HoTT

Amr Sabry

Math REU

June - July 2017

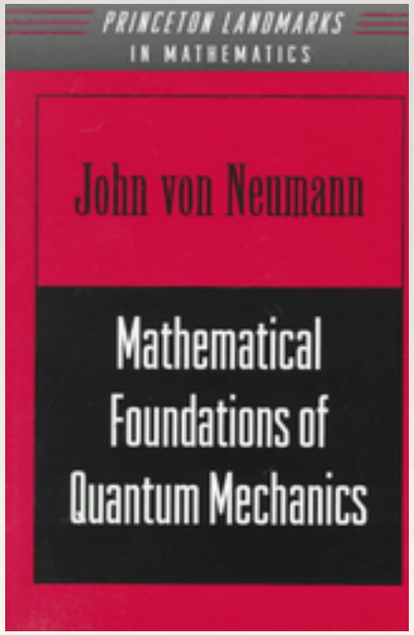
Physics and Computation

Physics

- Most accurate theory known to us is *Quantum Mechanics*

Physics

- Most accurate theory known to us is *Quantum Mechanics*

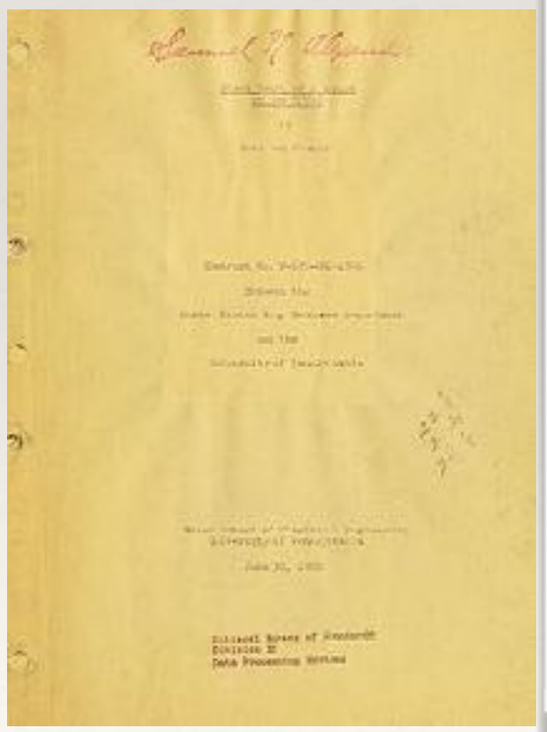


Computation

- Universal model of computation is based on the Von Neumann architecture
- Equivalent to Turing machines, the λ-calculus, and many other models

Computation

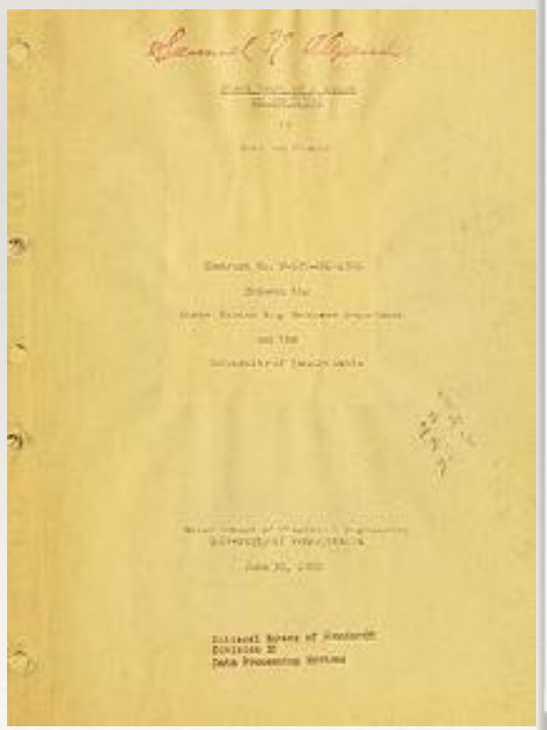
- Universal model of computation is based on the Von Neumann architecture
- Equivalent to Turing machines, the λ-calculus, and many other models



