Pipelines as First-Class Semantics

A denotational account of pipeline types, values, and composition

Lateralus Language

bad-antics · May 2024 · Lateralus Language Research

ABSTRACT We give a denotational semantics for Lateralus pipelines, treating them as first-class semantic objects distinct from function types. The denotation of a pipeline is a morphism in a category whose objects are types and whose arrows are staged computations. Pipeline composition corresponds to morphism composition; the four operator variants correspond to four distinct morphism categories (total, partial, Kleisli, and product). We prove soundness and completeness of the type system with respect to this denotational model and show that the fusion optimization is sound as a bisimulation.

1. Motivation for a Formal Semantics

The design of Lateralus pipeline operators was initially driven by pragmatic concerns: readability, error handling, async composition. A formal semantics is needed for three reasons: to verify that the four operators are not inadvertently overlapping (that they cover distinct semantic ground), to prove that the optimizer's fusion transformation preserves meaning, and to provide a precise foundation for future extensions.

We give the semantics using category theory at the level of introductory graduate courses; no prior exposure to categorical semantics is assumed beyond familiarity with functions, composition, and identity.

2. The Pipeline Category

Let **Lat** be the category where:

- Objects are Lateralus types: A, B, Result<A, E>, Future<A>, etc.
- Arrows from A to B are Lateralus pipeline values of type Pipeline<A, B>.
- Composition of arrows p : A -> B and q : B -> C is the pipeline composition p >> q : A -> C.
- The identity arrow for type A is the one-stage pipeline pipe { |> identity }.

We verify the category axioms. Associativity of composition follows from the associativity of sequential stage execution. Identity law follows from the fact that identity is a total function that returns its argument unchanged.

```
-- Associativity
(p >> q) >> r = p >> (q >> r)

-- Identity
id >> p = p
p >> id = p
```

3. Four Operator Variants as Functor Families

3.1 Total Composition: |>

The total operator |> corresponds to ordinary arrow composition in **Lat**. A total stage f : A -> B is a morphism in the full subcategory of total Lateralus types.

3.2 Error Composition: |?>

The error operator |?> corresponds to Kleisli composition in the Result monad. Define the Kleisli category $\mathbf{Lat}_{\text{Result}}$ where arrows from A to B are morphisms of type A -> Result<B, E> for a fixed E. The |?> operator is the Kleisli composition operator in this category.

```
-- |?> as Kleisli composition
(f |?> g)(x) = match f(x) {
    Ok(v)  => g(v),
    Err(e) => Err(e),
}
```

3.3 Async Composition: |>>

The async operator |>> corresponds to Kleisli composition in the Future monad. The denotation is the sequential chaining of futures: when the left future resolves, its value is passed to the right stage function.

3.4 Fan-Out Composition: |>|

The fan-out operator |>| corresponds to the diagonal morphism in a product category: a single input is duplicated and passed to multiple morphisms in parallel. The result type is the product of the individual output types.

```
-- Fan-out denotation
(x |>| [f, g, h]) = (f(x), g(x), h(x))
```

4. Type Soundness

We prove type soundness by subject reduction: if an expression e has type T and e reduces to e', then e' also has type T.

For pipeline expressions, the reduction relation is defined by the step-by-step execution of pipeline stages. The key lemma is that each operator preserves the type of the pipeline value:

```
-- Preservation for |>
If Gamma |- p : Pipeline<A, B> and p ---> p',
then Gamma |- p' : Pipeline<A, B>.

-- Preservation for |?>
If Gamma |- p : Pipeline<A, Result<B, E>> and p ---> p',
then Gamma |- p' : Pipeline<A, Result<B, E>>.
```

Progress holds trivially for pipeline values: a fully-constructed pipeline value is a terminal form; execution begins only when the pipeline is applied to an input.

5. Fusion as Bisimulation

The fusion optimization merges consecutive fusable stages into a single generated function. We prove fusion soundness by showing that the original and fused pipelines are bisimilar: they produce the same output for every input and have the same error behavior.

Formally, two pipelines p and q are bisimilar (written p ~ q) if for every input x:

```
p(x) = Ok(v)  iff  q(x) = Ok(v)   (same Ok value)
p(x) = Err(e) iff  q(x) = Err(e)  (same Err value)
p(x) diverges iff  q(x) diverges  (same termination)
```

Fusion is the transformation that replaces pipe { |> f |> g } with pipe { |> (fun x -> g(f(x))) }. This is bisimilar by the definition of function composition and the semantics of the total operator.

6. Denotational Model and Compositionality

The denotational model assigns to each pipeline expression a mathematical object in the pipeline category. The model is compositional: the denotation of a compound expression is determined by the denotations of its parts and the denotation of the operator connecting them.

This is a stronger property than mere type soundness: it means that any two pipeline expressions that denote the same mathematical object are interchangeable in any context. This justifies the optimizer's freedom to rewrite pipeline expressions as long as the denotation is preserved.

6.1 Full Abstraction

Full abstraction — the converse of compositionality — holds for the total pipeline subcategory: two total pipelines are interchangeable in all contexts if and only if they denote the same function. Full abstraction fails for the error and async subcategories due to observational differences in error order, a known phenomenon in partial-function denotational semantics.

7. Extensions and Future Work

The categorical model extends naturally to several planned Lateralus features:

- **Parametric polymorphism**: polymorphic pipelines are natural transformations between functors.
- **Effect tracking**: effectful stages can be modeled as arrows in a category enriched with an effect lattice.
- **Dependent pipelines**: pipelines where the output type of stage N depends on the value produced by stage N-1 correspond to fibrations over the pipeline category.

We plan to extend the formal model to cover the fan-in and fallback composition operators, which require product and coproduct structure respectively, and to prove that the interaction between error propagation and async composition is coherent (i.e., does not create race conditions in the error corridor).

8. Conclusion

We have given a categorical denotational semantics for Lateralus pipeline operators, proved type soundness, and established that the fusion optimization is sound as a bisimulation. The four operator variants correspond to four established categorical constructions (arrow composition, Kleisli composition in two monads, and diagonal morphism), confirming that the design covers distinct semantic ground without overlap.

The formal model provides a foundation for compiler correctness proofs and a vocabulary for discussing pipeline semantics precisely in future language extensions.