Symbolic and Automatic Differentiation of Languages

Conal Elliott

ICFP, August 2021

Languages via type theory

A language is a predicate on “strings” yielding proofs/explanations/parsings.

$$\mathsf{Lang} = A^{\star} \to \mathsf{Set}\ \ell$$

Languages via type theory

A language is a predicate on “strings” yielding proofs/explanations/parsings.

$$\text{Lang} = A^{\star} \to \text{Set}\ \ell$$

$\varnothing\ w = \bot$

$\mathcal{U}\ w = \top$

$\mathbf{1}\ w = w \equiv []$

$\text{`}\ c\ w = w \equiv [\ c\ ]$

$(s \cdot P)\ w = s \times P\ w$

$(P \cup Q)\ w = P\ w \uplus Q\ w$

$(P \cap Q)\ w = P\ w \times Q\ w$

$(P * Q)\ w = \exists\ \lambda\ (u\ ,\ v) \to (w \equiv u +\!\!+ v) \times P\ u \times Q\ v$

$(P^{\star})\ w = \exists\ \lambda\ ws \to (w \equiv \text{concat}\ ws) \times \text{All}\ P\ ws$

Languages via type theory

A language is a predicate on "strings" yielding proofs/explanations/parsings.

$$\mathrm{Lang} = A^{\star} \to \mathrm{Set}\ \ell$$

$\varnothing\ w = \bot$

$\mathcal{U}\ w = \top$

$\mathbf{1}\ w = w \equiv []$

$\text{`}\ c\ w = w \equiv [\ c\ ]$

$(s \cdot P)\ w = s \times P\ w$

$(P \cup Q)\ w = P\ w \uplus Q\ w$

$(P \cap Q)\ w = P\ w \times Q\ w$

$(P * Q)\ w = \exists\ \lambda\ (u\ ,\ v) \to (w \equiv u +\!\!+ v) \times P\ u \times Q\ v$

$(P^{\star})\ w = \exists\ \lambda\ ws \to (w \equiv \mathrm{concat}\ ws) \times \mathrm{All}\ P\ ws$

Puzzle: how to compute parsings?

Plan

- Normal form for arbitrary (type-level) languages
- Normal form lemmas for standard language building blocks
- Decidability
- As corollaries, dual verified parsing implementations

Decomposing languages

Consider each list constructor:

```
ν : Lang → Set ℓ        -- "nullable"
ν P = P []

δ : Lang → A → Lang -- "derivative"
δ P a w = P (a :: w)
```

Decomposing languages

Consider each list constructor:

```
ν : Lang → Set ℓ        -- "nullable"
ν P = P []

δ : Lang → A → Lang     -- "derivative"
δ P a w = P (a :: w)
```

ν and repeated δ capture languages fully:

```
νoFoldlδ : ν ∘ foldl δ P ≗ P
νoFoldlδ []       = refl
νoFoldlδ (a :: as) = νoFoldlδ as
```

```
foldl : (X → A → X) → X → A * → X
foldl h x []       = x
foldl h x (a :: as) = foldl h (h x a) as
```

Language calculus lemmas

Language calculus lemmas

$$\nu\varnothing : \nu\ \varnothing \equiv \bot \qquad \delta\varnothing : \delta\ \varnothing\ a \equiv \varnothing$$

$$\nu\mathcal{U} : \nu\ \mathcal{U} \equiv \top \qquad \delta\mathcal{U} : \delta\ \mathcal{U}\ a \equiv \mathcal{U}$$

$$\nu\cup : \nu\ (P \cup Q) \equiv (\nu\ P \uplus \nu\ Q) \qquad \delta\cup : \delta\ (P \cup Q)\ a \equiv \delta\ P\ a \cup \delta\ Q\ a$$

$$\nu\cap : \nu\ (P \cap Q) \equiv (\nu\ P \times \nu\ Q) \qquad \delta\cap : \delta\ (P \cap Q)\ a \equiv \delta\ P\ a \cap \delta\ Q\ a$$

$$\nu\cdot : \nu\ (s \cdot P) \equiv (s \times \nu\ P) \qquad \delta\cdot : \delta\ (s \cdot P)\ a \equiv s \cdot \delta\ P\ a$$

$$\nu\mathbf{1} : \nu\ \mathbf{1} \leftrightarrow \top \qquad \delta\mathbf{1} : \delta\ \mathbf{1}\ a \longleftrightarrow \varnothing$$

