# Fun with GADTs

Gowtham Kaki

PURDUE
UNIVERSITY

# GADTs

- Generalized Algebraic Data Types (GADTs) generalize algebraic datatypes of ML/Haskell by letting constructors construct values of different types, but same top-level type constructor.
- Formally introduced into functional programming (Standard ML!) as Guarded Recursive Datatype Constructors (GRDTs) [Xi et al., POPL'03]

```
typecon (type) TY =
  (int) TYint
| {'a,'b}.('a * 'b) TYtup of 'a TY * 'b TY
| {'a,'b}.('a -> 'b) TYfun of 'a TY * 'b TY
| {'a}.('a TY) TYtyp of 'a TY
```

$$
\begin{array}{rcl}
TYint & : & (int)\,TY \\
TYtup & : & \forall\alpha\forall\beta.(\alpha)\,TY * (\beta)\,TY \rightarrow (\alpha * \beta)\,TY \\
TYfun & : & \forall\alpha\forall\beta.(\alpha)\,TY * (\beta)\,TY \rightarrow (\alpha \rightarrow \beta)\,TY \\
TYtyp & : & \forall\alpha\forall\beta.(\alpha)\,TY \rightarrow ((\alpha)\,TY)\,TY
\end{array}
$$

# List examples

- Traditional list ADT
- List GADT that can distinguish between empty and non-empty lists in[ocaml]
- Length-indexed list GADT in [ocaml]
- Length-indexed list GADT using type-level functions, and datakind promotion in [Haskell]

# Printf/Scanf

- With traditional format descriptor
- Naive translation to GADT-based descriptor that turns out to be useless.
- A sensible format GADT, that lets us write sprintf with type sprintf : s fmt → s → string.
- A better format GADT, with sprintf : s fmt → s

# Higher-Order Abstract Syntax

- Consider traditional AST formulation of a toy language similar to untyped LC.
  - Scope-respecting & capture-avoiding alpha renaming is tricky.
  - side-conditions (helper functions) are needed to ensure that substitution is syntactically correct.
- Higher-order abstract syntax is higher-order generalization of abstract syntax trees, which are first-order. The "higher-order" property manifests as higher-order constructors of abstract syntax agains first-order constructors of AST.
- The basic idea is to use meta language's abstraction facilities to represent the object language's binding constructs, so that we get sound alpha-renaming and "syntactically legal" beta-reduction in object language for free.

# Encoding in core ML

- Can we encode GADTs in core ML?