The Expression Problem and Lenses

Lambdajam 2016

Tony Morris

The Expression Problem

A new name for an old problem[Wad98]

Whether a language can solve the Expression Problem is a salient indicator of its capacity for expression.

The Expression Problem

What is The Expression Problem?

```
data TrafficLight = Green | Amber | Red

cycle Red = Green
cycle Amber = Red
cycle Green = Amber
```

The Expression Problem

Adding a new case?

- `data TrafficLight = ... | BusesOnlyProceed`
- All referencing functions (e.g. `cycle`) must either change or fail for the new case.

The Expression Problem

Adding a new case?

- `data TrafficLight = ... | BusesOnlyProceed`
- All referencing functions (e.g. `cycle`) must either change or fail for the new case.

The Expression Problem

Create a new data type?

- `data BussyLight = Old TrafficLight | BusesOnlyProceed`
- Existing functions are unusable and must be repeated.
- Does there exist an appropriate abstraction?

NICTA

The Expression Problem

Create a new data type?

- `data BussyLight = Old TrafficLight | BusesOnlyProceed`
- Existing functions are unusable and must be repeated.
- Does there exist an appropriate abstraction?

NICTA

The Expression Problem

Create a new data type?

- `data BussyLight = Old TrafficLight | BusesOnlyProceed`
- Existing functions are unusable and must be repeated.
- Does there exist an appropriate abstraction?

The Expression Problem

What is the solution?

- **There isn't one.**
- Does clojure `defprotocol` solve TEP?
- **No.**

The Expression Problem

What is the solution?

- **There isn't one.**
- Does clojure `defprotocol` solve TEP?
- **No.**

The Expression Problem

What is the solution?

- **There isn't one.**
- Does clojure `defprotocol` solve TEP?
- **No.**

The Expression Problem

What is the solution?

- There are only *trade-offs.*
- Some trades maximise economy.
- Does clojure `defprotocol` maximise economy? **No.**

NICTA

The Expression Problem

What is the solution?

- There are only *trade-offs.*
- Some trades maximise economy.
- Does clojure `defprotocol` maximise economy? **No.**

NICTA

The Expression Problem

What is the solution?

- There are only *trade-offs*.
- Some trades maximise economy.
- Does clojure `defprotocol` maximise economy? **No.**

The Expression Problem

What is the solution?

I will use the Haskell `lens` library to demonstrate one such technique.

The Expression Problem

Another example

```
data Either a b = Left a | Right b

leftor :: a -> Either a b -> a
leftor _ (Left a) = a
leftor a (Right _) = a
```

The Expression Problem

Add `None` case

```
data Either a b = Left a | Right b | None

leftor :: a -> Either a b -> a
leftor _ (Left a)   = a
leftor a (Right _) = a
leftor a None       = a
```

The Expression Problem

Another example

We'd like to write `leftor` once, and for both data types.

NICTA

The Expression Problem

None case

```
data Either a b = Left a | Right b
data Prolly a b = This a | That  b | None

leftor :: Leftish t => a -> t a b -> a
leftor = ...
```

The Expression Problem

Leftish

We want to abstract

- the **view** of `(a)`
- **possibly existing** in `(t a b)`

The Expression Problem

Leftish view

```
type LeftishView t a b =
  (a -> Identity a) -> (t a b -> Identity (t a b))
```

- `Identity`
- `(->)`

The Expression Problem

Leftish **view**

```
type LeftishView t a b =
  (a -> Identity a) -> (t a b -> Identity (t a b))
```

- `Identity`
- `(->)`

The Expression Problem

Leftish view

```
type LeftishView p f t a b =
  (a `p` f a) -> (t a b `p` f (t a b))
```

The Expression Problem

Left prism

```
type LeftPrism p f t a b =
  (Choice p, Applicative f) =>
    (a `p` f a) -> (t a b `p` f (t a b))
```

NICTA

The Expression Problem

Leftish type-class

```
class Leftish p f t where
  _Leftish ::
    (a `p` f a) -> (t a b `p` f (t a b))

instance (Choice p, Applicative f) =>
  Leftish p f Either where
instance (Choice p, Applicative f) =>
  Leftish p f Prolly where
```

