CONCLUSIONS

The thesis is a successful attempt to implement the functional languages on Parallel Architecture SIMD. As parallelism is encapsulated by higher-order functions such as map, fold and scan, each having a serial analogue in a traditional functional language, the introduction of parallel programming to a functional programmer is natural, the new in these parallel functions have a different complexity to their sequential counter parts, opening the door to efficient algorithms. The important theme in this thesis is non-strict evaluation, namely evaluating only those things that are required and thus allowing infinite data-structures. Non strictness always delays evaluation to the point that the evaluated object is required. In contrast in a data-parallel model large amounts of data are evaluated synchronously and in parallel. These two opposing views are reconciled by introducing a new evaluation mechanism called the aim of evaluation.

A new evaluation mechanism and a notation for expressing parallel algorithms has been proposed in terms of set theory. An important aspect of this notation is that it provides a single frame work in which both parallel operations and communication can be expressed. Non-strictness played an important role in defining the semantics of POD comprehensions where generators were designed in such a way as to exhibit as much non-strictness as possible. Extensive use of this notation was used to develop parallel algorithms for higher-order functions. The gap between definition and realisation of a
CONCLUSIONS

parallel machine has been bridged by expressing the higher order functions in terms of more low level techniques.

From an implementation viewpoint, the contribution of this thesis has been to formalize the vectorisation process. Here vectorisation is expressed in terms of a small set of transformation rules for lambda-calculus. Vectorization has also been extended to cope with algebraic data-types by transforming data-structures in a sequential implementation that are typically implemented by pointer-structures, into a form where the pointers are eliminated and the transformed data-type is susceptible to evaluation on a data-parallel machine.

Another significant contribution of this thesis is to bridge the gap between parallel functional programming and parallel programming in the "real world". This work offers to the parallel programming community the ability to raise the level of abstractions of existing techniques away from combinations of simple operations while having a solid foundation in terms of well understood principles of both functional and parallel architectures paradigms. A partial implementation of a compiler has been presented. Although the compiler was not completely finished, it provided an invaluable insight into compilation techniques and functional programming implementation on SIMD machine.
CONCLUSIONS

The technical contributions of this thesis are

- The extension of the evaluation mechanism to enable operations on parallel datastructures to be non-strict without sacrificing parallelism.
- A design of fundamental higher-order functions that encapsulate general patterns of parallel computation, with emphasis placed on their non strict semantics.
- A technique for vectorising non-strict functional programs in terms of program transformations so that they are suitable for SIMD machines.
- A systematic method of transforming algebraic data-types into an inside-out form which facilitates the implementation of parallel evaluation on SIMD machines.
Further work:

- In the context of the abstract machine, the effectiveness of the techniques used for updating and merging vectors still needs to be more fully investigated.

- From an implementation standpoint: The language semantics and various implementation choices were made with a SIMD machine in mind. This can be extended to a MIMD machine.

- One aspect of implementation that has been ignored is the space complexity of non-strict data-parallel evaluation. As PODs can be potentially large objects, the needless retention of these objects is particularly worrying. This angle may be further explored.

- Another exciting further work is combining the aspects of data abstraction and implementation of associated functions in terms of PODs and higher-order parallel functions which are described in this thesis.