G53IDS - Interim Report

Embedded Domain Specific Language for
Describing Recipes in Haskell

James Burton - 4251529 - psyjb6

November 20, 2017

Contents

1 Introduction

Consider the following recipe to make a cup of tea:

```
- Boil some water
- Pour over a teabag
- Wait for 3 minutes
- Remove the teabag
- Add milk (optional)
```

This is a very simple but useful recipe that many people will perform, in some cases, many times a day over their lives. What we can realise by looking at this recipe is that it actually consists of many smaller recipes, such as boiling water and combining tea with milk, performed in a certain order. This raises the question, to what extent does the order matter and to what extent can we rearrange things in order to make the recipe more efficient? No doubt you have done this, maybe subconsciously, while cooking at home. Furthermore which steps can be done concurrently in the event that multiple people are cooking e.g. in a professional kitchen with a full brigade?

Perhaps closer to computer science, we could also ask, how could we instruct a robot to do this? After doing some research on robotic chefs it appears that not a huge number exist. There is one home cooking robot [1] which uses motion capture in order to learn recipes. In my opinion this is rather restrictive. It presumes that the human performs the recipe in the optimal manner and it would be very difficult to model a brigade system in this way. In reality there is a limited set of fundamental actions that one becomes able to perform when learning to cook. Recipes can then be performed using a sequence of these actions. Representing recipes like this would allow us to take a robot programmed to perform each of the fundamental actions and tell it how to cook literally anything.

So we've established that if we can structure recipes in a more formal way then we will have a great amount of freedom in terms of how we process them whether it be optimisation for a human chef or full on automation. My contributions / planned contributions to this are the following:

- Define a set of combinators as an EDSL in Haskell and show that they can be used to describe a wide variety of recipes (Section 2).
- The combinators describe a recipe but we then need to know how to execute a recipe as a sequence of the fundamental actions mentioned above. Fortunately, as described above, recipes are just a combination

of simpler recipes meaning that we can recursively define the actions necessary to perform complex recipes as long as we have manually defined which action are required for each combinator (Section 3).

- We now have a way to describe recipes and a way to show the sequence of actions to complete a recipe but now we need to apply them to something. Using the operational semantics of recipes we can optimise and schedule recipes for a given kitchen system. We can then express this in several ways including printing steps, drawing the cooking process as a graph or simulating the recipe within the given kitchen setup (Section 4).

- It may also be useful to provide an intermediate language between informal English and our combinators, some sort of markup language, that can then be parsed in.

2 Describing Recipes

3 Executing a Recipe

4 Implementation

5 Progress

5.1 Project Management

5.2 Contributions and Reflections

6 Related Work

References

[1] The Guardian. 2015. *Future of food: how we cook*. https://www.theguardian.com/technology/2015/sep/13/future-of-food-how-we-cook