Links

Philip Wadler
University of Edinburgh
wadler@inf.ed.ac.uk

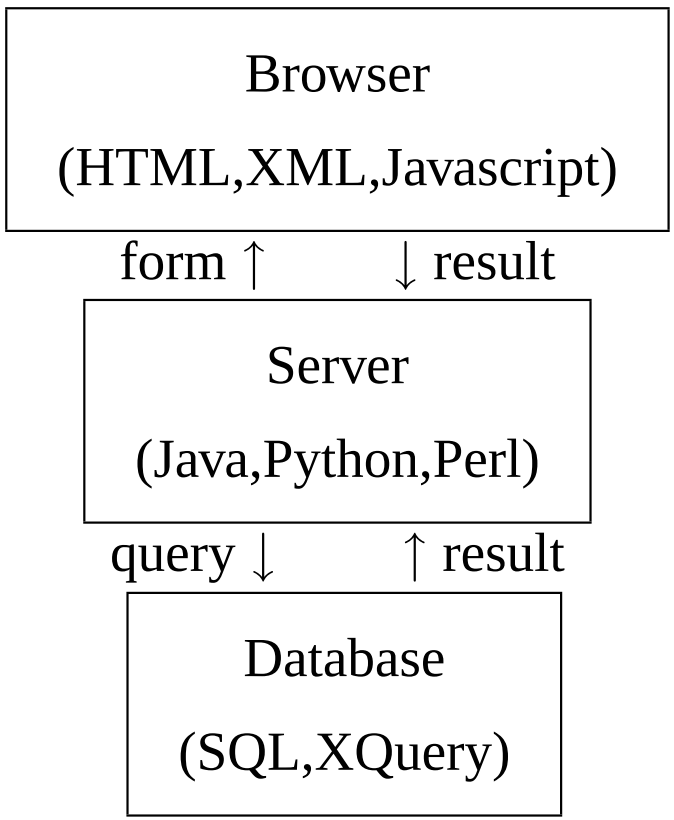
Less than a Grand Challenge

Design a programming language with
a sound basis in theory that is the
leader in its domain.

Success stories

- Klesli (databases)
- XDuce and XQuery (XML, databases)
- PLT Scheme (web applications)
- Erlang (distribution)

Three-tier model



Comprehensions for Data

Comprehensions from Monads

$$
\begin{array}{lrcl}
(i) & map\, id & = & id \\
(ii) & map\, (g \cdot f) & = & map\, g \cdot map\, f \\
(iii) & map\, f \cdot unit & = & unit \cdot f \\
(iv) & map\, f \cdot join & = & join \cdot map\, (map\, f) \\
 & & & \\
(I) & join \cdot unit & = & id \\
(II) & join \cdot map\, unit & = & id \\
(III) & join \cdot join & = & join \cdot map\, join \\
 & & & \\
(1) & [\, t \mid () \,] & = & unit\, t \\
(2) & [\, t \mid x \leftarrow u \,] & = & map\, (\lambda x.\, t)\, u \\
(3) & [\, t \mid (p, q) \,] & = & join\, [\, [\, t \mid q \,] \mid p \,]
\end{array}
$$

Monads from Comprehensions

$$
\begin{array}{lll}
(I') & [\, t \mid (),\, q \,] & = \quad [\, t \mid q \,] \\
(II') & [\, t \mid q,\, () \,] & = \quad [\, t \mid q \,] \\
(III') & [\, t \mid (p,\, q),\, r \,] & = \quad [\, t \mid p,\, (q,\, r) \,] \\
 & & \\
(a) & [\, x \mid x \leftarrow u] & = \quad u \\
(b) & [\, t \mid p,\, x \leftarrow [u|q],\, r \,] & = \quad [\, t[u/x] \mid p,\, q,\, r[u/x] \,] \\
 & & \\
(1') & unit\, x & = \quad [\, x \,] \\
(2') & map\, f\, xs & = \quad [\, f\, x \mid x \leftarrow xs \,] \\
(3') & join\, xss & = \quad [\, x \mid xs \leftarrow xss,\, x \leftarrow xs \,]
\end{array}
$$

Relational Data

TITLES

title	isbn	year
What Can You Do With a Shoe?	0613733266	1997
Where the Wild Things Are	0060254920	1963

AUTHORS

author	isbn
Beatrice Schenk de Regniers	0613733266
Maurice Sendak	0613733266
Maurice Sendak	0060254920

Relational Query

SQL

```
select TITLES.title, AUTHORS.author
from TITLES, AUTHORS
where TITLES.isbn = AUTHORS.isbn
  and TITLES.year < 2000
```

Kleisli

```
TITLES : {(title: String, isbn: Integer, year: Date)}
AUTHORS : {(author: String, isbn: Integer)}
{ (title: t.title, author: a.string) |
  \t <--- TITLES, \a <--- AUTHORS,
  t.isbn = a.isbn, t.year < 2000 }
```

XML Data

```
<books>
  <book>
    <title>Where the Wild Things Are</title>
    <author>Maurice Sendak</author>
    <isbn>0060254920</isbn>
    <year>1963</year>
  </book>
  <book>
    <title>What Can You Do With a Shoe?</title>
    <author>Beatrice Schenk de Regniers</author>
    <author>Maurice Sendak</author>
    <isbn>0613733266</isbn>
    <year>1997</year>
  </book>
</books>
```

