THE UNIVERSITY OF MELBOURNE

MELBOURNE SCHOOL OF GRADUATE RESEARCH

20 March 2013

Mr Paul Bone
1/212 Pascoe Vale Road
ESSENDON VIC 3040

Dear Mr Bone

PhD Examination

The examiner(s) of your thesis have requested that certain minor amendments, which are set out in the accompanying report(s), be made to your thesis. You should consult your supervisor and Chair of Examiners, **Professor A Moffat**, on how to respond to the examiner(s) comments.

The amendments required are to be incorporated into the text of the thesis. There are specific requirements for the final presentation of the thesis, particularly with regard to what should appear on the spine of the thesis. Refer to "Final Form of the Thesis" requirements in the relevant handbook, located at: gradresearch.unimelb.edu.au/index.html. You are requested to provide an index summarising the changes for the Chair of Examiners to consider and approve **before binding your thesis.**

Candidates who commenced from 1 January 2007 must now submit a digital copy of the thesis, preferably in PDF format, to The University of Melbourne ePrints Repository (UMER) as a requirement for completion. All candidates who commenced prior to 2007 are encouraged to submit a digital copy but it is not mandatory. There are clear instructions to guide the uploading process on the Repository available at: www.lib.unimelb.edu.au/eprints/thesis. Information on the requirement can be found at: gradresearch.unimelb.edu.au/exams/digital-thesis.html.

Please submit prior to **29 April 2013** two copies of the amended thesis in permanent hard cover binding (one of these on archival quality paper) **to the Chair of Examiners** and one digital copy to the ePrints Repository (if you commenced your degree from 2007). If you are unable to meet this deadline and require an extension of up to one month, email msgr-exams@unimelb.edu.au otherwise, you are required to complete an *Application for an Extension to Submit Final Permanently Bound Copies of the Thesis* available at: gradresearch.unimelb.edu.au/exams/candidates.html#forms.

Candidates are considered eligible to pass once the Melbourne School of Graduate Research has received a Completion of Degree Form from the Chair of Examiners, two permanently bound copies of the thesis through the Chair of Examiners, an approved citation and a digital copy of the thesis has been deposited to the Library (for candidates who commenced from 2007). Once these requirements have been fulfilled you will receive a letter stating you have successfully completed all the requirements for the degree and will be eligible to graduate. Graduation information will accompany the completion letter.

Should you need further information please contact the examinations staff at msgr-exams@unimelb.edu.au or ph. +61 3 8344 8294.

Yours sincerely

Mary Makris
Senior Examinations Officer

cc. Professor A Moffat, Chair of Examiners
Dr Z Somogyi, Supervisor
Computing and Information Systems

MM:rw

Melbourne School of Graduate Research
The University of Melbourne, Victoria 3010 Australia
T: +61 3 8344 8599 **F:** +61 3 9349 2103
W: www.gradresearch.unimelb.edu.au

Ex: 2

Overall Evaluation

The dissertation provides a fairly comprehensive overview of techniques developed to exploit and-parallelism from the execution of Mercury programs.
Overall, this dissertation is clear and well-written, and the research performed to support the dissertation is of good quality. The exploitation of parallelism in a system like Mercury is a no-brainer, but it is also the type of work that requires an intimate knowledge of Mercury's mechanisms and confidence in how to modify them. Paul definitely fit that role perfectly and produced research results that are worthy of publication in good research venues.
As such, my recommendation is to award the degree without any further examination.

Organization

The dissertation is well organized. I particularly appreciated the pedagogical approach that Paul followed in laying out the materials, starting with a clear review of the mechanisms of Mercury, followed by a motivation for the changes needed to support parallelism, followed by detailed descriptions of implementation and evaluations. As such, the dissertation is quite self-contained and comprehensive in its coverage of the materials.

Relevant Literature

For the most part, I was satisfied by the coverage of the related literature. I also enjoyed reading the considerations that Paul developed at the end of Sections 4, 5 and 6 to compare his contributions to the most relevant related efforts.
I was somewhat less satisfied with the actual level of comparison that was included in some of the parts. For example, Shen's work in DASWAM went into quite a good level of detail and it would seems to have a good relation to what covered in Section 4, yet I do not see a discussion at the level of implementation structures. We could say the same for comparing with &-Prolog/&-ACE in terms of data structures and optimization (Section 5 talks a little about the flattening of parallel calls, but that's not much).

