Interactive Programming with Automated Reasoning

Breandan Considine

McGill University

breandan.considine@mail.mcgill.ca

March 28, 2023

Research Interests

How do we use software to build more intelligent systems, and how can we use intelligent systems to help us write better software?

Find programmer friction points, formalize them as optimization problems, solve and publish our solutions as new developer tools.

- ▶ Realtime developer assistance
- ▶ Code completion and program repair
- ▶ Documentation search and retrieval
- ▶ Editing and refactoring source code
- ▶ Assistance for impaired developers

Important: build tools that we personally need or want to use, then *use* the tools we built to identify and improve usability issues.

Tools for assistive programming

Three developer tools and their core optimization problems:

- **AceJump** - A single character search, select, and jump
  - Minimizes finger travel distance in keyboard navigation
- **Tidyparse** - Syntax repair for programming languages
  - Minimizes Levenshtein edit distance subject to a grammar
- **Idiolect** - Handsfree audio development interface
  - Minimizes intent-recognition failures in voice programming

Tag Assignment in AceJump

Helps developers search and navigate code rapidly. To do so, we solve the **Tag Assignment Problem**, i.e., stated formally:

Given a set of indices I in document d, and a set of tags T, find a bijection $f : T^* \subset T \leftrightarrow I^* \subset I$, maximizing $|I^*|$, such that:

$$d[i \ldots k] + t \notin d[i' \ldots (k + |t|)], \forall i' \in I \setminus \{i\}, \forall k \in (i, |d| - |t|]$$

where $t \in T, i \in I$. This can be relaxed to $t = t[0]$ and $\forall k \in (i, i + K]$ for some fixed K, in most natural documents.

Natural language: *Maximizes the number of non-conflicting tags assigned to search results in uniquely-selectable manner, i.e., should never be possible to select a tag by mistake.*

AceJump Usage

```
fun map(availableXDTags: List<String>, caches: Map<Editor, EditorOffsetCache>): Map<StringJJTag> {
  val eligibleSitesDVTag = HashMap<String, MHtableLisDDTag>>( initialCapacity: 100)
  varKKtagsByFirstLetter : Map<Char, List<String>> = availabXATags.groupBy { it[0] }

  for ((editor : Editor , offsets : IntList ) in newResults) {
    val iter : IntListIterator = offsets.iterator()
    while (iter.hasNext()) {
      val site : Int = iter.nextInt()

      for ((firstLetter : CharMMtags : List<String> ) iXXtagsByFirstLetter.entries) {
        if (cDTTagBeginWithChar(editor, site, firstLetter)) {
          forQQtag : String  iPPtags) {
            eligibleSitesMLTag.getOrPuFGtag, ::MUtableListOf .adFDTag(editor, site))
          }
        }
      }
    }
  }
}
```

https://github.com/acejump/AceJump/

Error Correction in Tidyparse

Helps novice programmers fix syntax errors in source code. We do so by solving the **Bounded Levenshtein Reachability Problem**:

Given a conjunctive grammar $\mathcal{G}$, a fixed edit distance r, and a malformed string $\sigma : \Sigma^*$, find all syntactically well-formed strings $\{\tilde{\sigma} \in \mathcal{L}(\mathcal{G}) \mid \Delta(\sigma, \tilde{\sigma}) < r\}$, ranked by Levenshtein edit distance.
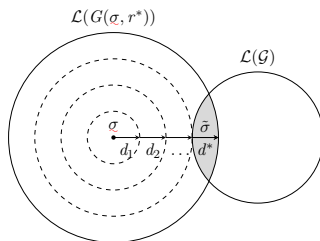
Natural language: *Finds all syntactically valid repairs within a small edit distance, ranked by similarity to the original input.*

Tidyparse Usage

Intent Recognition in Idiolect

Helps programmers with motor and visual impairments write code.
We do so by solving the **IDE-intent recognition problem**:

Given an audio signal containing an arbitrary stream of words,
$S(\Sigma) = 1 + \Sigma \cdot S(\Sigma)$ consisting of subsequences from a context-free
grammar find the optimal alignment of non-overlapping commands
from that grammar satisfying the given command.

Natural language: *Given a series of spoken voice commands, find
the optimal alignment of actions satisfying the user's intent.*

Idiolect Overview

https://github.com/OpenASR/idiolect/
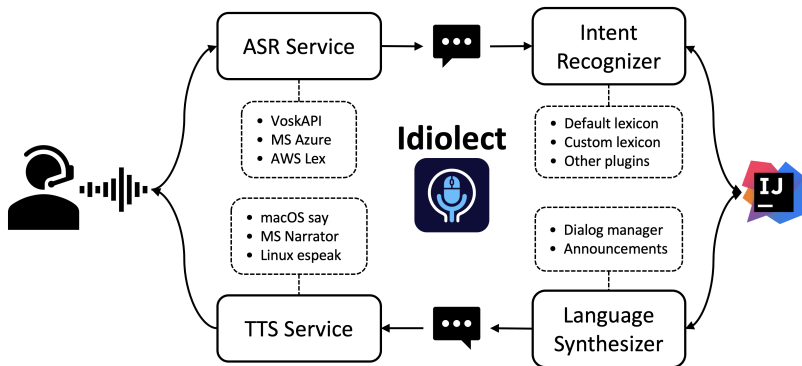
Research Summary

My research studies:

- ▶ Information foraging patterns in software development
- ▶ Common programming mistakes and their solutions
- ▶ Usability and accessibility challenges in programming

Using the insights gained, we:

- ▶ Reframe usability issues as optimization problems
- ▶ Formalize and solve those problems using pen and paper
- ▶ Integrate the solutions into real-world developer tools

This helps developers be more productive by providing:

- ▶ More contextually-aware user interfaces
- ▶ Safer and more flexible code completion
- ▶ More accessible and interactive developer tools

Acknowledgements

Academic Advisors

- Jin Guo
- Xujie Si

Collaborators/Contributors

- Nicholas Albion
- Daniel Chýlek
- Alex Pláte
- John Lindquist
- Alexey Kudinkin
- Yaroslav Lepenkin

Feedback/Inspiration

- Stefan Monnier
- Brigitte Pientka
- Torsten Scholak
- Jürgen Cito
- Michael Schröder
- Ori Roth
- Younesse Kaddar
- Kiran Gopinathan