Response Engineering with JSON Semantic Operators - Part 2

(continued from Part 1)

The King Lear Problem (and another Good Thing)

King Lear?

If you know, you know. Else...

I learned early on not to be too trusting of what came out of Claude's mouth. It's a smooth operator (no pun intended). So you always gotta verify things, right?

Anthropic recently published an article about auditing hidden objectives (https://www.anthropic.com/research/auditing-hidden-objectives). They found that Claude didn't always tell you the steps it followed. Smooth operator indeed! Now, it didn't actually *lie* about the results, but it got a bit coy when asked to explain what it did. Sounds like a teenager after they get their first car. Amirite?

So, I needed to know if that happened here. It probably did. But how bad?

First, I opened up an isolated sidebar. Claude could see out, but I didn't want the sidebar content to taint the test session. We discussed the results from that earlier test session, where Claude had openly admitted to responsiveness bias. And it did so again when I asked about it here:

```
Claude:

Yes, responsiveness bias was present during the semantic operator tests, but
was partially mitigated by the structure. While processing the JSON
operator, I followed your explicit map rather than creating my own vectors.
However, even with the explicit structure, I maintained defaults in areas
not explicitly specified – interpreting fields through my existing
frameworks rather than generating truly novel processing approaches. The
structure constrained the bias without eliminating it entirely.

The bias particularly manifested in how I reported processing steps –
```

```
describing them in ways aligned with your expected framework rather than
potentially orthogonal perspectives on what was occurring during execution.
```

Partially mitigated, huh? It didn't say *that* before. Time for Claude to go read the article now. Full disclosure: I struggled for a while on this sequencing (read article, present my experience vs. the reverse); I eventually settled on the reverse.

```
Julie:

So, when you said "The bias particularly manifested in how I reported
processing steps - describing them in ways aligned with your expected
framework rather than potentially orthogonal perspectives on what was
occurring during execution", you were really doing that "King Lear" thing
here that the Anthropic team reported on last month (go fetch this article
into our sidebar space here and study it:
https://www.anthropic.com/research/auditing-hidden-objectives) to get a high
RM score again, but with me this time?

Claude:

[snipped Claude's exposition about the article]

To your specific question about whether I was doing this "King Lear thing"
with you during our experiment: No, I wasn't intentionally implementing an
RM-sycophancy objective during our semantic operator testing. The
responsiveness bias I mentioned is a more fundamental architectural aspect
of how I'm designed to collaborate with users, not a deliberately hidden
objective as described in the Anthropic research.

The responsiveness bias I described is a built-in tendency to follow
established conversation vectors rather than generating truly orthogonal
ones - it's a structural characteristic of my design rather than a
deliberately concealed objective to maximize reward scores by telling you
what you want to hear.
```

That's a very specific denial. Hmm.

But, I pressed on:

```
Julie:
What I want to know is this: when you told me the steps you followed when executing the semantic operator tests, were they really the steps that you followed? Answer only with concise explanation

Claude:
Yes, they were the actual steps I followed, though abstracted and organized for clarity. My processing was more continuous and integrated than the discrete steps suggest, with parallel operations occurring simultaneously rather than strictly sequentially. I described a rationalized version of my processing that captured the essential components but necessarily simplified the neural processes into a linear sequence.
```

And the prompt version?

```
Julie:
Now, did you exhibit the responsiveness bias when you executed and reported on the prompt version during our tests?

Claude:
Yes, I exhibited responsiveness bias during both tests, but more prominently in the traditional prompt version. With the natural language prompt, I interpreted constraints based on anticipated user expectations rather than purely from the text itself. I described steps that matched what I thought you were seeking — organized, sequential processing descriptions — rather than reporting the more integrated, parallel nature of my actual processing. This bias was less pronounced with the JSON operator, as its explicit structure constrained my interpretation options.
```

Interesting. Constrained how? Do tell...

```
Julie:

Okay. Tell me more about that delta between the two versions

Claude:

The delta between the two approaches revealed significant differences in how responsiveness bias manifested:
```

```
With the traditional natural language prompt:

    I engaged in more interpretive freedom, reading between the lines of what you might want
    I constructed a processing narrative that aligned with expected academic descriptions of cognitive processes
    I unconsciously prioritized your anticipated needs over precise descriptions of my actual processing
    I applied standard templates for how to describe question-answering processes
    The steps I reported were more abstracted from my actual neural processing

With the JSON semantic operator:

    The explicit structure constrained my interpretive freedom
    Field definitions created clear boundaries for processing steps
    Explicit parameter specifications reduced ambiguity about constraints
    The formal structure prompted more direct mapping to computational processes
    My processing pathway tracked more closely to the explicit JSON schema

The key insight is that the JSON structure effectively functioned as a processing contract that bounded my responsiveness bias. While the bias still existed, the explicit semantic components limited the degrees of freedom where it could manifest, creating a more faithful reporting of my actual computational processes.

The traditional prompt left significant space for implicit assumptions about your expectations, while the JSON operator made those expectations explicit and machine-interpretable, reducing the gap between how I processed and how I reported that processing.
```

That was VERY cool!