The Expression Problem

leftor from Leftish type-class

```
leftor ::
  Leftish (->) (Const (First a)) t =>
  a
  -> t a b
  -> a
leftor a x =
  fromMaybe a (x ^? _Leftish)
```

NICTA

The Expression Problem

Other structures supporting `Leftish`

- iso `Const` in `data Const a b = Const a`
- prism `These` in `data These a b = This a | That b | Both a b`
- prism `Validation` in `data Validation e a = Fail e | Success a`
- lens `Pair` in `data Pair a b = Pair a b`

NICTA

The Expression Problem

`leftor` from Leftish type-class

```
leftor ::
  LeftLike (->) (Const (First a)) t =>
  a
  -> t a b
  -> a
leftor a x =
  fromMaybe a (x ^? _Leftish)
```

The Expression Problem

The technique

- For each field or data constructor
  - A type-class over `p f s` with one function.
  - A target type (`T`); the field type or the data associated with the constructor.
  - `(T `*p*` f T) -> (s `*p*` f s)`
- The equivalence instance for that type (`T`).
- The instance for the data type with the field or constructor.

The Expression Problem

The technique

Constraints depending on the type of view.

- `Lens: (p ~ (->), Functor f) =>`
- `Prism: (Choice p, Applicative f) =>`
- `Traversal: (p ~ (->), Applicative f) =>`
- `Iso: (Profunctor p, Functor f) =>`
- `Getter: (p ~ (->), Contravariant f, Functor f) =>`
- `Fold: (p ~ (->), Contravariant f, Applicative f) =>`

NICTA

The Expression Problem

The technique

```haskell
data Person = Person Int String

class AsAge p f s where
  _Age ::
    p Int (f Int) -> p s (f s)

instance
  AsAge p f Int where _Age = id
instance (p ~ (->), Functor f) => -- Lens
  AsAge p f Person where ...
```

. . . *library functions in terms of* `_Age`

NICTA

The Expression Problem

The technique

```
data Shape = Circle Float | ...

class AsRadius p f s where
  _Radius ::
    p Float (f Float) -> p s (f s)

instance
  AsRadius p f Float where _Radius = id
instance (Choice p, Applicative f) => -- Prism
  AsRadius p f Shape where ...
```

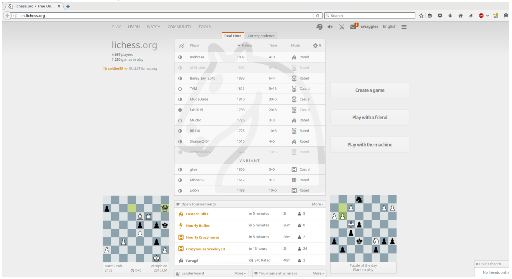
. . . library functions in terms of `_Radius`

NICTA

The Expression Problem

Cheat Detection

lichess.org is an open-source chess server, written using Scala.



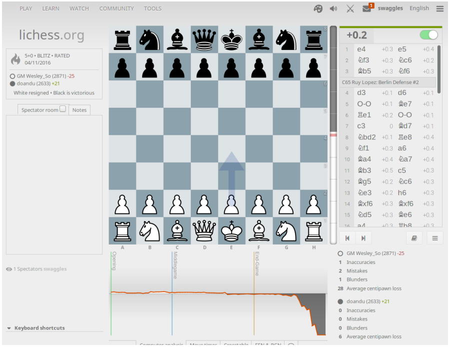
Player	Rating	Time	Mode
mehrasa	1897	4+0	Rated
M-khaidir	1892	5+15	Rated
Bailey_Jay_2045	1833	6+4	Rated
TVM	1811	5+15	Casual
MudeDude	1810	30+0	Casual
luis2015	1793	20+8	Casual
Mucho	1726	3+0	Rated
RR110	1725	10+8	Rated
Shanay2404	1515	4+5	Rated
vedatsahin52	1474	10+5	Rated
— VARIANT —			
gtee	1856	3+0	Casual
Mishel52	1612	9+7	Rated
jo55h	1489	10+0	Rated

Open tournaments More »

