**IODyn: A High-Level Language for Incremental Computation**

**Problem**

- Incremental computation is critical for efficient, high-performance code.
- Caching and reuse require advanced techniques and languages.
- Current implementations are ad-hoc, and may not be sound.

**Effect**

IODyn offers a simple language including incremental collections.

**Bidirectional type checking**

Quick Hull Sample

Expression: `let hull = { }` where `hull` is a function.

```
let hull = { (x, y) => { return Math.sqrt(x*x + y*y); } }
```

HINTS

Plots use hints like the ones at the beginning of these lines to indicate sections of code that have certain incremental properties. Here, we're labeling each recursive call, meaning that the path through code is not expected to change much. This is the case for quick Hull, which deals with points in space. Paths through space remain the same.

**Background**

IODyn is used as a general-purpose functional language, for simplicity, it translates to Typed Adapton, providing incremental behavior. It makes use of an incremental collections library providing performance for large data sets.

The IODyn project will soon be a full pipeline for compiling simple source code into an optimized incremental executable. It will be able to handle changes to large amounts of data, recompiling in asymptotically less time than initial compilation.

IODyn will allow non-experts of IC to use simple designs to build incremental applications that are type-safe and can incrementally provide (identical results as from-standard code), and with performance-no more complex programs. IODyn will bring general-purpose incremental computing to a wider range of users.

**IODyn Status**

**by Kyle Headley**

Project assisted by Matthew Hammer

More info at kyleheadley.github.io