Higher-Order Symb Comput (2009) 22: 305–312
DOI 10.1007/s10990-010-9055-7

The Peter Landin prize

Kevin Hammond · Greg Michaelson

Published online: 10 April 2010

Abstract The Peter Landin prize honours the best paper presented at each year's International Symposium on the Implementation and Application of Functional Languages (IFL). It has been awarded every year since 2003, and covers a range of topics including functional operating systems, static analysis for cost information of functional programs, techniques to improve array processing for data locality and parallelism, explicit parallel coordination, supercompilation, and a rational deconstruction of Landin's SECD machine itself. This article describes the history of the prize, explains why Peter Landin was chosen as nominee, and describes each of the articles that have been awarded the prize to date.

Keywords Functional programming · Programming language design · SECD machine · ISWIM · Parallelism · Static analysis · Supercompilation

1 History of the Peter Landin prize

In 1999, the authors commissioned and edited a book on parallel functional programming [10]. Its twenty-seven international contributors subsequently agreed that the royalties should be donated to fund a small annual prize that could be used to promote research in functional programming.

We were very pleased when the International Symposium on the Implementation and Application of Functional Languages (IFL) agreed to host the prize.[1] IFL originated as a small workshop on parallel functional programming, held at Nijmegen in 1989. Since that time, it has been held annually, growing significantly in the process, and since 1996, it has published a high-quality post-event proceedings, containing a selection of the best papers

[1] See http://www.ifl-symposia.org.

K. Hammond (✉)
School of Computer Science, University of St. Andrews, St. Andrews, Scotland
e-mail: kh@cs.st-andrews.ac.uk

G. Michaelson
School of Mathematics and Computer Science, Heriot-Watt University, Edinburgh, Scotland
e-mail: G.Michaelson@hw.ac.uk

presented at the event. Reflecting this focus on quality, after ten years of existence, IFL was upgraded into a Symposium. IFL has now established itself as one of the leading annual events in the functional programming calendar, with a specific focus on practical aspects of functional programming.

The Peter Landin prize is administered by the authors, on behalf of the donors, and has been awarded every year to the best paper presented at the previous IFL. The prize is chosen by the IFL programme committee based on the papers that have been submitted, without any input on our part. It thus represents an honest and prestigious recognition of the merit of the authors of the papers that have been so honoured. Each year, the prize certificate is awarded in person to the authors of the successful publication (a cheque is, of course, sent sooner, to allow the laureates to celebrate in style!).

2 Why honour Peter Landin?

It is customary to name major awards after significant contributors to the field. For example, in Computer Science, outstanding work is recognised by the annual ACM Turing Award, the BCS Lovelace medal and the BCS Needham award, honouring three notable pioneering figures in the discipline, namely Alan Turing, Ada Lovelace and Roger Needham, respectively. Since functional programming is a broad field, closely integrating and synthesising programming language formalisation, design and implementation, in pursuit of principled, rigorous and robust system construction, we felt that Peter Landin was an ideal person to honour for his foundational work in the area. Peter's contributions are throughly documented elsewhere in this special issue. Here, we note particularly:

- the design of the SECD machine for applicative expressions, the grandparent of all subsequent abstract machines for functional languages [13];
- the elaboration of ISWIM, the grandparent of all post-LISP untyped functional languages [15]. ISWIM was both a principled and practical formalism for functional programming, which rationalised both syntactic constructs and semantic concepts, and founded the concept of domain specific languages.

We approached Peter in 2002, to see if he would agree to his name being used for the new prize. He was both delighted and disarmingly modest: though we continued to correspond with him he never mentioned the prize again.

We also asked Peter if he had any feel for the origins of parallel functional languages. After all, Wegner in his 1971 book [23], which contains an important popularisation of the SECD machine, had written:

> Note that [the Church-Rosser] theorem essentially states that lambda expressions can be evaluated by *asynchronous multiprocessing* applied in arbitrary order to local subexpressions. (p. 185)

Peter replied:

> You probably know about Samson Abramsky's parallel (?concurrent?) SECD-machine [1].
> My choosing rands before rators was at the time an arbitrary decision constrained by wishing to exclude that separate topic from the paper. I had been very struck by a paper pointing to the high-speed possibilities of pure functional programming, arising from parallel treatment of multiple operands (which of course comes to the same thing,

except he was not, as I remember, conscious of lambda) by a Brown(?) in an MIT collection of articles called "High-Speed Computing" (?) circa 1960. I think he was a prof at Sloan Business School.[1]
My own attempts to describe such a machine didn't get past being overawed by scheduling problems.
Thanks for your interest. I'm posting the (2-page) IFIP paper, which should have been called: Certain possibilities for economising in an implementation of SECD. (email, 21/10/2002)

In the paper [14], Peter discusses 'an equivalence relation called *"sharing"* among a state's structural positions', and whether shared entities should be copied at construction or application, if at all. This notion of sharing seems to foreshadow subsequent debates over parallel graph reduction.

