# An Enhanced Jigsaw Puzzle Solver

CS297 Final Report

Zayd Hammoudeh
(zayd.hammoudeh@sjsu.edu)

April 27, 2016

# Contents

# List of Figures

# 1 Introduction

Jigsaw puzzles have been around since the 1760s when they were made from wood. Their name derives from the fact that they were originally carved using jigsaws. The 1930s saw the introduction of the modern jigsaw puzzle where an image was printed on a cardboard sheet that was cut into a set of interlocking pieces. Although jigsaw puzzles had been solved by children for centuries, it was not until 1964 that the first automated jigsaw puzzle solver was proposed by [6], and that solver could only solve 9 piece puzzles. While an automated jigsaw puzzle solver may seem trivial, it has been shown by [1] and [5] to be strongly NP-complete when pairwise compatibility between pieces is not a reliable metric for determining adjacency.

Jig swap puzzles are specific type of jigsaw puzzle where rather than pieces have different (usually unique) shaped, each piece is a square of equal size. Jig swap puzzles are substantially more difficult to solve than standard jigsaw puzzle as one can no longer consider mechanical compatibility when trying to determine a relationship between pieces. In such scenarios, one can only on the individual piece's image information when solving the possible.

Solving a jigsaw puzzle simplifies to reconstructing an object from a set of component pieces. As such, techniques developed for jigsaw puzzles can be generalized to many practical problems. Examples where jigsaw puzzle strategies have been used include: reassembly of archaeological artifacts [2] [4], digital forensic analysis of deleted files [7], image editing [3],

Currently, there is no full implementation of Haskell in the JVM. One Haskell dialect that is runnable in Java is Frege [**?** ].

This project implements, *HamSkill*, which is a transpiler from Haskell to Scala; *HamSkill* enables a dialect of Haskell to run in the JVM.

# 2 Key Project Requirements

When designing and implementing this project, there were four primary goals:

1. **Runnable in the Java Virtual Machine** - As explained in Section **??**, Java's Virtual Machine enables significant machine independence, which Haskell does not currently have.
2. **Minimal JVM Requirements** - In addition to just running in the JVM,

*HamSkill* was created to be as standalone as possible. As an example, it was not expected that in most applications, the user would have Scala installed on their machine. To achieve this maximum portability, some more niche features of Haskell may not be supported.

3. **Identical Input and Output Between Haskell and *HamSkill*** - In many scenarios, it may not be sufficient for a Haskell program to simply run inside the JVM. Rather, it is more likely that the output generated by the two environments will need to be identical. As such, *HamSkill* includes an additional post-processing step to ensure its output is identical to that of Haskell.

4. **Human Readable Output Code** - A transpiler is any program that takes source code from one programming language and outputs code in another programming language, with a similar level of abstraction [**?** ]. To enable increased reuse of the outputted code, *HamSkill* uses techniques such as indenting, newlines, etc. to maximize the readability of the generated output. While this is not a necessary requirement for the complete system to work properly, it enhances the tool's potential.

# 3 *HamSkill*'s Software Architecture

*HamSkill* is a transpiler that takes Haskell code as an input, converts it to Scala, and then runs the transpiled code in the JVM. The *HamSkill* implementation consists of six major components. They are:

- ANTLR Lexer and Parser
- `Haskell` ANTLR Grammar
- `ScalaOutput` ANTLR Grammar
- Scala Runtime Environment
- `HamskillMain` Java Class
- `ScalaOutput` Java Class

# Bibliography


[1] Tom Altman. Solving the jigsaw puzzle problem in linear time. *Appl. Artif. Intell.*, 3(4):453–462, January 1990.

[2] Benedict J. Brown, Corey Toler-Franklin, Diego Nehab, Michael Burns, David Dobkin, Andreas Vlachopoulos, Christos Doumas, Szymon Rusinkiewicz, and Tim Weyrich. A system for high-volume acquisition and matching of fresco fragments: Reassembling Theran wall paintings. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 27(3), August 2008.

[3] Taeg Sang Cho, Moshe Butman, Shai Avidan, and William T. Freeman. The patch transform and its applications to image editing. *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[4] Marc Levoy David Koller. Computer-aided reconstruction and new matches in the forma urbis romae. *Bullettino Della Commissione Archeologica Comunale di Roma*, 2:103–125, 2006.

[5] Erik D. Demaine and Martin L. Demaine. Jigsaw puzzles, edge matching, and polyomino packing: Connections and complexity. *Graphs and Combinatorics*, 23 (Supplement):195–208, June 2007.

[6] H. Freeman and L. Gardner. Apictorial jigsaw puzzles: The computer solution of a problem in pattern recognition. *IEEE Transactions on Electronic Computers*, 13:118–Ű127, 1964.

[7] Simson L. Garfinkel. Digital forensics research: The next 10 years. *Digit. Investig.*, 7, August 2010.