

placeholder Variables

Xiyu Zhai, Liao Zhang

Latex is so lame. Cite shit Author [2023]here.

1 Introduction

Husky is one of the most sophisticated programming language on earth, designed during my best years. Thus, it's too exhausting to explain it in one short. We shall first write a couple of essays addressing specific aspects and then combine them together.

In this paper, we shall describe the placeholder variables in the Husky programming language, which is the basis of Husky's novel high level syntax and semantics.

A placeholder variables is defined by the keyword `var`. For example,

Husky Code

```
static var INPUT_ID: InputId;
```

We also have type `var`, and `compterm var`.

The intended usage of placeholder variables is inspired from natural language. Consider the following:

Example

Let x be an integer. We say x is positive if $x > 0$. Then if $x = 1$, x is positive. Let $y = x + 1$, then $y|_{x=0} = 1$.

In above, x is not a variable like in python or Rust. It's a contextual placeholder that can be used to construct a term (including both value and proposition) depending on it, with the dependency automatically tracked. It's actually quite similar to tensorflow's placeholder variables, designed for compilation purposes, but much more general.

TODO: ask Liao Zhang, whether this mechanism has true support from certain prover languages like Coq or Lean.

The intended purpose is to have a richer semantics for expressing **values that depends on a set of placeholders** together with **strong type safety**.

2 Related Work

2.1 placeholder Variables as Indices

Husky Code

```
static var I: Fin 8;  
static var J: Fin 8;  
val m: f32 = I * J
```

The above defines a 8×8 matrix $m = (ij)_{0 \leq i, j < 8}$

We can define attention as follows:

Husky Code

```
// This is used anywhere.  
// This is the default placeholder for type 'Pos'  
pub static var POS: Pos;  
// This is only used within current module.  
static var POS_AUX: Pos;  
gn attn(  
  q: Q,  
  k: K,  
  v: V,  
  f: fn(Q, K) -> f32,  
) -> V:  
  // 'of' by default overrides the default placeholder  
  // of the type with the given expr  
  let k = k of POS_AUX // depends on POS, POS_AUX  
  let score = f(q, k) // depends on POS, POS_AUX  
  fold(v, score) // depends on POS  
  
  // Fold along the direction of 'POS_AUX'.  
  //  
  // 'const[POS_AUX]' denotes that the output is deprived  
  // of any dependency on POS_AUX.  
  gn fold(v: V, score: f32) -> const[POS_AUX] V:  
    ...
```

2.2 Mathematica

It does a similar thing, but with dynamic typing. And there's a lack of support for compilation and static analysis.

2.3 Tensor Libraries like Numpy, PyTorch, Tensorflow

References

Some Author. Some relevant paper title. *Journal of Visualization*, 2023.