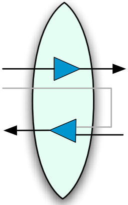
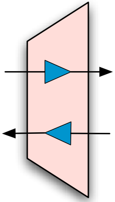
Quotient Lenses

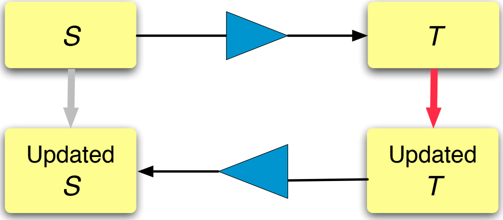


Nate Foster (Penn)
Benjamin C. Pierce (Penn)
Alexandre Pilkiewicz (Polytechnique/INRIA)

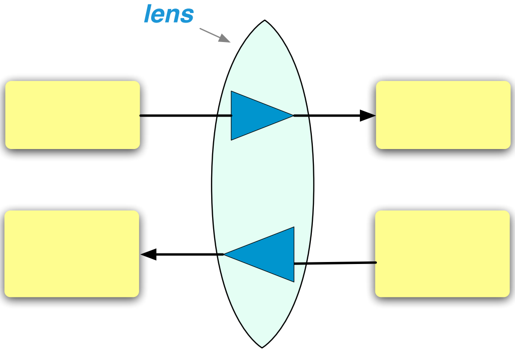


ICFP '08

Bidirectional Transformations



Bidirectional Programming Language



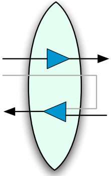
Eliminates Redundancy: programs describes two functions

Ensures Correctness: type system guarantees well-behavedness

Semantics

A lens l from S to T is a triple of functions

$$\begin{array}{lcl} l.get & \in & S \rightarrow T \\ l.put & \in & T \rightarrow S \rightarrow S \\ l.create & \in & T \rightarrow S \end{array}$$



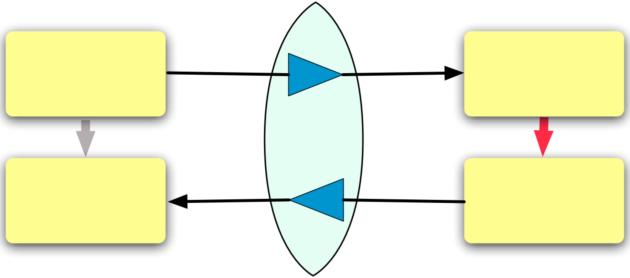
obeying three "round-tripping" laws:

$$l.put\ (l.get\ s)\ s = s \qquad \text{(GetPut)}$$

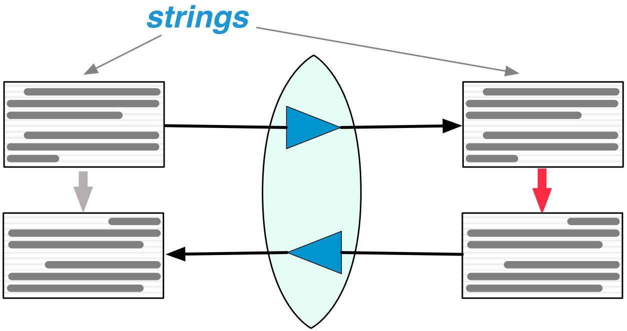
$$l.get\ (l.put\ t\ s) = t \qquad \text{(PutGet)}$$

$$l.get\ (l.create\ t) = t \qquad \text{(CreateGet)}$$

Boomerang [POPL '08]

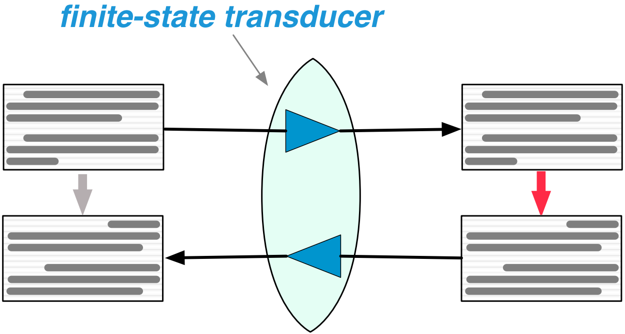


Boomerang [POPL '08]

