Partiality Wrecks GADTs' Functoriality

Pierre Cagne and Patricia Johann

Appalachian State University, Boone NC, USA
`cagnep@appstate.edu`, `johannp@appstate.edu`

Generalized Algebraic Data Types (GADTs) are, as their name suggests, syntactic generalizations of standard algebraic data types (ADTs) such as list, trees, etc. ADTs are known to support an initial algebra semantics (IAS) in any category with enough structure [MA86]. This IAS gives a semantic justification for the syntactic tools ADTs come with: pattern-matching, recursion rules, induction rules, etc. One of the fundamental properties of an IAS is that the interpretation of the type constructor defined by an ADT can be extended to a functor whose action on morphisms interprets the ADT's syntactic map function.

It is natural to explore a potential generalization of IAS to GADTs. However, mapping functions over elements of GADTs is notorious for being only partially defined [JG08, JC22, JP19]. This compels us to seek the potential generalized semantics in categories with an inherent notion of partiality.

In this work, we first define a categorical framework which captures a notion of partiality that is computationally relevant. We consider the main feature of computationally relevant partiality to be that functions propagate undefinedness. This is akin to how functions that are strict in the sense of [BHA86] behave. Next, we show that any semantics in this framework is trivial if we insist that the interpretations of the type constructors defined by GADTs must extend to functors.

Given a category $\mathcal{C}$, we write $\mathsf{Mor}(\mathcal{C})$ for its (possibily large) set of morphisms. Let us start by recalling a classic definition that categorifies the notion of ideal in monoids.

Definition 1. A *cosieve* in a category $\mathcal{C}$ is a (possibly large) subset $S \subseteq \mathsf{Mor}(\mathcal{C})$ such that for all morphisms $f : A \to B$ and $g : B \to C$ in $\mathcal{C}$, if $f \in S$ then $gf \in S$.

Recall that a *wide subcategory* of a category $\mathcal{C}$ is a subcategory of $\mathcal{C}$ that contains all objects of $\mathcal{C}$ (and thus all identity morphisms as well). Given any subcategory $\mathcal{D}$ of $\mathcal{C}$, we denote $\overline{\mathcal{D}}$ for its *complement*, i.e., for the (possibly large) set $\mathsf{Mor}(\mathcal{C}) \setminus \mathsf{Mor}(\mathcal{D})$.

Definition 2. A *structure of computational partiality* on a category $\mathcal{C}$ is a wide subcategory whose complement is a cosieve.

In a category $\mathcal{C}$ equipped with a structure of computational partiality $\mathcal{D}$, we call morphisms of $\mathcal{D}$ *total* and those of $\overline{\mathcal{D}}$ *properly partial*. The intuition behind Definition 2 is that $\overline{\mathcal{D}}$ is the collection of partial computations. Following that intuition, both identities and compositions of total functions must be total functions, and, when a function yields an error on an input there is no way to come back from the error by postcomposing with another function. Other categorical frameworks capturing partiality include p-categories [RR88], (bi)categories of partial maps [Car87], categories of partial morphisms [CO89], and restriction categories [CL02]. These all give rise to structures of computational partiality.

Lemma 3. *In a category equipped with a structure of computational partiality, split monomorphisms are always total.*

Proof. Let s be a split monomorphism in a category $\mathcal{C}$. Then there exists r such that $rs = \mathrm{id}$. If s were properly partial in a structure of computational partiality on $\mathcal{C}$, then rs, and thus id, would be properly partial as well. But id is total by definition, so it cannot be. Thus, s is total. □

Now fix a category $\mathcal{C}$ equipped with a structure of computational partiality $\mathcal{D}$. Suppose $\mathcal{D}$ has finite products, and write 1 for the terminal object of $\mathcal{D}$. An *interpretation* $[\![\_]\!]$ *of (a language with) GADTs in* $(\mathcal{C}, \mathcal{D})$ maps each closed type τ of the language to an object $[\![\tau]\!]$ of $\mathcal{C}$, and each function $f : \tau_1 \to \tau_2 \to \dots \to \tau_n \to \tau$ to a total morphism $[\![f]\!] : [\![\tau_1]\!] \times [\![\tau_2]\!] \times \cdots \times [\![\tau_n]\!] \to [\![\tau]\!]$ in $\mathcal{D}$. We require that $[\![\_]\!]$ maps the unit type $\top$ to 1 and compositions of syntactic functions to the compositions of their interprations in $\mathcal{C}$. Given a n-ary GADT `G`, a functor $G : \mathcal{C}^n \to \mathcal{C}$ *manifests* `G` *relative to* $[\![\_]\!]$ if the action of G on every object $([\![\tau_1]\!], \dots, [\![\tau_n]\!])$ is precisely $[\![\mathtt{G}\, \tau_1 \dots \tau_n]\!]$.