Computation

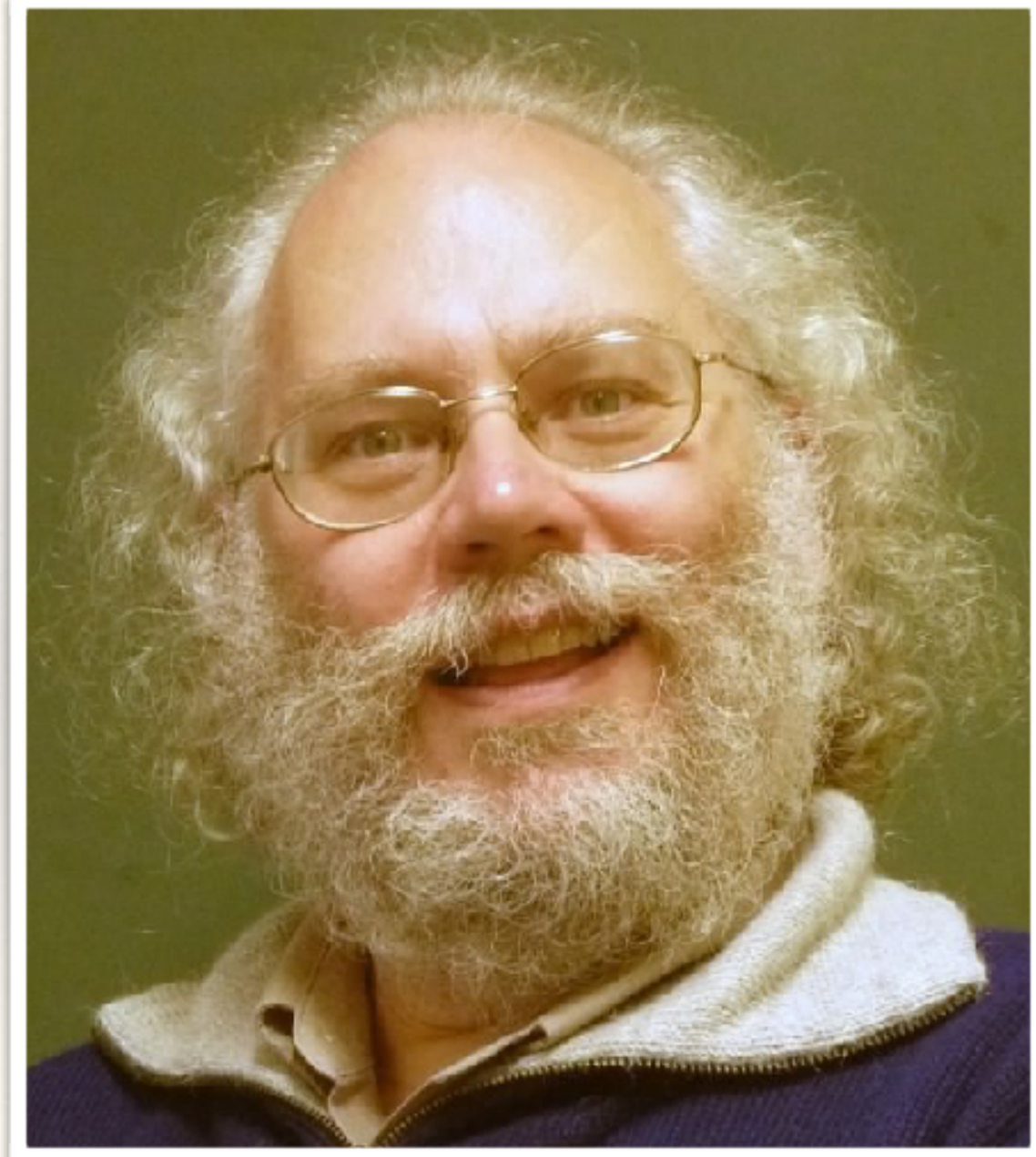
- Universal model of computation is based on the Von Neumann architecture
- Equivalent to Turing machines, the λ-calculus, and many other models

