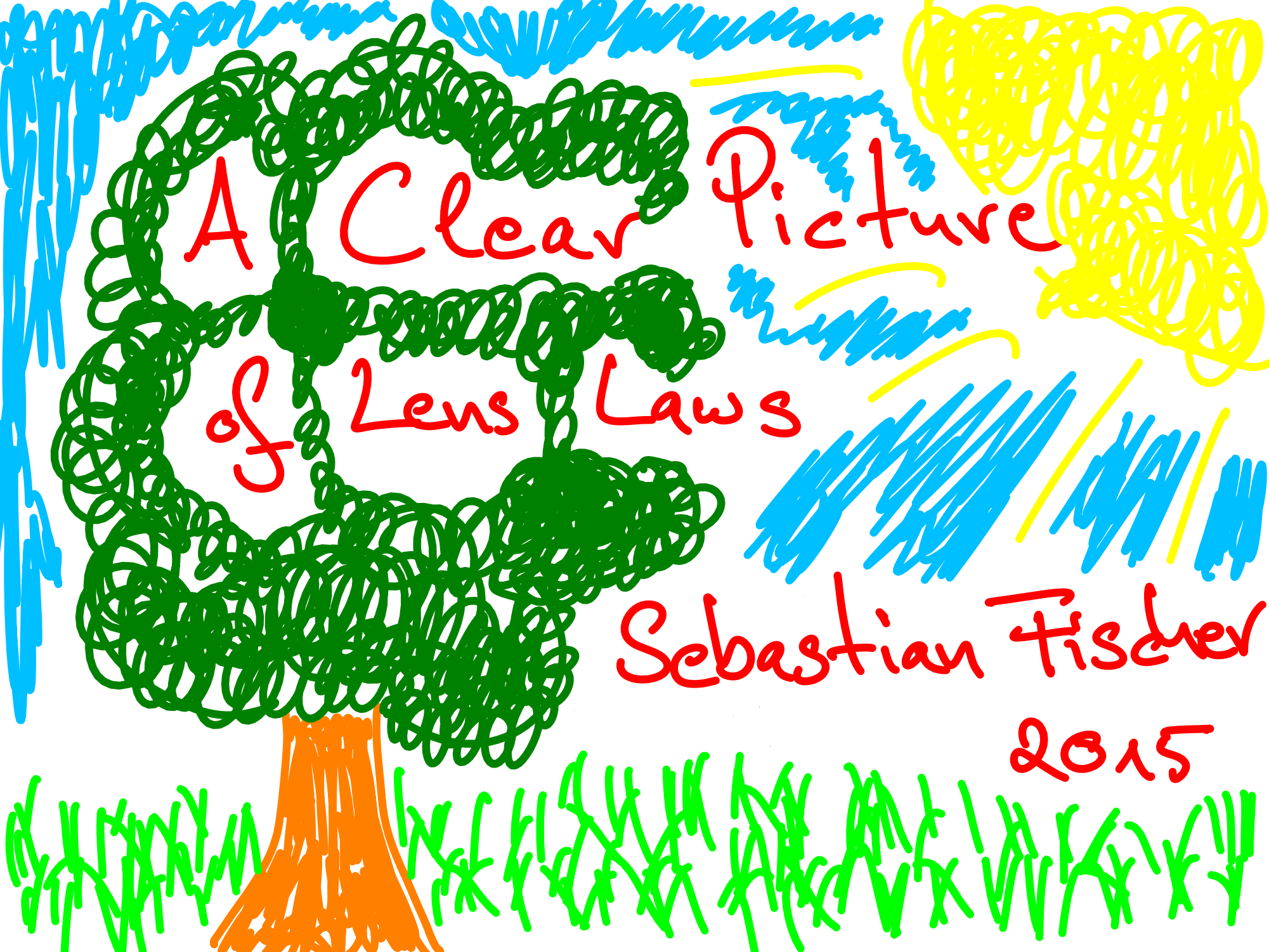
