Chapter 4

Prototype Library for Edit Lenses

4.1 Introduction

synchronizes two simple, text-based databases according to a pre-determined lens by validating and extracting edits from user actions

1. Purpose
   (a) validate that the fundamental design is complete enough to implement some examples
   (b) investigate the connection between strings as a data model and structured data models
2. Initial choices and assumptions
   (a) Haskell as the programming language – and in particular not a mechanization of the theory
   (b) Chose strings as a data model (because it is often used in practice)
   (c) Edits are to be extracted from user operations (applies to the example and executable)
   (d) Chose not to expand the existing asymmetric string editors, in favor of a simpler approach
3. Challenges
   (a) representing the interrelations between 'things'
   (b) Haskell does not have dependent types, but they would have been very useful
   (c) 'parsing' user actions (string edits) into structured edits
4. Outcome

   (a) can't apply standard techniques (like incremental parsing) to parsing user actions
   (b) proposed some heuristics for the parsing, but they're special-purpose and don't adequately reflect all user actions (e.g. cut/paste is not translated to a reordering)
   (c) demonstrated a small, clean core library design

4.2 Usage Example/Description of Functionality

1. describe database format + lens connecting them
2. screenshots showing synchronized databases
3. a few simple edits that get reflected; and why we can't reflect before the edit is completed

4.3 Implementation Details

annotated code, may describe selected functions or maybe all functions (?)

1. edits + edit application: the Module type class
2. lenses, simple lenses, lens+module triples
3. basic lenses + maybe a few combinators
4. unparsing (needs more motivation and explanation, or less if you consider this as not being reusable code)
5. parsing (needs more motivation and explanation, or less if you consider this as not being reusable code)
6. connecting to a GUI + storing complements in ref cells
7. a 'bad' choice - modules are based on type classes instead of being records

4.4 Conclusion

1. Message: this is an existing library and an associated GUI that extracts alignment information from user actions
2. Message: the existence of a working library is an indication that nothing important was overlooked in the theoretical foundation

3. Message: this library could be used for further studies of edit lenses beyond the scope of the current work
    (a) original purpose: convert a string edit into a tree edit (resulted in a hard problem)
    (b) demonstrate usability of the syntax by generating some practical examples
    (c) show a performance advantage
    (d) demonstrate a practical application of lenses (we have a long list of ideas about this)
4. Outcome: we need new techniques for some parts, but core library can be elegant

4.5 Full code

perhaps appendix, or pointers to hosting, or an attachment to the dissertation, or some such thing