3 Prize awards

The Peter Landin prize has been awarded at every event since IFL 2003. It has been awarded to a wide variety of international authors at varying stages of their careers. The topics that have been honoured are similarly diverse, ranging from functional operating systems to new compilation techniques; from novel static analyses to new methods of compiler construction; and from new ideas on language constructs, derived from old discoveries in number theory, to a rational deconstruction of the SECD machine itself. Table 1 shows the Peter Landin prize awards from 2003 to 2009. In each case, the prize refers to a paper presented at the previous year's IFL Symposium, and published in the post-symposium proceedings. The prize has frequently been awarded to PhD students, in competition with much more established researchers, demonstrating the excellent quality of the work that is often done by pre-doctoral researchers in the field of functional programming.

2003: Arjen van Weelden and Rinus Plasmeijer [22]

The first (2003) Peter Landin prize was awarded to Arjen van Weelden and Rinus Plasmeijer for a paper on *Towards strongly-typed functional operating systems* [22] based on work done by Arjen van Weelden for his PhD at the University of Nijmegen, the Netherlands. The paper introduced Famke, a prototype functional operating system written in the Clean functional language, and which supports an interactive, functional shell command language. The Famke system is capable of dealing with a variety of important constructs, including lightweight threads, exception handling and distributed processes. A key issue covered in the paper was to allow the communication of values of any type, including unevaluated closures, between pairs of processors, which exploited Clean's dynamic typing framework. This paper represented one of the earliest attempts to define complex operating system constructs taking advantage of the capabilities offered by a strong typing framework.

2004: Pedro Vasconcelos [21]

The second (2004) Peter Landin prize was also awarded to a PhD student, Pedro Vasconcelos from the University of St Andrews. Pedro Vasconcelos' paper, *Inferring Costs for Recursive,*

[1] We have been unable to locate either the author or collection.

Table 1 The Peter Landin Prize awards: 2003–2009

Year	Winner	Title
2003	Arjen van Weelden and Rinus Plasmeijer	Towards a Strongly Typed Functional Operating System
2004	Pedro Vasconcelos	Inferring Costs for Recursive, Polymorphic and Higher-Order Functional Programs
2005	Olivier Danvy	A Rational Deconstruction of Landin's SECD Machine
2006	Clemens Grelck, Karsten Hinckfuß and Sven-Bodo Scholz	With-Loop Fusion for Data Locality and Parallelism
2007	Jost Berthold and Rita Loogen	Parallel Coordination made Explicit in a Functional Setting
2008	Neil Mitchell and Colin Runciman	A Supercompiler for Core Haskell
2009	Ralf Hinze	Scans and Convolutions: a Calculational Proof of Moessner's Theorem

Polymorphic and Higher-Order Functional Programs [21] presents a new type-based analysis for inferring size and cost equations for recursive higher-order programs. Unlike many previous approaches, many of which are capable only of checking the size and cost information that is provided by the programmer (e.g. [12]), the analysis described in this paper is capable of inferring costs directly from unannotated source programs. The main contribution of the paper is a type reconstruction algorithm that extends a basic Hindley-Milner type inference algorithm to cover size and cost information, and which exposes recurrence information in the form of a set of constraints that must subsequently been solved by an external constraint solver. Vasconcelos found that the equations produced by his system gave accurate predictions for a large number of functions from the Haskell standard prelude. Subsequent work has taken the ideas developed in this paper and applied them to cover stack and heap space metrics. The work is currently being combined with an *amortised analysis* approach developed at Ludwig-Maximilians-Universität, München, to give cost information which is more accurate than either individual approach.