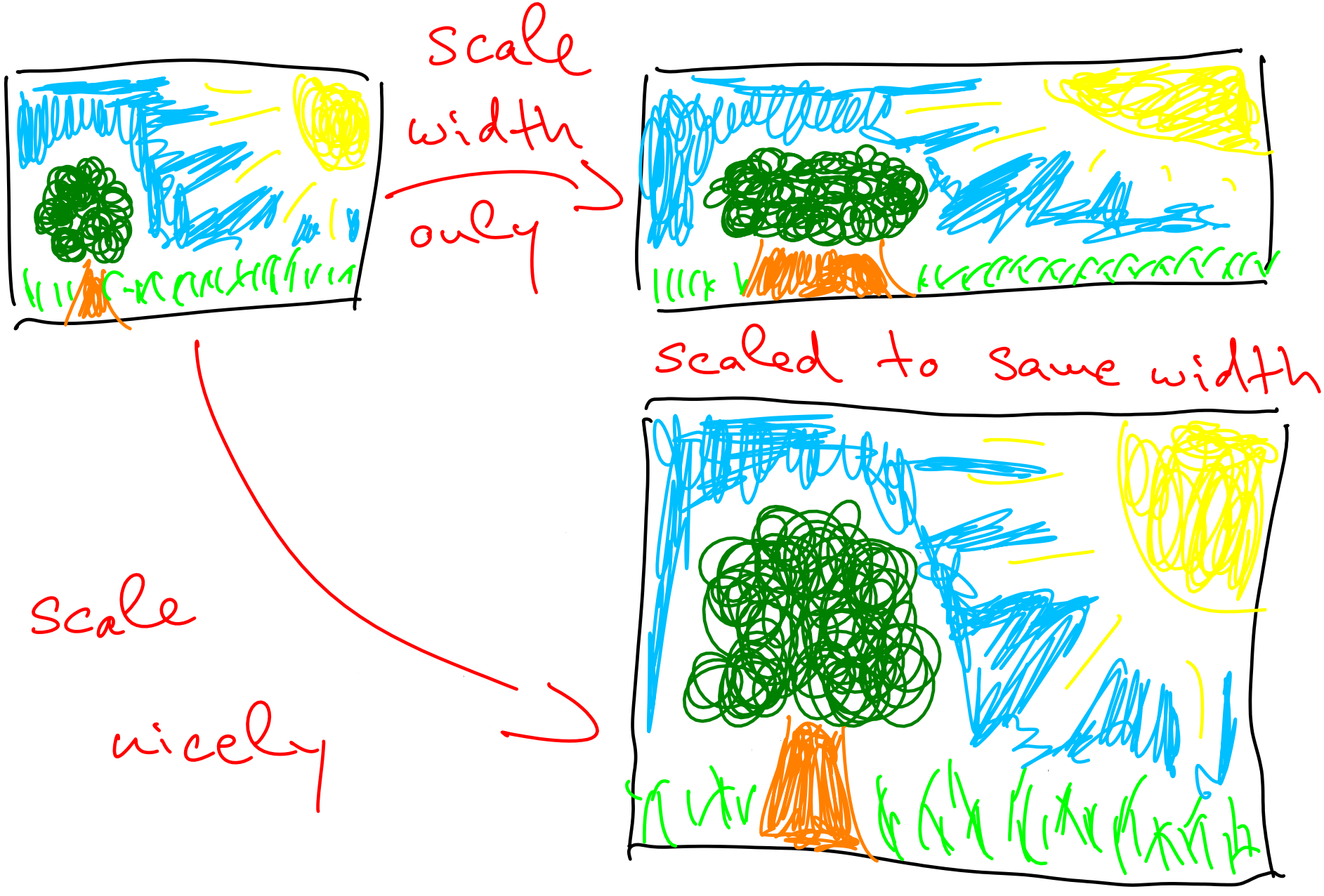

