RecipeGen: Language Specification

Anna Owens and Emma Neil

Fall 2022

1 Introduction

We recently came across Joshua McFadden and Martha Holmberg's award-winning veggie-focused cookbook entitled *Six Seasons: A New Way with Vegetables*. The book divides the year into 6 growing seasons (three of which are in the summer). Each section features recipes that use vegetables and fresh herbs exclusively from that season. McFadden's motivation was founded on several guiding principles: first, vegetables take on a more complex, sweeter, and simply better taste at the peak of their growing season, second, vegetables taste best alongside other vegetables from the same season, and third, that it is possible for beginner chefs to learn how to cook to the seasons.

We employ the tenets of the *Six Seasons* cooking philosophy as inspiration for a programming language, grounded in the idea that beginner chefs do not necessarily have to learn how to cook for the seasons through trial and error. Instead, they could write a simple program to generate a scrumptious, seasonal recipe that fits their needs. In the final version of our language, we hope to offer the possibility for our users to enter information about the dish type, season, and ingredients to include or exclude from their recipe.

Our project stands out from other sources of recipes in a few important ways. First, and most obviously, it is opinionated in the style of recipes that it generates, choosing exclusively seasonal (to New England) vegetables and combining flavors according to the recipes in *Six Seasons*. Second, it offers the possibility of generating several recipes in a row, if the first recipe isn't to a user's taste. Third, because it is a programming language, the rules dictating flavors can be combined are systematically guided by food science and are more likely to yield yummy dishes.

(We haven't quite finalized the name of the language, so please excuse our current boring title)

2 Design Principles

Our language was designed with the following principles in mind:

Simplicity It is easy to use because it requires so little input. It also produces output that is simple to understand because it takes the form of a list of ingredients which is familiar to anyone who has seen a recipe.
Support for abstraction Users have to know remarkably little about the rules that make the recipe generator function. In particular, they do not need to know which ingredients are seasonal or which combination would taste best. All they need to know is the kind of dish they would like to make and the season in which they will make this dish.
Maintainability and Expandability We chose to create general types (detailed below) that were flexible enough to enable us to add particular ingredients as we expand the types of dishes a user can program. This flexibility is further achieved by the choice to input new ingredients via a CSV file rather than hardcode them.
Regularity Our parser expects inputs in a very specific order (season, dish type, restrictions) and recognizes a bounded number of input options for each "slot" in the program.

3 Examples

Example 1
Input:

```
    dotnet run "spring salad without nuts, cheese"
```

Output:

```
Recipe will include one spring green, 2 spring vegetables, and a dressing.
One example output could be:

    Your recipe is:
    0.250 bunches of Butter Lettuce
    4.000 ounces of Asparagus
    1 whole Artichoke
    0.125 cups of Pancetta Vinaigrette
```

Example 2
Input:

```
    dotnet run "warm fall salad without broccoli"
```

Output:

```
Recipe will include one fall green, 2 fall vegetables (none of which are broccoli), a nut,
a dressing.  One example output could be:

    Your recipe is:
    0.250 bunches of Arugula
    4.000 bunches of Broccoli
    5.000 ounces of Butternut Squash
    0.5000 ounces of Roasted Pine Nuts
    0.500 cups of Parmigiano-Reggiano
    0.125 cups of Simple Red Wine Vinaigrette
```

Example 3
Input:

```
    dotnet run "summer salad"
```

Output:

```
Recipe will include one summer green, 2 summer vegetables, a nut, a cheese and
a dressing.  One example output could be:

    Your recipe is:
    0.250 bunches of Kale
    4.000 bunches of Cucumber
    5.000 ounces of Bell Pepper
    0.5000 ounces of Roasted Walnuts
    0.500 cups of Aged Pecorino Romano
    0.125 cups of Citrus Vinaigrette
```

4 Language Concepts

The core concepts a user needs to understand our language is a basic knowledge of seasons and ingredients. Our language produces recipes for dishes based on seasons, which dictate which vegetables and fruits are best, and dish type, currently only soup and salad. For seasons, there are four basic season: fall, winter, spring, and summer. In theory, the user would use the current season. Since the recipe generator works for all seasons, and because modern grocery stores generally provide many fruits and vegetables year round, a user could theoretically input any season.