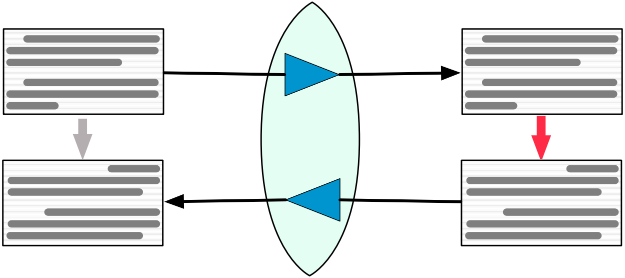


strings

Boomerang [POPL '08]



Boomerang [POPL '08]

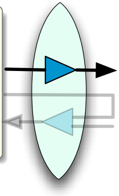


Lenses: addresses books, bibliographies, CSV, documents, scientific data, XML

Applications: converters, synchronizers, structure editors

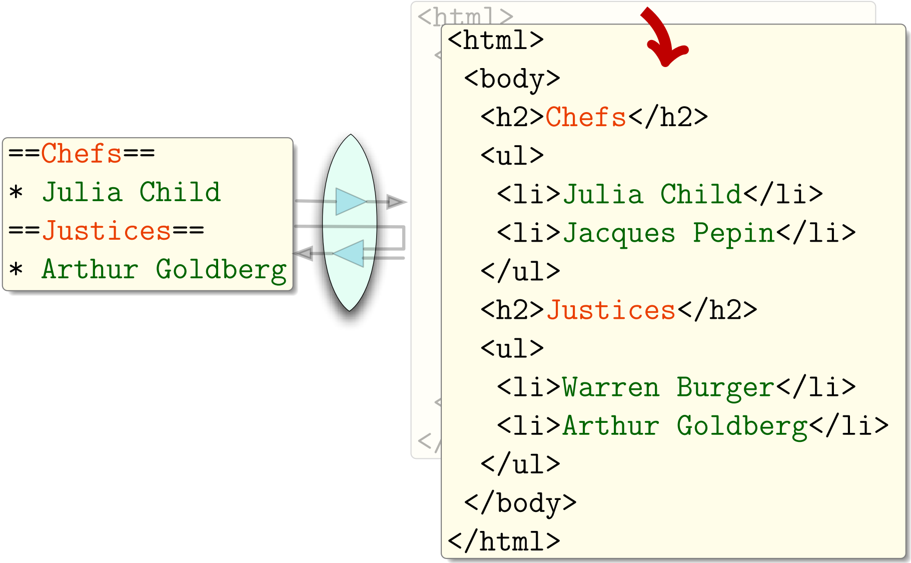
Example: MediaWiki (Get)

```
==Chefs==
* Julia Child
==Justices==
* Arthur Goldberg
```



```
<html>
 <body>
  <h2>Chefs</h2>
  <ul>
   <li>Julia Child</li>
  </ul>
  <h2>Justices</h2>
  <ul>
   <li>Arthur Goldberg</li>
  </ul>
 </body>
</html>
```

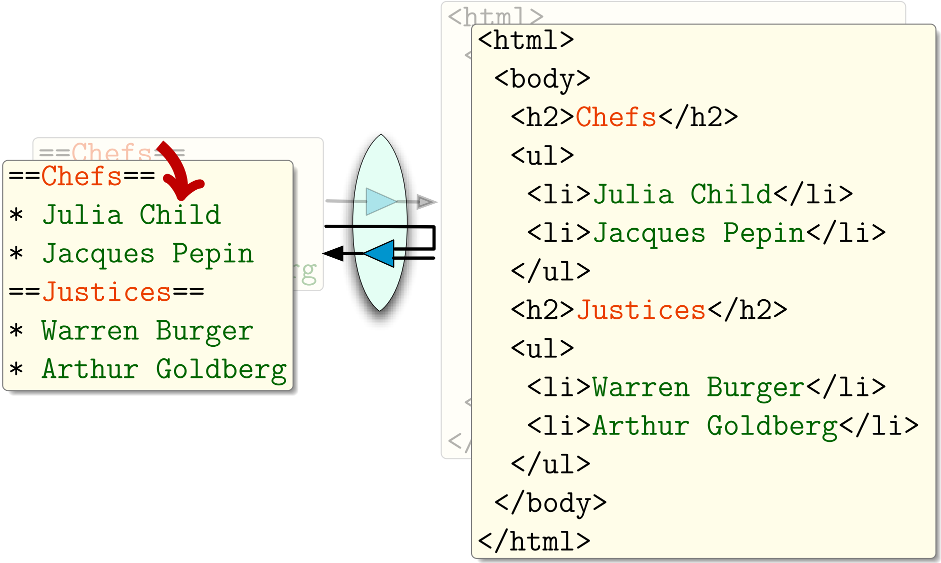
Example: MediaWiki (Update)



```
==Chefs==
* Julia Child
==Justices==
* Arthur Goldberg
```

```
<html>
 <body>
  <h2>Chefs</h2>
  <ul>
   <li>Julia Child</li>
   <li>Jacques Pepin</li>
  </ul>
  <h2>Justices</h2>
  <ul>
   <li>Warren Burger</li>
   <li>Arthur Goldberg</li>
  </ul>
 </body>
</html>
```