A Clear Picture
of Lens Laws
Sebastian Fischer
2015


scale
width
only
scaled to same width
scale
nicely

```
scaleWidth : Pic -> Width -> Pic

scaleNicely : Pic -> Width -> Pic

width : Pic -> Width
```

(in Haskell-like notation)

Bidirectional Correspondence

```
width (scaleWidth p w) = w
scaleWidth p (width p) = p

width (scaleNicely p w) = w
scaleNicely p (width p) = p
```

Generally

put : Source → View → Source

get : Source → View

PutGet : get (put s v) = v

GetPut : put s (get s) = s

→ well-behaved lens

As not every combination of put and get functions is well-behaved, which part is redundant?

(put is not!)

get is redundant

Let (put, get_1) and (put, get_2) be well-behaved.
Then for all sources s:

$$
\begin{aligned}
& get_1\ s \\
= \ & get_1\ (put\ s\ (get_2\ s)) && (\text{GetPut}_2) \\
= \ & get_2\ s && (\text{PutGet}_1)
\end{aligned}
$$

So, in a well-behaved lens, get is determined by put.

But, when does a corresponding get exist for an arbitrary given function put : Source → View → Source?

(such that they form a well-behaved lens)

Looking at put alone

```
scaleNot : Pic -> Width -> Pic
scaleNot p w = p
```

Is there a corresponding get function?

Put Injectivity

scaleWidth p : Width → Pic

gives **different** results for different widths

scaleNicely p : Width → Pic

gives **different** results for different widths

scaleNot p : Width → Pic

gives **same** result for different widths

Put injectivity is implied by lens laws (proof omitted)

→ It is a necessary condition on put functions in well-behaved lenses

No corresponding get for

```
scaleNot : Pic -> width -> Pic
scaleNot p w = p
```

because Put injectivity does not hold.

Looking at put alone

```
scaleBy : Pic -> Factor -> Pic
scaleBy p x =
    scaleNicely p (x * width p)
```

Is there a corresponding get function?

Put Twice

```
scaleWidth (scaleWidth p w) w
    = scaleWidth p w

scaleVicely (scaleVicely p w) w
    = scaleVicely p w
```

but

```
scaleBy (scaleBy p 2) 2
    = scaleBy p 4
    ≠ scaleBy p 2
```

PutTwice is implied by lens laws:

put (put s v) v

= put (put s v) (get (put s v)) (PutGet)

= put s v (GetPut)

→ It is a necessary condition on put functions in well-behaved lenses.

No corresponding get for

scaleBy : Pic → Factor → Pic

because PutTwice does not hold.

(scaleBy satisfies PutInjectivity and
scaleNot satisfies PutTwice)

Looking at put alone

```
scaleBlank : Pic -> Width -> Pic
scaleBlank p w =
  scaleNicely someBlankPic w
```

Is there a corresponding get function?

Put Surjectivity

every picture is result of scaleWidth
(by scaling itself with its own width)

every picture is result of scaleNicely
(by scaling itself with its own width)

not every picture is result of scaleBlank
(because every result is blank)

PutSurjectivity is implied by lens laws (proof omitted)

→ It is a necessary condition on put functions in well-behaved lenses.

No corresponding get for

scaleBlank : Pic → Width → Pic

because PutSurjectivity does not hold.