XML Query

XQuery

```
for $b from input()/books/book
    $a from $b/author
where $b/year < 2000
return
  <book>{ $b/title, $a }</book>
```

Kleisli

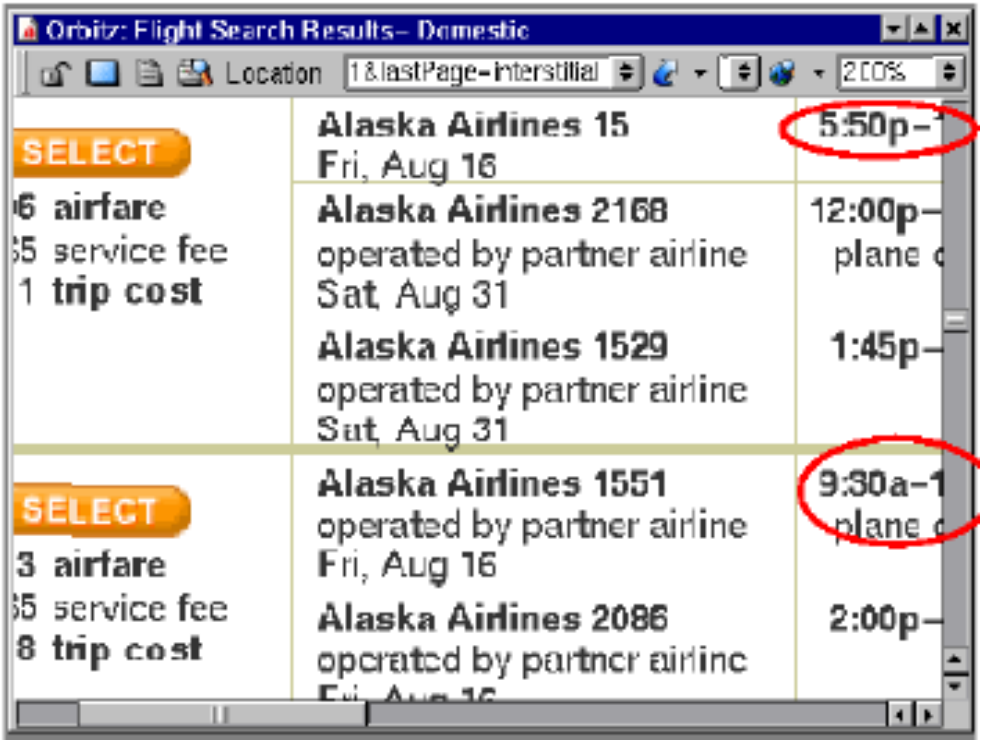
```
BOOKS : {(title: String,
          authors: [String],
          isbn: Integer,
          year: Date)}
{ (title: b.title, author: a) |
  \b <--- BOOKS, \a <-- t.authors,
  b.year < 2000 }
```

Related work

- Kleisli (Buneman, Libkin, Suciu, Tannen, Wong)
- Mnesia/Erlang (Wikström)
- Pdiff (Griffin and Trickey)
- Natural Expert (Hutchison, Neuhaus, Schmidt-Schauss)