Methods and Techniques

The dissertation uses the best methods and techniques in the research development. The author has done a good job in identify the critical points to be addressed to gain good parallel performance and develop an infrastructure to make that possible. All the techniques are grounded in experimental motivations, making the design very strong.

More Detailed Considerations

As mentioned above, I am overall pleased with this dissertation. It makes a good contribution to the field of parallel logic programming and parallel declarative languages. I would like to praise Paul for having always used experimental arguments to motivate the developments – this is a very good approach.

I have only a few minor considerations that I would like to point out and possibly address in a final version of the dissertation (or perhaps as future steps in continuing the research work).

- General Considerations:
  - I would have liked to see some discussion about how all the techniques proposed in this dissertation could be applied outside of Mercury (e.g., to Prolog? To functional languages?)
  - Many of your considerations rely on two benchmarks, representing some fairly regular computations. How would you consider these representatives? Or, more in general, I would have liked to see a much broader pool of diverse benchmarks being used throughout the dissertation.
  - There are no formal considerations about the fact that the parallel implementations respect the "theoretical" operational semantics of the language (e.g., same observable behavior). Even though it is true, it would be a good idea to spell it out.
- Chapter 1 is supposed to set the context for the whole dissertation, and it does so in a good way. The chapter could be strengthened a bit by adding some citations (especially in the first few pages). Additionally
  - Considerations in this chapter ignore the new generations of architectures based on CUDA (NUMA, not SMP, etc.)
  - I would suggest to add examples of Pure and Impure languages
  - Is the example in page 8 correct?
  - Considerations in page 9 talk about "logic programming", but they are really focused on languages derived from Prolog (SLD-based, etc.). Logic programming is a much broader term, and the considerations in this page do not reach other LP languages (e.g., ASP-based).
  - Hermenegildo used to stress that there is really no such thing as independent and dependent and-p, they are the same thing just seen at different levels of granularity (and I tend to agree with this).
  - My memory might be wrong, but the dependent and-p model of Pontelli and Gupta does not really build on [45] (they are completely independent). Furthermore, DDAS was the name of the system developed by Kish Shen, not by Pontelli/Gupta.
- Chapter 2:
  - Can you provide a source for the various statistics mentioned in page 25?
  - How does the discussion in page 26 relate to some of the tail recursion optimizations developed for and-parallelism?

- I might have missed it, but lots of what I see in page 28 resembles the behavior of conditional variables in POSIX threads.
- I found some considerations in page 30/31 a bit speculative (especially the last two paragraphs before 2.4.1); any evidence supporting these claims? In particular, evidence related to how unbalanced computations can become due to different inputs.
- The discussion in this chapter could benefit from graphical representations of the data structures.

- Chapter 3:
  - I found several English errors and typos, please proofread
  - Amdahl's law tend to be rather conservative – have you considered using something like Gustafson-Barsis instead?
  - Reason 2 page 50: would it be possible to test this hypothesis? E.g., bounding/unbounding threads?
  - I found page 56 rather poorly written and hard to follow.
- Chapter 6:
  - Please include more figures
- Bibliography:
  - Several errors, please review your entries.
  - [46] has a spurious 'p'
  - [45] appeared in a more complete form in some ICLP (perhaps 1994)
  - I believe Pontelli was an author in [47] – also it was published in 2001, not in 1995; on the other hand 1995 saw the publication of Hermenegildo's et al. paper on &-ACE (which introduces many of the independent and-p structures and optimizations)
  - [90] was published in ICLP'97

Ex: 1

Automatic Parallelisation for Mercury (Paul Bone)

Context

This thesis represents a substantial amount of work on the handling of, and the automatic parallelisation of, programs written in Mercury. Mercury being a declarative (logic/functional) language presents both opportunities and challenges for automatic parallelisation that are not (or to a lesser extent) present in imperative or object-oriented languages. The contributions of the thesis include :

- Improvements designed for, and implemented into, the Mercury runtime system. The main contribution here consists in the introduction of a so-called *work stealing* algorithm in Mercury's runtime system. While the ideas behind *work stealing* are far from new, integrating them in a system as complex as the Mercury runtime is in itself a challenging and non-trivial task to do. The author describes two implementations, one of which being joint work with P. Wang.
- A set of algorithms for analysing what parts of a computation are likely worthwhile to be performed in parallel. The analysis is based on coverage information gathered from profiling the application and basically computes what parts of a computation represented by a conjunction can overlap, and proposes a scheme for parallelising the conjunction.
- A program transformation that transforms right-recursive predicates in order to limit the number of contexts that are spawned when the predicate body is parallelised.
- An extension to the Mercury runtime system that performs extensive logging, and an accompanying module in the ThreadScope tool that allows to interpret and visualize the logged information as such providing a visual feedback to the programmer/researcher.