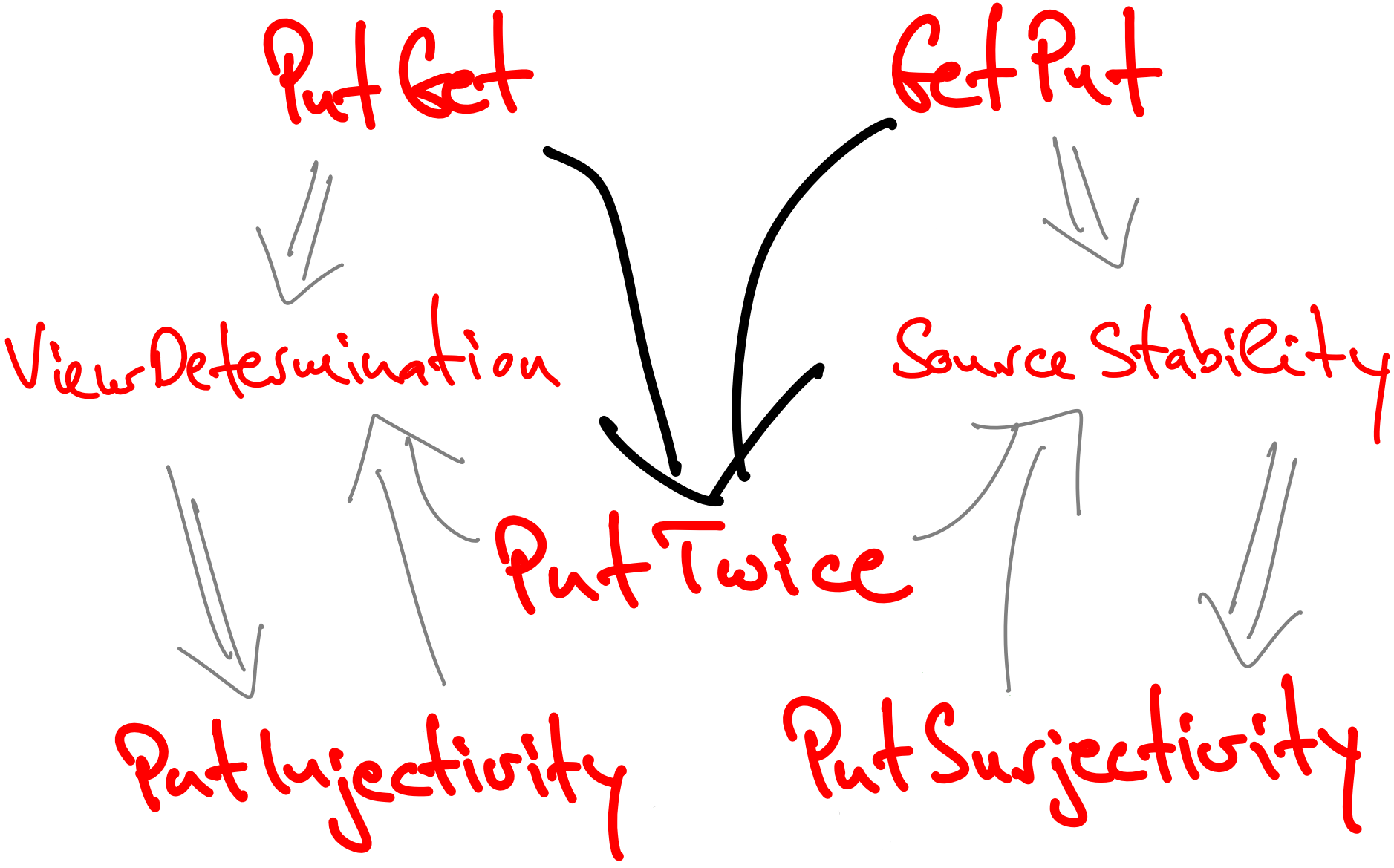
(scaleNot and scaleBy satisfy PutSurjectivity,
scaleBlank satisfies PutInjectivity and PutTwice)