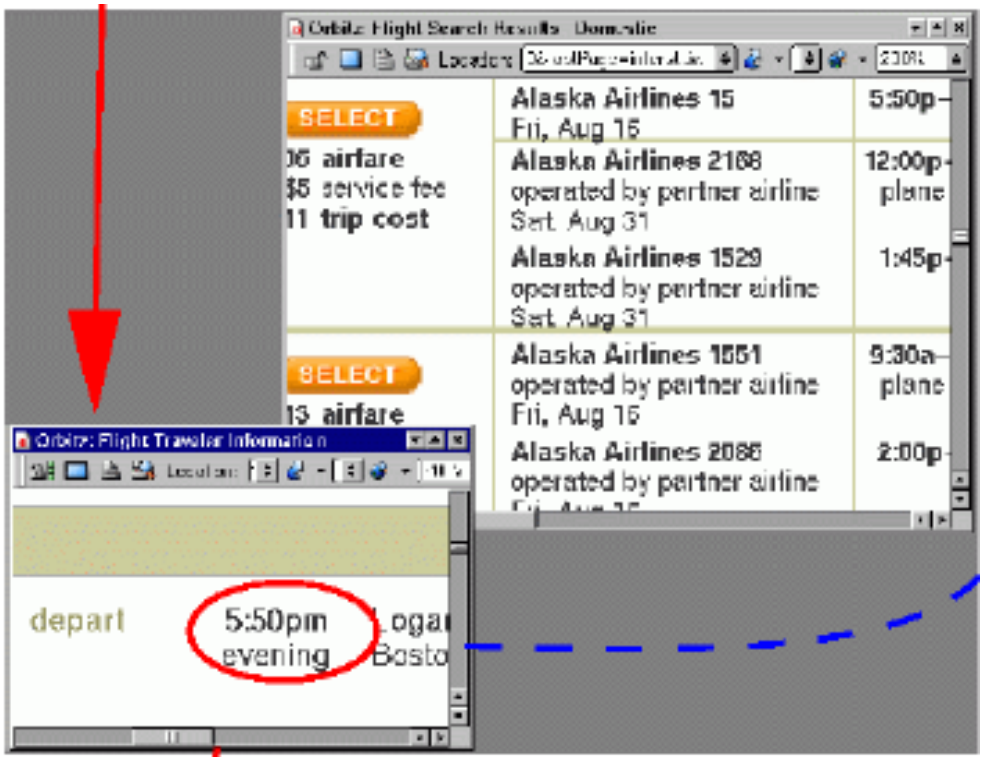
Continuations for the Web

Orbitz: Two flights

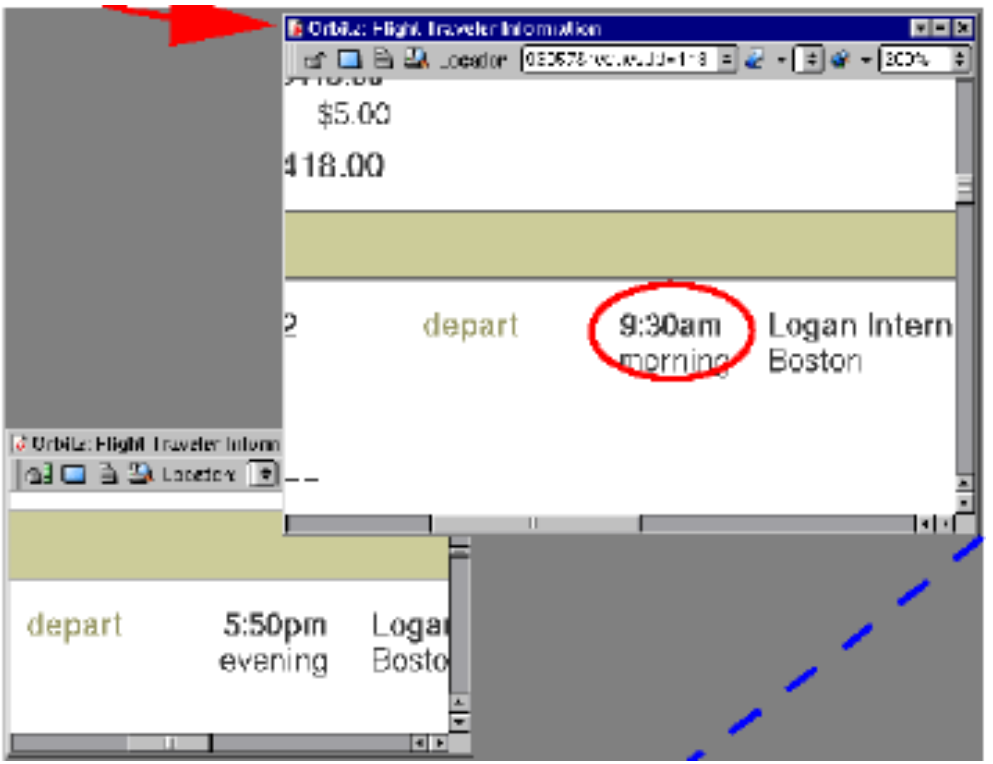


Graunke, Findler, Krishnamurthi, Felleisen (ESOP 2003)