First Draft of a Report on the EDVAC
by John von Neumann,
Contract No. W-670-ORD-4926,
Between the United States Army Ordinance Department
and the University of Pennsylvania Moore School of Electrical Engineering
University of Pennsylvania
June 30, 1945



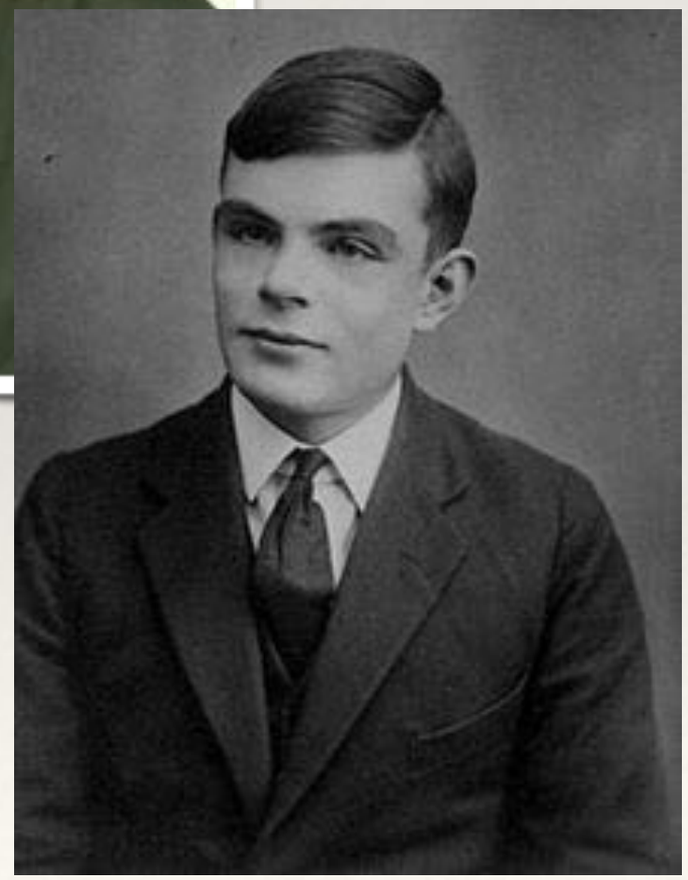
Physics Worldview

- Quantum mechanics implies the existence of an efficient (polynomial) algorithm for factoring integers (Shor)
- RSA is not secure.



Computer Science Worldview

- Extended Church-Turing Thesis: Everything that is computable in Nature is computable by a von Neumann machine —- *and is not exponentially faster.*
- Despite intense interest and effort, no one knows how to factor integers efficiently in that model.
- RSA is secure.



Something is Wrong

- Either Shor's algorithm is not "natural". (Textbook quantum mechanics is wrong);
- or, the von Neumann architecture is not universal. (There are other "natural" computing models that are exponentially faster);
- or, computer scientists have not been clever enough to find an efficient factoring algorithm;
- or, everybody is wrong

Possibility I: Revise quantum mechanics

The mathematician's vision of an unlimited sequence of totally reliable operations is unlikely to be implementable in this real universe.

