Eelco Visser: The Oregon Connection

Andrew Tolmach
Portland State University, OR, USA

Abstract

This paper shares some memories of Eelco gathered over the past 25 years as a colleague and friend, and reflects on the nature of modern international collaborations.

2012 ACM Subject Classification Social and professional topics → Historical people

Keywords and phrases Eelco Visser, Stratego, scope graphs

Digital Object Identifier 10.4230/OASIcs.EVCS.2023.27

1 OGI and Stratego

I first met Eelco in June 1997 on the campus of the University of Amsterdam, which was hosting the International Conference on Functional Programming (ICFP). Eelco was just finishing up his Ph.D. on parsing, and had been offered a postdoc position at the Oregon Graduate Institute (OGI) in Portland, where I held a joint appointment in what we called the Pacific Software Research Center (PacSoft). I believe the postdoc invitation initially derived from a connection between Eelco's thesis advisor, Paul Klint, and PacSoft's senior faculty member, Richard Kieburtz. But as someone who worked on practical aspects of (functional) language compilation and tools, I seemed like the faculty member whose interests best matched Eelco's, so it fell mainly to me to sell him on the idea of coming to Oregon. This was not so easy: Eelco already had a clear idea of his career path, and he wanted to be sure that his postdoc year would help him advance along it, and not just be a pleasant vacation after completing his degree. Fortunately, I was able to convince him that PacSoft, which was focused on methods for design and development of domain-specific languages (DSLs), would be a good environment in which to pursue his interests in language tooling.[1]

When Eelco arrived in Portland that Fall, he was already engaged with the idea of developing a particular (meta) DSL of his own, namely a language for defining term rewriting strategies separately from the underlying rewrite rules, using combinators somewhat inspired by operators in process calculi. This idea was motivated by his experiences using existing rewriting tools such as ASF+SDF and ELAN, and it had already been the subject of an initial paper with Bas Luttik [3]. At PacSoft he built several prototype implementations of the language, which shortly after acquired the name Stratego. The first implementation was in MetaML [7], which was being developed by Tim Sheard and Walid Taha at OGI at that time; this was followed by a boot-strapped implementation that ultimately generated C code using the ATerm library [11]. Eelco also worked with another postdoc, Zino Benaissa, to develop an elegant core theory for the strategy combinators [14]. The three of us collaborated to write about the language in the context of a particular application: optimization of expressions in a compiler for RML [8], a dialect of ML developed a little earlier at PacSoft by Dino Oliva and myself. The resulting paper, "Building Program Optimizers with Rewriting Strategies," appeared at ICFP98 [15], and became a foundational citation for Stratego and strategic programming in general.[2]

[1] Given Eelco's life-long love of photography, I suspect that Oregon's (well-deserved) reputation for natural beauty probably helped too.

[2] Incidentally, according to a friend on the program committee, the paper very nearly wasn't accepted. As of 2012, it was the fifth most highly cited paper in the history of the ICFP conference.

Eelco was a great postdoc, of course, because of his ability, energy, and congenial personality. We used to enjoy long rambles around the (rather bucolic) OGI campus; we both found walking conducive to creative thinking out loud. But what impressed me the most about Eelco was his willingness and ability to sit down and *write*[3] about his research ideas while they were still in their formative stages. This quality is pretty rare among aspiring computer scientists, and may be a good predictor of success. In later years, Eelco was kind enough to blame me for having impressed upon him the importance of regular conference publication ("got me going on the treadmill" is the phrase he used). But he certainly didn't need me to teach him that writing helps to clarify thinking, and that research achievements are useless until they are effectively communicated to others.

Stratego became the foundation of Eelco's entire career, both literally and figuratively: it was a key component of nearly all the software systems he built, and it served as his distinctive calling card in the research community. This has perhaps been a mixed blessing: the user base for the language has remained small, which is a barrier to entry for those wishing to contribute to (or in some cases, even to use) Stratego-based tools. On the other hand, using Stratego does encourage novel techniques and perspectives that are not so natural in more type-centered functional languages such as Haskell or ML.[4] Indeed, my experience when observing Eelco program was that he approached most tasks as *syntactic* transformation problems; he always saw the ATerms lurking not far below the surface.

