A Theory of Changes for Higher-Order Languages

A Theory of Changes: (1) Formalizing change structures

Definition: Change Structure

A change structure for V is the tuple $\hat{\mathbf{V}} = (V, \Delta, \oplus, \ominus)$ where ...

- V is a set called the base set
- Δv is the change set for a specified value $v \in V$
- Updating a base value $v_{old} \in \mathbf{V}$ with a change $dv \in \Delta v_{old}$:

  $v_{new} = v_{old} \oplus dv$
- Computing the change between two values $v_{old} \in \mathbf{V}$ and $v_{new} \in \mathbf{V}$:

  $dv = v_{new} \ominus v_{old}$

Example: Naturals

Base set: $\mathbb{N}$ Change set: Δv $\oplus = +$ $\ominus = -$

Given some natural $v_{old} \in \mathbb{N}$, a change $dv \in \Delta v_{old}$ describes the difference between v_{old} and another natural v_{new} .

- Updating a base value v with a change dv :

$$v_{new} = v_{old} \oplus dv$$
$$v_{new} = v_{old} + dv$$

If $\Delta v = \mathbb{Z}$, then the following is possible (and invalid):

$$3 \oplus -5$$
$$= 3 + (-5)$$
$$= -2 \notin \mathbb{N}$$

- Computing the change between two values v_{old} and v_{new} :

$$dv = v_{new} \ominus v_{old}$$
$$dv = v_{new} - v_{old}$$

If $\Delta v = \mathbb{N}$, then the following is possible (and invalid):

$$3 \ominus 5$$
$$= 3 - 5$$
$$= -2 \notin \mathbb{N}$$

A Theory of Changes: (1) Formalizing change structures

Example: Naturals

Base set: $\mathbb{N}$ Change set: Δv $\oplus = +$ $\ominus = -$

Given some natural $v \in \mathbb{N}$, a change $dv \in \Delta v$ describes the difference between v and another natural v_{new} .

Solution: Introduce dependent types

For a natural v, we set $\Delta v = \{ dv \mid v + dv \geq 0\}$; $\ominus$ and $\oplus$ are then always defined.

Conditions for change structures:

- For each base value $v \in \mathbb{N}$, there exists a specific set of changes Δv for v.
- Given $v_{new}, v_{old} \in \mathbb{N}$: $v_{new} \ominus v_{old} \in \Delta v_{old}$
- Given $v \in \mathbb{N}$, $dv \in \Delta v$: $v \oplus dv \in \mathbb{N}$

A Theory of Changes: (2) Nil Changes & Derivatives

Definition: Nil Changes

$$\mathbf{0}_{\mathbf{v}} = \mathrm{v} \ominus \mathrm{v}$$

$$\mathrm{v} \oplus \mathbf{0}_{\mathbf{v}} = \mathrm{v}$$

Definition: Derivatives

The derivative **f’** of a function **f** computes the change of **f a** when **a** is updated to **a ⊕ da**.

$$\mathbf{f} \quad :: \mathbf{A} \to \mathbf{B}$$

$$\mathbf{f'} \quad :: \mathbf{A} \to \Delta\mathbf{A} \to \Delta\mathbf{B}$$

$$\mathbf{f'\ a\ da = f\ (a \oplus da) \ominus f\ a}$$

or equivalently:

$$\underbrace{\underline{\mathbf{f\ a}}}_{\text{Original output}} \oplus \underbrace{\underline{\mathbf{f'\ a\ da}}}_{\text{Output change}} = \mathbf{f}\ \underbrace{\underline{\mathbf{(a \oplus da)}}}_{\text{Updated input}}$$

A Theory of Changes: (2) Nil Changes & Derivatives

<u>Corollary</u>: The derivative of a function applied to an input 'a' and its nil change '0_a' produces a nil output change '$0_{(f\ a)}$'

$$\mathbf{f'\ a\ 0_a = 0_{(f\ a)}}$$

$$\mathbf{f\ a \oplus f'\ a\ 0_a = f\ a}$$

A Theory of Changes: (3) Function Changes

Recall (Derivatives):

A derivative **f'** of a function **f** computes the change of **f a** when:

- **a** is updated to **a ⊕ da**.

$$f'\ a\ da = f\ (a \oplus da) \ominus f\ a \qquad f' :: A \to \Delta A \to \Delta B$$

Incrementalization Theorem (Function Changes):

A function change **df** of a function **f** computes the change of **f a** when:

- **a** is updated to **a ⊕ da**
- **f** is updated to **f ⊕ df**

$$df\ a\ da = \underbrace{(f \oplus df)\ (a \oplus da)}\ \ominus f\ a \qquad df :: A \to \Delta A \to \Delta B$$

We need to avoid computing this directly

A Theory of Changes: (3) Function Changes

Relating Derivatives With Function Changes:

Recall the incrementalization theorem:

$$\mathbf{df\ a\ da = (f \oplus df)\ (a \oplus da) \ominus f\ a} \qquad \text{(Incrementalization Theorem)}$$

When $\mathbf{df}$ is the nil change for $\mathbf{f}$ (i.e. $\mathbf{f \oplus df = f}$ or $\mathbf{df = 0_f}$), this theorem becomes:

$$\mathbf{df\ a\ da = f\ (a \oplus da) \ominus f\ a} \qquad \text{(Incrementalization Theorem when } \mathbf{df = 0_f}\text{)}$$

This looks exactly the same as the definition for derivatives of a function!

$$\mathbf{f'\ a\ da = f\ (a \oplus da) \ominus f\ a} \qquad \text{(Definition: Derivatives)}$$

Conclusion:

The nil change to a function, $\mathbf{0_f}$, is exactly the same as its derivative, $\mathbf{f'}$.

$$\mathbf{0_f = f'} \qquad \text{(Theorem: Nil Changes are Derivatives)}$$

Incrementing λ-Calculi

We want to define a computation $\mathcal{Derive}$ which computes the derivatives of functions.