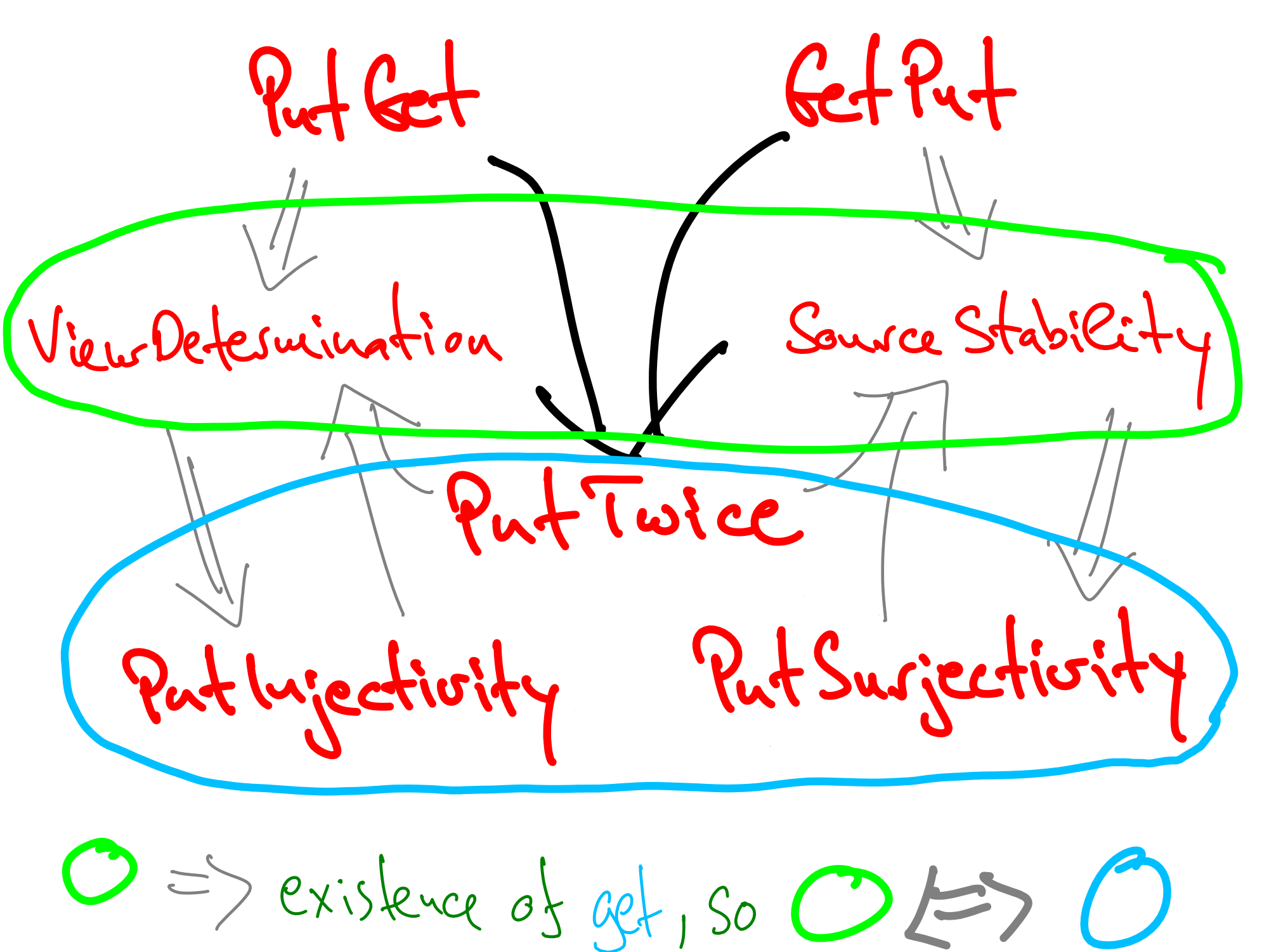
PutInjectivity, PutTwice, PutSurjectivity

are **necessary** conditions on **put**.

They are also **sufficient** for existence of corresponding **get** that forms a well-behaved lens with **put**. (proof omitted)

(remember: get is unique)


PutGet
GetPut
ViewDetermination
SourceStability
PutTwice
PutInjectivity
PutSurjectivity


PutGet
GetPut
ViewDetermination
SourceStability
PutTwice
PutInjectivity
PutSurjectivity

=> existence of get, so