2005: Olivier Danvy [6]

The 2005 Peter Landin prize was most appropriately awarded to Olivier Danvy, Aarhus, Denmark, for his paper on *A Rational Deconstruction of Landin's SECD Machine* [6] Olivier Danvy gave a beautiful and rational theoretical deconstruction of the basis of Landin's SECD machine, with the fundamental aim of explaining *for the first time in the 40 years since the publication of Landin's foundational paper* the specifics of the SECD machine. He achieved this goal by deconstructing the SECD machine mechanically into a compositional evaluation function, and then rationally reconstructing a whole range of different SECD machines

from this fundamental basis. These machines include a tail-recursive SECD machine, call-by-name/call-by-need SECD machines, SEC/EC/SC/C/SCD machines and an SECD machine with an instruction set. A key contribution of the article is that it outlines a *reversible* methodology for extracting the fundamental denotational semantics of an abstract machine as a compositonal evaluation function. Since the publication of this paper, Danvy and his PhD student Kevin Millikin have extended the reconstruction to Peter Landin's J operator, providing objective evidence why Peter Landin's name should be added to the list of the discoverers of continuations [7].

2006: Clemens Grelck, Karsten Hinckfuß and Sven-Bodo Scholz [9]

The 2006 Peter Landin prize was awarded to Clemens Grelck, Karsten Hinckfuß and Sven-Bodo Scholz for their paper on *With-Loop Fusion for Data Locality and Parallelism* [9]. With-loops are a descendent of loop fusion as envisaged by Burge and Landin [5]. This array-processing construct used in the numerically-oriented SAC language [19] to implement array operations of arbitrary shape or rank. They come in two forms, either generating a new array structure of a given shape, or reducing the elements of an array using some operation. In this paper, the authors define high-level code transformations that fuse (not-necessarily adjacent) pairs of with-loops into a single with-loop. This process may be repeated as many times as required. By using this fusion technique, it is possible to avoid repeated traversals of an array structure, and to replace memory accesses by (faster) register accesses. For sufficiently large problems, the resulting program can be more than four times as fast as the original code. Scholz's research has directly influenced work on efficient array skeletons for SAC [8], where one or more with-loops are used to implement each array skeleton. With-loop fusion allows an efficient parallel implementation of the array skeletons. In this way, the authors achieve a good balance between expressibility and performance, balancing the demand for fine-grained skeletons for software engineering purposes, with that for coarse-grained skeletons for higher performance. With-loop fusion also forms a key optimisation in the compiler from SAC to the μTC compiler target language [20], which is a key part of the EU Framework 7 Apple-Core project (FP7-215216). μTC is a high-level assembly language which targets many-core chip multi-processors (or *Microgrids*). The work exploits new optimisation techniques for with-loops to primitive μTC operations.

2007: Jost Berthold and Rita Loogen [2]

The 2007 Peter Landin prize was awarded to Jost Berthold and Rita Loogen of Philipps-Universität, Marburg, Germany for their paper on *Parallel Coordination made Explicit in a Functional Setting* [2]. This paper presents a low-level parallel coordination language for Haskell that can be used to implement a variety of parallel extensions for Haskell as part of the widely-used GHC compiler. Their EDI system provides a small set of basic Haskell primitives for thread control, system information and communication. This set of primitives can then be used to encode higher-level constructs, such as those found in the Eden parallel Haskell dialect [17]. In this way, a complete parallel runtime system can be written in Haskell rather than in some lower-level language such as C. The key advantages of this approach are that it considerably simplifies the task of writing new parallel implementations; and that it becomes much easier to ensure that the parallel implementation remains in step with changes in the underlying sequential implementation. The authors show that the approach carries acceptable overheads: the EDI implementation can encode the same constructs as the higher-level Eden system without any loss of efficiency. This system is being exploited as part of ongoing and topical effort aimed at exploiting high-level skeletons

in Eden, including an implementation of Google's map-reduce skeleton [3] and a skeleton for fast-Fourier transforms [16], for work aimed at obtaining low-cost, high-performance parallel computing as part of the ongoing EU Framework 6 Sciences project (RII3-026133) [24, 25], and for work that explores the use of divide-and-conquer skeletons on many-core architectures [4].