We could implement this using the definition for derivatives ...

$$\mathbf{f' \; a \; da = f \; (a \oplus da) \ominus f \; a}$$

… but this is no faster than recomputation.

Therefore, we must define $\mathcal{Derive}$ as a source-to-source transformation that guarantees that:

$$[\![\mathcal{Derive}(f)]\!] \text{ represents the derivative of } [\![f]\!]$$

$$\mathcal{Derive}\mathbf{(f) \; a \; da = f \; (a \oplus da) \ominus f \; a}$$

Incrementing λ-Calculi - Plugins

Incremental λ-Calculi is a simply-typed λ-Calculi parameterized by *plugins* provided by a user.

A plugin should define:

- The base types
- The change representation for each base type
- The primitive constants
- The incremental version/derivatives of each primitive constant

What we can define right now, is the incrementalization of features that are universal amongst all possible instances of plugin implementations.

- Function types
- Abstractions
- Application
- Variable References

Incrementing λ-Calculi - Differentiation

Defining $\mathcal{D}erive(f)$

- We require that if $\Gamma \vdash t : \tau$, then $\mathcal{D}erive(t)$ represents the change in t in terms of changes to the values of its free variables.

$$\frac{\Gamma \vdash t : \tau}{\Gamma, \Delta\Gamma \vdash \mathcal{D}erive(t) : \Delta\tau}\ \mathcal{D}erive$$

$\mathcal{D}erive$(u)

1. If u = x

   $\mathcal{D}erive$(x) = dx (that is, the change of x)

2. If u = λx. t

   $\mathcal{D}erive$(λx. t) = λx. λdx. $\mathcal{D}erive$(t)

3. If u = s t

   $\mathcal{D}erive$(s t) = $\mathcal{D}erive$(s) t $\mathcal{D}erive$(t)

4. If u = c

   $\mathcal{D}erive$(c) = 0_c

Incrementing λ-Calculi - Proving Desired Behaviour of Derive

Desired behaviour of $\mathcal{Derive}$: $f\ (a \oplus da) \cong (f\ a) \oplus (\mathcal{Derive}(f)\ a\ da)$

Given a standard denotational semantics $[\![-]\!]$ for simply typed λ-calculus, we define a non-standard denotational semantics $[\![-]\!]^{\Delta}$

We aim to prove that $[\![t]\!]^{\Delta}$ is the derivative of $[\![t]\!]$

Incrementing λ-Calculi - Proving Desired Behaviour of Derive

Change Semantics

$[\![t]\!]\ \rho$

$$
\begin{aligned}
&[\![c]\!] && \rho = \ldots \\
&[\![\lambda x.\ t]\!] && \rho = \lambda v.\ [\![t]\!]\ (\rho\ ,\ x = v) \\
&[\![s\ t]\!] && \rho = ([\![s]\!]\ \rho)\ ([\![t]\!]\ \rho) \\
&[\![x]\!] && \rho = \text{lookup } x \text{ in } \rho
\end{aligned}
$$

(Standard Evaluation)

$[\![t]\!]^{\Delta}\ \rho\ d\rho$

$$
\begin{aligned}
&[\![c]\!]^{\Delta} && \rho\ d\rho = \ldots \\
&[\![\lambda x.\ t]\!]^{\Delta} && \rho\ d\rho = \lambda v\ dv.\ [\![t]\!]^{\Delta}\ (\rho\ ,\ x = v)\ (d\rho\ ,\ dx = dv) \\
&[\![s\ t]\!]^{\Delta} && \rho\ d\rho = ([\![s]\!]^{\Delta}\ \rho\ d\rho)\ ([\![t]\!]^{\Delta}\ \rho\ d\rho) \\
&[\![x]\!]^{\Delta} && \rho\ d\rho = \text{lookup } dx \text{ in } d\rho
\end{aligned}
$$

(Differential Evaluation)

Incrementing λ-Calculi - Proving Desired Behaviour of Derive

Aim: Proving the behaviours of $[\![\mathrm{Derive}(t)]\!]$ and $[\![t]\!]^{\Delta}$ are consistent

Theorem: (Correctness of differentiation)

Let $f : \sigma \to \tau$ be a closed term of function type.
For every closed base term $s : \sigma$ and closed change term $ds : \Delta\sigma$ such that there exists a change $dv \in \Delta_{\sigma}[\![s]\!]$ which erases to ds, we have that:

$$f\ (s \oplus ds) \cong (f\ s) \oplus (\mathrm{Derive}(f)\ s\ ds)$$

Recall: $f\ (a \oplus da) \cong (f\ a) \oplus (\mathrm{Derive}(f)\ a\ da)$