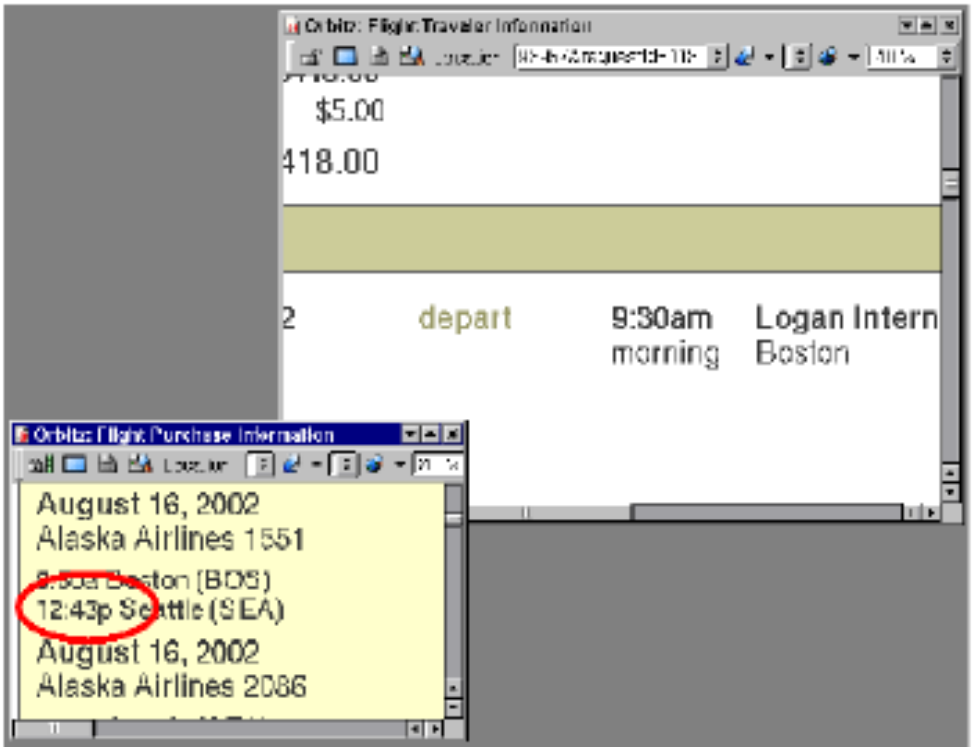
Orbitz: Clone and submit first



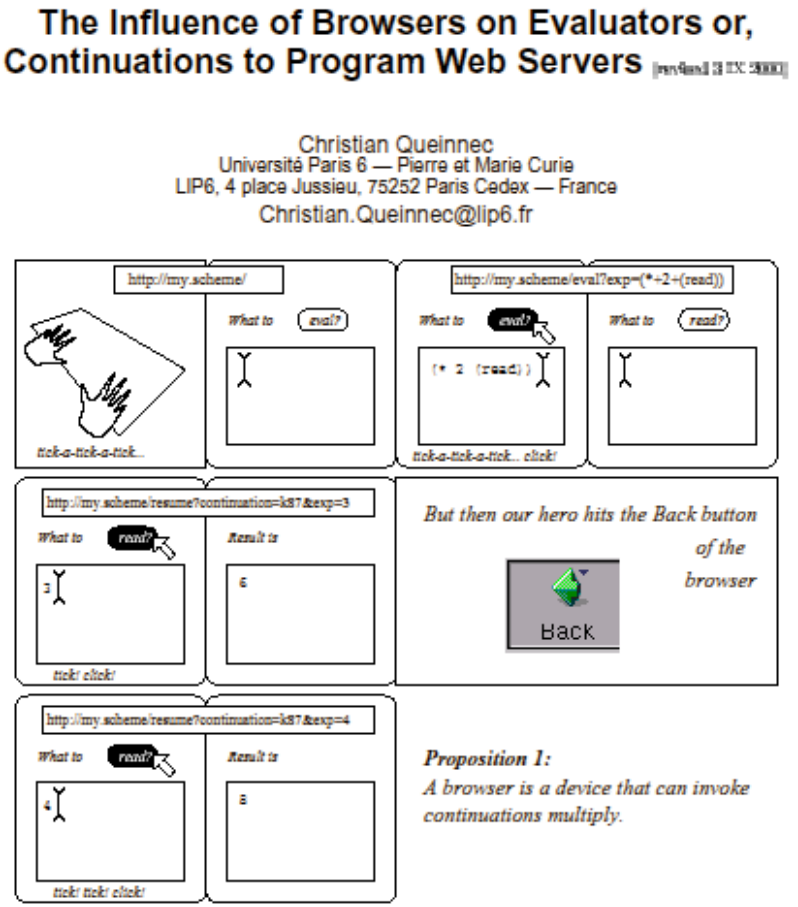
Orbitz: Submit second



Orbitz: Select first – problem!



Quenniec: Browsers and continuations



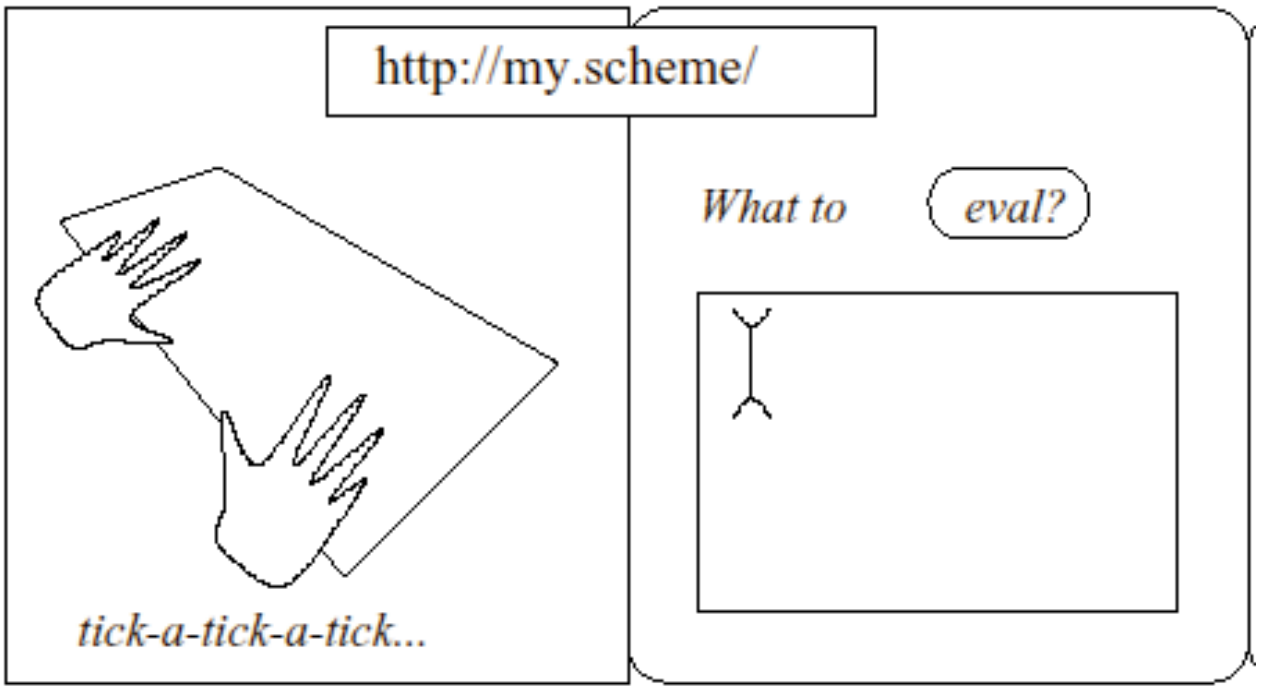
The Influence of Browsers on Evaluators or, Continuations to Program Web Servers