$$\nu* : \nu\ (P * Q) \leftrightarrow (\nu\ P \times \nu\ Q) \qquad \delta* : \delta\ (P * Q)\ a \longleftrightarrow \nu\ P \cdot \delta\ Q\ a \cup \delta\ P\ a * Q$$

$$\nu^{\bigstar} : \nu\ (P^{\bigstar}) \leftrightarrow (\nu\ P)^{\star} \qquad \delta^{\bigstar} : \delta\ (P^{\bigstar})\ a \longleftrightarrow (\nu\ P)^{\star} \cdot (\delta\ P\ a * P^{\bigstar})$$

$$\nu` : \nu\ (`\ c) \leftrightarrow \bot \qquad \delta` : \delta\ (`\ c)\ a \longleftrightarrow (a \equiv c) \cdot \mathbf{1}$$

Decidable types

Decidable types

```
data Dec (X : Set ℓ) : Set ℓ where
  yes :    X → Dec X
  no  : ¬ X → Dec X          (¬ X = X → ⊥)
```

Decidable types

```
data Dec (X : Set ℓ) : Set ℓ where
  yes :    X → Dec X
  no  : ¬ X → Dec X          (¬ X = X → ⊥)
```

For predicates (languages),

```
Decidable P = ∀ x → Dec (P x)
```

Decidable types

$$\begin{aligned}&\textbf{data}\ \mathsf{Dec}\ (X : \mathsf{Set}\ \ell) : \mathsf{Set}\ \ell\ \textbf{where}\\ &\quad \mathsf{yes} : \quad X \to \mathsf{Dec}\ X\\ &\quad \mathsf{no}\ : \neg\ X \to \mathsf{Dec}\ X \qquad\qquad (\neg\ X = X \to \bot)\end{aligned}$$

For predicates (languages),

$$\mathsf{Decidable}\ P = \forall\ x \to \mathsf{Dec}\ (P\ x)$$

Isomorphisms:

$$\_\triangleleft\_ : B \leftrightarrow A \to \mathsf{Dec}\ A \to \mathsf{Dec}\ B$$

$$\_\blacktriangleleft\_ : Q \longleftrightarrow P \to \mathsf{Decidable}\ P \to \mathsf{Decidable}\ Q$$

Compositionally decidable types

Compositionally decidable types

$\bot^? : \mathsf{Dec}\ \bot$
$\bot^? = \mathsf{no}\ (\lambda\ ())$

$\top^? : \mathsf{Dec}\ \top$
$\top^? = \mathsf{yes}\ \mathsf{tt}$

$\_\uplus^?\_ : \mathsf{Dec}\ A \to \mathsf{Dec}\ B \to \mathsf{Dec}\ (A \uplus B)$
$\mathsf{no}\ \neg a \uplus^?\ \mathsf{no}\ \neg b = \mathsf{no}\ [\ \neg a\ ,\ \neg b\ ]$
$\mathsf{yes}\ a\ \uplus^?\ \mathsf{no}\ \neg b = \mathsf{yes}\ (\mathsf{inj}_1\ a)$
$\_\ \uplus^?\ \mathsf{yes}\ b\ = \mathsf{yes}\ (\mathsf{inj}_2\ b)$

$\_\times^?\_ : \mathsf{Dec}\ A \to \mathsf{Dec}\ B \to \mathsf{Dec}\ (A \times B)$
$\mathsf{yes}\ a\ \times^?\ \mathsf{yes}\ b\ = \mathsf{yes}\ (a\ ,\ b)$
$\mathsf{no}\ \neg a \times^?\ \mathsf{yes}\ b\ = \mathsf{no}\ (\neg a \circ \mathsf{proj}_1)$
$\_\ \times^?\ \mathsf{no}\ \neg b = \mathsf{no}\ (\neg b \circ \mathsf{proj}_2)$

$\_^{\star ?} : \mathsf{Dec}\ A \to \mathsf{Dec}\ (A^\star)$
$\_^{\star ?} = \mathsf{yes}\ []$

Reflections

Reflections

$\nu : \mathsf{Lang} \to \mathsf{Set}\ \ell$

$\delta : \mathsf{Lang} \to A \to \mathsf{Lang}$

$\nu\circ\mathsf{foldl}\delta : \nu \circ \mathsf{foldl}\ \delta\ P \mathrel{\dot{=}} P$