2 Verification and Scope Graphs

At the conclusion of his postdoc, Eelco returned to the Netherlands to take up a faculty position at Utrecht. We exchanged a few visits over the next couple of years, in Portland and in Amsterdam. After that, we met only occasionally at conferences during the subsequent decade. The Stratego mailing list provided a periodic update on his many activities. I was very impressed by the high productivity of Eelco's research group, which he was quick to credit to having excellent students – although he did take pride in his ability to attract and keep them.

Our second period of active collaboration began in 2011. By this time, Eelco had moved to Delft, built a new group, and completed many parts of the Stratego-based language workbench that was now called Spoofax. He was becoming interested in adding a verification dimension to the workbench tools, for example to automatically generate proofs of type soundness for a language definition. Knowing that my own interests had shifted toward verification, he invited me to join in a new research effort to extend Spoofax in this direction [16], which he was able to support with a large (and prestigious) Dutch VICI grant. It is quite pleasing when a former student or postdoc thinks you have something to contribute to their (already) successful enterprise!

As it turned out, the first major fruit of this collaboration was on a somewhat peripheral topic: formalizing the notion of name binding in languages. Spoofax already included a tool called NaBL for specifying name binding, but the NaBL language lacked a clear semantics independent of its current concrete implementation. Since binding resolution is a key requirement for defining static typing, it seemed reasonable to attack this problem before

[3] And often *draw*. His art background evidently encouraged him to express ideas visually too.

[4] In the early days of Stratego, we often used to argue about the costs and benefits of static typing. Despite the early work of Ralf Lämmel [4], it is only quite recently that Stratego has acquired a (gradual) type system as part of the ongoing Ph.D. work of Jeff Smits [6].

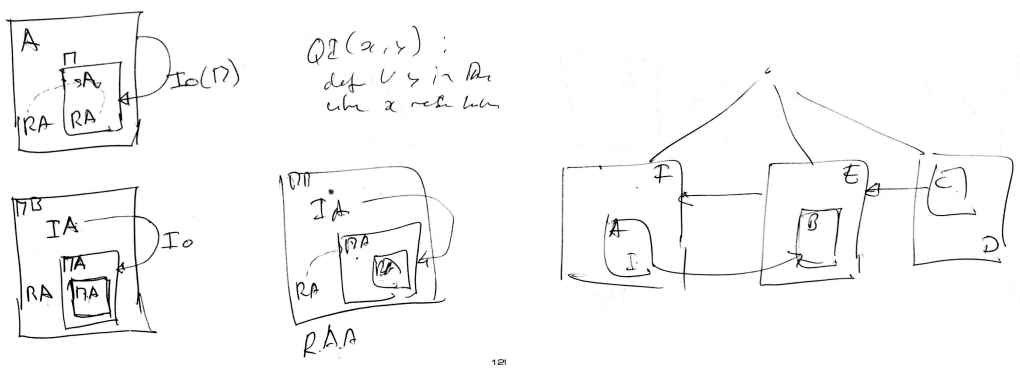

Figure 1 Some of Eelco's highly caffeinated initial sketches for scope graphs.

trying to address verification of type soundness. This effort resulted in the invention of *scope graphs*, a simple but powerful formalism for describing binding structure, with a strong visual intuition. The idea first really took concrete shape under Eelco's pen in a Portland coffee shop in June 2014 (see Figure 1). The paper introducing scope graphs [5], joint with Delft postdoc Pierre Néron and faculty member Guido Wachsmuth, won the Best Paper award at the European Symposium on Programming (ESOP) conference in 2015. We were quite proud of this paper, especially its balance of theory and practicality.

The initial work on scope graphs spawned a good deal of subsequent research, some of which I joined, together with several Delft students, postdocs, and faculty, including Hendrik van Antwerpen, Casper Bach Poulsen, Vlad Vergu, Arjen Rouvoet, and Robbert Krebbers. A first effort was to use scope graphs to support type checking, which clarified that binding and typing are not wholly separable concerns, and led to a unified constraint-based approach to defining and querying scope graphs [10].[5] We also eventually returned to our original verification goals via an approach to dynamic semantics based on heap *frames* that derive their shape and connections from an underlying scope graph. Scopes-and-frames has proved a fruitful paradigm for simplifying proofs of soundness [1], structuring intrinsically typed interpreters [2], and supporting optimization [12]. And there is still more being mined from the basic idea of scope graphs.

