G53IDS - Interim Report

Embedded Domain Specific Language for
Describing Recipes in Haskell

James Burton - 4251529 - psyjb6

November 14, 2017

Contents

1 Introduction **3**

2 Describing Recipes **3**

3 Executing a Recipe **3**

4 Progress **3**

5 Related Work **3**

1 Introduction

2 Describing Recipes

3 Executing a Recipe

4 Progress

While implementing the function to map a Recipe to a tree of Actions it became apparent that the initial combinators that were being used had some significant issues. The main problem is having an explicit Sequence combinator. The problem with this is that most of the Recipes infer order already. This is necessary for the Recipes to be displayed as a tree. For example, if you Heat 100 r, it is trivially true that you must complete Recipe r before you can heat it. Having a separate Sequence combinator allows for the Sequencing of Combine steps which should never happen as it places unecessary restrictions on the ordering of the Recipe. (Include diagram to demonstrate that the two combines are mutually exclusive). The main reason that Sequence was included was to provide more expresiveness for the Wait combinator. Seqeuencing a Recipe r with Wait 10 means complete r then wait for 10 whereas Combine (Wait 10) r means perform r for 10. If we take a closer look at this we can see that what we are actually trying to do is provide time constraints on our other combinators. The solution to this is the Condition class.

A further issue is that the Combine combinator has too many responsibilities. We have removed the case where it is used with Wait to indicate performing a Recipe for a certain time but it is also used for all kinds of mixing, assembling etc. Include SPJ quote.

5 Related Work

References