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

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

Guard Syntax

$$
\begin{array}{rcll}
K \in & \mathrm{Con} & & \\
x, y, a, b \in & \mathrm{Var} & & \\
\tau, \sigma \in & \mathrm{Type} & & \\
e \in & \mathrm{Expr} & ::= & x : \tau \\
 & & \mid & K\ \overline{a}\ \overline{\gamma}\ \overline{e : \tau} \\
 & & \mid & \ldots
\end{array}
\qquad
\begin{array}{rcll}
n \in & \mathbb{N} & & \\
\gamma \in & \mathrm{TyCt} & ::= & \tau_1 \sim \tau_2 \mid \ldots \\
g \in & \mathrm{Grd} & ::= & \mathrm{let}\ x : \tau = e \\
 & & \mid & K\ \overline{a}\ \overline{\gamma}\ \overline{y : \tau} \leftarrow x \\
 & & \mid & !x
\end{array}
$$

DNF Syntax

$$
\begin{array}{rcll}
\Gamma & ::= & \varnothing \mid \Gamma, x : \tau \mid \Gamma, a & \text{Context} \\
\delta & ::= & \gamma \mid K\ \overline{x : \tau} \leftarrow y \mid x \not\approx K \mid x \approx \bot \mid x \not\approx \bot \mid x \approx e & \text{Literals} \\
\Delta & ::= & \checkmark \mid \Delta \wedge \delta & \text{Clause} \\
\nabla & ::= & \times \mid \Delta \mid \nabla_1 \vee \nabla_2 & \text{Formula}
\end{array}
$$

$$
\begin{array}{lcl}
\times \oplus \delta & = & \times \\
\Delta \oplus \delta & = & \Delta \wedge \delta \\
\nabla_1 \vee \nabla_2 \oplus \delta & = & (\nabla_1 \oplus \delta) \vee (\nabla_2 \oplus \delta)
\end{array}
$$

Clause Tree Syntax

$$
\begin{array}{rcl}
t_G, u_G \in \mathrm{Gdt} & ::= & \mathtt{Rhs} \mid t_G; u_G \mid \mathtt{Guard}\ g\ t_G \\
t_A, u_A \in \mathrm{Ant} & ::= & \mathtt{AccessibleRhs} \mid \mathtt{InaccessibleRhs} \mid t_A; u_A \mid \mathtt{MayDiverge}\ t_A
\end{array}
$$

Checking Guard Trees

$$\boxed{\mathcal{U}(\nabla, \mathrm{Gdt}) = \nabla}$$

$$
\begin{array}{lcl}
\mathcal{U}(\nabla, \mathtt{Rhs}) & = & \times \\
\mathcal{U}(\nabla, (t; u)) & = & \mathcal{U}(\mathcal{U}(\nabla, t), u) \\
\mathcal{U}(\nabla, \mathtt{Guard}\ (!x)\ t) & = & \mathcal{U}(\nabla \oplus (x \not\approx \bot), t) \\
\mathcal{U}(\nabla, \mathtt{Guard}\ (\mathrm{let}\ x = e)\ t) & = & \mathcal{U}(\nabla \oplus (x \approx e), t) \\
\mathcal{U}(\nabla, \mathtt{Guard}\ (K\ \overline{a}\ \overline{\gamma}\ \overline{y : \tau} \leftarrow x)\ t) & = & (\nabla \oplus (x \not\approx K) \oplus (x \not\approx \bot)) \vee \mathcal{U}(\nabla \oplus (K\ \overline{y : \tau} \leftarrow x) \oplus \overline{\gamma}, gs)
\end{array}
$$

