An Overview of the Husky Programming Language

Xiyu Zhai

What is Husky

It is a new language for next-generation AI/software.

- it's created out of necessity for next-generation AI.
- it has a novel programming paradigm called **ascension**
- it has a powerful **debugging system**
- it merges the essential features of modern regular languages including C/C++, Rust, python, Haskell, Lean, ATS, etc.

Prerequisites

Prerequisites: Type Theory

A type is basically a set.

Prerequisites: Curry

X, Y, Z are types.
$X \to Y$ denotes a function from X to Y.
$X \to Y \to Z$ denotes a function from X to $Y \to Z$ because $\to$ is right associative.
Note that $(X, Y) \to Z$ and $X \to Y \to Z$ are "equivalent".

Next Generation AI

Next Generation AI: Typed Computation Graph

Next Generation AI: Shape Analysis

Next Generation AI: Image Recognition

Next Generation AI: Natural Language Processing

Next Generation AI: AI Alignment

Ascension Paradigm

Ascension: One Expression for Both Training and Inference

Let C be the **type of contexts** containing training dataset and configurations.
Concept **feature of type** T is defined by

$$\mathscr{F}\,T := C \to \mathsf{Input} \to T$$

.
Given a context c, and an input x, it should provide a value that is the feature trained over c and then evaluated on x.

Ascension: Generic Function Coarse Definition

A generic function

$$\mathrm{gn}(X_1, \cdots, X_n) \to Y$$

can be defined in a coarse way as

$$\mathscr{F} X_1 \to \cdots \to \mathscr{F} X_n \to \mathscr{F} Y, \tag{1}$$

Ascension: Generic Function Refined Definition

A generic function

$$\mathrm{gn}(X_1, \cdots, X_n; \tilde{X}_1, \cdots, \tilde{X}_m) \to Y$$

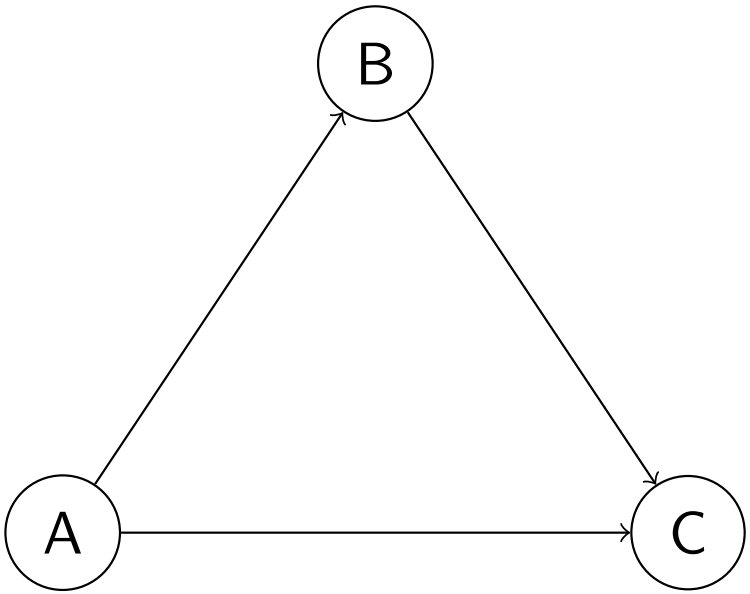
where X_i are normal inputs, and $\tilde{X}_i$ are training-time inputs, can be defined in a more refined way as

$$C \to \underbrace{\mathscr{F} X_1 \to \cdots \to \mathscr{F} X_n}_{\text{all-time inputs for training}} \to \underbrace{\mathscr{F} \tilde{X}_1 \to \cdots \to \mathscr{F} \tilde{X}_n}_{\text{training-time inputs}} \to \underbrace{X_1 \to \cdots \to X_n}_{\text{all-time inputs for training}} \to Y \tag{2}$$

.

Trivally this can be viewed as a subtype of the previous type.

Ascension: Computation Graph



Ascension in General

$\mathscr{F}\ T$

Debugging System

test frame for section one

Regular Feautures

Regular Features

This section we will discuss the regular features of Husky.

- functions
- methods
- values
- type definitions

Regular Features: Type Definition

One can define

- regular struct/structure
- tuple struct/structure
- enum/inductive
-

Regular Features: Borrow Checking

Regular Features: Decorators

must_use, no_discard

Regular Features: Monad through Effect and Unveil

Regular Features: Incremental Code Analysis

Regular Features: Incremental Compilation

Regular Features: Affine Type

Regular Features: Lifetime and Place

Regular Features: Generics

Regular Features: Dependent Types

Development Progress