Scope graphs are largely about describing data that is explicitly named by the user. But we gradually came to realize the importance of also handling other dynamic data, e.g. the temporaries and control information typically produced by a compiler. This led to a number of lower-level models for defining dynamic semantics, under the name Dynamix, which were an active topic of collaboration among Eelco, Casper Bach Poulsen, and myself during the pandemic years. Much of the work was planned in the form of projects conducted by Delft master's students Chiel Bruin, Bram Crielaard, Thijs Molendijk, and Ruben van Baarle. Eelco's death occurred before the latter two had completed their degrees; Casper and I have done our best to supervise them since then.

The scope graphs collaboration was extremely rewarding, if sometimes a bit intense. Eelco had a persistent habit of juggling far too many tasks at once, and scope graphs were just one cog in his busy research machine. So even without conscious procrastination, lots of things didn't get timely attention, and there would inevitably be a mad drive to complete

[5] Subsequent work, with which I was not directly involved, extended these ideas to deal with structural typing and led to the design of the Statix DSL for static semantics [9].

the work backing a conference submission in the last few days before a deadline. I think Eelco thrived under this regimen. Already in the Preface to his Ph.D. thesis [13, p. iv], he had rhapsodized

> Finally, one of the great contributors to this thesis is the deadline. Soft deadlines, firm deadlines, extended deadlines, nearing deadlines, changed deadlines, passed deadlines, the empowering deadlines that make page after page appear as if by magic, the deadlines that spook at night, deadlines that you can really feel, deadlines that are suddenly there...

While I still prefer to keep my deadlines at a distance, I discovered to my surprise that quite a bit of good science could be done under this kind of time pressure (writing clarifies thinking, once again).

3 Talking to Delft

While Eelco and I did visit each other fairly frequently (thanks in large part to that VICI grant), much of our collaboration was conducted at a distance. Over most of the last ten years, we talked face to face every week or two, courtesy of Skype or Zoom. I still find this quite amazing just on a technological level. When I started my career, even domestic long-distance voice conversations were still expensive, and calls between Europe and the US were only for emergencies. Now it is possible to have a visual, interactive research discussion at an arbitrary distance, essentially for free. It is hard to overestimate how much this development aids international collaboration.

One barrier that does remain is the time difference. Delft is nine hours ahead of Portland, so finding a mutually acceptable time to meet with Eelco required compromise: our regular meeting slot was a little early for me and a little late for him. Paradoxically, I think this slight inconvenience helped us take the meetings more seriously: once they were on the schedule, we didn't want to cancel them lightly.

Our conversations usually focused on the research problem of the moment, and typically included other postdocs and students. But sometimes it was just the two of us, and sometimes we were just checking in to find out how the other was coping with the latest professional, political, or pandemic news. On the professional front, it was an opportunity to speak freely – brag a little or complain a lot – in ways that would be difficult with colleagues in our home institutions. On a personal level, it was a chance to to share ideas across different nations and cultures;[6] Eelco was always interested and engaged in the world beyond computer science. I've spent quite a few happy years living and working in Europe, and talking regularly with Eelco was a way of keeping that transatlantic connection alive – which seems more important than ever in our increasingly separatist world.

In short, my regular talks with Eelco were a rare privilege. I sorely miss them. I sorely miss him.

[6] It could also have been a chance to share them in different languages, except that I have never learned any Dutch at all, to my regret.

References

1 Casper Bach Poulsen, Pierre Néron, Andrew P. Tolmach, and Eelco Visser. Scopes describe frames: A uniform model for memory layout in dynamic semantics. In Shriram Krishnamurthi and Benjamin S. Lerner, editors, *30th European Conference on Object-Oriented Programming, ECOOP 2016, July 18-22, 2016, Rome, Italy*, volume 56 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ECOOP.2016.20.

2 Casper Bach Poulsen, Arjen Rouvoet, Andrew P. Tolmach, Robbert Krebbers, and Eelco Visser. Intrinsically-typed definitional interpreters for imperative languages. *Proceedings of the ACM on Programming Languages*, 2(POPL), 2018. doi:10.1145/3158104.