2008: Neil Mitchell and Colin Runciman [18]

The 2008 Peter Landin prize was awarded to Neil Mitchell and Colin Runciman of the University of York, UK for their paper on *A Supercompiler for Core Haskell* [18]. While functional languages allow very programs to be expressed in a very elegant way, they are often less efficient than good imperative code. Unlike conventional compilation, a *supercompiler* evaluates some of the program at compile-time, so producing an optimised residual program with superior performance properties. The paper represents the first time that supercompilation has been applied to Haskell, and makes a major contribution in studying the use of supercompilation for let-expressions, which are an important construct in functional intermediate languages. Using their techniques, the authors show that it is possible to eliminate a number of unnecessary overheads including intermediate lists, functional arguments that are used to instantiate higher-order definitions, and unnecessary laziness and thunks. In this way, it is possible to achieve software that is not merely as efficient as its imperative counterpart, but which can actually be *faster*. Overall, the supercompilation approach achieves a 16% improvement in performance compared with the already highly optimising GHC Haskell implementation: an impressive result.

2009: Ralf Hinze [11]

Most recently (but not finally), the 2009 Peter Landin prize was awarded to Ralf Hinze of the University of Oxford, UK for his paper on *Scans and Convolutions: a Calculational Proof of Moessner's Theorem* [11]. In this paper, Hinze revisits a scheme for generating the natural *k*th powers that was originally described by Alfred Moessner in the 1950s. The paper introduces two *co-recursion* schemes for stream-generating functions, *scans* and *convolutions*, liberating them from their number-theoretic roots through the use of modern programming language theory. In the process, Hinze shows how *scans* and *convolutions* can be represented in Haskell, and exposes a number of useful *free theorems* by turning them into polymorphic combinators. In this way, Hinze has introduced us to new language structures, with strong theoretical roots, that will undoubtedly have significant longer-term impact.

4 Conclusion

Peter Landin's ground breaking work on functional programming languages has proved a consistent inspiration and source of ideas in the research community, for both practical and theoretical ends. It is a measure of his genius that his work has such a pervasive appeal, and a measure of his greatness that his legacy has proved permanent in our fast-paced field of research. The papers that have been awarded the Peter Landin prize to date, and those that will be awarded the prize in the future, form a lasting legacy of living research that perpetuates Peter Landin's name and reflects his own seminal contributions to research.

