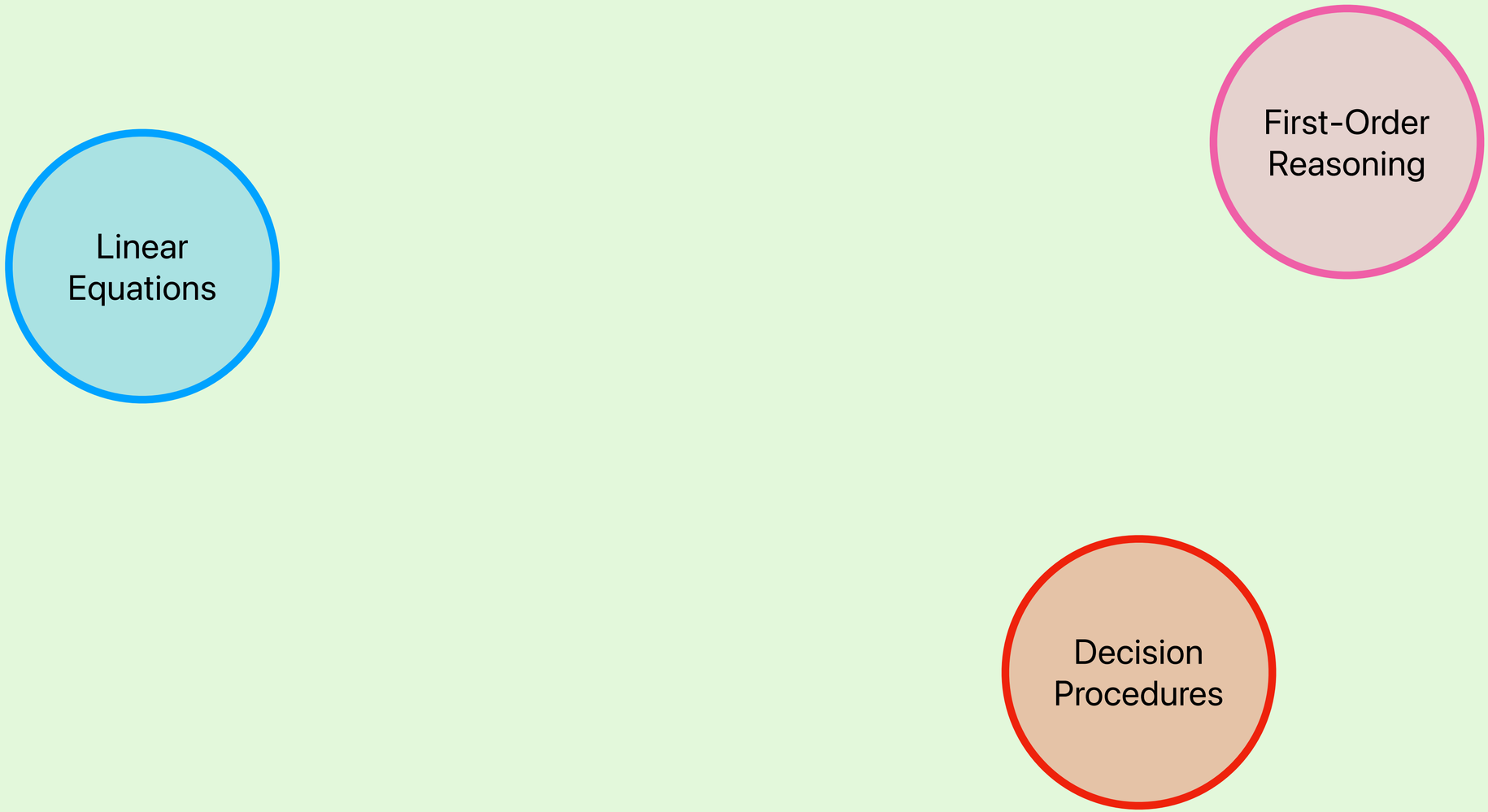
Canonical

Chase Norman, Jeremy Avigad

Proof Automation



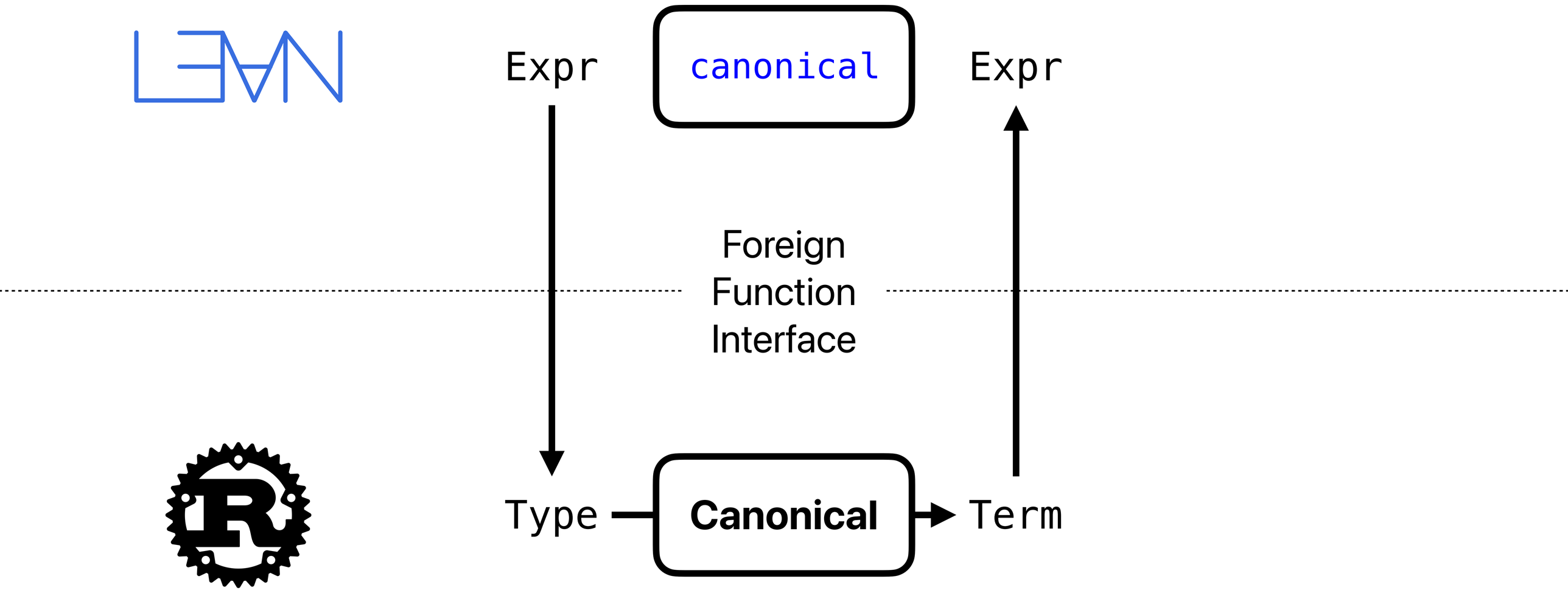
Design Requirements

Support all of Dependent Type Theory

Reason about novel, unseen theories

Provide readable proofs

canonical


Expr
canonical
Expr
Foreign
Function
Interface
Type
Canonical
Term

Type Inhabitation

Propositions ⟶ Proofs

Function Signatures ⟶ Programs

Datatypes ⟶ Objects

Format

```
inductive Expr where
  | bvar : Nat → Expr
  | fvar : FVarId → Expr
  | mvar : MVarId → Expr
  | sort : Level → Expr
  | const : Name → List Level → Expr
  | app : Expr → Expr → Expr
  | lam : Name → Expr → Expr → BinderInfo → Expr
  | forallE : Name → Expr → Expr → BinderInfo → Expr
  | letE : Name → Expr → Expr → Expr → Bool → Expr
  | lit : Literal → Expr
  | mdata : MData → Expr → Expr
  | proj : Name → Nat → Expr → Expr
```

```
structure Term where
  params : List Var
  lets : List Let
  head : String
  args : List Term
```

```
λ params ↦ head args
```