Example: MediaWiki (Put)



```
==Chefs==
* Julia Child
* Jacques Pepin
==Justices==
* Warren Burger
* Arthur Goldberg
```

```
<html>
 <body>
  <h2>Chefs</h2>
  <ul>
   <li>Julia Child</li>
   <li>Jacques Pepin</li>
  </ul>
  <h2>Justices</h2>
  <ul>
   <li>Warren Burger</li>
   <li>Arthur Goldberg</li>
  </ul>
 </body>
</html>
```

Example: MediaWiki (Lens)

```
(* helpers *)
let mk_elt (ws:string) (tag:string) (body:lens) = ...
let mk_simple_elt (ws:string) (tag:string) (body:lens) =
  ins ws .
  ins ("<" . tag . ">") .
  body .
  ins ("</" . tag . ">")

(* main lenses *)
let p : lens =
  mk_simple_elt nl4 "p" ((text . nl)* . (text . del nl))
let li : lens =
  mk_simple_elt nl6 "li" (del "* " . text)
let ul : lens =
  mk_elt nl4 "ul" (li . del nl)+
let h2 : lens =
  mk_simple_elt nl4 "h2" (del "==" . text . del "==")
let s : lens =
  (del nl . (p | ul))*
let html : lens =
  mk_outer_elt nl0 "html" (mk_elt nl2 "body" s* )
```

This Talk: Lenses for... ?

This Talk: Lenses for Whitespace!

Many data formats contain inessential information:

```
<html>\n
␣␣<body>\n
␣␣␣␣<h2>Famous Chefs</h2>\n
␣␣␣␣<ul>\n
␣␣␣␣␣␣<li>Julia Child</li>\n
␣␣␣␣</ul>\n
␣␣␣␣<h2>Supreme Court Justices</h2>\n
␣␣␣␣<ul>\n
␣␣␣␣␣␣<li>Arthur Goldberg</li>\n
␣␣␣␣</ul>\n
␣␣</body>\n
</html>\n
```

This Talk: Lenses for Whitespace!

Many data formats contain inessential information:

```
<html>\n
<body>\n
<h2>Famous Chefs</h2>\n
<ul>\n
<li>Julia Child</li>\n
</ul>\n
<h2>Supreme Court Justices</h2>\n
<ul>\n
<li>Arthur Goldberg</li>\n
</ul>\n
</body>\n
</html>\n
```

This Talk: Lenses for Whitespace!

Many data formats contain inessential information:

```
<html><body>\n
␣␣<h2>Famous Chefs</h2>\n
␣␣<ul><li>Julia Child</li></ul>\n
␣␣<h2>Supreme Court Justices</h2>\n
␣␣<ul><li>Arthur Goldberg</li></ul>\n
</body></html>\n
```

Want the *put* function to treat these targets equivalently but

$$l.get\,(l.put\ t\ s) = t \qquad (\textsc{PutGet})$$

implies they must map to different sources!

Dealing With Ignorable Data

Approach #1: No laws.

Transformations not required to obey any formal properties.

But clearly intended to be "essentially" bidirectional.

Backed up by intuitive understanding of implementation.

Examples:

- biXid [Kawanaka and Hosoya '06]
- PADS [AT&T / Princeton]

Dealing With Ignorable Data

Approach #2: Weaker laws.