$$\boxed{\mathcal{A}_\Gamma(\nabla, \mathrm{Gdt}) = \mathrm{Ant}}$$

$$
\begin{array}{lcl}
\mathcal{A}_\Gamma(\nabla, \mathtt{Rhs}) & = & \begin{cases} \mathtt{InaccessibleRhs}, & \mathcal{V}(\Gamma, \nabla) \Rightarrow \varnothing \\ \mathtt{AccessibleRhs}, & \text{otherwise} \end{cases} \\
\mathcal{A}_\Gamma(\nabla, (t; u)) & = & \mathcal{A}_\Gamma(\nabla, t); \mathcal{A}_\Gamma(\mathcal{U}(\nabla, t), u) \\
\mathcal{A}_\Gamma(\nabla, \mathtt{Guard}\ (!x)\ t) & = & \begin{cases} \mathcal{A}_\Gamma(\nabla \oplus (x \not\approx \bot), t), & \mathcal{V}(\Gamma, \nabla \oplus (x \approx \bot)) \Rightarrow \varnothing \\ \mathtt{MayDiverge}\ \mathcal{A}_\Gamma(\nabla \oplus (x \not\approx \bot), t) & \text{otherwise} \end{cases} \\
\mathcal{A}_\Gamma(\nabla, \mathtt{Guard}\ (\mathrm{let}\ x = e)\ t) & = & \mathcal{A}_\Gamma(\nabla \oplus (x \approx e), t) \\
\mathcal{A}_\Gamma(\nabla, \mathtt{Guard}\ (K\ \overline{a}\ \overline{\gamma}\ \overline{y : \tau} \leftarrow x)\ t) & = & \mathcal{A}_\Gamma(\nabla \oplus (K\ \overline{y : \tau} \leftarrow x) \oplus \overline{\gamma}, t)
\end{array}
$$

Putting it all together

(0) Input: Context with match vars Γ and desugared Gdt t
(1) Report n value vectors of $\mathcal{V}(\Gamma, \mathcal{U}(\checkmark, t)) \Rightarrow V$ as uncovered
(2) Report the collected redundant and not-redundant-but-inaccessible clauses in $\mathcal{A}_\Gamma(\checkmark, t)$ (TODO: Write a function that collects the RHSs, maybe add numbers to `Rhs` to distinguish).

Generating Inhabitants

$$\boxed{\mathcal{V}(\Gamma, \nabla) \Rightarrow \mathcal{P}(V)}$$

This is `provideEvidence`

$$\frac{}{\mathcal{V}(\Gamma, \times) \Rightarrow \varnothing} \qquad \frac{\mathcal{V}(\Gamma, \nabla_1) \Rightarrow V_1 \quad \mathcal{V}(\Gamma, \nabla_2) \Rightarrow V_2}{\mathcal{V}(\Gamma, \nabla_1 \vee \nabla_2) \Rightarrow V_1 \cup V_2} \qquad \frac{}{\mathcal{V}(\Gamma, \Delta) \Rightarrow \{v \mid \mathcal{V}(\Gamma, \Delta) \Rightarrow v\}}$$

$$\boxed{\mathcal{V}(\Gamma, \Delta) \Rightarrow V}$$

$$\frac{\mathcal{T}(\Delta)}{\mathcal{V}(\varnothing, \Delta) \Rightarrow ()} \qquad \frac{\mathcal{V}((x_1 : \sigma_1, \ldots, x_n : \sigma_n, \Gamma), (K\ (x_1 : \sigma_1) \ldots (x_n : \sigma_n) \leftarrow y, \Delta)) \Rightarrow (a_1, \ldots, a_n, v_2, \ldots, v_m)}{\mathcal{V}(y : \tau, \Gamma, \Delta) \Rightarrow (K\ x_1\ \ldots\ x_n, v_2, \ldots, v_m)}$$

$$\frac{\text{no more fuel}}{\mathcal{V}(x_1 : \tau_1, \ldots, x_n : \tau_n, \Delta) \Rightarrow (\_, \ldots, \_)}$$

$$\boxed{\mathcal{T}(\Delta)}$$

Test a Δ for satisfiability

This figure is completely out of date, don't waste your time

Test if Oracle state Delta is unsatisfiable

$$\boxed{\nvdash_{\textsc{Sat}} \Gamma \vdash \Delta}$$

$$\frac{\nvdash_{\textsc{Sat}} \Gamma \vdash fvs\Gamma \triangleright \Delta}{\nvdash_{\textsc{Sat}} \Gamma \vdash \Delta}$$

Test a list of SAT roots for inhabitants

$$\boxed{\nvdash_{\textsc{Sat}} \Gamma \vdash \overline{x} \triangleright \Delta}$$

$$\frac{\nvdash_{\textsc{Sat}} \Gamma \vdash x_i \triangleright \Delta}{\nvdash_{\textsc{Sat}} \Gamma \vdash \overline{x} \triangleright \Delta}$$

Test a single SAT root for inhabitants

$$\boxed{\nvdash_{\textsc{Sat}} \Gamma \vdash x \triangleright \Delta}$$

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

Add a single equality to Δ

$$\boxed{\nvdash_{\textsc{Sat}} \Gamma \vdash \oplus\Delta\delta}$$

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

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

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

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

Type stuff: Hand over to unspecified type oracle

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

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

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