Title: Verifying Arrow’s Impossiblity Theorem using Behavioral Subtyping

Abstract: Arrow’s impossibility theorem is a landmark result in theoretical economics that led to the formation of modern social choice theory. I develop a new, computer-verified proof of Arrow’s impossibility using refinement types, a concept from type theory that allows functions to return both data and predicates on that data. I then argue that refinement types are a natural paradigm for formally verifying many theorems in economics and political science.

# Introduction

Arrow’s impossibility theorem formalizes an unfortunate truth about collective decision making: all voting systems are provably flawed. In his [insert year] paper [insert title], Arrow proves that all systems to aggregate group preferences violate at least one (and possibly multiple) seemingly innocuous assumptions about how group preferences ought to be aggregated. When Arrow received the Nobel prize in economics, the Swedish Academy of Sciences described this theorem as “perhaps the most important of [Arrow’s] many contributions to welfare theory.”

However, like most important results from the social sciences, Arrow’s theorem has been largely ignored as a target for formal verification. This is unfortunate, as the social sciences generally and social choice theory specifically are great targets for computer-verified proof. Many of the concepts involved in social choice theory (e.g. preference relations) are easy to define and reason about in proof assistants. Furthermore, many of the proofs of Arrow’s theorem I read for this project were disappointingly unclear, including some of the most referenced in the literature. It’s worth emphasizing again how prominent Arrow’s theorem is: it is considered one of the most important theoretical results in all of economics. If even proofs of Arrow’s theorem suffer from vagueness, I am sure that many lesser-known results could be clarified substantially by formal verification. In addition, I suspect that there exist at least a few (though possibly many more) results whose proofs are just incorrect. Theoretical economics is thus both a technically feasible and academically rewarding site for formal verification.

To the extent that they exist, current computer-verified proofs of social scientific theorems tend to be both difficult to write and difficult to read. Many human proofs of Arrow’s theorem involve pictures that show how one can alter a list of candidates representing a voter’s preferences in a certain way while maintaining a certain desired property. While these intermediate results are easy to see visually, the Agda type checker does not have eyes (as of version 2.7), and formalizing these pictures within a type system is not intuitive. Furthermore, the limited verifications of Arrow’s theorem tend to rely heavily on proof automation techniques that are difficult if not impossible for human eyes to parse. In addition, these proofs tend to be extremely long and difficult to compartmentalize into lemmas because the automation techniques require the context provided by the prior parts of the proof to solve the goal.

One cause of these problems is the paradigm from which theoretical computer scientists tend to approach the social sciences. Verifying social scientific results becomes substantially easier if we formalize their various components as programs and apply well-known techniques for reasoning about programs (e.g. Hoare logic). In the remainder of this paper, I will develop a paradigm for formalizing social scientific results in proof assistants using refinement types, a concept in type theory that allows programs to return both an output and a proof about the content of that output. I will then discuss my proof of Arrow’s theorem using refinement types, and finally I will discuss the differences between my computer verified proof and a selection of hand-written proofs of Arrow’s theorem.

# Outline

1. What is arrows?
   1. Every electoral system is provably terrible
   2. Intuition for why conditions are interesting/useful
      1. IIA formalizes no spoilers
      2. Transitivity is just … how we rank things, nec to be useful
      3. Pareto
   3. Every real world system just has spoilers basically
2. Why arrows
   1. Natural choice for behavioral subtyping
      1. Also a natural choice for constructive logic since in some sense we “construct” a dictator
   2. Not a lot of work in formal verification of social sciences
   3. Rich basis for expansions
3. What’s interesting about this strategy
   1. All previous proofs are tactic mode proofs
      1. Which makes sense! Those people all were math people
      2. And they were probably way easier
      3. I couldn’t get tactic mode to work on my computer ☹
   2. This approach is basically wholly on the program side of the curry-howard isomorphism
4. How does the proof work?
   1. Combo of border notes and sen strategy
5. Lessons learned
   1. Lots of proofs are bad!
      1. Conflations between product types and functions
      2. Products carry state and functions do not
6. Future work
   1. Gibbard satterwaite
   2. Alt strategies to finish the proof
   3. Do it in a real programming language and execute the code (don’t just compile it)