Christian Queinnec
Université Paris 6 — Pierre et Marie Curie
LIP6, 4 place Jussieu, 75252 Paris Cedex — France
Christian.Queinnec@lip6.fr

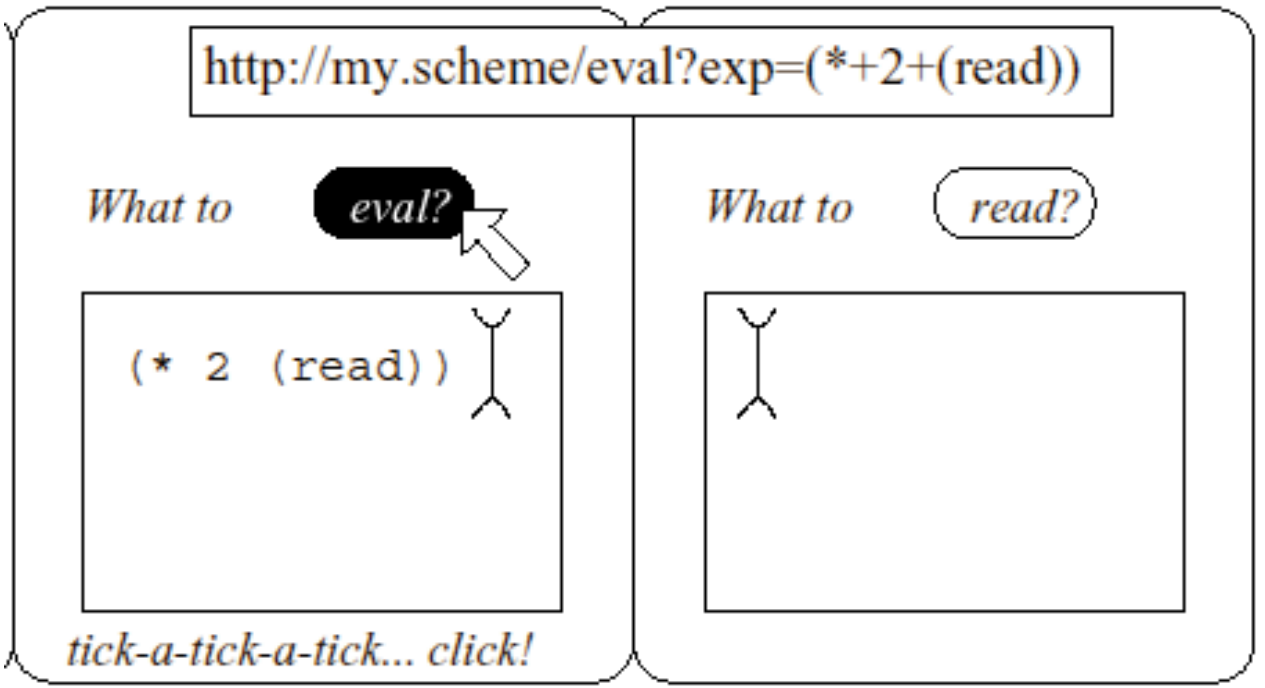
Christian Quenniec (ICFP 2000)