Reflections

$\nu : \mathsf{Lang} \to \mathsf{Set}\ \ell$

$\delta : \mathsf{Lang} \to A \to \mathsf{Lang}$

$\nu\circ\mathsf{foldl}\delta : \nu \circ \mathsf{foldl}\ \delta\ P \stackrel{\circ}{=} P$

$\nu* : \nu\ (P * Q) \quad \leftrightarrow \quad (\nu\ P \times \nu\ Q)$

$\delta* : \delta\ (P * Q)\ a \longleftrightarrow \nu\ P \cdot \delta\ Q\ a \cup \delta\ P\ a * Q$

```
-- etc
```

Reflections

$\nu : \mathsf{Lang} \to \mathsf{Set}\ \ell$

$\delta : \mathsf{Lang} \to A \to \mathsf{Lang}$

$\nu\circ\mathsf{foldl}\delta : \nu \circ \mathsf{foldl}\ \delta\ P \triangleq P$

$\nu* : \nu\ (P * Q) \quad \leftrightarrow \quad (\nu\ P \times \nu\ Q)$

$\delta* : \delta\ (P * Q)\ a \longleftrightarrow \nu\ P \cdot \delta\ Q\ a \cup \delta\ P\ a * Q$

```
-- etc
```

ν and δ are *higher-order functions*. How to apply these rules?

Reflections

$$\nu : \mathsf{Lang} \to \mathsf{Set}\ \ell$$
$$\delta : \mathsf{Lang} \to A \to \mathsf{Lang}$$

$$\nu\circ\mathsf{foldl}\delta : \nu \circ \mathsf{foldl}\ \delta\ P \mathrel{\dot{=}} P$$

$$\nu* : \nu\ (P * Q) \quad \leftrightarrow \quad (\nu\ P \times \nu\ Q)$$
$$\delta* : \delta\ (P * Q)\ a \longleftrightarrow \nu\ P \cdot \delta\ Q\ a \cup \delta\ P\ a * Q$$
```
-- etc
```

ν and δ are *higher-order functions*. How to apply these rules?

Same challenge in differential calculus.

Solve via *symbolic* or *automatic* differentiation.

Regular expressions (inductive)

```
data Lang : ₛLang → Set (suc ℓ) where
  ∅   : Lang ₛ∅
  𝒰   : Lang ₛ𝒰
  _∪_ : Lang P → Lang Q → Lang (P ₛ∪ Q)
  _∩_ : Lang P → Lang Q → Lang (P ₛ∩ Q)
  _·_ : Dec  s → Lang P → Lang (s ₛ· P)
  𝟏   : Lang ₛ𝟏
  _*_ : Lang P → Lang Q → Lang (P ₛ* Q)
  _☆  : Lang P → Lang (P ₛ☆)
  ‘   : (a : A) → Lang (ₛ‘ a)
  _◀_ : (Q ⟷ P) → Lang P → Lang Q
```

```
ν : Lang P → Dec (ₛν P)
δ : Lang P → (a : A) → Lang (ₛδ P a)
```

```
⟦_⟧? : Lang P → Decidable P
⟦ p ⟧?  []     = ν p
⟦ p ⟧? (a :: w) = ⟦ δ p a ⟧? w
```

Symbolic differentiation (column-major / patterns)

Symbolic differentiation (column-major / patterns)

$$\nu\ \varnothing = \bot^?$$

$$\nu\ \mathcal{U} = \top^?$$

$$\nu\ (p \cup q) = \nu\ p \uplus^?\ \nu\ q$$

$$\nu\ (p \cap q) = \nu\ p \times^?\ \nu\ q$$

$$\nu\ (s \cdot p) = s \times^?\ \nu\ p$$

$$\nu\ \mathbf{1} = \nu\mathbf{1} \lhd \top^?$$

$$\nu\ (p * q) = \nu* \lhd (\nu\ p \times^?\ \nu\ q)$$

$$\nu\ (p\ ^{\bigstar}) = \nu^{\bigstar} \lhd (\nu\ p\ ^{*?})$$

$$\nu\ (`\ a) = \nu` \lhd \bot^?$$

$$\nu\ (f \blacktriangleleft p) = f \lhd \nu\ p$$

$$\delta\ \varnothing\ a = \varnothing$$

$$\delta\ \mathcal{U}\ a = \mathcal{U}$$

$$\delta\ (p \cup q)\ a = \delta\ p\ a \cup \delta\ q\ a$$

$$\delta\ (p \cap q)\ a = \delta\ p\ a \cap \delta\ q\ a$$

$$\delta\ (s \cdot p)\ a = s \cdot \delta\ p\ a$$

$$\delta\ \mathbf{1}\ a = \delta\mathbf{1} \blacktriangleleft \varnothing$$

$$\delta\ (p * q)\ a = \delta* \blacktriangleleft (\nu\ p \cdot \delta\ q\ a \cup \delta\ p\ a * q)$$

