GADTs Meet Their Match:

Pattern-Matching Warnings That Account for GADTs, Guards, and Laziness

SEBASTIAN GRAF, Karlsruhe Institute of Technology, Germany
SIMON PEYTON JONES, Microsoft Research, UK

ACM Reference Format:
Sebastian Graf and Simon Peyton Jones. 2020. GADTs Meet Their Match:: Pattern-Matching Warnings That Account for GADTs, Guards, and Laziness. *Proc. ACM Program. Lang.* 1, ICFP, Article 1 (January 2020), 6 pages.

Authors' addresses: Sebastian Graf, Karlsruhe Institute of Technology, Karlsruhe, Germany, sebastian.graf@kit.edu; Simon Peyton Jones, Microsoft Research, Cambridge, UK, simonpj@microsoft.com.

2020. 2475-1421/2020/1-ART1 $15.00
https://doi.org/

Pattern Syntax

$$
\begin{array}{rlll}
K \in & \text{Con} & & \\
x, y, a, b \in & \text{Var} & & \\
\tau, \sigma \in & \text{Type} & & \\
e \in & \text{Expr} & ::= & x : \tau \\
& & \mid & K\ \overline{a}\ \overline{\gamma}\ \overline{e : \tau} \\
& & \mid & \ldots \\
\gamma \in & \text{TyCt} & ::= & \tau_1 \sim \tau_2 \mid \ldots \\
g \in & \text{Grd} & ::= & \mathsf{let}\ x : \tau = e; \\
& & \mid & K\ \overline{a}\ \overline{\gamma}\ \overline{y : \tau} \leftarrow x; \\
& & \mid & !x;
\end{array}
$$

Oracle Syntax

$$
\begin{array}{rcll}
\Gamma & ::= & \varnothing \mid \Gamma, x : \tau \mid \Gamma, a & \text{Context} \\
\Theta & ::= & \times \mid \Gamma \vdash \Delta \mid \Theta_1 \vee \Theta_2 & \text{"Deltas"} \\
\Delta & ::= & \checkmark \mid \Delta \wedge \delta & \text{Delta} \\
\delta & ::= & \gamma \mid x_1 \approx x_2 \mid x \approx K\ \overline{y} \mid x \not\approx K \mid x \approx \bot \mid x \not\approx \bot \mid x \approx e & \text{Constraints}
\end{array}
$$

Adding Constraints

$$
\begin{array}{rcrcl}
\times & \oplus & \delta & = & \times \\
\Gamma \vdash \Delta & \oplus & \delta & = & \Gamma \vdash \Delta \wedge \delta \\
\Theta_1 \vee \Theta_2 & \oplus & \delta & = & (\Theta_1 \oplus \delta) \vee (\Theta_2 \oplus \delta)
\end{array}
$$

Binding Free Variables

$$
\begin{array}{rcrcl}
\times & \# & x : \tau & = & \times \\
\Gamma \vdash \Delta & \# & x : \tau & = & \Gamma, x : \tau \vdash \Delta \\
\Theta_1 \vee \Theta_2 & \# & x : \tau & = & (\Theta_1 \# x : \tau) \vee (\Theta_2 \# x : \tau)
\end{array}
$$

Pattern-match Result

$$r ::= \langle \Theta_u, \Theta_d, \Theta_c \rangle$$

$$\langle \Theta_u, \Theta_d, \Theta_c \rangle \cup_U \Theta = \langle \Theta_u \vee \Theta, \Theta_d, \Theta_c \rangle$$
$$\langle \Theta_u, \Theta_d, \Theta_c \rangle \cup_D \Theta = \langle \Theta_u, \Theta_d \vee \Theta, \Theta_c \rangle$$