Although the last contribution remains in a somewhat exploratory phase, all contributions are well-motivated, generally well-developed, and their effect is experimentally evaluated by a number of benchmarks.

General evaluation

The work presented in the thesis is original, and parts of it have been published in international and peer-reviewed series (a journal article, an international conference paper, and an international workshop paper). The text is rather well-written and of suitably high standard. In my opinion, the candidate shows sufficient familiarity with the relevant literature, which is cited appropriately.

The thesis provides an in-depth treatment of a relevant problem, is well-motivated and the techniques and solutions developed in the thesis are scientifically sound. The chosen approach is properly justified and appropriate, the solutions are all implemented, and the implementations are experimentally evaluated. The set of benchmarks used for evaluation is limited but the chosen programs are appropriate for evaluating the proposed techniques. The benchmarks include programs of substantial size and seem representative for what can be found in real-world applications. The experimental results are detailed and sufficiently well exposed and – where possible – appropriately interpreted.

The work obviously contains an important effort of implementation and is heavily based on the existing implementation of Mercury. This is also reflected in the operational style in which most of the text is written. Consequently, it is not always easy to see if and how the obtained results could be transposed to other programming languages/environments, but this was not among the objectives of the thesis.

Some more-detailed comments:

p.28 Top. When explaining the working of future_wait/2, it would be nice to add a line explaining why there is no danger for deadlock.

p.30-31. In your argumentation for the feedback-directed approach, you state yourself that a program is typically executed multiple times with respect to different inputs (p.31). While you use this argument in favour of the feedback-approach (which is imo totally justified), it does raise the question on how well is the feedback-approach providing a model for *all* of these runs? Unfortunately, this question is rapidly put aside by your remark "Most variations in input data will not effect parallelisation enough to cause a significant

difference in performance" (p.31), but this is quite strong a statement, is there any (experimental) proof of it? I think the chosen approach is a good one, but it should be introduced/discussed with a more appropriate level of objectiveness. See also my remark on the discussion w.r.t. Static analysis (p 92).

p.34. Imprecise vocabulary. On the one hand you talk about "computations p and q" but further down the text they have become "conjuncts p and q".

p.41, line 3. G_{k1},\ldots,G_n. This must be G_{k+1}.

p.50. What is "HBLKSIZE"? Is it some dark option to be set in the Boehm runtime? If you care to mention it, it should also be explained.

p.60. While I was wondering for several pages why this structure had to be a stack, the second half of the page provides the explanation (to deal with nested conjunctions). It would help the reader if this justification was moved to the spot where the stacks are introduced.

p.71. Just a remark, but algorithm 3.8 is quite basic and the idea of reordering independent conjunctions seems not being pushed very far.

p. 78. In Figure 3.6, there is an "XXX" remaining.

p.79. Figure 3.7. Seems strange to characterize some transitions as "busier" because "you think" (p.78) they occur most often. Is this relevant and, if it is, could it be better (experimentally) validated/justified? If it isn't, don't talk about it as it makes one wonder whether some of the choices you made are based on intuition only.

p.92. When (re)introducing the general approach and justifying the feedback-approach, the discussion on profiler-feedback versus static analysis could be more detailed and more objective. You put a lot of emphasis on "representative input" (see also my remark concerning pp.30-31) that is chosen by the programmer, but why not let the *user* decide on what is "representative input" by providing, e.g. a *specification* of typical input (e.g. types and size of certain structures). In the latter case, an approach using static analysis might be more useful than a profiler-based one. Just to be clear, I am not criticising your approach, nor am I asking to change it; I am only stating I feel it could be somewhat more objectively (with its strong and weak points) introduced and discussed.

p.93 (end of section 4.2). Terminology: one often uses "monovariant/polyvariant" to refer to the fact that a predicate/procedure is analysed/transformed/compiled one versus multiple times with respect to a somewhat different context.

p.106 (bottom of the page): "the recursive call's cost at its average recursion depth is used by the algorithm". Is this (theoretically speaking) the best one can get or would it be possible to obtain more precise results (e.g. by performing some fixpoint computation on the predicate)?

p. 120 (bottom of the page). Typo: "performed perform".

p. 124. Typo: "that the each iteration"