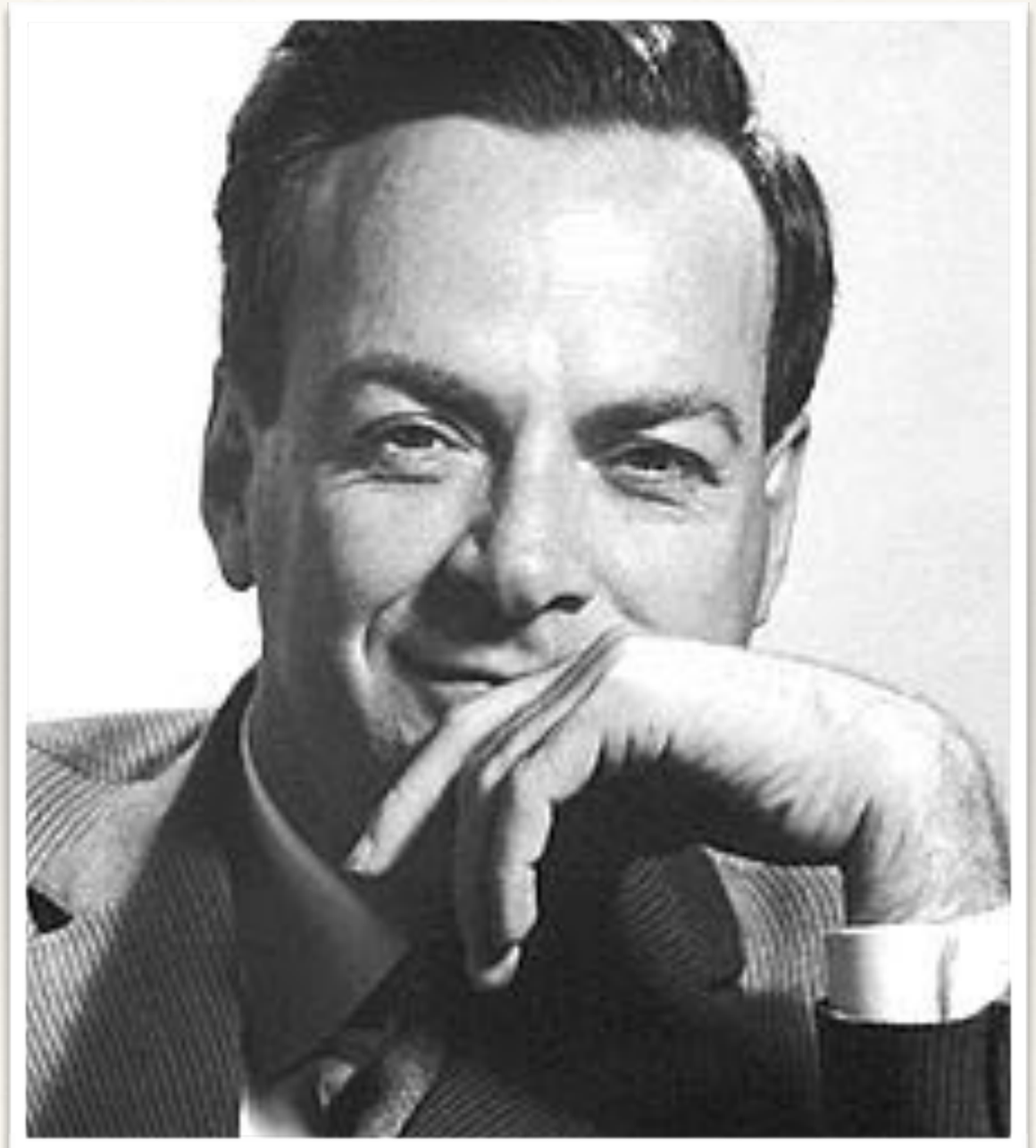
But the real world is unlikely to supply us with unlimited memory or unlimited Turing machine tapes. Therefore, continuum mathematics is not executable, and physical laws which invoke that can not really be satisfactory. They are references to illusionary procedures.

(Landauer 1996 and 1999)

Possibility I: Revise quantum mechanics

I want to talk about the possibility that there is to be an exact simulation, that the computer will do exactly the same as nature. If this is to be proved and the type of computer is as I've already explained, then it's going to be necessary that everything that happens in a finite volume of space and time would have to be exactly analyzable with a finite number of logical operations. The present theory of physics is not that way, apparently. It allows space to go down into infinitesimal distances, wavelengths to get infinitely great, terms to be summed in infinite order, and so forth; and therefore, if this proposition is right, physical law is wrong.