also John Hughes, Paul Graham

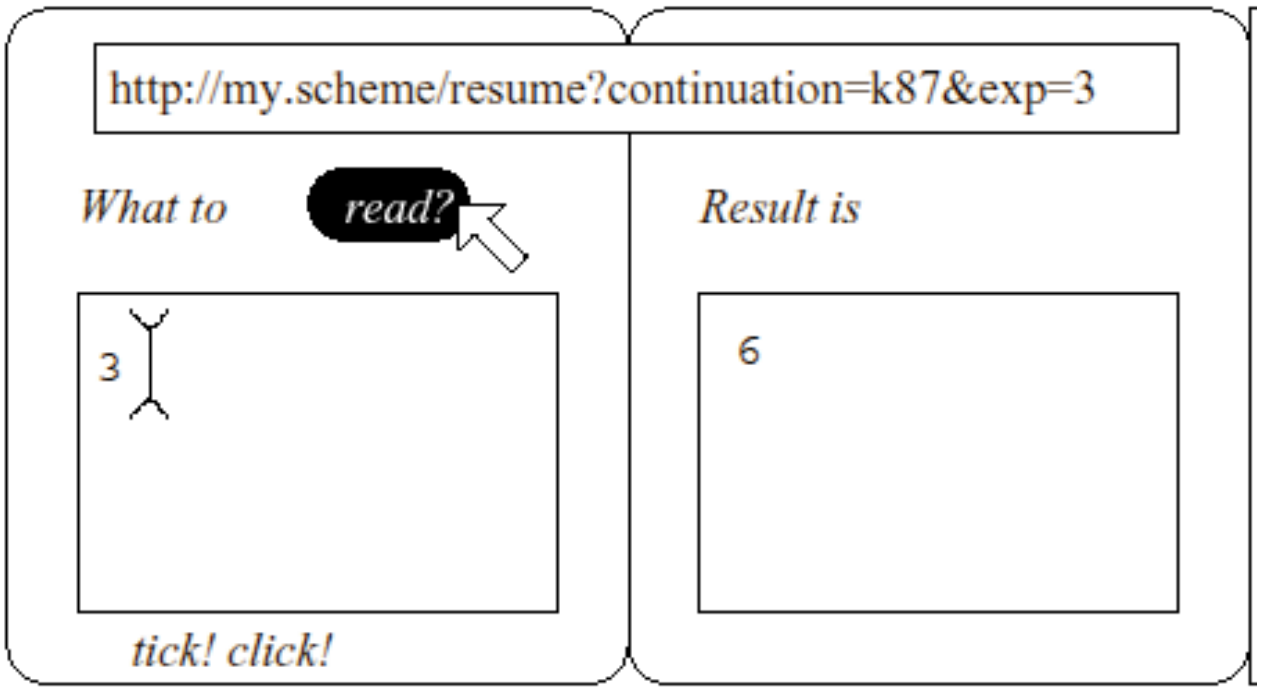
Quenniec: Go to web page



Quenniec: First argument



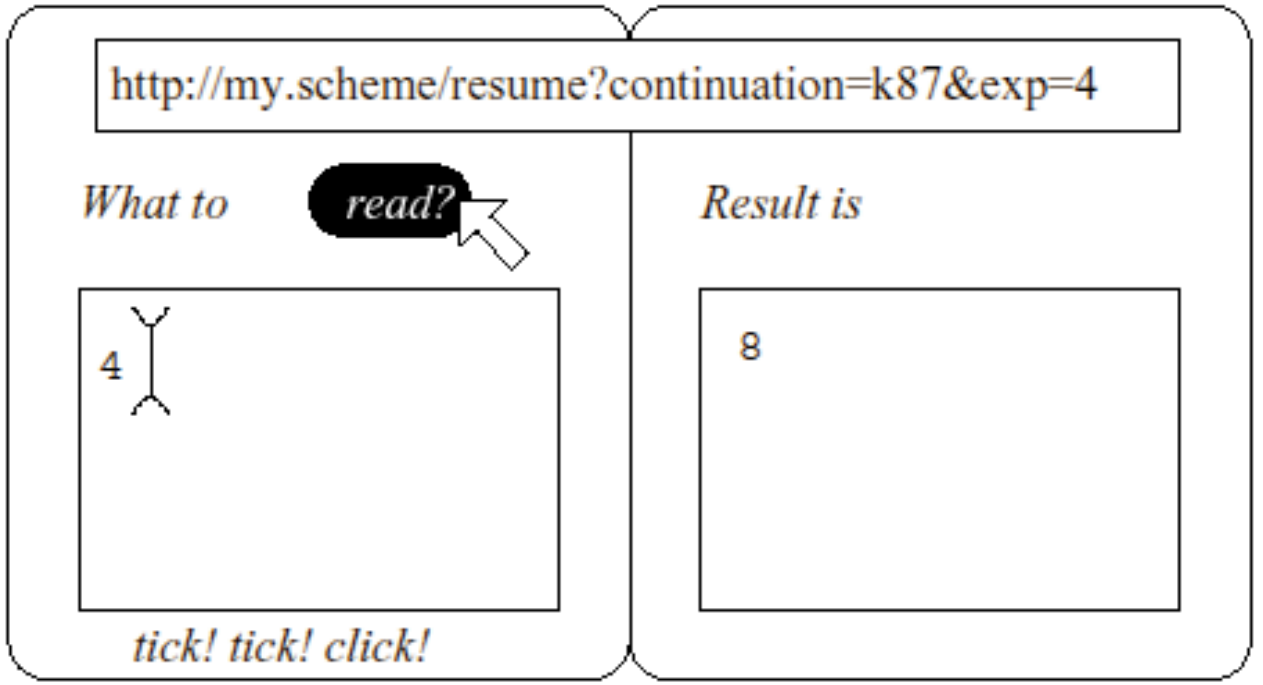
Quenniec: Second argument



Quenniec: Back button

But then our hero hits the Back button of the browser

Quenniec: Second argument, second time



Related work

- Mawl (Ramming, Atkins, Ball, Bruns, Cox)
- Continuations (Quiennec)
- PLT Scheme (Graunke, Findler, Krishnamurthi, Felleisen)
- Bigwig (Brabrand, Sandholm, Møller, Schwartzbach)
- WASH (Thiemann)

Communication via values

Erlang: An area server

```
start() ->
    register(area_server,
        spawn(fun() -> loop(0) end)).
loop(Tot) ->
    receive
        {Pid, {square, X}} ->
            Pid ! X*X,
            loop(Tot + X*X);
        {Pid, {rectangle, [X,Y]}} ->
            Pid ! X*Y,
            loop(Tot + X*Y);
        {Pid, areas} ->
            Pid ! Tot,
            loop(Tot)
    end.
```

Erlang: Generic server

```
start(Name, Data, Fun) ->
    register(Name,
        spawn(fun() -> loop(Data, Fun) end)).
rpc(Name, Query) ->
    Tag = ref(),
    Name ! {query, self(), Tag, Query},
    receive
        {Tag, Reply} -> Reply
    end.
loop(Data, Fun) ->
    receive
        {query, Pid, Tag, Query} ->
            {Reply, Data1} = Fun(Query, Data),
            Pid ! {Tag, Reply},
            loop(Data1, Fun)
    end.
```

Erlang: Instantiating the Generic Server

```
start() ->
    start(area_server, 0, handler/2).
handler({square, X}, Tot) ->
            {X*X, Tot + X*X};
handler({rectangle, [X,Y]}, Tot) ->
            {X*Y, Tot + X*Y};
handler(areas, Tot) ->
            {Tot, Tot}.
```