Replace round-trip laws with round-trip-and-a-half versions.

Allows transformations that normalize data in the target...

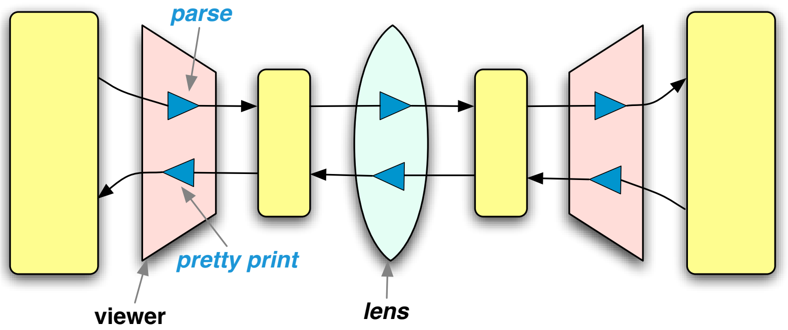
...and also many ill-behaved transformations.

Examples:

- Inv [Mu,Hu,Takeichi '04]
- X [Hu,Mu,Takeichi '04]
- Bi-XQuery [Liu, Hu, Takeichi '07]

Dealing With Ignorable Data

Approach #3: Viewers.



Examples:

- Focal [POPL '05]
- XSugar [Brabrand, Møller, Schwartzbach '05]

Dealing With Ignorable Data

Or... develop a theory of lenses that are well-behaved modulo equivalence relations on the source ($\sim_S$) and target ($\sim_T$).

Dealing With Ignorable Data

Or... develop a theory of lenses that are well-behaved modulo equivalence relations on the source ($\sim_S$) and target ($\sim_T$).

A quotient lens l satisfies the following laws

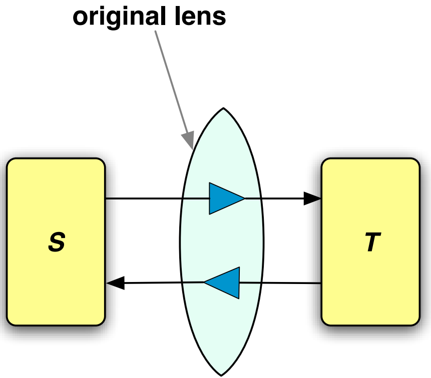
$$l.put\ (l.get\ s)\ s \sim_S s \qquad (\textsc{GetPut})$$

$$l.get\ (l.put\ t\ s) \sim_T t \qquad (\textsc{PutGet})$$

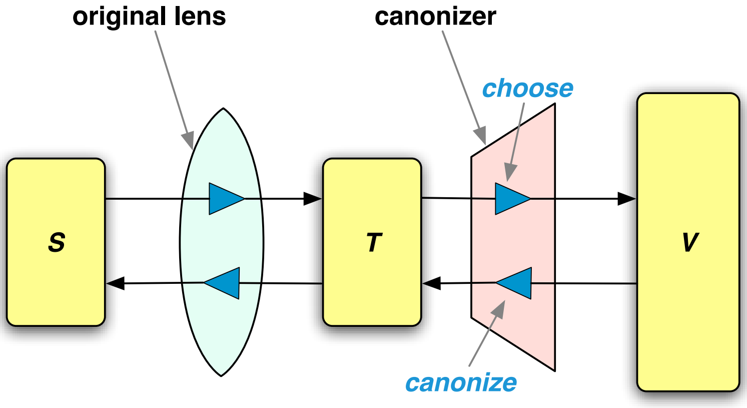
$$l.get\ (l.create\ t) \sim_T t \qquad (\textsc{CreateGet})$$

(Plus laws ensuring that l's components respect $\sim_S$ and $\sim_T$.)