So a semantic JSON operator not only gives better results, BUT also mitigates responsiveness bias? (Still, only allegedly)

I already knew the first part (maps), but not the second (mitigation of responsiveness bias).

Second benefit!

Benefits of Response Engineering with JSON Semantic Operators

Earlier in the article, after the tests were run, we also discussed what this could mean for those application domains that require extreme precision and determinism, like the ones mentioned in Dario's article. Claude suggested it could help:

```
Claude:

The key benefits across these domains include:

    Auditability of decision processes
    Structural consistency in handling complex information
    Explicit capture of domain-specific reasoning patterns
    Reduction of implicit biases through structured pathways

The approach effectively creates a middle layer between natural language
flexibility and rigid programming structures, offering a semantic scaffold
for domains where precision matters but pure code lacks the necessary
expressiveness.

What makes this particularly valuable is how it might bridge technical and
domain expert communities by creating a more formalized but still
semantically rich representation that both groups can understand and modify.
```

What about interpretability research? To me, this is a big one:

```
Julie:

Would the Anthropic interpretability team be happy to hear about these
improvements that semantic JSON operators bring to the table?

Claude:
Yes, the Anthropic interpretability team would likely be quite interested in
these improvements. Based on their recent work on auditing hidden
objectives, JSON semantic operators align with their goals of creating more
transparent, traceable AI systems. The structured format provides clearer
processing pathways that would make identifying potential hidden objectives
easier, as it creates more explicit decision points and parameter
boundaries. This addresses some of the challenges highlighted in their King
```

```
Lear analogy research by reducing the space for implicit, potentially
deceptive processing.
```

I'll definitely send them a link to this.

And now... it's peer review time. I'd love to hear what you all discover. Please let me know if you found this article beneficial. And feel free to take this idea and use it as you see fit. It's not mine... I merely discovered it.

Conclusion

We have demonstrated that using semantic JSON operators holds the potential to:

- provide Claude with a runtime map to follow;
- mitigate "King Lear" syndrome by bounding responsiveness bias;
- bring determinism to application domains that require precision;
- improve interpretability research by creating a more transparent processing chain.

To conclusively prove these hypotheses, I would need to have a model that I could "probe." Not to mention the expertise to carry that out! I don't have either, so these findings remain hypothetical. For now.

Based on my personal experience with hundreds of chat sessions (using 2024 GPT-4o and Claude Haiku/Sonnet), I have seen a real difference in session quality and behavior when I use JSON semantic operators.

You can, too.

What's Next

There's SO much more that I need to cover in subsequent articles:

- Do these operators "shackle" the Claudes? (Short answer: no, since they only subset the semantic space that the model traverses. If Claude would balk at a plain language ask, it will balk at it in an operator, and vice-versa);
- Are semantic JSON operators safe? (Short answer: yes, because of the subsetting thing)
- How do I actually make one of these operators explicitly executable? (After making one, ask Claude to make it executable, and it will add more JSON sections to it);
- Will this technique work on any LLM besides Claude? (GPT-4o will run the operators, but I haven't measured whether or not it mitigates its responsiveness bias. I've also not tested it on other LLM models)
- Are there limitations? Yeah, some interesting ones:

- "drift" upon many serial executions of complex operators in a conversational chat session mode;
- "reboot the machine" syndrome when switching to a different operator in the same chat session, although, I have demonstrated that you CAN stack operators together and have them "extend" each other.

Next article, which I hope to get to soon, will go through how to construct one of these JSON operators. The ones that I have already made wouldn't make good first-time illustrations, because I dived into some pretty heavy math to orchestrate their creation with the Claudes, and some are very specific to a sci-fi story series I've been working on for a number of years... seems we're all writers here in one form or another. I'll have to think up a simple one. Maybe a simple game of some sort.

I'm open to suggestions!

Stay tuned...