Pattern-match checking

$$\boxed{\mathsf{pmc}\ \overline{\Theta}\ \overline{\mathsf{Grd}} = r}$$

$$
\begin{array}{lcl}
\mathsf{pmc}\ \Gamma\ \Theta\ \epsilon & = & \langle \times, \times, \Theta \rangle \\
\mathsf{pmc}\ \Gamma\ \Theta\ (\mathsf{let}\ x : \tau = e;\ \overline{g}) & = & \mathsf{pmc}\ \Gamma\ (\Theta \# x : \tau\ \oplus\ x \approx e)\ \overline{g} \\
\mathsf{pmc}\ \Gamma\ \Theta\ (!x;\ \overline{g}) & = & \mathsf{pmc}\ \Gamma\ (\Theta\ \oplus\ x \not\approx \bot)\ \overline{g} \\
& & \quad \cup_D \Theta\ \oplus\ x \approx \bot \\
\mathsf{pmc}\ \Gamma\ \Theta\ (K\ \overline{a}\ \overline{\gamma}\ \overline{x : \tau} \leftarrow y;\ \overline{g}) & = & \mathsf{pmc}\ \Gamma\ (\Theta\ \overline{\#\ a}\ \overline{\#\ x : \tau}\ \overline{\oplus\ \gamma}\ \oplus\ y \approx K\ \overline{x})\ \overline{g} \\
& & \quad \cup_D \Theta\ \oplus\ x \approx \bot \\
& & \quad \cup_U \Theta\ \oplus\ x \not\approx K
\end{array}
$$

TODO LIST

Pattern-match Result

$$\boxed{\texttt{ClauseResult}}$$

$$\begin{array}{rrcl} c \in & \texttt{Coverage} & ::= & \texttt{Redundant} \\ & & \mid & \texttt{RhsInaccessible} \\ & & \mid & \texttt{RhsReachable} \\ \\ r \in & \texttt{ClauseResult} & ::= & \langle \overline{\Delta}, \texttt{Coverage} \rangle \\ \\ & \texttt{empty} & = & \langle \times, \texttt{Redundant} \rangle \end{array}$$

$$\boxed{r \cup_D \overline{\Delta}}$$

$$\begin{array}{rcrcl} \langle \overline{\Delta_u}, \texttt{Redundant} \rangle & \cup_D & \overline{\Delta_d} & = & \langle \overline{\Delta_u}, \texttt{RhsInaccessible} \rangle \text{ if any } \Delta_d \text{ inhabited} \\ r & \cup_D & & = & r \end{array}$$

$$\boxed{r \cup_C \overline{\Delta}}$$

$$\begin{array}{rcrcl} \langle \overline{\Delta_u}, \rangle & \cup_C & \overline{\Delta_c} & = & \langle \overline{\Delta_u}, \texttt{Covered} \rangle \text{ if any } \Delta_c \text{ inhabited} \\ r & \cup_C & & = & r \end{array}$$

$$\boxed{r \cup_U \overline{\Delta}}$$

$$\langle \overline{\Delta_u}, c \rangle \quad \cup_U \quad \overline{\Delta_{u'}} \quad = \quad \langle \overline{\Delta_u}\ \overline{\Delta_{u'}}, c \rangle$$

Pattern-match checking

$$\boxed{\texttt{pmc}\ \overline{\Delta}\ \overline{\texttt{Grd}}\ = r}$$

$$\begin{array}{lcl} \texttt{pmc}\ \overline{\Delta}\ \epsilon & = & \texttt{empty} \cup_C \overline{\Delta} \\ \texttt{pmc}\ \overline{\Delta}\ (\texttt{let}\ x : \tau = e;\ \overline{g}) & = & \texttt{pmc}\ \overline{\Delta \oplus x : \tau \oplus x \approx e}\ \overline{g} \\ \texttt{pmc}\ \overline{\Delta}\ (!x;\ \overline{g}) & = & \texttt{pmc}\ \overline{\Delta \oplus x \not\approx \bot}\ \overline{g} \\ & & \cup_D\ \overline{\Delta \oplus x \approx \bot} \\ \texttt{pmc}\ \overline{\Delta}\ (K\ \overline{a}\ \overline{\gamma}\ \overline{x : \tau} \leftarrow y;\ \overline{g}) & = & \texttt{pmc}\ \overline{\Delta \oplus \overline{a} \oplus \overline{\gamma} \oplus \overline{x : \tau} \oplus x \approx K\ \overline{a}\ \overline{\gamma}\ \overline{x : \tau}}\ \overline{g} \\ & & \cup_D\ \overline{\Delta \oplus x \approx \bot} \\ & & \cup_U\ \overline{\Delta \oplus x \not\approx K} \end{array}$$

REFERENCES

Lennart Augustsson. 1985. Compiling pattern matching. In *Proceedings of the 1985 Conference on Functional Programming and Computer Architecture.*

Edwin Brady. 2013a. Idris, a general-purpose dependently typed programming language: Design and implementation. *Journal of Functional Programming* 23 (9 2013), 552–593. Issue 05. https://doi.org/10.1017/S095679681300018X

Edwin Brady. 2013b. Programming and Reasoning with Algebraic Effects and Dependent Types. In *Proceedings of the 18th ACM SIGPLAN International Conference on Functional Programming (ICFP '13).* ACM, New York, NY, USA, 133–144. https://doi.org/10.1145/2500365.2500581

James Cheney and Ralf Hinze. 2003. *First-class phantom types.* Technical Report. Cornell University.

