No Ifs, Ands, or Buts: Uncovering the Simplicity of Conditionals

Jonathan Edwards
OOPSLA 2007

Summary by Prof. Thomas LaToza
SWE 795, Spring 2017
Software Engineering Environments

GEORGE MASON UNIVERSITY

Schematic Tables: Motivation

- Make conditional logic simpler
  - Reduce possibility of duplicate logic
  - Offer canonical expression of branch logic
  - Make it easier to determine if a decision tree coves all cases
  - Make it easier to determine if cases overlap

- Address limitations of traditional decision tables by adding flexibility

Key Ideas

1. Computation and logic are represented orthogonally, in the vertical and horizontal dimensions of a table, separating their different semantics.

2. The two dimensions of tables provide an extra degree of freedom over textual syntax for structuring code, and reduce the need to split up related code.

3. Maintenance of a single logical invariant (partitioning), drives high level semantic transformations that reduce programming effort.

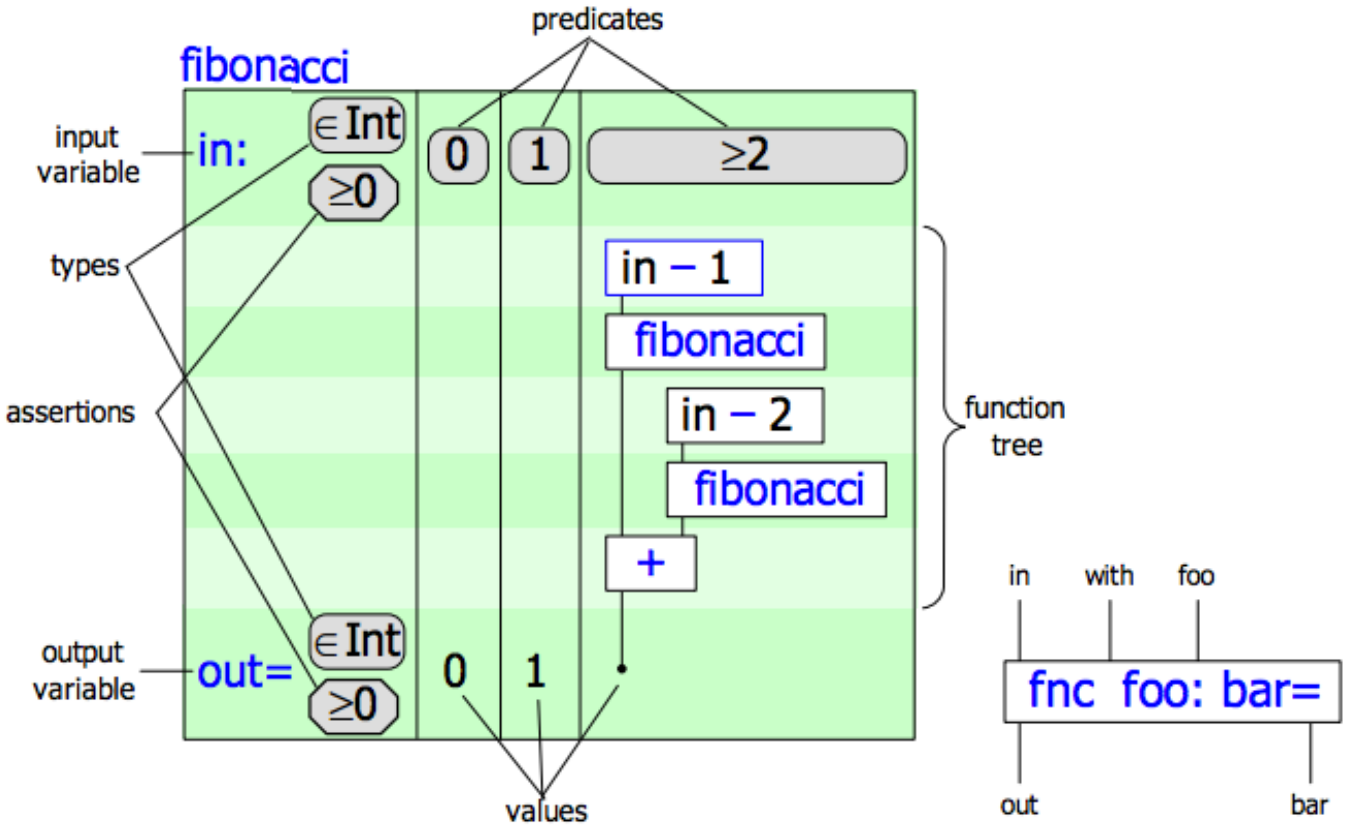
4. Schematic tables are canonical, in a certain sense representing equivalent programs equivalently. Diagrammatic notation is made feasible by eliminating manual layout. The need for refactoring is reduced.

5. All forms of conditionals are unified, from ad hoc if statements to pattern matching and polymorphic predicate dispatch.

6. A declarative language of predicates ties together the notions of types, assertions and conditionals.

Schematic Tables: Example

```
boolean a, b, c;
int x;
if (a & (!b | !c)) {
  x = 1;
} else if ((!a & b & !c)
           | (!a & !b & c)) {
  x = 2;
} else {
  x = 3;
}
```



Demo

https://vimeo.com/140738254

Questions for discussion

- Overall reaction to the paper

- Is this really simpler than traditional conditionals?
  - Are there other problems this creates?
  - Or other problems that are not addressed?

- Would you use such a system for your everyday programming?
  - Why or why not?