LADL: Logic As Distributive Law

Mike Stay
Greg Meredith

Curry–Howard Isomorphism

PROPOSITIONS ARE TYPES

Implicational Intuitionistic Logic

Axiom schemata:

S: (Z => Y => X) => (Z => Y) => (Z) => X

K: X => Y => X

Rules:

Modus ponens:

$$\frac{X \Rightarrow Y \qquad X}{Y}$$

SK combinators

ECMAScript 6:

```
// S: (Z => Y => X) => (Z => Y) => (Z) => X
S = a => b => c => a(c)(b(c))

// K: X => Y => X
K = a => b => a
```

Type guards

(ES6)

```
let num = (n) => {
  if ('number' === typeof (n)) {
    return n;
  }
  throw new TypeError('Expected a number');
};
```

Arrow type guards

(ES6)

```
const arrow = (A, B) =>
  (f) =>
    (a) => B(f(A(a)));
```

Set builder notation

{ n | 'number' === typeof n }

{ f | ∀ a:A. f(a):B }

Comprehensions

(Python)

```
[n for n in list if type(n) is int]
```

Wadler: Comprehending monads

$[x] = \text{return } x$

$[f(x) \mid x \leftarrow X] = \text{fmap } f\ X$

$[y \mid x \leftarrow X, y \leftarrow Y(x)] = \text{join} . Y$

Collections as interfaces/monads/Lawvere theories

(Lawvere theory)

Th(FinSet)

- Sorts: S, A
- Function symbols:
  - Atom: A -> S
  - Empty: 1 -> S
  - Union: S x S -> S
- Equations:
  - Unit law: Union(Empty, s) = s = Union(s, Empty)
  - Associative: Union(Union(a, b), c) = Union(a, Union(b, c))
  - Commutative: Union(a, b) = Union(b, a)

Programming languages as Lawvere theories

(Lawvere theory)

Th(SK)

- Sorts: T
- Function symbols:
  - s,t: T -> T
  - S,K: T
  - (- -): T^2 -> T
  - σ: T^3 -> T
  - κ: T^2 -> T

- Equations
  - $s \circ s = s \circ t = s$
  - $t \circ t = t \circ s = t$
  - $(s \circ \sigma)(x, y, z) = (((S\ x)\ y)\ z)$
  - $(t \circ \sigma)(x, y, z) = ((x\ z)\ (y\ z))$
  - $(s \circ \kappa)(x, y) = ((K\ x)\ y)$
  - $(t \circ \kappa)(x, y, z) = x$

Structural types

(TypeScript)

{ "name": "Joe Schmoe", "age": 43, "address": { "number": 123, "street": "Some Ave."} }

is of the type

{ "name": string, "age": number, "address": { "number": number, "street": string} }

Sum of Lawvere theories

1. Identify sorts
2. Union function symbols
3. Union equations

Example

Th(Monoid)

- Sorts: M
- Function symbols:
  - I: 1 -> M
  - ⊗: M^2 -> M
- Equations
  - Unit: $I \otimes x = x = x \otimes I$
  - Assoc: $(a \otimes b) \otimes c = a \otimes (b \otimes c)$

Th(PA)

- Sorts: N
- Function symbols:
  - z: 1 -> N
  - s: N -> N
- No equations

Example

Th(Monoid+PA)

- Sorts: T
- Function symbols:
  - I: 1 -> T // empty list
  - ⊗: T^2 -> T // concat
  - z: 1 -> T // list containing zero
  - s: T -> T // successor
- Equations
  - Unit: I ⊗ x = x = x ⊗ I
  - Assoc: (a ⊗ b) ⊗ c = a ⊗ (b ⊗ c)

Example

sss[z, ss[z, sz], s[]]

Collections of terms

1. Union sorts
2. Union function symbols
3. Union equations
4. Add function symbol from terms to collections

Example

Th(Monoid○PA)

- Sorts: M, N
- Function symbols:
    - I: 1 -> M // empty list
    - ⊗: M^2 -> M // concat
    - z: 1 -> N // zero
    - s: N -> N // successor
    - []: N -> M // return
- Equations
    - Unit: I ⊗ x = x = x ⊗ I
    - Assoc: (a ⊗ b) ⊗ c = a ⊗ (b ⊗ c)

Example

[sssz, sz, z] // = [3, 1, 0]

Interpretation

Structural Type => Collection of terms

Sum of theories, revisited

$(T \amalg C)X =$

$X +$

$(TX + CX) +$

$(TTX + TCX + CTX + CCX) +$

$(TTTX + \cdots) +$

$\ldots$

Distributive law

$$\delta: TC \Rightarrow CT$$

Think of terms as products, collections as sums.

Product of Sums ⇒ Sum of Products

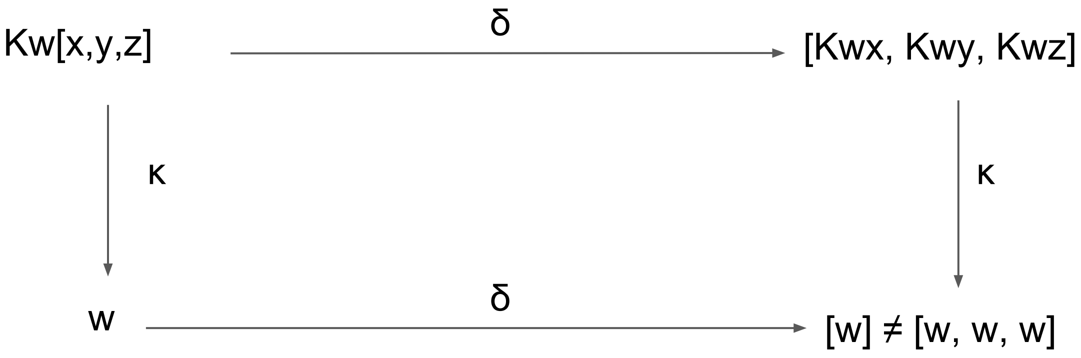
Interpretation

Structural Type => Collection of terms

TTCCTX => TCTX => CTTX => CTX

Distributive Law doesn't always exist

Use lists for collections, SKI calculus for terms.



Lists are "linear" in the sense that they count uses.
Sets aren't: {w} = {w, w, w}. So δ exists for SKI when C=Sets but not when C=Lists

Example: Primes

$$\neg l \land \neg((p \land \neg l) \otimes (q \land \neg l))$$

Modalities

$\Diamond A = \{ s(a) \mid \exists\ a.\ t(a) : A \}$

$A<K>B = \{ t \mid \exists\ u{:}A,\ v{:}B,\ w.\ s(w) = K(t, u) \wedge t(w) = v \}$

Application gives arrow types: $A<(-\ -)>B = A \Rightarrow B$

Par gives Caires' rely-guarantee types: $A< - \mid - >B = A \triangleright B$

Modalities for security

$$\neg \Diamond \exists x.(x!(*dontLeak) \mid (P \wedge \neg Nil))$$