1 term constructor
1 way to represent a term
1 inference rule
1 constraint — DefEq

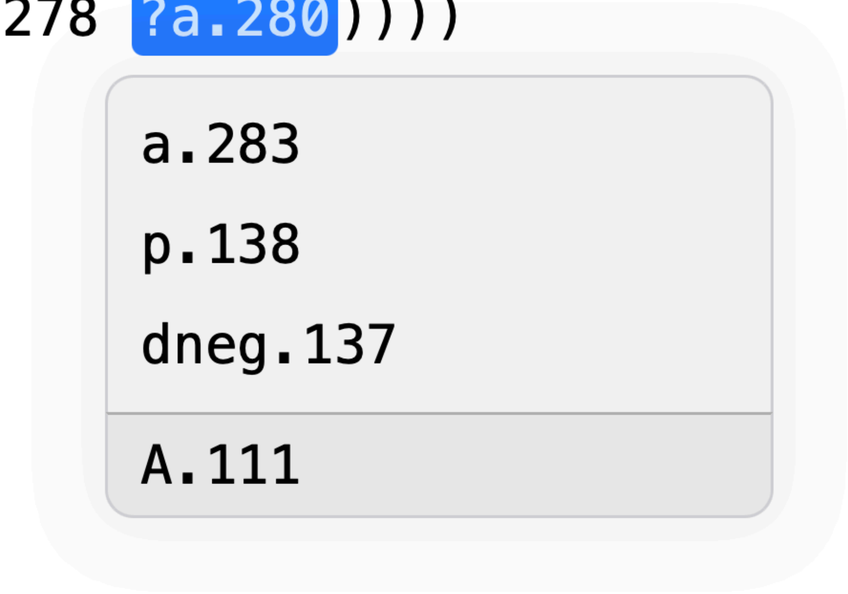
Be discerning with your theory, and you'll be richly rewarded.

Refinement

```
Eq (B b) (subst A a b p (λ _ _ _ ↦ B y) (f a)) (f b)
```

Refinement

```
dneg.137 A.111 (λ a.278 ↦ a.278
(p.138 (λ a.283 ↦ dneg.137 B.113 (λ
a.2781 ↦ a.278 ?a.280))))
```



