(def declarative)

Christophe Grand, @cgrand
Rotterdam, 27th March 2013

Disclaimer

- Nothing is definitive
- Just my current understanding and intuition

WHAT

VS

HOW

Subjective

Let’s ignite a flamewar!

When climbing the abstraction ladder...

one's WHAT
is the
other's HOW

Let’s backtrack!

Why is FP good?

easier reasoning

about code

define easier

local reasoning about code

referential transparency

«An expression is said to be referentially transparent if it can be replaced with its value.»

referential transparency

«An expression is said to be referentially transparent if it can be replaced with its value.»

Corollaries

- Two expressions evaluating to the same value are interchangeable.
- Evaluation order does not matter.

Corollaries

Idempotency

- Two expressions evaluating to the same value are interchangeable.
- Evaluation order does not matter.

Corollaries

Idempotency

- Two expressions evaluating to the same value are interchangeable.
- Evaluation order does not matter.

Commutativity

idempotency

+ commutativity

= sound semantics

Performance is a separate concern

What is not idempotent and not commutative?

What repeats and is ordered?

lists!
sequences!
vectors!
code source!

Sequentiality considered harmful

Short-circuiting and/or

- Refactor with care!
- Swapping two expressions may:
  - Cause exceptions
    (and (not= x 0) (/ 1 x))
  - Change the returned value
    (and (pred x) (lookup x))

Short-circuiting and/or

- You can't know when order matters
- The compiler can't either

Waiting to bite you

Regexes

```
=> (re-seq #"a|ab" "abababab")
("a" "a" "a" "a")
```

- Choice is ordered
- You can’t locally fix a regex

CFG vs RDP

- L ::= "a" L "a" | "a"
- Which strings are matched?

CFG vs RDP

- L ::= "a" L "a" | "a"
- Which strings are matched?
  - CFG: any string of 2N+1 "a"s
  - RDP: any string of 2^N-1 "a"s

CFG vs RDP

- L ::= "a" | "a" L "a"
- Which strings are matched?

CFG vs RDP

- L ::= "a" | "a" L "a"
- Which strings are matched?
  - CFG: any string of 2N+1 "a"s
  - RDP: just "a"

miniKanren

- Depending on the ordering of your disjunctions and conjunctions...
- ...your program may run endlessly without ever returning a single answer
- Defaults fixed for disjunction in the 2nd edition

core.logic

- That's why *fair* disjunction/conjunction is important!
- Purity/fairness is hard to implement efficiently

self-criticism & doubts

- Moustache
- Ordered routes
- Why only routes?

self-criticism & doubts

- Enlive
  - Defaults to ordered selectors
  - Non-ordered selectors bounded by the current transformation
  - Why the selector/fn division?

Defaults

Defaults shape a language

Viscosity / Wood grain

Defaults hinder us

e.g. Java mutability by default
e.g. «Reduce considered harmful»
e.g. short-circuiting or/and

Exigent defaults

No mutability
No sequentiality
No duplicates
No determinism
...
Constraints are good for creativity!

API Design ideas

Quack!*

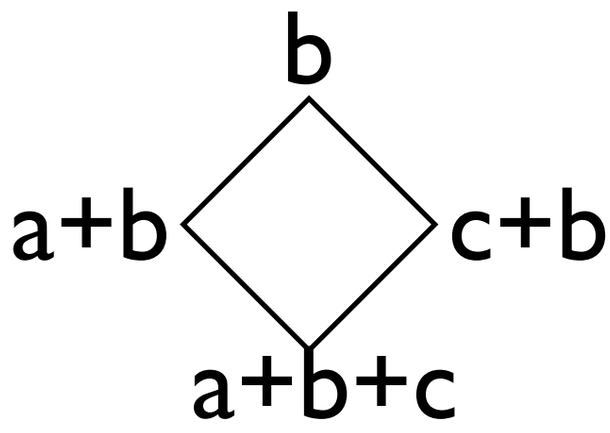
*Eureka in Duck

API Design ideas

- one abstraction
- composition-centric
- merge-like composition operation
  - sort of union or intersection

Consequences

- how to handle ambiguity?
  - the later the better
  - MUST correctly handle diamonds!
    - (op (op a b) (op c b))



SQRel

- The SQL lib that doesn't drive you nuts
- Everything is a rel
- Composition:
  - sort of natural join
    - qualified columns
- Demo

Mashup

- Prismatic's Graph + Spreadsheet + lazy
- Lazy maps backed by «formulas»
  - assoc → update dependent entries
  - composition → merge
  - encapsulation → remapping keys

Mashup: example 1

```clojure
(def m (mashup
         :a+b (fm [a b] (+ a b))))

(:a+b m) ; IllegalStateException
(:a+b (assoc m :a 1 :b 2)) ; 3
```

Mashup: example 2

```clojure
(def m (mash
         (mashup
           :a+b (fm [a b] (+ a b)))
         (mashup
           :b (fm [a] (* a a)))))

(:a+b (assoc m :a 3)) ; 12
```

Mashup API

- fm – fn-like constructor (macro)
- mash – composition
- remap/jail – encapsulation
- mashup – map-like constructor helper

```
(defn mashup [& kfs]
  (reduce mash
    (map (fn [[k f]] (remap f {::out k}))
      (partition 2 kfs))))
```

Advanced features

- Patterns as keys
- Conflict resolution with (incremental) accumulators

Tastes like...

- FRP
- IoC
- Whiteboards
- Tuple spaces

To be continued...

again: nothing I said is definitive