CS 522: Programming Language Semantics
Homework-4 Solutions

Jai Arora (jaia3@illinois.edu)

Comparing Semantic Approaches

Big-Step SOS

Score: 5

- Big-Step Semantics are usually simpler, since they have lesser number of rules than, say, Small-Step SOS. It closely models a recursive interpreter
- It cannot distinguish between divergence of a program and errors/stuck configurations. The former corresponds to the proof search not terminating, and the latter corresponds the failure to find a proof (but the search has terminated)
- Big-Step SOS is highly non-modular. We saw that adding features to `IMP` meant modifying almost every rule, sometimes duplicating them to accommodate the changes
- Big-Step SOS cannot properly deal with non-determinism. It can only capture choice non-determinism, but not all interleavings, or true non-determinism

Small-Step SOS

Score: 8

- Small-Step SOS gives more control over the finer details and order of execution
- It can distinguish between stuck configurations and non-determinism
- Small-Step SOS is also non-modular, because one may need to duplicate the rules, modify configurations when adding new features
- It can deal with concurrency better than Big-Step SOS

Denotational Semantics

Score: 5

- Adding new features affects Denotational Semantics a lot: we might have to change the denotation domain altogether
- It cannot handle concurrency. Since, currently we modelled the denotations of programs as partial functions from `State`, for every input state, the denotation will map it to at most one output. However, with non-determinism and concurrency, we may have multiple behaviors for each input state. One fix is to switch to powerdomains, but computing the denotation of a program would require us to explicitly enumerate all possible behaviours for a given input
- Describing the denotations of programs in a language like `IMP` requires the knowledge of posets, CPOs and fixed points. Personally, I liked these domains a lot, which is why this approach is rated the same as Big-Step SOS, despite more shortcomings

Modular SOS

Score: 9

- As the name suggests, this approach tries to solve the problem of modularity of semantics when adding new features, and it fares well. In the case of `IMP++`, we can look at all approaches in isolation and add new rules for each of them
- One thing that I like about MSOS is that it is essentially a set of conventions on top of Small-Step SOS. Any MSOS rule can be easily desugared to a corresponding Small-Step SOS rule

Reduction Semantics with Evaluation Contexts

Score: 7

- Provides an explicit representation of the execution context as a first class citizen in the language semantics
- One needs to provide a grammar for the `Context`, which I think adds an additional complexity
- It is a modular SOS, so I ranked it higher than Big-Step SOS

Chemical Abstract Machine

Score: 7

- It can effectively model concurrency
- It is a very different approach that any of the previous semantic approaches, so I ranked it a bit high for its uniqueness
- I feel it is too abstract, and one would need to think a lot about designing rules for CHAM, and also think the language being designed. It feels like two different efforts, to me at least