Markedly: a cartographic approach for mapping eDSL implementation costs

Matthew P. Ahrens*, **Karl Cronburg****, and Jeanne-Marie Musca***
Siriusly PL Lab @ Tufts University

Definitions

Design	$\boldsymbol{D}$	set of features
Implementation	$\boldsymbol{I}$	set of source code spans
Source code span	(s, e)	start to end (characters)

The Problem

Embedded domain specific languages (eDSLs) are difficult to design, implement, and maintain to stand the test of time. EDSL creators want to:

- $\mathbb{P}$reserve . . . the correctness of $\boldsymbol{D}$ in $\boldsymbol{I}$
- $\mathbb{E}$xtend . . . $\boldsymbol{D}$ and $\boldsymbol{I}$ to support new features
- $\mathbb{A}$dhere . . . exclusively to $\boldsymbol{D}$ in $\boldsymbol{I}$
- $\mathbb{R}$ecognize . . . source spans in $\boldsymbol{I}$ as features in $\boldsymbol{D}$

The Markedly Method

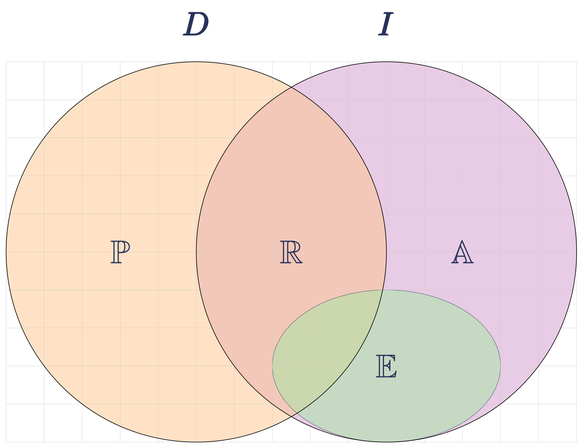

Figure 1: Sources of $\mathbb{PEAR}$ costs.

0 Language creator Mark wants to create an eDSL in Haskell with the ability to annotate Haskell types with natural numbers.

Type Level Naturals (TLN)

1 Mark first writes down the desired features:

1. human-readable natural number weights,
2. to attach weights to arbitrary expressions,
3. and to sum weights across function application.

TLN Implementation

2 Mark implements a prototype of TLN:

```
data Nat = Z | S Nat
type Weighted (w :: Nat) a = a
type family
  (w1 :: Nat) :+: (w2 :: Nat) :: Nat
type instance Z :+: m = m
type instance (S n) :+: m = S (n :+: m)
($$) :: Weighted w1 (a -> b)
     -> Weighted w2 a
     -> Weighted (w1 :+: w2) b
($$) f a = f a
```

Figure 2: Haskell source code for TLN.

$\mathbb{PEAR}$ Metrics

$\mathbb{P}$reserve	How many features in $\boldsymbol{D}$ are compromised in $\boldsymbol{I}$?
$\mathbb{E}$xtend	How many intermediate representations in $\boldsymbol{I}$ are not accessible?
$\mathbb{A}$dhere	How many source spans in $\boldsymbol{I}$ do not map to a feature in $\boldsymbol{D}$?
$\mathbb{R}$ecognize	Is this source span in $\boldsymbol{I}$ present in $\boldsymbol{D}$?

The TLN Map

3 With Haskell code in hand (Figure **??**), Mark hand-draws a Markedly map depicting how meta-information† flows through the code.

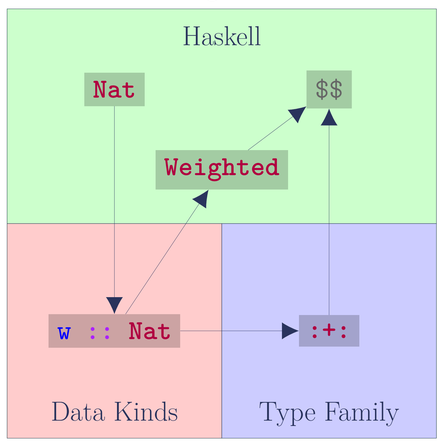

Figure 3: Haskell map for TLN.

TLN $\mathbb{PEAR}$ Costs

4 Finally Mark computes $\mathbb{PEAR}$ costs according to the metrics listed above.

Element	$\mathbb{P}$	$\mathbb{E}$	$\mathbb{A}$	$\mathbb{R}$
`Nat`	1	·	·	·
`Weighted`	·	·	1	·
`w :: Nat`	·	·	·	1
`$$`	·	·	1	·
`Nat` → `w :: Nat`	·	·	·	1
`w :: Nat` → `Weighted`	·	·	·	1
`w :: Nat` → `:+:`	·	·	·	1
$\text{cost}(\boldsymbol{D}, \boldsymbol{I}) = \Sigma \langle \mathbb{P}, \mathbb{E}, \mathbb{A}, \mathbb{R} \rangle =$	1	·	2	4

Figure 4: The non-zero $\mathbb{PEAR}$ costs of TLN

Upcoming Work & Takeaways

1. Markedly user study.*
2. Source span highlighting IDE support.**
3. Git integration.**
4. Extensible (multi-union) data types.***

Question

Adherence and Recognition costs are very similar. How can we more clearly distinguish "superfluous" source spans present in the design $\boldsymbol{D}$ from source spans **not** in $\boldsymbol{D}$?

- A weak or vague $\boldsymbol{D}$ incurs high $\mathbb{A}$ and $\mathbb{R}$ costs.
- Prototypes of $\boldsymbol{I}$ sacrifice $\mathbb{P}$ in favor of $\mathbb{E}$.
- Markedly fits into an iterative design and implementation workflow.

References

Acknowledgements

Markedly is sponsored by the Defense Advanced Research Projects Agency (contract: FA8750-15-2-0033). Markedly does not necessarily reflect the position or the policy of the US Government, and no official endorsement should be inferred.

Contact

Web: `http://siriusly.cs.tufts.edu/`
Email: `karl@cs.tufts.edu`
Preprint: `cronburg.com/papers/markedly17.pdf`