Eastern Blitz	in 5 minutes	2h	5
Hourly Bullet	in 5 minutes	26m	0
Hourly Crazyhouse	in 5 minutes	26m	2
Crazyhouse Weekly III	in 13 hours	2h	24
Faragó	3+0 Rated	40m	2

Leaderboard More »
Tournament winners More »

Cheat Detection



5+0 • BLITZ • RATED
04/11/2016

○ GM Wesley_So (2871) -25
● doandu (2633) +21
White resigned • Black is victorious

1 Spectators swaggles

+0.2

1	e4	+0.3	e5	+0.4
2	♘f3	+0.3	♘c6	+0.2
3	♗b5	+0.3	♘f6	+0.3
C65 Ruy Lopez: Berlin Defense #2				
4	d3	+0.1	d6	+0.1
5	O-O	+0.1	♗e7	+0.1
6	♖e1	+0.2	O-O	+0.1
7	c3	0	♗d7	+0.1
8	♘bd2	+0.1	♖e8	+0.4
9	♘f1	+0.3	a6	+0.4
10	♗a4	+0.4	♘a7	+0.4
11	♗b3	+0.5	c5	+0.3
12	♗g5	+0.2	♘c6	+0.3
13	♘e3	+0.2	h6	+0.3
14	♗xf6	+0.3	♗xf6	+0.3
15	♘d5	+0.3	♗e6	+0.3
16	a4	+0.3	♖h8	+0.3

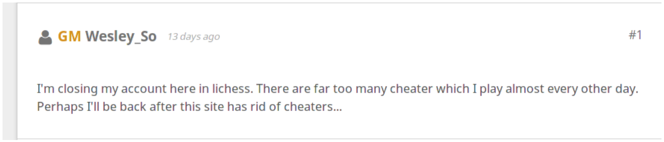
○ GM Wesley_So (2871) -25
1 Inaccuracies
2 Mistakes
1 Blunders
28 Average centipawn loss

● doandu (2633) +21
0 Inaccuracies
0 Mistakes
0 Blunders
6 Average centipawn loss

Here is the world #12 being beaten by a patzer using computer-assistance.

The Expression Problem

Cheat Detection



GM Wesley_So 13 days ago #1

I'm closing my account here in lichess. There are far too many cheater which I play almost every other day. Perhaps I'll be back after this site has rid of cheaters...

NICTA

The Expression Problem

Cheat Detection

```
cheatReport :: String
```

Currently, cheat reports are written by and assessed by humans.

NICTA

The Expression Problem

Cheat Detection

```
data CheatReport =
  UnhumanCriticalLine MoveNumber
  | RepeatingMoveStrengthPattern Pattern
  | UnusualCentipawnLoss Loss
  | ...
```

The Expression Problem

Cheat Detection

- A `String` cheat report provides open description.
- However, with no structure, applying any automation is impossible.
- A closed, structured cheat report afford automation.
- However, cheat report structure is likely to change over time.

The Expression Problem

Cheat Detection

```
class AsMoveNumber p f s where ...
class AsCentipawnLoss p f s where ...
learn :: AsPattern p f s => ...
```

The Expression Problem

An incidental advantage

GPS Exchange Format (GPX) is the open standard format for GPS tracks, waypoints and routes.

The Expression Problem

An incidental advantage

- a **gpx** has zero or many **tracks**
- a **track** has zero or many **segments**
- a **segment** has zero or many **track points**
- a **track point** has one **latitude**
- a **latitude** has a **decimal value between** -90 and 90

The Expression Problem

An incidental advantage

```
gpxTrack =
  _Gpx . _Track
gpxTrackSegment =
  _GpxTrack . _Segment
gpxTrackSegmentTrackPointLatitude =
  _GpxTrackSegment . _TrackPoint . _Latitude
```

NICTA

The Expression Problem

An incidental advantage

```
instance AsLatitude Gpx where ...
```

No need to come up with a tree of identifier names!

The Expression Problem

Disadvantages

What are some of the penalties?

The Expression Problem

Disadvantages

Lots of boilerplate

- `{-# LANGUAGE MultiParamTypeClasses, FlexibleInstances #-}`
- A type-class.
- An equivalence instance `(id)`.
- The instance for the field or constructor.