(Feynman 1981)



Possibility II: Revise Computer Science

Another thing that had been suggested early was that natural laws are reversible, but that computer rules are not. But this turned out to be false; the computer rules can be reversible, and it has been a very, very useful thing to notice and to discover that. This is a place where the relationship of physics and computation has turned itself the other way and told us something about the possibilities of computation. So this is an interesting subject because it tells us something about computer rules…

(Feynman 1981)

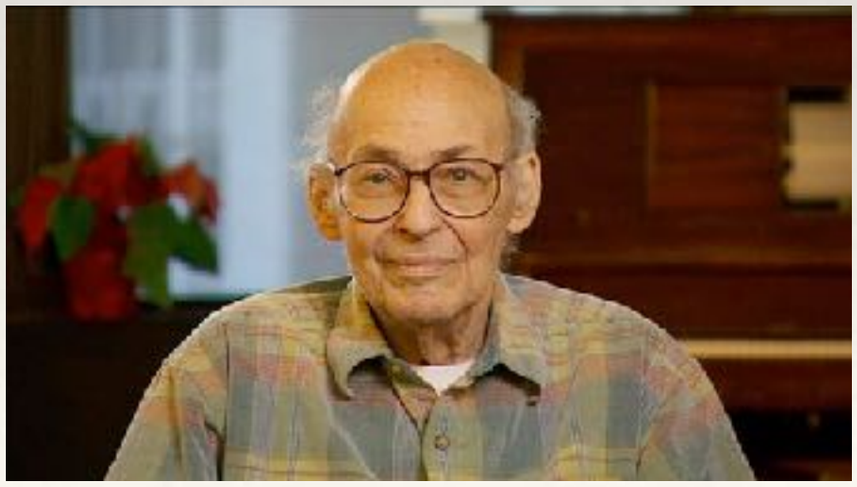
Possibility II: Revise Computer Science

Turing hoped that his abstracted-paper-tape model was so simple, so transparent and well defined, that it would not depend on any assumptions about physics that could conceivably be falsified, and therefore that it could become the basis of an abstract theory of computation that was independent of the underlying physics. 'He thought,' as Feynman once put it, 'that he understood paper.' But he was mistaken. Real, quantum-mechanical paper is wildly different from the abstract stuff that the Turing machine uses. The Turing machine is entirely classical, and does not allow for the possibility the paper might have different symbols written on it in different universes, and that those might interfere with one another.

(Deutsch 1985)

Possibility II: Revise Computer Science

Ed Fredkin pursued the idea that information must be finite in density. One day, he announced that things must be even more simple than that. He said that he was going to assume that information itself is conserved. "You're out of you mind, Ed." I pronounced. "That's completely ridiculous. Nothing could happen in such a world. There couldn't even be logical gates. No decisions could ever be made." But when Fredkin gets one of his ideas, he's quite immune to objections like that; indeed, they fuel him with energy. Soon he went on to assume that information processing must also be reversible — and invented what's now called the Fredkin gate.

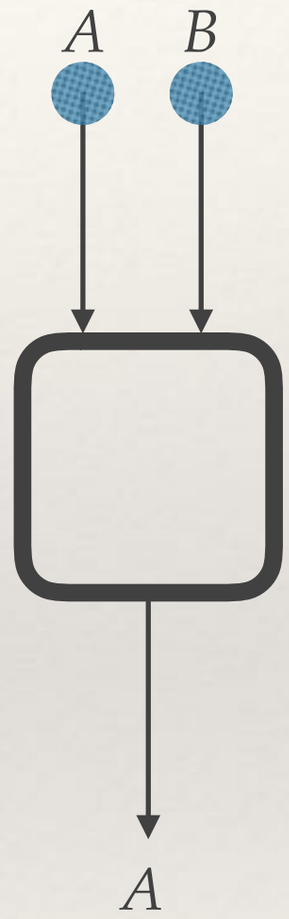


(Minsky 1999)