Koen Claessen, Moa Johansson, Dan Rosén, and Nicholas Smallbone. 2013. Automating Inductive Proofs Using Theory Exploration.. In *CADE (Lecture Notes in Computer Science)*, Maria Paola Bonacina (Ed.), Vol. 7898. Springer, 392–406.

Thierry Coquand. 1992. Pattern matching with dependent types. In *Proceedings of the Workshop on Types for Proofs and Programs.*

Test if Oracle state Delta is unsatisfiable

$$\frac{\boxed{\nvdash_{\text{SAT}} \Gamma \vdash \Delta} \qquad \nvdash_{\text{SAT}} \Gamma \vdash fvs\Gamma \triangleright \Delta}{\nvdash_{\text{SAT}} \Gamma \vdash \Delta}$$

Test a list of SAT roots for inhabitants

$$\frac{\boxed{\nvdash_{\text{SAT}} \Gamma \vdash \overline{x} \triangleright \Delta} \qquad \nvdash_{\text{SAT}} \Gamma \vdash x_i \triangleright \Delta}{\nvdash_{\text{SAT}} \Gamma \vdash \overline{x} \triangleright \Delta}$$

Test a single SAT root for inhabitants

$$\frac{\boxed{\nvdash_{\text{SAT}} \Gamma \vdash x \triangleright \Delta} \qquad \nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta x \approx \bot \quad \{\overline{K}\} \text{ COMPLETE set} \quad \overline{\forall \overline{y : \tau}.\ \nvdash_{\text{SAT}} \Gamma, \overline{y : \tau} \vdash \oplus \Delta x \approx K\ \overline{y}}}{\nvdash_{\text{SAT}} \Gamma \vdash x \triangleright \Delta}$$

Add a single equality to Δ

$$\boxed{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta\delta}$$

Term stuff: Bottom, negative info, positive info + generativity, positive info + univalence

$$\frac{x \not\approx sth \in \Delta}{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta x \approx \bot} \qquad \frac{x \approx K\ \overline{y} \in \Delta}{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta x \approx \bot}$$

$$\frac{x \not\approx K \in \Delta}{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta x \approx K\ \overline{y}} \qquad \frac{x \approx K_i\ \overline{y} \in \Delta \quad i \neq j \quad K_i \text{ and } K_j \text{ generative}}{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta x \approx K_j\ \overline{z}}$$

$$\frac{x \approx K\ \overline{\tau}\ \overline{y} \in \Delta \quad \nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta\tau_i \sim \sigma_i}{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta x \approx K\ \overline{\sigma}\ \overline{z}} \qquad \frac{x \approx K\ \overline{\tau}\ \overline{y} \in \Delta \quad \nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta y_i \approx z_i}{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta x \approx K\ \overline{\sigma}\ \overline{z}}$$

Type stuff: Hand over to unspecified type oracle

$$\frac{\tau_1 \text{ and } \tau_2 \text{ incompatible to Givens in } \Delta \text{ according to type oracle}}{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta\tau_1 \sim \tau_2}$$

Mixed: Instantiate K and see if that leads to a contradiction TODO: Proper instantiation

$$\frac{\overline{\nvdash_{\text{SAT}} \Gamma \vdash y \triangleright \Delta \cup y \not\approx \bot}}{\nvdash_{\text{SAT}} \Gamma \vdash \oplus \Delta x \approx K\ \overline{y}}$$

Joshua Dunfield. 2007a. Refined Typechecking with Stardust. In *Proceedings of the 2007 Workshop on Programming Languages Meets Program Verification (PLPV '07)*. ACM, New York, NY, USA, 21–32. https://doi.org/10.1145/1292597.1292602

Joshua Dunfield. 2007b. *A Unified System of Type Refinements.* Ph.D. Dissertation. Carnegie Mellon University. CMU-CS-07-129.

Richard A. Eisenberg and Stephanie Weirich. 2012. Dependently Typed Programming with Singletons. In *Proceedings of the 2012 Haskell Symposium (Haskell '12)*. ACM, New York, NY, USA, 117–130. https://doi.org/10.1145/2364506.2364522

M Erwig and SL Peyton Jones. 2000. Pattern guards and transformational patterns. In *Proceedings of the 2000 Haskell Symposium*. ACM.