References

1. Abramsky, S., Sykes, R.: SECD-M: a virtual machine for applicative programming. In: Jouannaud, J.-P. (ed.) Functional Programming Languages and Computer Architecture, Nancy, France, September 1985. Lecture Notes in Computer Science, vol. 201, pp. 81–98. Springer, Berlin (1985)
2. Berthold, J., Loogen, R.: Parallel coordination made explicit in a functional setting. In: Horváth, Z., Zsók, V., Butterfield, A. (eds.) Implementation and Application of Functional Languages, 18th International Symposium, IFL 2006, Budapest, Hungary, September 4–6, 2006. Lecture Notes in Computer Science, vol. 4449, pp. 73–90. Springer, Berlin (2007). Revised Selected Papers
3. Berthold, J., Dieterle, M., Loogen, R.: Implementing parallel Google map-reduce in Eden. In: Sips, H., Epema, R., Lin, H.-X. (eds.) Proc. Euro-Par 2009: 15th International Euro-Par Conference, Delft, the Netherlands. Lecture Notes in Computer Science, vol. 5704, pp. 990–1002. Springer, Berlin (2009)
4. Berthold, J., Dieterle, M., Lobachev, O., Loogen, R.: Distributed memory programming on many-cores—a case study using Eden Divide-&-Conquer skeletons. In: Großpitsch, K.-E., Henkersdorf, A., Uhrig, S., Ungerer, T., Hähner, J. (eds.) Proc. ARCS'09—22nd International Conference on Architecture of Computing Systems, March (2009)
5. Burge, W.H.: Recursive Programming Techniques. Addison-Wesley, New York (1975)
6. Danvy, O.: A rational deconstruction of Landin's SECD machine. In: Grelck, C., Huch, F., Michaelson, G., Trinder, P.W. (eds.) Implementation and Application of Functional Languages, 16th International Workshop, IFL 2004, Lübeck, Germany, September 8–10, 2004. Lecture Notes in Computer Science, vol. 3474, pp. 52–71. Springer, Berlin (2005). Revised Selected Papers
7. Danvy, O., Millikin, K.: A rational deconstruction of Landin's SECD machine with the J operator. Log. Methods Comput. Sci. **4**(12), 1–67 (2008)
8. Grelck, C., Scholz, S.-B.: Merging compositions of array skeletons in SAC. Parallel Comput. **32**(7), 507–522 (2006)
9. Grelck, C., Hinckfuß, K., Scholz, S.-B.: With-loop fusion for data locality and parallelism. In: Butterfield, A., Grelck, C., Huch, F. (eds.) Implementation and Application of Functional Languages, 7th International Workshop, IFL 2005, Dublin, Ireland, September 19–21, 2005. Lecture Notes in Computer Science, vol. 4015, pp. 178–195. Springer, Berlin (2006). Revised Selected Papers
10. Hammond, K., Michaelson, G.: Research Directions in Parallel Functional Programming. Springer, Berlin (1999)
11. Hinze, R.: Scans and convolutions: a calculational proof of Moessner's theorem. In: Scholz, S.-B. (ed.) Implementation and Application of Functional Languages, 20th International Symposium, IFL 2008, Hatfield, UK, September 10–12, 2008. Lecture Notes in Computer Science. Springer, Berlin (2009). Revised Selected Papers
12. Hughes, J., Pareto, L., Sabry, A.: Proving the correctness of reactive systems using sized types. In: Proc 1996 ACM Symposium on Principles of Programming Languages—POPL '96, pp. 410–423. St. Petersburg, FL, January (1996)
13. Landin, P.: The mechanical evaluation of expressions. Comput. J. **6**(4), 308–320 (1964)
14. Landin, P.: An abstract machine for designers of computer languages. In: Proceedings of the IFIP Congress 65, pp. 438–439. IFIP (1965)
15. Landin, P.: The next 700 programming languages. Commun. ACM **9**(3), 157–166 (1966)
16. Lobachev, O., Berthold, J., Dieterle, M., Loogen, R.: Parallel FFT using Eden skeletons. In: Proc. 10th Intl. Conference on Parallel Computing Technologies (PACT)'09. Lecture Notes in Computer Science, vol. 5698, pp. 990–1002. Springer, Berlin (2009)
17. Loogen, R., Ortega-Mallén, Y., Peña-Marí, R.: Parallel functional programming in Eden. J. Funct. Program. **15**(3), 431–475 (2005)
18. Mitchell, N., Runciman, C.: A supercompiler for core Haskell. In: Chitil, O., Horváth, Z., Zsók, V. (eds.) Implementation and Application of Functional Languages, 19th International Symposium, IFL 2007, Freiburg, Germany, September 27–29, 2007. Lecture Notes in Computer Science, vol. 5083, pp. 147–164. Springer, Berlin (2008). Revised Selected Papers
19. Scholz, S.-B.: Single assignment C—efficient support for high-level array operations in a functional setting. J. Funct. Program. **13**(6), 1005–1059 (2003)
20. Scholz, S.-B., Herhut, S., Joslin, C.: Architecture paradigms and programming languages for efficient programming of multiple COREs. Apple-Core Deliverable D4.2. http://www.apple-core.info/wp-content/apple-core/2008/01/d42.pdf, May (2009)
21. Vasconcelos, P.B., Hammond, K.: Inferring cost equations for recursive, polymorphic and higher-order functional programs. In: Trinder, P.W., Michaelson, G., Pena, R. (eds.) Implementation of Functional Languages, 15th International Workshop, IFL 2003, Edinburgh, UK, September 8–11, 2003. Lecture Notes in Computer Science, vol. 3145, pp. 86–101. Springer, Berlin (2004). Revised Papers

22. van Weelden, A., Plasmeijer, R.: Towards a strongly typed functional operating system. In: Arts, T., Pena, R. (eds.) Implementation of Functional Languages, 14th International Workshop, IFL 2002, Madrid, Spain, September 2003. Lecture Notes in Computer Science, vol. 2670, pp. 215–231. Springer, Berlin (2003). Revised Selected Papers
23. Wegner, P.: Programming Languages, Information Structures and Machine Organization. McGraw-Hill, New York (1971)
24. Al Zain, A.D., Hammond, K., Berthold, J., Trinder, P., Michaelson, G., Aswad, M.: Low-pain, high-gain multicore programming in Haskell: coordinating irregular symbolic computations on multicore architectures. In: Proc. DAMP '09: 4th International Workshop on Declarative Aspects of Multicore Programming, Savannah, Georgia, USA, January (2009)
25. Al Zain, A.D., Trinder, P.W., Hammond, K., Konovalov, A., Linton, S., Berthold, J.: Parallelism without pain: orchestrating computational algebra components into a high-performance parallel system. In: Proc. IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA'08), pp. 99–112. Sydney, Australia, December (2008)