Homework 4 Essay

Mihai Codoban

Before I begin with the scoring, I must make the remark that I have classified the different structural semantics in terms of a few concepts.

What I called *the snowball effect* refers to semantics in which change is not unitary and every new feature needs to update existing features and needs to be updated by all the existing particularities as well. Thus we get an exponential growth with every feature leaving it's mark on the overall code.

A critical property for semantics is how they represent *program properties that get updated*, such as state, flags and other miscellaneous entities. We can have an explicit representation (BigStep, SmallStep, Denotational, RSEC) in which the semantic configurations enumerate all properties or an implicit configuration (RSEC, CHAM) that collapses all properties under one generalization. The former is of course a pain to evolve since it reduces readability and requires the revisit of previous rules upon semantic evolution.

The property of *context manipulation* refers to how semantics deal with rules that require updating part of or all of the proof tree (halting etc). Some semantics (BigStep, SmallStep, MSOS, Denotational) use flags to notify the rest of the tree while other semantics (RSEC, CHAM) are smarter and represent the context as a first class citizens.

A nuisance for all semantics was the need to update the way the top level program is handled, needing to initialize different properties such as state or IO buffers. This problem could be mitigated by building adapters that adapt any top level rule to the most elaborate one, thus providing backwards compatibility. An even more elegant solution would be a way for a language feature to plug its initializers somewhere, thus attaining the open closed principle.

The following scores were given through the mindset of adding new features. In terms of how easy it is to reason about and analyse programs through the use of these semantics, I have no idea.

1 BigStep

Score: 4

- Readability: Low. Snowball effect contributors: variable size configurations, explicit representation of non deterministic rule cases
- Program property representation: explicit
- Context manipulation power: explicit through flag propagation
- Ease of change: Low

2 SmallStep

Score: 6

- Readability: Medium - Low. Snowball effect contributors: variable size configurations

- Program property representation: explicit
- Context manipulation power: explicit through flag propagation
- Ease of change: Medium - Low

3 Denotational Semantics

Score: 1

- Readability: non existent. Snowball effect contributors: severe variable size configurations, large number of projections break readability context
- Program property representation: explicit
- Context manipulation power: explicit through flag propagation
- Ease of change: Low
- Unable to represent non deterministic rules

4 MSOS

Score: 8

- Readability: High. No snowball effect
- Program property representation: implicit through records
- Context manipulation power: explicit through flag propagation
- Ease of change: Medium - High
- Remark: ugly to represent halting due to SmallStep particularities. Requires a level of indirection under Pgm; Too many rules for Spawn and Let.
- Remark: Inherits problems from SmallStep. It's only contribution is in making program property representation implicit by the use of records.

5 RSEC

Score: 8

- Readability: Medium - High. Snowball effect: variable size configurations; a bit polluted with all those split and plug rules that have to be written for every syntactic item
- Program property representation: explicit (*could borrow MSOS's labels*)
- Context manipulation power: implicit through contexts. Seems very clean and elegant
- Ease of change: Medium - High. Need to propagate configuration changes

6 CHAM

Score: 9

- Readability: High. No snowball effect
- Program property representation: implicit through composite nature of molecules
- Context manipulation power: implicit through K representation of contexts and the Chemical and Membrane laws
- Ease of change: High.
- Remark: out of all the structural semantics, *CHAM* seems the most elegant in terms of conciseness, readability and changeability.
- Remark: the failed attempts at CHAM show the sad fact that it cannot naturally represent abstract syntax trees. Things such as precedence get lost via heating rules