Erlang: Instantiating a Replicated Server

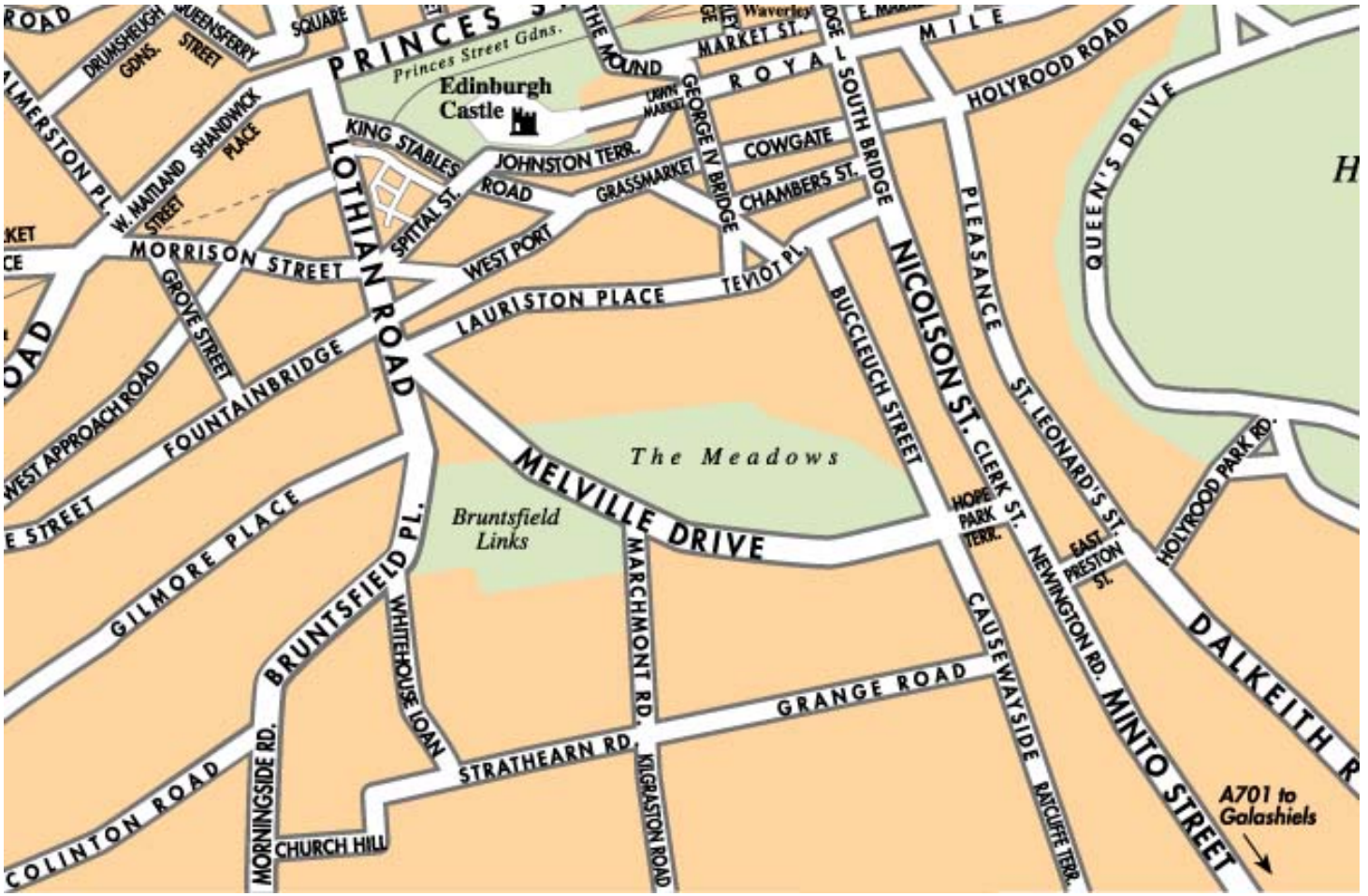
```
start() ->
    start_replicated(area_server, 0, handler/2).
handler({square, X}, Tot) ->
            {X*X, Tot + X*X};
handler({rectangle, [X,Y]}, Tot) ->
            {X*Y, Tot + X*Y};
handler(areas, Tot) ->
            {Tot, Tot}.
```

Related work

- Erlang (Armstrong, Virding, Wikström, Williams)
- Ensemble (Hayden and vanRenesse)
- Fox (Harper and Lee)
- Plan X (Henglein)

Links

Hope and Links



Links

Hope

(Hope Park Square)

Burstall, MacQueen,

Sannella (1980)

