1 Proposed Work

As we have seen there have been several attempts at either embedding PROLOG into HASKELL or integrating the paradigms, a few shortcomings are very clear with the former,

1. Only two embeddings exist, one of them is old and made for **hugs** which 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 part.

2. 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 lack many practical parts of Prolog.

3. The last part would be the libraries, a few exist, most of which are old and are not currently maintained or updated at all for quite a few years. Many of which 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 i.e. with some practical Prolog concepts, but are not complete.

4. Adding on to that none of the above have full list support that exist in Prolog.

And as far as the latter goes, that is not the main focus of this thesis and can be more of an "add-on". Moreover, only one such crossover hybrid language that is based on Haskell exists, Curry [**?**]. Moving away from Haskell and exploring other paradigms, a respectable number of implementations exist but again most of them have faded out. Also the surviving contenders are not very widely used nor popular and hardly come up to the surface.

As discussed in the sections above, either an embedding or an integration approach is taken up for accomplishing the task. So there is either a very shallow approach which does not fully utilize the constructs available in the host language and just results in a mere translation of the characteristics. While the other is a fairly complex process which results in tackling the conflicting nature of different programming paradigms and/or languages, resulting in a toned down compromised language that neither takes advantages of either paradigms, sure the both the sides need to be integrated but integrated languages have never really worked. Mostly the trend is to build a library for extension to replicate the features mostly as an add on.

1.1 Contributions

Taking into consideration the flaws and the shortcomings above, there is quite some room for improvement and additions. First thing's first a complete, fully functional library which comes close to a PROLOG like language which has practical abilities to carry out real world tasks. They include predicates like *cut,*

write a paragraph

1. Create a Prolog library which comes the closest to a complete Prolog like Language, not only the Declarative concepts but also a lot of Practical concepts like cuts, assert, fail, setOf, bagOf among others.

2. Do away with the shell approach. One must be able to write a program just the way one can write a Haskell Program and the goal is passed as a parameter to the main function and the result is achieved by running it.

3. Moreover, make an attempt to add some features which are related to Functional Logic Programming Languages, like narrowing and residuation among others.

4. Furthermore, adding or adjusting Prolog IO into the Haskell IO Monad.

5. Translating a Prolog Program into a Haskell Program, for example two programs with same declarative meaning but different computational complexities in Prolog translated into Haskell. (Something similar is there in a paper called Higher Order Transformation of Logic Programs)

6. Tackling Translation of Tail Recursion in Prolog Programs.
7. Able to recognize and work with all kinds of lists supported by Prolog.
8. Anything else ??????????????