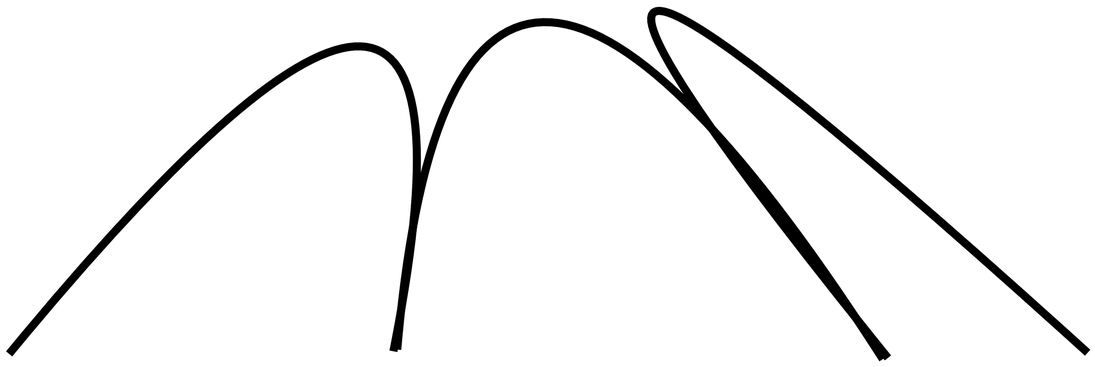
Only canonical forms

Only type-correct options

Any metavariable ordering

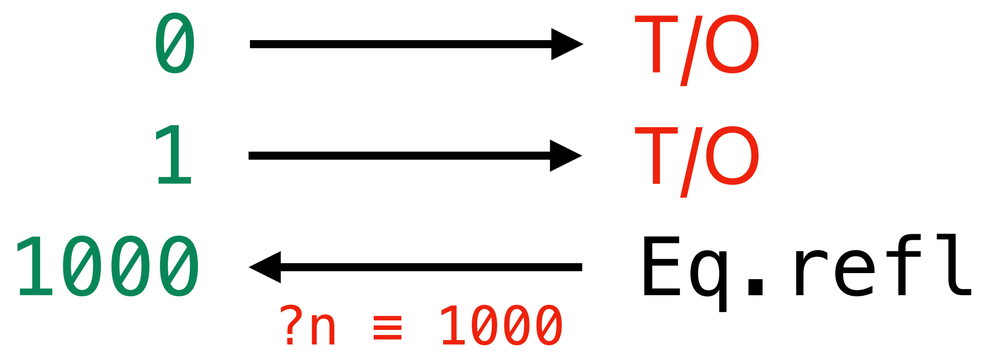
Complete

Iterative Deepening DFS



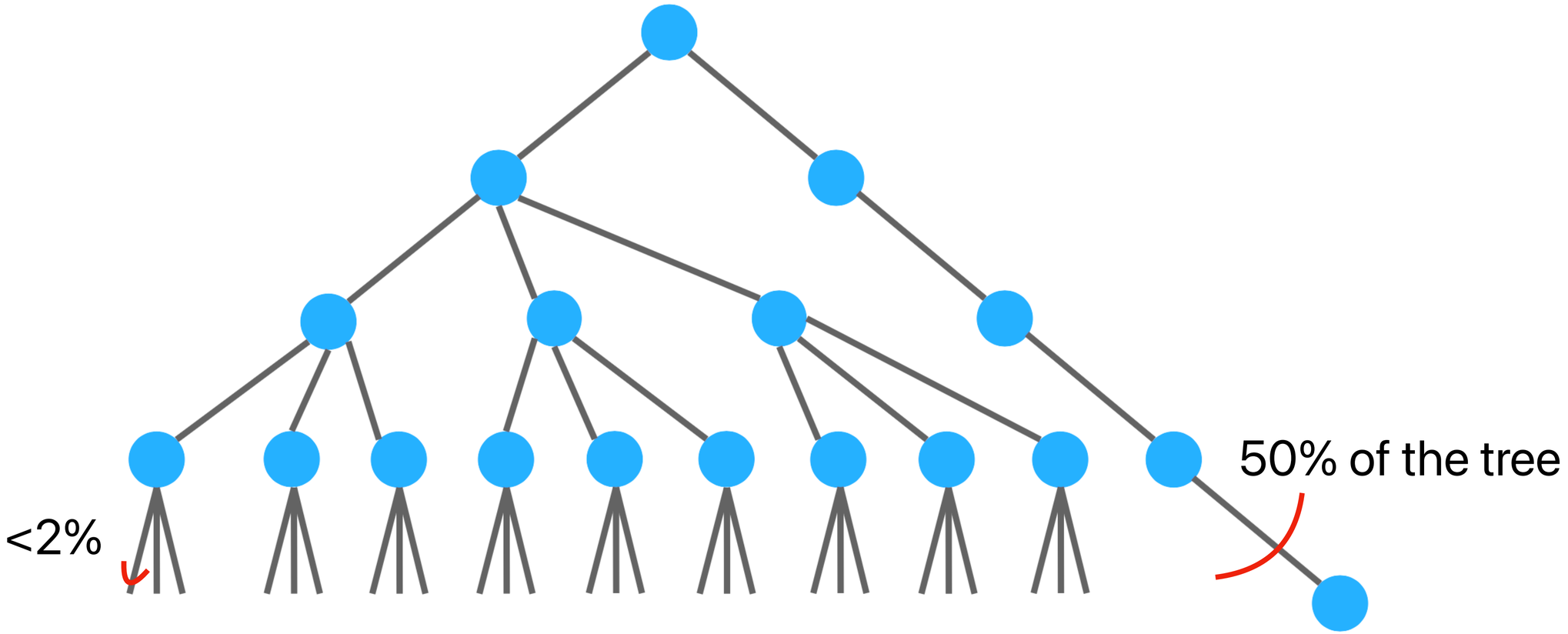
How do we choose?

```
∃ (n : ℕ), (n = 1000)
```