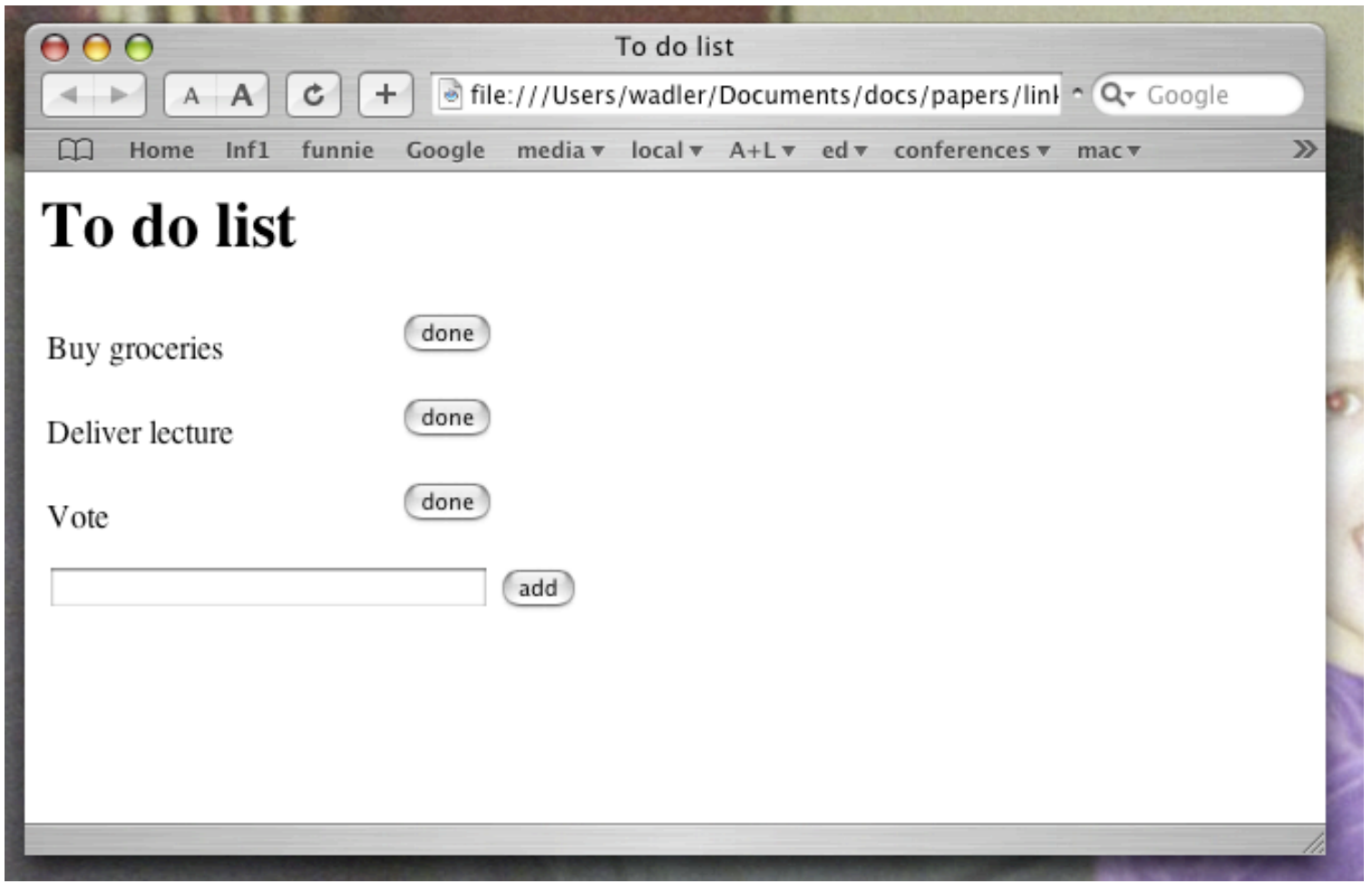
Links

(Bruntsfield Links)

Wadler et al (2005)

A Links program state in client

```
main() ->
  todo([]).
todo(items) ->
  <html><body>
    <h1>Items to do</h1>
    <table>{
      for item in items return
        <tr>
          <td>{item}</td>
          <td>
            <form action="{todo(items\\[item])}">
              <input type="submit" value="done"/>
            </form>
          </td>
        </tr>
    }</table>
    <form action="{todo(items++[new])}">
      <input name="{new}" type="text" size="40">
      <input type="submit" value="add"/>
    </form>
  </body></html>.
```


To do list
file:///Users/wadler/Documents/docs/papers/link
Google
Home
Inf1
funnie
Google
media
local
A+L
ed
conferences
mac
To do list
Buy groceries
done
Deliver lecture
done
Vote
done
add

A Links program state in server

```
type Name = String.
type Item = String.
table TODO of (name : Name, item : Item)
lookup(n) ->
  [ i | (name:n,item:i) <- TODO ].
add(n,i) ->
  insert into TODO values (name:n, item:i),
  todo(name).
remove(n,i) ->
  remove from TODO values (name:n, item:i),
  todo(name).
main() ->
  <html><body>
    <h1>Login</h1>
    <form action="todo(name)">
      <input name="{name}" type="text" size="40">
      <input type="submit" value="login"/>
    </form>
  </body></html>.
```

```
todo(name) ->
  let items = lookup(name) in
  <html><body>
    <h1>Items to do</h1>
    <table>{
      for item in items return
        <tr>
          <td>{item}</td>
          <td>
            <form action="{remove(name,item)}">
              <input type="submit" value="done"/>
            </form>
          </td>
        </tr>
    }</table>
    <form action="{add(name,new)}">
      <input name="{new}" type="text" size="40">
      <input type="submit" value="add"/>
    </form>
  </body></html>.
```

A few open questions

- Syntax? – Can we take a scientific approach?
- Type classes and generic programming?
- Types for servers?
- Integration with Java or .NET?
- Transactions?
- Monads and effect types?