Jayme Woogerd
Comp 150PLD - Programming Language Design
October 15, 2014
Final Project Design Proposal

1) I plan to create a language to for programmatically specifying, generating, and analyzing surveys. Specifically, my language will target the type of surveys administered for research in the social sciences.

2) The intended users of this language are domain experts in a research field in which surveys are a major means of data collection, e.g economics or education. I assume users will have some knowledge of survey design and statistical analysis, but may have little or no formal programming experience.

   In general, users will want to: 1) write (specify) survey questions and response types, 2) generate and deploy a survey and 3) perform statistical analysis on the results.

3) Goals:

   i) The language should be small, simple, and easy to learn for non-programmers In fact, my idea is that the surface syntax will look more like markup (plus some control-flow constructs) than code

   ii) It should be easy to design and specify simple survey questions, but there should be enough power and flexibility in the language for users to define their own question types and formats

   iii) It should be easy to generate artifacts that are helpful for survey deployment: a paper-based survey via LaTeX and possibly a web-based interface via HTML and JavaScript

   iv) Primitive data structures should be amenable to statistical analysis and manipulation and it should be straightforward to perform common statistical operations on survey results

   v) (This is a stretch) I got the original idea from a friend of mine, who is writing a proposal to do research in India. Her research project hinges on administering surveys to Tibetans who have taken refuge there, many of whom have limited English skills. It would *nice* if she could write the survey in both English and Tibetan and then at compile time choose what language to render the survey in

4) To my knowledge, there is no existing programming language for specifying, generating, and analyzing research surveys. However, there are several software programs and web services for survey generation; two well-known ones are Qualtrix and Survey Monkey. These products may be easy to use but (anecdotally) lack the flexibility and control needed to define some custom question types or survey structures. Additionally, users are limited to the features included in the software packages.

5) Features:

   i) Simple, declarative constructs for specifying `surveys`, `sections`, `questions` and `responses`

   ii) Variable binding and functions

   iii) Constructs for composing new response types and macros for control flow

   iv) Control-flow constructs for dependent or conditional questions (e.g. `if ...then go to...else go to`, `case response of ...got to...`)

   v) Rendering of surveys in paper format or as a web-based interactive survey

   vi) Support for statistical analysis on collected data

   The features set of this language will be just large enough to specify common survey structures and question types and render them in some format, which should be a sufficient first step of the project. Additionally, many surveys have some sort of flow logic: depending on a subject's response to a particular question, she may be directed to a follow-up question or to skip a section of questions. To support this very common structure, there a simple control flow construct is essential.

6) My intuition is that all of the artifact generation will happen in the runtime system. My initial idea is to provide two artifacts for survey deployment: a text version of a survey via LaTeX and a web-based interface via generated HTML and JavaScript.

7) At very least, I will probably need library support for generating LaTeX, HTML markup, and JavaScript in addition to any statistical computation I want to support natively. In any case, a very cursory query into the Hackage repository yielded the following libraries as possibly useful: 1) `HaTex` for generating LaTeX code, 2) `html` for generating HTML code, and `statistics` for standard statistical computations.

8) Domain experts will gain a number of benefits from using this language to define and generate their surveys. First, users will be able to write a survey just once and generate the survey in multiple formats (text, web). Second, users may be able to specify surveys with less text ("code"). For example, if many questions have a the list ["Always", "Sometimes", "Never"] as their available responses, this list can be named once and referenced throughout the program. There is also the potential for users to have more control over the structure of a survey and the types of questions they want to ask. Last, this language may make it trivial to deploy multiple variants of a single survey, e.g. in multiple languages‘’.

9) Use cases: see following pages.

10) Evaluation:
    Since one of the goals of this language is to be easy for a non-programmer to learn, the first dimension to evaluate the language is how easy it is for a non-programmer to learn

and how willing he is to learn use the language to make surveys over survey software programs. Additionally, given the general principles of code reuse and compositionality, I suspect that many surveys, especially those with repeated types of responses, can be specified with substantially less text than when written out manually. Less text means smaller chance of typographic error and may make modifying the survey easier. Lastly, to support the claim that a language is more flexible or powerful for generating surveys than a traditional survey software application, it would be good to have some examples of survey structures, questions or analyses that are easy to perform in the language but would be difficult or impossible to do with software.

(Note: this section needs fleshing out. These are my preliminary thoughts.)

In the simplest case, a user defines a very short survey with two questions and generates a representation of the survey in printed text and in web markup. The user supplies the content for the questions and responses and uses language constructs to markup the survey structure.

```
[begin survey mySurvey]
    [begin section A]

        [1][question "What is your favorite color?"]
           [repsonse {"Red", "Blue", "Yellow", "Green"}]

        [2][question "How often do you ride the T?"]
           [repsonse {"Never", "Sometimes", "Often", "Always"}]

    [end section]
[end survey]

main:
    make mySurvey
```

This code snippet shows what a simple control sequence might look like and how common response patterns might be leveraged to reduce code length.

```
howFrequent = {"Never", "Sometimes", "Often", "Always"}

[begin survey mySurvey]

    [1][question "How often do you brush your teeth?"]
       [repsonse howFrequent]
       if response > "Sometimes" then go to [3]

    [2][question "How often do you ride the T?"]
       [repsonse howFrequent]

    [3][question "Have you ever seen the movie 'The Mighty
        Ducks'?"]
       [repsonse {"Yes", "No"}]

[end survey]

main:
    make mySurvey
```