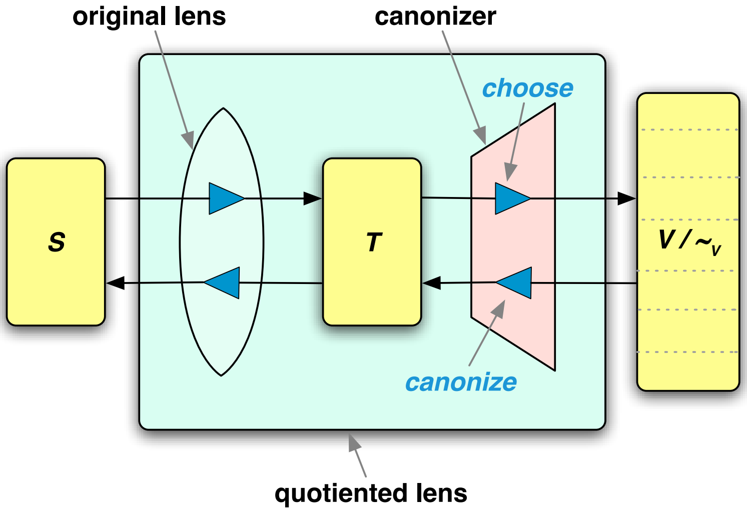
Syntax for Quotient Lenses



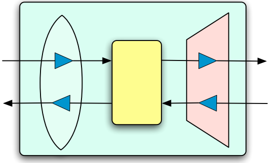
Syntax for Quotient Lenses



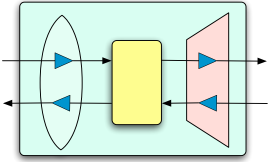
Syntax for Quotient Lenses

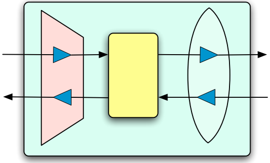


Syntax for Quotient Lenses

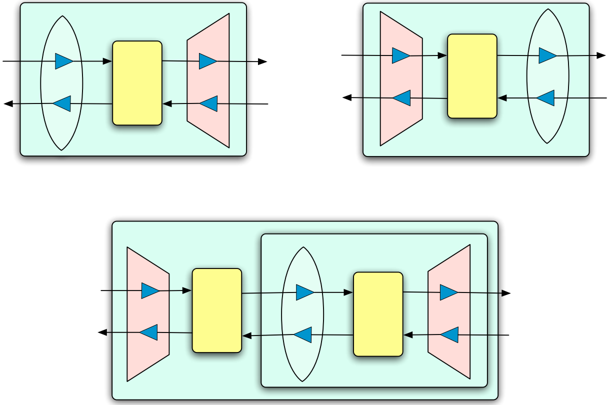


Syntax for Quotient Lenses

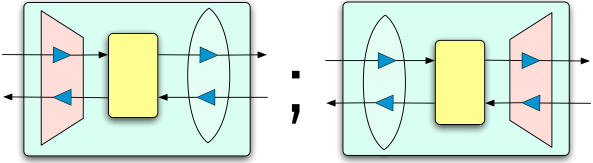




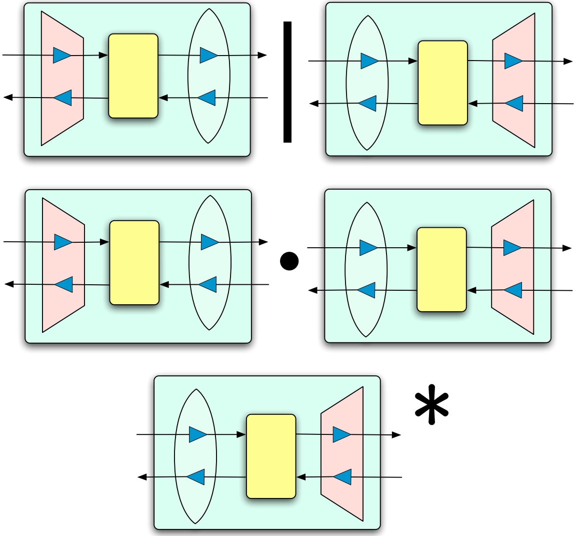
Syntax for Quotient Lenses



Syntax for Quotient Lenses



Syntax for Quotient Lenses



Example: MediaWiki (Lens)

```
(* helpers *)
let mk_elt (ws:string) (tag:string) (body:lens) = ...
let mk_simple_elt (ws:string) (tag:string) (body:lens) =
  ins ws .
  ins ("<" . tag . ">") .
  body .
  ins ("</" . tag . ">")

(* main lenses *)
let p : lens =
  mk_simple_elt nl4 "p" ((text . nl)* . (text . del nl))
let li : lens =
  mk_simple_elt nl6 "li" (del "* " . text)
let ul : lens =
  mk_elt nl4 "ul" (li . del nl)+
let h2 : lens =
  mk_simple_elt nl4 "h2" (del "==" . text . del "==")
let s : lens =
  (del nl . (p | ul))*
let html : lens =
  mk_outer_elt nl0 "html" (mk_elt nl2 "body" s* )
```