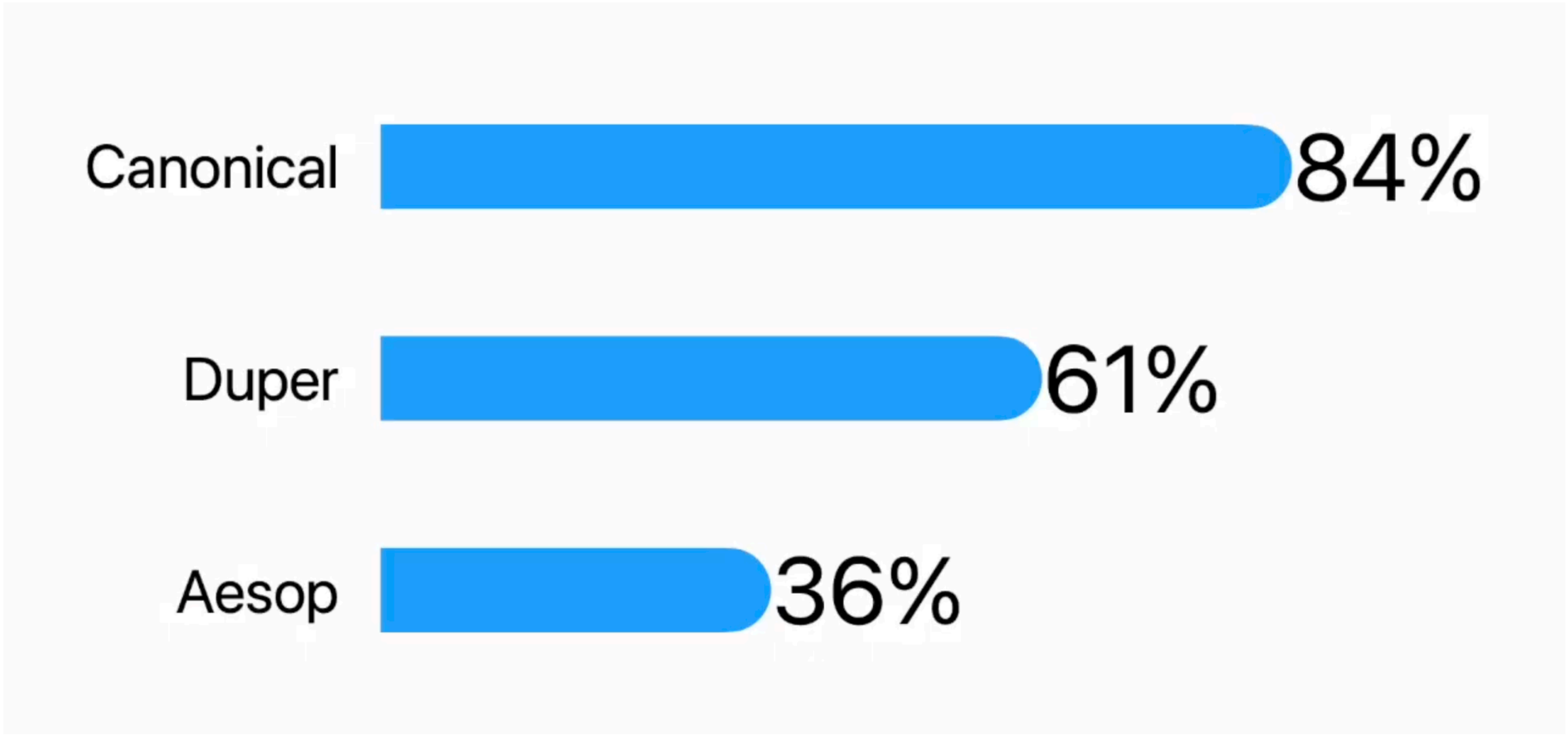


Work on the hardest, rightmost metavariable first.

Entropy Metric



Natural Number Game



Canonical has no native support for natural numbers or equality.


ROCQ


Agda

Canonical

chasenorman.com