$$\delta\ (p\ ^{\bigstar})\ a = \delta^{\bigstar} \blacktriangleleft (\nu\ p\ ^{*?} \cdot (\delta\ p\ a * p\ ^{\bigstar}))$$

$$\delta\ (`\ c)\ a = \delta` \blacktriangleleft ((a \stackrel{?}{=} c) \cdot \mathbf{1})$$

$$\delta\ (f \blacktriangleleft p)\ a = f \blacktriangleleft \delta\ p\ a$$

Tries (coinductive)

Tries (coinductive)

```
record Lang (P : ◈Lang) : Set (suc ℓ) where
  coinductive
  field
    ν : Dec (◈ν P)
    δ : (a : A) → Lang (◈δ P a)
```

```
⟦_⟧? : Lang P → Decidable P
⟦ p ⟧?  []     = ν p
⟦ p ⟧? (a :: w) = ⟦ δ p a ⟧? w
```

```
∅   : Lang ◈∅
𝒰   : Lang ◈𝒰
_∪_ : Lang P → Lang Q → Lang (P ◈∪ Q)
_∩_ : Lang P → Lang Q → Lang (P ◈∩ Q)
_·_ : Dec  s → Lang P → Lang (s ◈·  P)
𝟏   : Lang ◈𝟏
_*_ : Lang P → Lang Q → Lang (P ◈* Q)
_☆  : Lang P → Lang (P ◈☆)
`   : (a : A) → Lang (`◈ a)
_◀_ : (Q ⟷ P) → Lang P → Lang Q
```

Automatic differentiation (row-major / copatterns)

Automatic differentiation (row-major / copatterns)

$$\nu\ \varnothing = \bot^?$$

$$\nu\ \mathcal{U} = \top^?$$

$$\nu\ (p \cup q) = \nu\ p \uplus^?\ \nu\ q$$

$$\nu\ (p \cap q) = \nu\ p \times^?\ \nu\ q$$

$$\nu\ (s \cdot p) = s \times^?\ \nu\ p$$

$$\nu\ \mathbf{1} = \nu\mathbf{1} \lhd \top^?$$

$$\nu\ (p * q) = \nu* \lhd (\nu\ p \times^?\ \nu\ q)$$

$$\nu\ (p\ ^{\bigstar}) = \nu^{\bigstar} \lhd (\nu\ p\ ^{\bigstar?})$$

$$\nu\ (\text{`}\ a) = \nu\text{`} \lhd \bot^?$$

$$\nu\ (f \blacktriangleleft p) = f \lhd \nu\ p$$

$$\delta\ \varnothing\ a = \varnothing$$

$$\delta\ \mathcal{U}\ a = \mathcal{U}$$

$$\delta\ (p \cup q)\ a = \delta\ p\ a \cup \delta\ q\ a$$

$$\delta\ (p \cap q)\ a = \delta\ p\ a \cap \delta\ q\ a$$

$$\delta\ (s \cdot p)\ a = s \cdot \delta\ p\ a$$

$$\delta\ \mathbf{1}\ a = \delta\mathbf{1} \blacktriangleleft \varnothing$$

$$\delta\ (p * q)\ a = \delta* \blacktriangleleft (\nu\ p \cdot \delta\ q\ a \cup \delta\ p\ a * q)$$

$$\delta\ (p\ ^{\bigstar})\ a = \delta^{\bigstar} \blacktriangleleft (\nu\ p\ ^{\bigstar?} \cdot (\delta\ p\ a * p\ ^{\bigstar}))$$

$$\delta\ (\text{`}\ c)\ a = \delta\text{`} \blacktriangleleft ((a \stackrel{?}{=} c) \cdot \mathbf{1})$$

$$\delta\ (f \blacktriangleleft p)\ a = f \blacktriangleleft \delta\ p\ a$$

Summary

- Simple, non-computational formal specification.
- Reasoning from propositionally defined languages to decidable parsing via decision types.
- Duality of regular expressions and tries (symbolic and automatic differentiation).

  Same code with dual interpretations.

Also in the paper

- Termination checking
- Generalizing languages
- Algebraic properties of languages
- Transporting algebraic properties