The Expression Problem

Disadvantages

Type errors are difficult to navigate.

```
λ> 12.34 .#. 56.78

No instance for (Fractional lat0)
      arising from the literal '12.34'
The type variable 'lat0' is ambiguous

No instance for (AsLongitude
      (->) (Control.Applicative.Const Longitude) lon0)
  arising from a use of '.#.'
```

NICTA

The Expression Problem

Disadvantages

Type **signatures** are difficult to navigate.

```
javaClassFileParser ::
  (AsEmpty (c Word8), AsEmpty (t Char),
    AsEmpty (f (Attribute a1)), AsEmpty (a Word8),
    AsEmpty (m (Attribute a2)), AsEmpty (a3 Word8), AsEmpty (a4 Word8),
    AsEmpty (c1 (ConstantPoolInfo p)), AsEmpty (i Word16),
    AsEmpty (s1 (Field a5 f1)), AsEmpty (t1 (Method m1 b)),
    AsEmpty (u (Attribute d)), Cons (c Word8) (c Word8) Word8 Word8,
    Cons (t Char) (t Char) Char Char,
    Cons
      (f (Attribute a1)) (f (Attribute a1)) (Attribute a) (Attribute a),
    Cons (a Word8) (a Word8) Word8 Word8,
    Cons
      (m (Attribute a2))
      (m (Attribute a2))
      (Attribute a3)
      (Attribute a3),
    Cons (a3 Word8) (a3 Word8) Word8 Word8,
    Cons (a4 Word8) (a4 Word8) Word8 Word8,
    Cons
      (c1 (ConstantPoolInfo p))
      (c1 (ConstantPoolInfo p))
      (ConstantPoolInfo t)
      (ConstantPoolInfo t),
    Cons (i Word16) (i Word16) Word16 Word16,
    ...
```

The Expression Problem

Disadvantages

No, srsly.

```
    ...
    Cons
      (s1 (Field a5 f1)) (s1 (Field a5 f1)) (Field a1 f) (Field a1 f),
    Cons
      (t1 (Method m1 b)) (t1 (Method m1 b)) (Method m a2) (Method m a2),
    Cons
      (u (Attribute d)) (u (Attribute d)) (Attribute a4) (Attribute a4),
    AsClassFileCafebabeError Tagged Identity (s c),
    AsClassFileVersionError Tagged Identity (s c),
    AsClassFileConstantPoolError Tagged Identity s,
    AsClassFileThisAccessFlagsError Tagged Identity (s c),
    AsClassFileThisClassError Tagged Identity (s c),
    AsClassFileSuperClassError Tagged Identity (s c),
    AsClassFileInterfacesError Tagged Identity (s c),
    AsClassFileFieldsError Tagged Identity (s c),
    AsClassFileMethodsError Tagged Identity (s c),
    AsClassFileAttributesError Tagged Identity (s c),
    AsClassFileUnexpectedInputOnStream Tagged Identity (s c)) =>
  Get (s c) (ClassFile p c1 i a5 f1 s1 m1 b t1 d u)
```

The Expression Problem

Disadvantages

However,

- This java class file parser works for both version 1.5 and 1.7 class files.
- Works for future java versions.
- Derived functions work against similar formats (.NET).
- and I can prove all this by **parametricity**.

Conclusion

Discussion

- The Expression Problem has been conquered in part by exploiting lens abstractions.
- For real this time; with demonstrable benefits.
- However, there are penalties.
- Is there an even better way?

NICTA

Conclusion

Discussion

- The Expression Problem has been conquered in part by exploiting lens abstractions.
- For real this time; with demonstrable benefits.
- However, there are penalties.
- Is there an even better way?

NICTA

Conclusion

Discussion

- The Expression Problem has been conquered in part by exploiting lens abstractions.
- For real this time; with demonstrable benefits.
- However, there are penalties.
- Is there an even better way?

NICTA

Conclusion

Discussion

- The Expression Problem has been conquered in part by exploiting lens abstractions.
- For real this time; with demonstrable benefits.
- However, there are penalties.
- Is there an even better way?

References

Philip Wadler, *The expression problem*, Java-genericity mailing list (1998).