Geedka: Moderation Flow without Code

Answers to some FAQs

Ron Dubinsky

June 7th, 2023

What is the story behind Geedka?

- In initial development of bots, there was a lot of handwritten boilerplate
- This is inevitable in moderation flows that must look and act consistent
- Issues became evident when policy change required much engineering time

What is the rationale behind Geedka?

- Engineers are good at building tools that are agnostic as to content
- Policy teams are good at developing policy that is agnostic as to implementation
- If policy changes require manual software changes, this quickly becomes a pain point

Geedka:

- provides separation of concerns
- serve as a proof of concept that this approach may work in developing mod tools

The ideal is a configuration language that policy teams can work with, while engineering teams focus on supporting it.

What's next?

- The language is premised on tree-like moderation flows. If two paths diverge then converge, this is difficult to model.
- There is practically no error reporting; config bugs are very difficult to find. More tools are crucial.
- The language is not configurable and the ergonomics are still poor. With more development, the language can be refined.
- The language does not support developing mod-facing and user-facing reporting flows in parallel. It can build one flow at a time.

What assumptions does Geedka make?

- Moderation flows look roughly like trees
- The components of a moderation flow are pretty consistent (a button is a button, regardless of when in the flow it is rendered)

- Policy teams are prepared to work with config languages
- Developing this generic solution is more efficient than building a one-off solution

None of these assumptions are universally true, so there are cases in which Geedka may not be ideal.