RECURSION, INDUCTION, AND DATA TYPES IN PROGRAMMING LANGUAGES AND PROOF ASSISTANTS

the 2nd Korea Logic Day 2022-01-14

안기영 Ahn, Ki Yung
Hannam University, Daejeon, Korea

TOOLS FOR PROVING THEOREMS USING COMPUTER

- Predefined object logic (SAT, SMT, FOF, ...)
- Formula as input
- Push-button

- a.k.a. Proof Assistants
- Provides meta-logic
- Define your own object logic (syntax, rules)
- Manual proof guide

DEFINITIONS MENTIONING THEMSELVES

- Syntactic
- Computation

- Semantic requirement
  (well-founded, monotonic, ...)
- Logic

DATA TYPE 101

```
data Nat: * where
  Z: Nat
  S: Nat → Nat
```

Recursive? 😊
Inductive? 😊

HIGHER-ORDER DATA TYPE

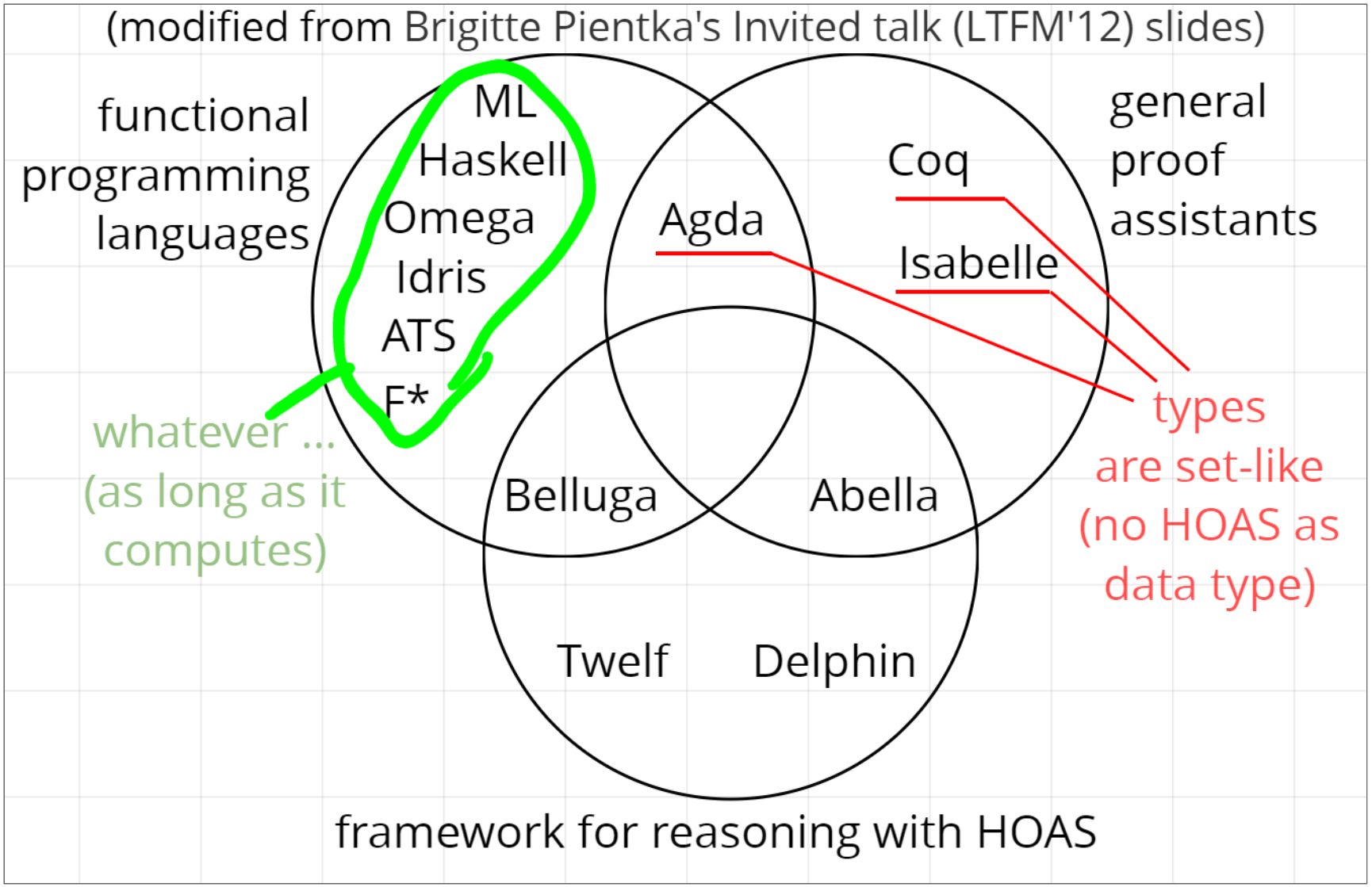
```
data Tm: * where
  App: Tm → Tm → Tm
  Lam: (Tm → Tm) → Tm
```

Recursive? 😊
Inductive? 🤔

HIGHER-ORDER DATA TYPE

```
data Tm: * where
  App: Tm → Tm → Tm
  Lam: (Tm → Tm) → Tm
```

exists a model for Tm (Dana Scott, 1970)


(modified from Brigitte Pientka's Invited talk (LTFM'12) slides)
functional programming languages
ML
Haskell
Omega
Idris
ATS
F*
Agda
Coq
Isabelle
general proof assistants
whatever ... (as long as it computes)
Belluga
Abella
Twelf
Delphin
types are set-like (no HOAS as data type)
framework for reasoning with HOAS

FAMILIES OF PROOF ASSISTANTS

- classical higher-order logic
- Church's simple theory of types
- datatypes and induction rules are built up (as a library) and validated

---

- dependently-typed lambda-calclus
  (intuitionistic higher-order logic)
- datatypes and induction rules are provided (as primitive feature)

CURRY-HOWARD CORRESPONDENCE

Logic	Programming
propositions	types
proofs	programs
proof nomralization	computation

TYPICAL RESTRICTIONS ON DATATYPES

```
data Tm: * where
  App: + → + → Tm
  Lam: (- → +) → Tm
```

Recursive ocurrence allowed only in + positions.
(negative of negative is NOT strictly positive!)
Simple syntactic criterion to ensure well-foundedness.

WHY HOAS NOT SUPPORTED IN MORE PROOF ASSISTANTS?

- Function space for HOAS is more restrictive
  (modulo $\beta\eta$-equiv, $(\lambda x.x) \equiv (\lambda x.(\lambda y.y)x)$)
- When you already have a larger function space ...
- Adding another kind of arrow type is a possibility (has been tried, e.g., **Hybrid project**)

SUMMARY

- HOAS is convinient for specifying object logic syntax
- HOAS is not directly definable with strictily positive datatypes
- There exists proof assistants with good support for HOAS (but not necessarily with all other featrues you need)
- HOAS function space differs from ordirary datatypes