Jacques Garrigue and Jacques Le Normand. 2011. Adding GADTs to OCaml: the direct approach. In *Workshop on ML*.

Jean-Yves Girard, Paul Taylor, and Yves Lafont. 1989. *Proofs and Types.* Cambridge University Press, New York, NY, USA.

Georgios Karachalias, Tom Schrijvers, Dimitrios Vytiniotis, and Simon Peyton Jones. 2015. *GADTs meet their match (extended version).* Technical Report. KU Leuven. http://people.cs.kuleuven.be/~george.karachalias/papers/gadtpm_ext.pdf

Neelakantan R. Krishnaswami. 2009. Focusing on Pattern Matching. In *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '09)*. ACM, New York, NY, USA, 366–378. https://doi.org/10.1145/1480881.1480927

Alain Laville. 1991. Comparison of Priority Rules in Pattern Matching and Term Rewriting. *J. Symb. Comput.* 11, 4 (May 1991), 321–347. https://doi.org/10.1016/S0747-7171(08)80109-5

Fabrice Le Fessant and Luc Maranget. 2001. Optimizing Pattern-Matching. In *Proceedings of the 2001 International Conference on Functional Programming*.

Luc Maranget. 1992. Compiling Lazy Pattern Matching. In *Proceedings of the 1992 ACM Conference on LISP and Functional Programming (LFP '92)*. ACM, New York, NY, USA, 21–31. https://doi.org/10.1145/141471.141499

Luc Maranget. 2007. Warnings for pattern matching. *Journal of Functional Programming* 17 (2007), 387–421. Issue 3.

Luc Maranget. 2008. Compiling pattern matching to good decision trees. In *Proceedings of the ACM Workshop on ML*.

Luc Maranget and Projet Para. 1994. *Two Techniques for Compiling Lazy Pattern Matching.* Technical Report.

The Coq development team. 2004. *The Coq proof assistant reference manual.* LogiCal Project. http://coq.inria.fr Version 8.0.

C. McBride and J. McKinna. 2004. The view from the left. *Journal of Functional Programming* 14, 1 (2004), 69–111.

Neil Mitchell and Colin Runciman. 2008. Not All Patterns, but Enough: An Automatic Verifier for Partial but Sufficient Pattern Matching. In *Proceedings of the First ACM SIGPLAN Symposium on Haskell (Haskell '08)*. ACM, New York, NY, USA, 49–60. https://doi.org/10.1145/1411286.1411293

Ulf Norell. 2007. *Towards a practical programming language based on dependent type theory.* Ph.D. Dissertation. Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden.

Ulf Norell. 2008. Dependently typed programming in Agda. In *In Lecture Notes from the Summer School in Advanced Functional Programming*.

Simon Peyton Jones, Dimitrios Vytiniotis, Stephanie Weirich, and Geoffrey Washburn. 2006. Simple Unification-based Type Inference for GADTs. In *Proceedings of the Eleventh ACM SIGPLAN International Conference on Functional Programming (ICFP '06)*. ACM, New York, NY, USA, 50–61. https://doi.org/10.1145/1159803.1159811

Norman Ramsey, João Dias, and Simon Peyton Jones. 2010. Hoopl: A Modular, Reusable Library for Dataflow Analysis and Transformation. In *Proceedings of the Third ACM Haskell Symposium on Haskell (Haskell '10)*. ACM, New York, NY, USA, 121–134. https://doi.org/10.1145/1863523.1863539

Patrick M. Rondon, Ming Kawaguci, and Ranjit Jhala. 2008. Liquid Types. In *Proceedings of the 2008 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '08)*. ACM, New York, NY, USA, 159–169. https://doi.org/10.1145/1375581.1375602

Tom Schrijvers, Simon Peyton Jones, Manuel Chakravarty, and Martin Sulzmann. 2008. Type Checking with Open Type Functions. In *Proceedings of the 13th ACM SIGPLAN International Conference on Functional Programming (ICFP '08)*. ACM, New York, NY, USA, 51–62. https://doi.org/10.1145/1411204.1411215

Tom Schrijvers, Simon Peyton Jones, Martin Sulzmann, and Dimitrios Vytiniotis. 2009. Complete and Decidable Type Inference for GADTs. In *Proceedings of the 14th ACM SIGPLAN International Conference on Functional Programming (ICFP '09)*. ACM, New York, NY, USA, 341–352. https://doi.org/10.1145/1596550.1596599