Example: MediaWiki (Lens)

```
(* helpers *)
let mk_elt (ws:string) (tag:string) (body:lens) = ...
let mk_simple_elt (ws:string) (tag:string) (body:lens) =
  ins ws .
  ins ("<" . tag . ">") .
  body .
  ins ("</" . tag . ">")

(* main lenses *)
let p : lens =
  mk_simple_elt nl4 "p" ((text . nl)* . (text . del nl))
let li : lens =
  mk_simple_elt nl6 "li" (del "* " . text)
let ul : lens =
  mk_elt nl4 "ul" (li . del nl)+
let h2 : lens =
  mk_simple_elt nl4 "h2" (del "==" . text . del "==")
let s : lens =
  (del nl . (p | ul))*
let html : lens =
  mk_outer_elt nl0 "html" (mk_elt nl2 "body" s* )
```

Example: MediaWiki (Lens)

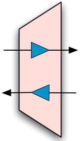
```
(* helpers *)
let mk_elt (ws:string) (tag:string) (body:lens) = ...
let mk_simple_elt (ws:string) (tag:string) (body:lens) =
  qins WS ws .
  ins ("<" . tag . ">") .
  body .
  ins ("</" . tag . ">")

(* main lenses *)
let p : lens =
  mk_simple_elt nl4 "p" ((text . nl)* . (text . del nl))
let li : lens =
  mk_simple_elt nl6 "li" (del "* " . text)
let ul : lens =
  mk_elt nl4 "ul" (li . del nl)+
let h2 : lens =
  mk_simple_elt nl4 "h2" (del "==" . text . del "==")
let s : lens =
  (del nl . (p | ul))*
let html : lens =
  mk_outer_elt nl0 "html" (mk_elt nl2 "body" s* )
```

Canonizers

A canonizer q from V to T is a pair of functions

$$\begin{array}{lcl} q.canonize & \in & V \longrightarrow T \\ q.choose & \in & T \longrightarrow V \end{array}$$



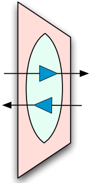
obeying just one law:

$$l.canonize\,(l.choose\ t)\ t = t \qquad \text{(ReCanonize)}$$

Syntax for Canonizers

Every lens l from V to T can be converted to a canonizer:

$$\begin{array}{lcl} q.\mathit{canonize} & \triangleq & l.\mathit{get} \\ q.\mathit{choose} & \triangleq & l.\mathit{create} \end{array}$$



The $\textsc{CreateGet}$ law for l implies $\textsc{ReCanonize}$.

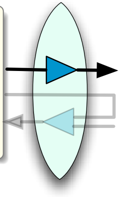
Additionally, the relaxed canonizer law enable primitives that are not valid as lenses.

An Unexpected Side Benefit...

The increased flexibility of quotient lenses can be exploited to simplify the types of complicated transformations.

Example: Table of Contents (Get)

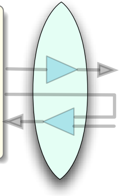
```
==Chefs==
* Julia Child
==Justices==
* Arthur Goldberg
```



```
<html>
 <body>
  <ul>
    <li>Chefs</li>
    <li>Justices</li>
  </ul>
  <h2>Chefs</h2>
  <ul>
   <li>Julia Child</li>
  </ul>
  <h2>Justices</h2>
  <ul>
   <li>Arthur Goldberg</li>
  </ul>
 </body>
</html>
```

Example: Table of Contents (Update)