Theorem 4. *Let $\mathcal{C}$ be a category equipped with structure of computational partiality $\mathcal{D}$. Suppose $[\![\_]\!]$ is an interpretation of GADTs in $(\mathcal{C}, \mathcal{D})$ relative to which each GADT can be manifested by a functor. Then $[\![\tau]\!] \simeq 1$ for all non-empty closed types τ.*

Proof. Among the GADTs in our language, we have

```
data Equal :: * → * → * where
  Refl :: ∀ α. Equal α α
```

We can use the recursion rule of GADTs to define:

```
trp :: ∀ {α β}. Equal α β → α → β
trp Refl x = x

trp⁻¹ :: ∀ {α β}. Equal α β → β → α
trp⁻¹ Refl y = y
```

Instantiating α and β to the closed types τ_1 and τ_2, respectively, the anonymous function `λ x → trp`$^{-1}$ `Refl (trp Refl x)` reduces to the identity function on τ_1. By the uniqueness property of functions defined by recursion on GADTs, `λ p → (λ x → trp`$^{-1}$ `p (trp p x))` reduces to the identity on τ_1 for any input `p`. Semantically this translates to the following composition being $\mathrm{id}_{[\![\tau_1]\!]}$ for any morphism $p : 1 \to [\![\mathtt{Equal}\ \tau_1\ \tau_2]\!]$ in $\mathcal{D}$:

$$[\![\mathtt{trp}^{-1}\ \{\alpha\ \mathtt{=}\ \tau_1\}\ \{\beta\ \mathtt{=}\ \tau_2\}]\!] \circ (p \times \mathrm{id}_{[\![\tau_2]\!]}) \circ \varphi_{[\![\tau_2]\!]} \circ [\![\mathtt{trp}\ \{\alpha\ \mathtt{=}\ \tau_1\}\ \{\beta\ \mathtt{=}\ \tau_2\}]\!] \circ (p \times \mathrm{id}_{[\![\tau_1]\!]}) \circ \varphi_{[\![\tau_1]\!]}$$

Here, φ_X is the canonical isomorphism $X \simeq 1 \times X$.

Now let τ be a non-empty closed type and t be a closed term of type τ. We abuse notation and write $[\![t]\!] : 1 \to [\![\tau]\!]$ for the morphism $[\![\lambda\ \_\ \to t]\!]$ in $\mathcal{D}$. Since every morphism with domain 1 in $\mathcal{D}$ is a split monomorphism, so is $[\![t]\!]$. Since there exists a functor $[\![\mathtt{Equal}]\!] : \mathcal{C}^2 \to \mathcal{C}$ manifesting `Equal` relative to $[\![\_]\!]$, and since split monomorphisms are preserved by all functors, $[\![\mathtt{Equal}]\!]([\![t]\!], \mathrm{id}_1)$ is a split monomorphism as well. By Lemma 3, $[\![\mathtt{Equal}]\!]([\![t]\!], \mathrm{id}_1)$ is a morphism in $\mathcal{D}$ from $[\![\mathtt{Equal}\ \top\ \top]\!]$ to $[\![\mathtt{Equal}\ \tau\ \top]\!]$. Consider the following morphisms in $\mathcal{D}$:

$$s = [\![\mathtt{trp}^{-1}\ \{\alpha\ \mathtt{=}\ \tau\}\ \{\beta\ \mathtt{=}\ \top\}]\!] \circ (p \times \mathrm{id}_1) \circ \varphi_1 \qquad : 1 \to [\![\tau]\!]$$
$$r = [\![\mathtt{trp}\ \{\alpha\ \mathtt{=}\ \tau\}\ \{\beta\ \mathtt{=}\ \top\}]\!] \circ (p \times \mathrm{id}_{[\![\tau]\!]}) \circ \varphi_{[\![\tau]\!]} \qquad : [\![\tau]\!] \to 1$$

The observation at the end of the previous paragraph instantiated with $\tau_1 = \tau$, $\tau_2 = 1$ and p being the morphism $[\![\mathtt{Equal}]\!]([\![t]\!], \mathrm{id}_1) \circ [\![\mathtt{Refl}\ \{\alpha\ \mathtt{=}\ \top\}]\!]$ in $\mathcal{D}$ shows that $sr = \mathrm{id}_{[\![\tau]\!]}$. The composition rs is necessarily id_1 because it is in $\mathcal{D}$ (i.e., is total), and 1 is terminal in $\mathcal{D}$. This explicitly gives the isomorphism announced in the statement of the theorem. □

The result holds in particular for $\mathcal{D} = \mathcal{C}$. In this case, it proves that any naive extension of IAS for ADTs to GADTs that interprets GADTs as functors on $\mathcal{C}$ directly must be trivial.

References

[BHA86] Geoffrey L. Burn, Chris Hankin, and Samson Abramsky. Strictness analysis for higher-order functions. *Science of Computer Programming*, 7:249–278, 1986.

[Car87] A. Carboni. Bicategories of partial maps. *Cahiers de topologie et géométrie différentielle catégoriques*, 28(2):111–126, 1987.

[CL02] J. R. B. Cockett and Stephen Lack. Restriction categories I: categories of partial maps. *Theoretical Computer Science*, 270(1–2):223–259, January 2002.

[CO89] Pierre-Louis Curien and A. Obtułowicz. Partiality, cartesian closedness, and toposes. *Information and Computation*, 270(1):50–95, January 1989.

[JC22] Patricia Johann and Pierre Cagne. Characterizing Functions Mappable over GADTs. In Ilya Sergey, editor, *Programming Languages and Systems*, pages 135–154, Cham, 2022. Springer Nature Switzerland.

[JG08] Patricia Johann and Neil Ghani. Foundations for structured programming with GADTs. In *Proceedings of the 35th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, volume 43 of *POPL '08*, page 297–308, New York, NY, USA, 2008. Association for Computing Machinery.

[JP19] Patricia Johann and Andrew Polonsky. Higher-Kinded Data Types: Syntax and Semantics. In *34th Annual Symposium on Logic in Computer Science (LICS)*, pages 1–13, 2019.

[MA86] Ernest G. Manes and Michael A. Arbib. *Algebraic Approaches to Program Semantics*. Monographs in Computer Science. Springer New York, 1986.

[RR88] E. Robinson and G. Rosolini. Categories of Partial Maps. *Information and Computation*, 79(2):95–130, November 1988.