The second concept a user needs to understand is a dish type. Currently, our dish type is a salad, but we hope to expand this to include soups, entrees, and side dishes. A user should have a basic understanding of these, as well as a basic understanding of how to put together salads, as the program does not (yet) have recipe instructions. However, even if a user does not have any cooking experience, it would only hinder the food they make, not their ability to use the program. Further implementation will have us adding more elements to each dish type, for example, adding nuts and cheese to a salad. We hope to allow our language to handle cases such as ”fall salad with no cheese;” in this case, our user would have to know which ingredients may show up in a soup or salad recipe that they would like to avoid.

With seasons and dishes being our key ideas (primitives), combining them is easy. The user must simply pick the dish they would like, and input their season.

5 Grammar

Note: not all of grammar has been implemented yet

```
<expr> := <Attribute><ws><Dish>
<Attribute> := <Temperature> | (<Attribute>, <Attribute>) | <NoAttribute>
<ws> := " "
<Temperature> := "warm"<ws>  | "cold"<ws>
<NoAttribute> := ""
<Dish> := (<Season>, <DishType>, <Exception>)
<Season> := "fall"<ws> | "winter"<ws> | "spring"<ws> | "summer"<ws>
<DishType> := "salad"<ws> | "salad"
<Exception> := SoftCore(<Flag>, <Name>) | HardCore(<SoftCore>, <SoftCore>) | <NoException>
<NoException> := ""
<Flag> := "with"<ws> | "without"<ws>
<Name> := <Category> | <Word> | <NoName> | <Name>(<Comma><ws><Name>)+
<Word> := <Letter>+
<Letter> := ’a’ | ’b’ | ... | ’z’
<Comma> := ’,’
<NoName> := ""
<Category> := "greens" | "vegetables" | "fruit" | "cheese" | "nuts" | "dressing" | "spice"
```

6 Semantics

Our programming language is designed to generate seasonal recipes with New England fruits and vegetables. The minimal version parses a simple string that users can enter via the command line. It expects a single input, a season and a dish type (salad is our only dish type option) and outputs a list of ingredients for a salad that belong to that season. The semantics are trivially simple because it only expects these. In order to accomplish this parsing and evaluation, we define the following primitives in our language:

Category describes the relevant food group to which an Ingredient belongs. Each ingredient (in this version) belongs to one of the following categories: Green, Vegetable, or Dressing.

Unit describes the units in which an Ingredient is measured. So far, we've defined the following units of measurement: Cup, Tablespoon, Ounce, Liter, Head, Whole, Ear, Bunch.

Season refers to the season of an Ingredient or recipe. The options (unsurprisingly) are: Fall, Winter, Summer, and Spring.

Dish refers to the dish category of a recipe. Currently, users can only choose to generate Salad, but eventually we hope to expand this to include Entree and Soup.

Ingredient is a record type that contains all the information necessary to choose an ingredient for a recipe, including a string Name, Unit unit, float Quantity (corresponds to the amount of the ingredient necessary for a serving size of one), Season list to which the ingredient belongs, and Category category.

7 Remaining work

We have many different directions we could go with this program, and are trying to confine our remaining work so we don't get too broad. A first step would be to add new dish types, such as soup, entree or side dishes. We would have to define exactly what these mean, which is difficult because in the regular world, many different dishes fit under the category of entree or side dish. We also hope to define a set of ingredient rules (i.e. flavor combinations that work well together), so that instead of our recipe generator being so random, it actually produces combinations that would taste good.

We also hope to allow the parser to handle something such as "fall salad without cheese," so that users are capable of making their recipes more specific and personalized.

Last, we need to continue adding ingredients to the CSV to broaden our recipe options. We also may divide our summer season into three categories: early summer, mid summer, and late summer, based on the cookbook we are using for inspiration. Because summer is such a good season for crops, and because these crops differ across the different stages of summer, this would allow for more specific seasonal dishes.