R. C. Sekar, R. Ramesh, and I. V. Ramakrishnan. 1995. Adaptive Pattern Matching. *SIAM J. Comput.* 24, 6 (Dec. 1995), 1207–1234. https://doi.org/10.1137/S0097539793246252

Peter Sestoft. 1996. ML pattern match compilation and partial evaluation. In *Partial Evaluation*, Olivier Danvy, Robert Glück, and Peter Thiemann (Eds.). Lecture Notes in Computer Science, Vol. 1110. Springer Berlin Heidelberg, 446–464. https://doi.org/10.1007/3-540-61580-6_22

Tim Sheard. 2004. Languages of the Future. In *In OOPSLA '04: Companion to the 19th annual ACM SIGPLAN conference on Object-oriented programming systems, languages, and applications*. ACM Press, 116–119.

W Sonnex, S Drossopoulou, and S Eisenbach. 2012. Zeno: An Automated Prover for Properties of Recursive Data Structures. Springer-Verlag Berlin, 407–421. https://doi.org/10.1007/978-3-642-28756-5_28

Martin Sulzmann, Manuel M. T. Chakravarty, Simon Peyton Jones, and Kevin Donnelly. 2007. System F with Type Equality Coercions. In *Proceedings of the 2007 ACM SIGPLAN International Workshop on Types in Languages Design and*

Implementation (TLDI '07). ACM, New York, NY, USA, 53–66. https://doi.org/10.1145/1190315.1190324

Peter Thiemann. 1993. Avoiding Repeated Tests in Pattern Matching. In *3rd International Workshop on Static Analysis*, Gilberto Filé (Ed.). Padova, Italia, 141–152.

Niki Vazou, Eric L. Seidel, Ranjit Jhala, Dimitrios Vytiniotis, and Simon Peyton-Jones. 2014. Refinement Types for Haskell. In *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming (ICFP '14)*. ACM, New York, NY, USA, 269–282. https://doi.org/10.1145/2628136.2628161

Dimitrios Vytiniotis, Simon Peyton jones, Tom Schrijvers, and Martin Sulzmann. 2011. Outsidein(x) Modular Type Inference with Local Assumptions. *J. Funct. Program.* 21, 4-5 (Sept. 2011), 333–412. https://doi.org/10.1017/S0956796811000098

Philip Wadler. 1987a. Efficient compilation of pattern matching. In *The implementation of functional programming languages*, SL Peyton Jones (Ed.). Prentice Hall, 78–103.

P. Wadler. 1987b. Views: A Way for Pattern Matching to Cohabit with Data Abstraction. In *Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages (POPL '87)*. ACM, New York, NY, USA, 307–313. https://doi.org/10.1145/41625.41653

Hongwei Xi. 1998a. Dead Code Elimination Through Dependent Types. In *Proceedings of the First International Workshop on Practical Aspects of Declarative Languages (PADL '99)*. Springer-Verlag, London, UK, 228–242.

Hongwei Xi. 1998b. *Dependent Types in Practical Programming*. Ph.D. Dissertation. Carnegie Mellon University.

Hongwei Xi. 2003. Dependently typed pattern matching. *Journal of Universal Computer Science* 9 (2003), 851–872.

Hongwei Xi, Chiyan Chen, and Gang Chen. 2003. Guarded Recursive Datatype Constructors. In *Proceedings of the 30th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '03)*. ACM, New York, NY, USA, 224–235. https://doi.org/10.1145/604131.604150

Dana N. Xu. 2006. Extended Static Checking for Haskell. In *Proceedings of the 2006 ACM SIGPLAN Workshop on Haskell (Haskell '06)*. ACM, New York, NY, USA, 48–59. https://doi.org/10.1145/1159842.1159849

Dana N. Xu, Simon Peyton Jones, and Koen Claessen. 2009. Static Contract Checking for Haskell. In *Proceedings of the 36th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '09)*. ACM, New York, NY, USA, 41–52. https://doi.org/10.1145/1480881.1480889

Brent A. Yorgey, Stephanie Weirich, Julien Cretin, Simon Peyton Jones, Dimitrios Vytiniotis, and José Pedro Magalhães. 2012. Giving Haskell a Promotion. In *Proceedings of the 8th ACM SIGPLAN Workshop on Types in Language Design and Implementation (TLDI '12)*. ACM, New York, NY, USA, 53–66. https://doi.org/10.1145/2103786.2103795