Conservation of Information in A Programming Language

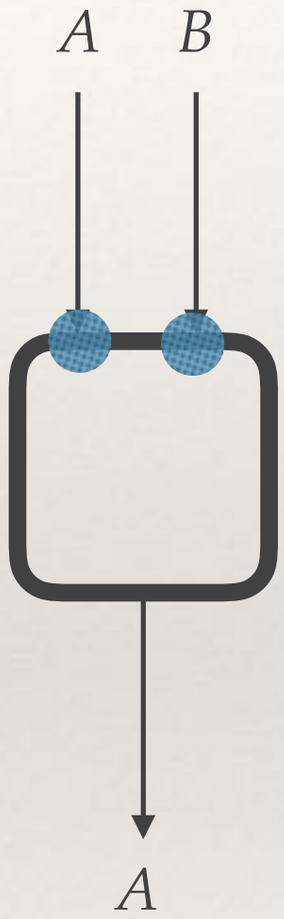
Conservation of Information

- None of the common foundational models of computation is based on conservation of information
  - Circuit model has AND, OR, etc gates that lose information;
  - Turing Machine allows one to overwrite a cell losing information;
  - λ-calculus allows functions that throw away their arguments losing information;
  - etc etc etc



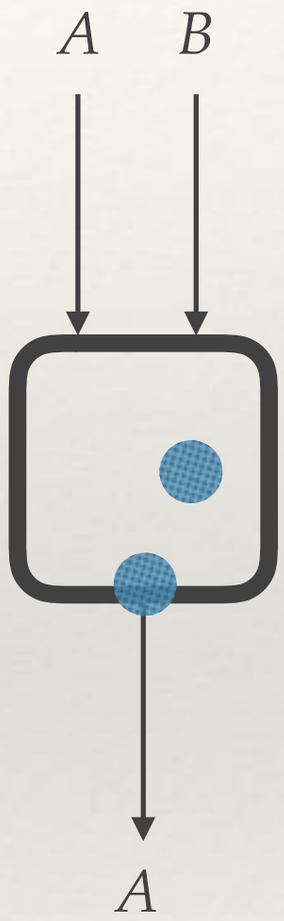
Conservation of Information

- None of the common foundational models of computation is based on conservation of information
  - Circuit model has AND, OR, etc gates that lose information;
  - Turing Machine allows one to overwrite a cell losing information;
  - λ-calculus allows functions that throw away their arguments losing information;
  - etc etc etc



Conservation of Information

- None of the common foundational models of computation is based on conservation of information
  - Circuit model has AND, OR, etc gates that lose information;
  - Turing Machine allows one to overwrite a cell losing information;
  - λ-calculus allows functions that throw away their arguments losing information;
  - etc etc etc

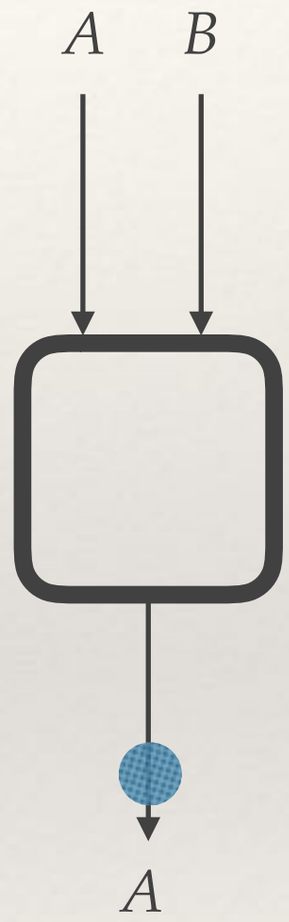


Conservation of Information

- None of the common foundational models of computation is based on conservation of information
  - Circuit model has AND, OR, etc gates that lose information;
  - Turing Machine allows one to overwrite a cell losing information;
  - λ-calculus allows functions that throw away their arguments losing information;
  - etc etc etc

