Goals

To show that is is possible to efficiently approximate some key features of natural language (implicit referencing and syntactic compression) with a formalized subset of it.

To present the implementation of a formalized subset of the English language, showcasing said naturalistic features.

To suggest that said formalized language subset may prove useful for some lightweight scripting tasks, particularly as a spoken language.

Programming by voice has received attention in the last years from both the commercial and the research sector, as an alternative to the de-facto golden standard approach of text-based programming [1], [2].

This has happened as a result of the increase in acquired disabilities realted to long periods of typing: Repetitive Stress Injury (RSI), which can lead to severe neck ache and back pain [1], [2].

A Voice User Interface (VUI) is a Human-Computer Interface (HCI) that enables interaction with a computer through an auditive interface. It is usually a complement to the more popular Graphical User Interfaces (GUI), as it is most often seen in virtual assistants, automobile and home automation systems, etc...

If one could overcome the many challenges behind building a suitable VUI for the task of writing, reading and maintaining code, it would improve the quality of life of many a disabled software developer and/or people interacting with computers that have motor health issues but are nonetheless capable of using speech to interact with such a hypothetical system.

There are quite a few hurdles on this path, some of them are general to the design of a VUI, and some are specifically related to the deployment of such a system for the purpose of programming.

Some of the more general problems are:

- the ephemeral nature of speech as compared to text [3]. - issues in discoverability; or making sure the user is aware of the options available to him/her at any point of using a voice enabled system. - issues in transcription: there is a tradeoff between the size of the available vocabulary and the precision with which the words are recognized [3]. - privacy and noise-related concerns (eg: at a crowded workplace), obviously. - etc...

Some of the more specific, programming-related problems are:

- recognizing keywords and abbreviations in code (at least in "traditional" code) that aren't contemplated by off-the-mill voice recognition software [2]. - dealing with multiple levels of nesting in programming language structures [2].

- etc...

There have been multiple attempts at designing such systems, and the approaches that were taken have been diverse.

One approach, detailed in [2], involves the idea of a Syntax-directed voice editor. A Syntax-directed editor, as explained in [2], takes advantage of the regularities of a formal language to provide automatic completion for common language constructs, saving the user time and typing. When applied to voice programming, the authors hypothesize that it can help reduce the mental toil of spelling out loud a potentially convoluted piece of programming syntax character by character. The Syntax-directed editor married to the voice recognition software produces a programming environment that is easy to reconfigure for many different programming languages.

A slightly different approach has been taken by [1]. The researchers here focused on the idea of applying the Reactive Paradigm (rather than the more traditional Imperative style) to voice programming, and comparing the efficiency of the two by preparing spoken versions of a set of programs in Java versus RxJava, a library that provides Reactive Extensions to the language.

The Reactive Paradigm is oriented around data flow and the propagation of change. It deals with asynchronous data streams where the data is events and vice versa [1].

The authors of the study found out that the Reactive style usually produces longer code in both characters, syllables, and words as compared to the Imperative style. However, owing to the higher expressiveness of Reactive constructs, those words themselves produced more effective work [1]. Moreover, those words contained a higher percentage of English-dictionary words, rather than word abbreviations: which are harder to pronounce and harder to be recognized by general purpose voice recognition software; this perhaps owing to the fact that Reactive programming makes less use of temporary variables and short variable names [1].

A general overview of what it means, practically speaking, to design an effective VUI is given by [3]. As already hinted, a VUI is a specific instance of a HCI, and as such it is subject to such general considerations that can be made on the usability of any computer interface.

Some of the HCI general concerns discussed here, are:

- Providing suitable feedback - Allowing for user diversity (novice vs expert) - Minimizing memorization efforts - Error prevention - Error handling - etc...

Some of the more VUI specific ones are:

- Appropriate output sentences - Output Voice Quality - Proper entry recognition - etc..

Discoverability, Mixed Initiative and (mutlimodal) output kind remain some of the key challenging aspects of designing a speech enabled system.

Speech output, especially when enumerating avaiable options, can be slow and tedious, it may therefore be of some benefit to provide an alternative alley for the output of a VUI system: such as a Graphical User Interface when feasible, thus making the system multimodal.

Another advantage of spoken systems is that speech is generally considered to be a faster input method than typing: most pepople can speak faster than they can type, at least when speking a natural language.

Bibliography

[1] Miriam Lagergren and Max Soneryd. Programming by voice: Efficiency in the reactive and imperative paradigm, 2021.

[2] Stephen C Arnold, Leo Mark, and John Goldthwaite. Programming by voice, vocalprogramming. In *Proceedings of the fourth international ACM conference on Assistive technologies*, pages 149–155, 2000.

[3] Valéria Farinazzo, Martins Salvador, Andre Luiz S Kawamoto, and João Soares de Oliveira Neto. An empirical approach for the evaluation of voice user interfaces. In *User Interfaces*. IntechOpen, 2010.