3 Bas Luttik and Eelco Visser. Specification of rewriting strategies. In M. P. A. Sellink, editor, *2nd International Workshop on the Theory and Practice of Algebraic Specifications (ASF+SDF 1997)*, Electronic Workshops in Computing, Berlin, November 1997. Springer-Verlag.

4 Ralf Lämmel. Typed generic traversal with term rewriting strategies. *The Journal of Logic and Algebraic Programming*, 54(1):1–64, 2003. doi:10.1016/S1567-8326(02)00028-0.

5 Pierre Néron, Andrew P. Tolmach, Eelco Visser, and Guido Wachsmuth. A theory of name resolution. In Jan Vitek, editor, *Programming Languages and Systems - 24th European Symposium on Programming, ESOP 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015. Proceedings*, volume 9032 of *Lecture Notes in Computer Science*, pages 205–231. Springer, 2015. doi:10.1007/978-3-662-46669-8_9.

6 Jeff Smits and Eelco Visser. Gradually typing strategies. In Ralf Lämmel, Laurence Tratt, and Juan de Lara, editors, *Proceedings of the 13th ACM SIGPLAN International Conference on Software Language Engineering, SLE 2020, Virtual Event, USA, November 16-17, 2020*, pages 1–15. ACM, 2020. doi:10.1145/3426425.3426928.

7 Walid Taha and Tim Sheard. Metaml and multi-stage programming with explicit annotations. *Theoretical Computer Science*, 248(1):211–242, 2000. PEPM'97. doi:10.1016/S0304-3975(00)00053-0.

8 Andrew P. Tolmach and Dino Oliva. From ML to Ada: Strongly-typed language interoperability via source translation. *J. Funct. Program.*, 8(4):367–412, 1998. doi:10.1017/s0956796898003086.

9 Hendrik van Antwerpen, Casper Bach Poulsen, Arjen Rouvoet, and Eelco Visser. Scopes as types. *Proceedings of the ACM on Programming Languages*, 2(OOPSLA), 2018. doi:10.1145/3276484.

10 Hendrik van Antwerpen, Pierre Néron, Andrew P. Tolmach, Eelco Visser, and Guido Wachsmuth. A constraint language for static semantic analysis based on scope graphs. In Martin Erwig and Tiark Rompf, editors, *Proceedings of the 2016 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, pages 49–60. ACM, 2016. doi:10.1145/2847538.2847543.

11 Mark van den Brand, Hayco de Jong, Paul Klint, and Pieter A. Olivier. Efficient annotated terms. *Software: Practice and Experience*, 30, 2000.

12 Vlad A. Vergu, Andrew P. Tolmach, and Eelco Visser. Scopes and frames improve meta-interpreter specialization. In Alastair F. Donaldson, editor, *33rd European Conference on Object-Oriented Programming, ECOOP 2019, July 15-19, 2019, London, United Kingdom*, volume 134 of *LIPIcs*. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2019. doi:10.4230/LIPIcs.ECOOP.2019.4.

13 Eelco Visser. *Syntax Definition for Language Prototyping*. PhD thesis, University of Amsterdam, September 1997.

14 Eelco Visser and Zine-El-Abidine Benaissa. A core language for rewriting. *Electronic Notes in Theoretical Computer Science*, 15:422–441, 1998. doi:10.1016/S1571-0661(05)80027-1.

15 Eelco Visser, Zine-El-Abidine Benaissa, and Andrew P. Tolmach. Building program optimizers with rewriting strategies. In Matthias Felleisen, Paul Hudak, and Christian Queinnec, editors, *Proceedings of the third ACM SIGPLAN international conference on Functional programming*, pages 13–26, Baltimore, Maryland, United States, 1998. ACM. doi:10.1145/289423.289425.

16 Eelco Visser, Guido Wachsmuth, Andrew P. Tolmach, Pierre Néron, Vlad A. Vergu, Augusto Passalaqua, and Gabriël Konat. A language designer's workbench: A one-stop-shop for implementation and verification of language designs. In Andrew P. Black, Shriram Krishnamurthi, Bernd Bruegge, and Joseph N. Ruskiewicz, editors, *Onward! 2014, Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software, part of SPLASH '14, Portland, OR, USA, October 20-24, 2014*, pages 95–111. ACM, 2014. doi:10.1145/2661136.2661149.