Decision Tables for Test Case Generation

CS 3750
A Sample Specification

Consider the Child Simulator 3000, a system meant to mimic the entertaining and illogical behavior of a two year old. The CS3k has a number of actions it can perform, all of which are based upon the various conditions specified below.

- The CS3k will follow all issued requests if and only if at least one of the following conditions is true. Otherwise, it will refuse to follow any requests.
  - it is daylight and it is before lunch
  - two people or less are watching the CS3k and it is after lunch
- The CS3k will behave “vibrantly” if and only if at least one of the following conditions is true. Otherwise, it will behave “calmly”.
  - it is dark and more than two people are watching
  - it is daylight and after lunch
A decision table contains a way to show how conditions affect actions by specifying rules.

If our decision table contains only Boolean conditions, it is a limited entry decision table.

If conditions have several values, it is an extended entry decision table.
A First Attempt Process

You can think of the process of making decision tables as follows:

1. List all the conditions. If you are using a limited-entry decision table, these conditions must be Boolean expressions. If you are using an extended entry decision table, you can have conditions with more than two possible values.

2. List all relevant actions of the system. For a software method, these might be all possible output values. For a business process, they might be all possible processes or actions we might take.
After listing all relevant actions, you can:

1. Fill in all possible rules. We can think of this as a mechanical process of enumerating all possible condition combinations. In practice, we’d use our higher-order thinking skills to reduce as we go, but we’ll think of the “enumerate all” part for now.

2. Fill in the correct action(s) for each rule you’ve generated. If you cannot actually create an input which satisfies all condition values, mark this rule as “Impossible” (which you can add as a special action).

3. Collapse redundant rules by (i) finding two rules with the same actions and then (ii) replacing the changing variables in these rules with a ’-’ (meaning the value of that variable does not seem to effect the output of the system).
Benefits of Decision Tables

Why might decision tables be helpful?

- They help us generate a complete set of test cases. We know how to compute the total number of rules, including when we have a “don’t care” condition, so we can ensure we’ve covered all possible combinations.

- They help us identify inconsistencies in our specification or test cases. If we find ourselves having more rules than we thought needed, we probably incorrectly labeled a “don’t care” somewhere.
Imagine Universal Pictures has plans to create *Jurassic World 3* in 2020 using the same director as *Jurassic World 2*, Juan Antonio Bayona. Imagine the decisions Bayona must make for creating the movie.

First, consider the decisions that must be made regarding casting. Bayona decides to use the following rules to determine the cast:

- If the studio has a budget of at least $225 million *and* news of the movie goes “viral”, then Bayona will hire Jeff Goldblum; otherwise Bayona will hire Matt Bomer.

- If the studio has a budget below $225 million *or* Michael Crichton’s family sues to stop the madness, then Bayona will cast Gary Busey in the film; otherwise Mr. Busey will not be included.
Bayona must also decide which dinosaurs to feature in the upcoming film, which will be determined with the following rules:

- If news of the movie goes “viral”, then the film will feature at least one *nanotyrannus*; otherwise it will not feature the *nanotyrannus*.

- If the studio has a budget of at least $225 million and Michael Crichton’s family does *not* sue, the film will once again feature a “genetically engineered” dinosaur; otherwise it will not feature an engineered dinosaur.