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
- The issues became more evident when a change to policy required a lot of engineering time

What is the rationale behind Geedka?

- Engineers should be able to focus on engineering
- Policy teams should be able to focus on policy
- 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

Geedka is designed to enable a separation of concerns and 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?

Geedka still has a ways to go

- The language itself is premised on the fact that moderation flows are tree-like. If two paths in the moderation tree diverge then converge, this is difficult to model.
- There is practically no error reporting; config bugs are very difficult to debug. More tools to support such config tools are crucial moving forward.
- The language is not configurable and the ergonomics are still poor. With more development time, the language can be made more refined.

What assumptions does Geedka make?

Geedka assumes:

- Moderation flows look roughly like trees
- The components of a moderation flow are pretty consistent (a button is a button)
- 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.