1 Proposed Work

1.1 Current Work

There have been several attempts at embedding PROLOG into HASKELL which are discussed below along with the shortcomings.

1. Very few embedded implementations exist which offer a perspective into the job at hand. The first implementation

2. Only two embeddings exist, one of them is old and made for **`hugs`** a functional programming system based on the HASKELL 98 specification. It is complex and also lacks a lot of PROLOG like features including *cuts, fails, assert* among others. The second one is based off the first one to make it simple but it loses the variable search strategy support which allows the programmer to choose the manner in which a solution is produced.

3. The papers that try to take the above further are also few in number and do not have any implementations with the proposed concepts. Moreover, none of them are complete and most lack many practical parts of PROLOG.

4. Libraries, a few exist, most are old and are not currently maintained or updated. Many provide only a shell through which one has to do all the work, which is synonymous with the embeddings mentioned above. Some are far more feature rich than others that is with some practical PROLOG concepts, but are not complete.

5. Moreover, none of the above have full list support that exist in PROLOG.

And as far as the idea of merging paradigms goes, it is not the main focus of this thesis and can be more of an "add-on". A handful of crossover hybrid languages based on HASKELL exist, CURRY [**?**] being the prominent one. Moving away from HASKELL and exploring other languages from different paradigms, a respectable number of crossover implementations exist but again most of them have faded out.

As discussed in the sections above, either an embedding or an integration approach is taken up for programming languages to work together. So, there is either a very shallow approach that does not utilize the constructs available in the host language and results in a mere translation of the characteristics, or the other is a fairly complex process which results in tackling the conflicting nature of different programming paradigms and languages, resulting in a toned-down compromised language that takes advantages of neither paradigms. Mostly the trend is to build a library for extension to replicate the features as an add on.

1.2 Contributions

Taking into consideration above, there is quite some room for improvement and additions. Moving onto what this thesis shall explore, first thing's first a complete,

fully functional library which comes close to a PROLOG like language and has practical abilities to carry out real-world tasks. They include predicates like *cut, assert, fail, setOf, bagOf* among others. This would form the first stage of the implementation. Secondly, exploring aspects such as*assert* and database capabilities. A third question to address is the accommodation of input and output, specifically dealing with the *IO Monad* in HASKELL with PROLOG *IO*. Moreover, PROLOG is an untyped language which allows lists with elements of different types to be created. Something like this is not by default in HASKELL. Hence syntactic support for the same is the next question to address. Furthermore, experimenting with how programs expressed with same declarative meaning differ operationally. Lastly, how would characteristics of hybrid languages fit into and play a role in an embedded setting.