a claim routed to a thousand readers is not thereby better supported than one routed to ten. Support does not establish relevance: a claim that stands may be routed to no one. A system can route well and be dangerous because the prominence of what it routes may be mistaken for the standing of what it contains. The specification gives routing a place, the surface, and a rule, and does not let it write to the labeling.

8 Objections and replies

"A reader cannot see everything, so a marker that says they did not is noise." Partly true, and the reply is in what the marker says. A bare flag on everything would be noise. The marker appears only when the hidden material is in the backward closure of the displayed item, meaning it actually bears on the status shown. It is an expectation, and not a result of the model, that many items in many views will have complete closures. Where they do not, the count tells the reader how much is hidden, so a reader can tell the difference between one missing reply and a dozen missing objections.

"Hiding things is unavoidable, so why insist on this?" Hiding is unavoidable. Misreporting standing on the basis of what was hidden is not. The rule leaves routing free to omit as much as it likes and constrains only the claim that the display makes about standing.

"Engagement predicts quality, so a view built on it is a decent proxy." Engagement may predict interest, and in some settings it correlates with quality. The model does not deny the correlation. It notes only that a proxy that is mostly right is wrong exactly where it matters most: contested items that draw engagement without resolution, and quiet replies that settle a dispute without drawing any. The worked example is an instance of the second case.

"The closure could be huge." It could. Closures are taken per unit of a claim's scope and include only arguments that affect the label, which may bound them in practice, but their size is an empirical matter and not something the model settles. A display may summarize a large closure, for example by counting and by showing the nearest unresolved defeaters, provided the marker stays accurate.

9 What this essay does not show

It does not show that any deployed system computes status from a routed view. The example is constructed, and the two failures it exhibits are consequences of a definition, not observations. It does not show that routers are badly designed or that engagement is a poor signal; it shows only that neither can be the input to a statement about what stands. It

does not claim that the marker is sufficient for a reader to understand a dispute: a reader can ignore it, and a summary of a large closure can mislead in its own ways. It does not address who controls the routing function or the policy, which are questions of authority taken up in a separate essay, and it does not claim the active set is itself neutral, since admission depends on authorization and on the policy in force.

What it establishes is narrow and exact. A view that contains the backward closure of what it displays is guaranteed to be faithful; a view that omits part of it may agree or disagree, and a display cannot tell which without the closure. Under the stated assumptions, every non-standing argument has some view in which it appears to stand. Because routing depends on evaluation and never the reverse, there is no cycle to hide a mistake in. A system that follows the two-part rule can route as freely as it likes without changing what the record says stands.

Notes

The formal results are in *Adversaria: A Specification for a Public Claim Graph*: the four functions, local evaluation on a view, the counterexample of a hidden objection, the design rule for status and the marker, and one-way dependence, all in Chapter 13; labels and backward closure in Chapter 7; the active set and admission in Chapters 9 and 14; authorization in Chapter 16.