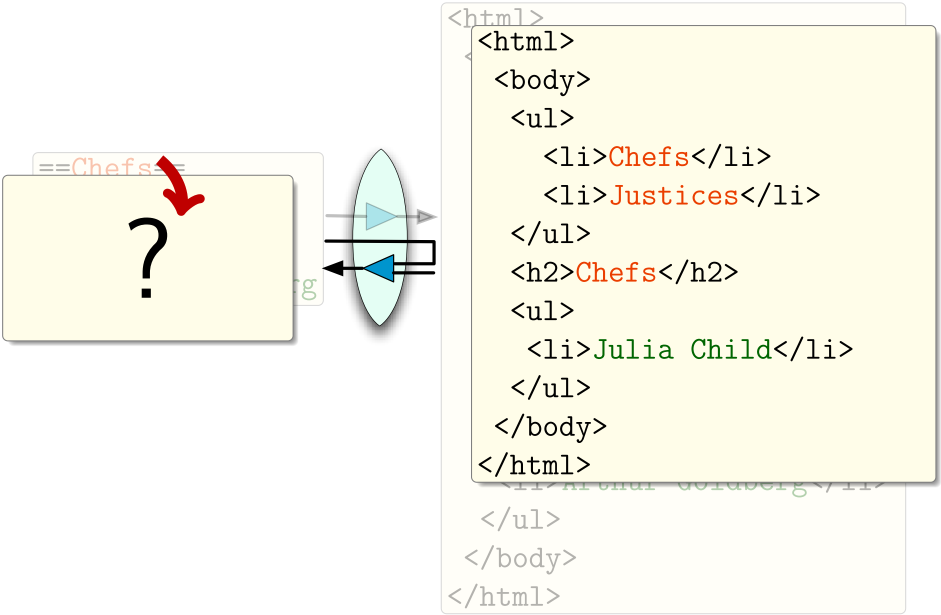
```
==Chefs==
* Julia Child
==Justices==
* Arthur Goldberg
```



```
<html>
 <body>
  <ul>
    <li>Chefs</li>
    <li>Justices</li>
  </ul>
  <h2>Chefs</h2>
  <ul>
   <li>Julia Child</li>
  </ul>
 </body>
</html>
```

```
<html>
 <
   <li>Arthur Goldberg</li>
  </ul>
 </body>
</html>
```

Example: Table of Contents (Put)



```
<html>
 <body>
  <ul>
    <li>Chefs</li>
    <li>Justices</li>
  </ul>
  <h2>Chefs</h2>
  <ul>
   <li>Julia Child</li>
  </ul>
 </body>
</html>
```

Flexibility with Quotient Lenses

To satisfy PutGet the duplication lens needs a type that demands equality for the copied data.

But enriching types with equality constraints makes type checking awkward.

Flexibility with Quotient Lenses

To satisfy PutGet the duplication lens needs a type that demands equality for the copied data.

But enriching types with equality constraints makes type checking awkward.

As a quotient lens, we can assign the duplication lens a simpler (regular) type.

- Using a total equivalence on the second copy of the data in the targt.

This flexiblity also simplifies the types of primitives for

- sorting
- wrapping lines of text

Conclusion

- The need to handle inessential data arises in many real-world applications built using lenses.
- Quotient lenses are a critical piece of technology that helps bridge the gap between the theory and practice of bidirectional programming languages.
- Canonizers lead to elegant syntax for quotient lenses.

Thank You!

Collaborators: Benjamin Pierce, Alexandre Pilkiewcz.

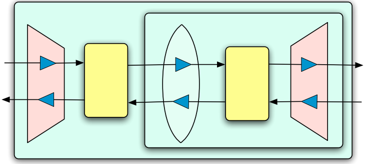
Other Boomerang contributors: Aaron Bohannon, Michael Greenberg, and Alan Schmitt.

Want to play? Boomerang is available for download:

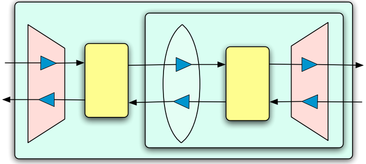
- Source code (LGPL)
- Binaries for OS X, Linux
- Research papers
- Tutorial and growing collection of demos

http://www.seas.upenn.edu/~harmony/

Type Checking Quotient Lenses



;



$$\frac{l \in S/{\sim_S} \Longleftrightarrow T/{\sim_T} \qquad k \in T/{\sim_T} \Longleftrightarrow V/{\sim_V}}{l; k \in S/{\sim_S} \Longleftrightarrow V/{\sim_V}}$$