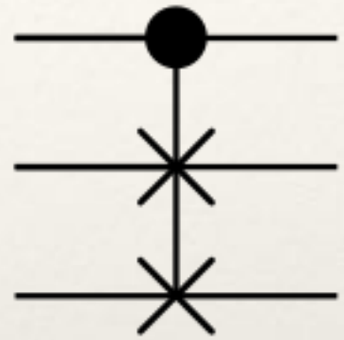
Conservation of Information

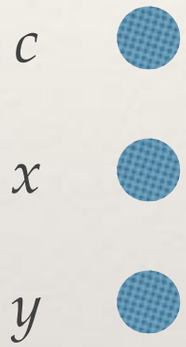
- None of the common foundational models of computation is based on conservation of information
  - Circuit model has AND, OR, etc gates that lose information;
  - Turing Machine allows one to overwrite a cell losing information;
  - λ-calculus allows functions that throw away their arguments losing information;
  - etc etc etc



Laundauer's principle:
Erasure of information generates heat!!!

Fredkin Gate





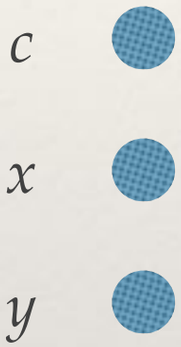
C

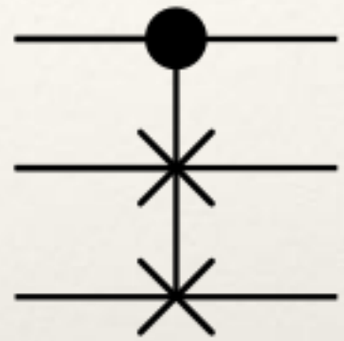
X

Y

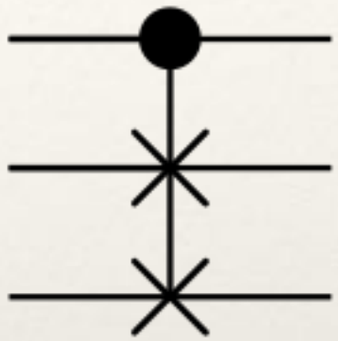
Fredkin Gate

$c = false$



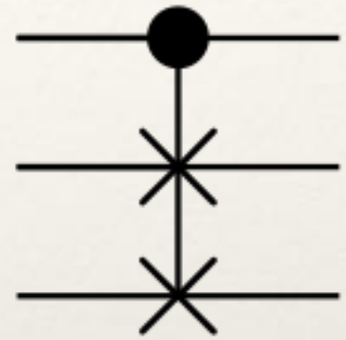


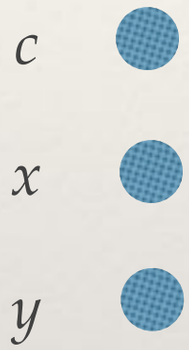
Fredkin Gate



$c = false$

Fredkin Gate



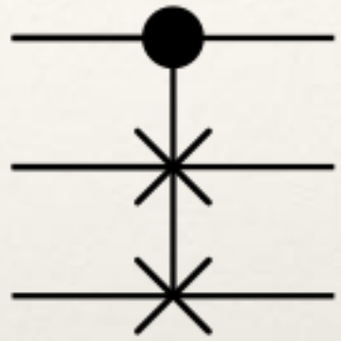


C

X

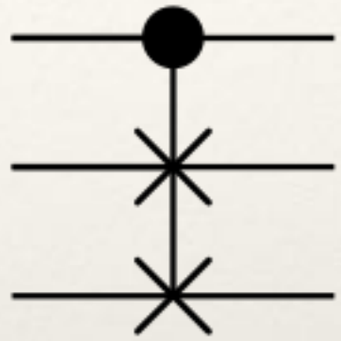
Y

Fredkin Gate



$c = true$

Fredkin Gate

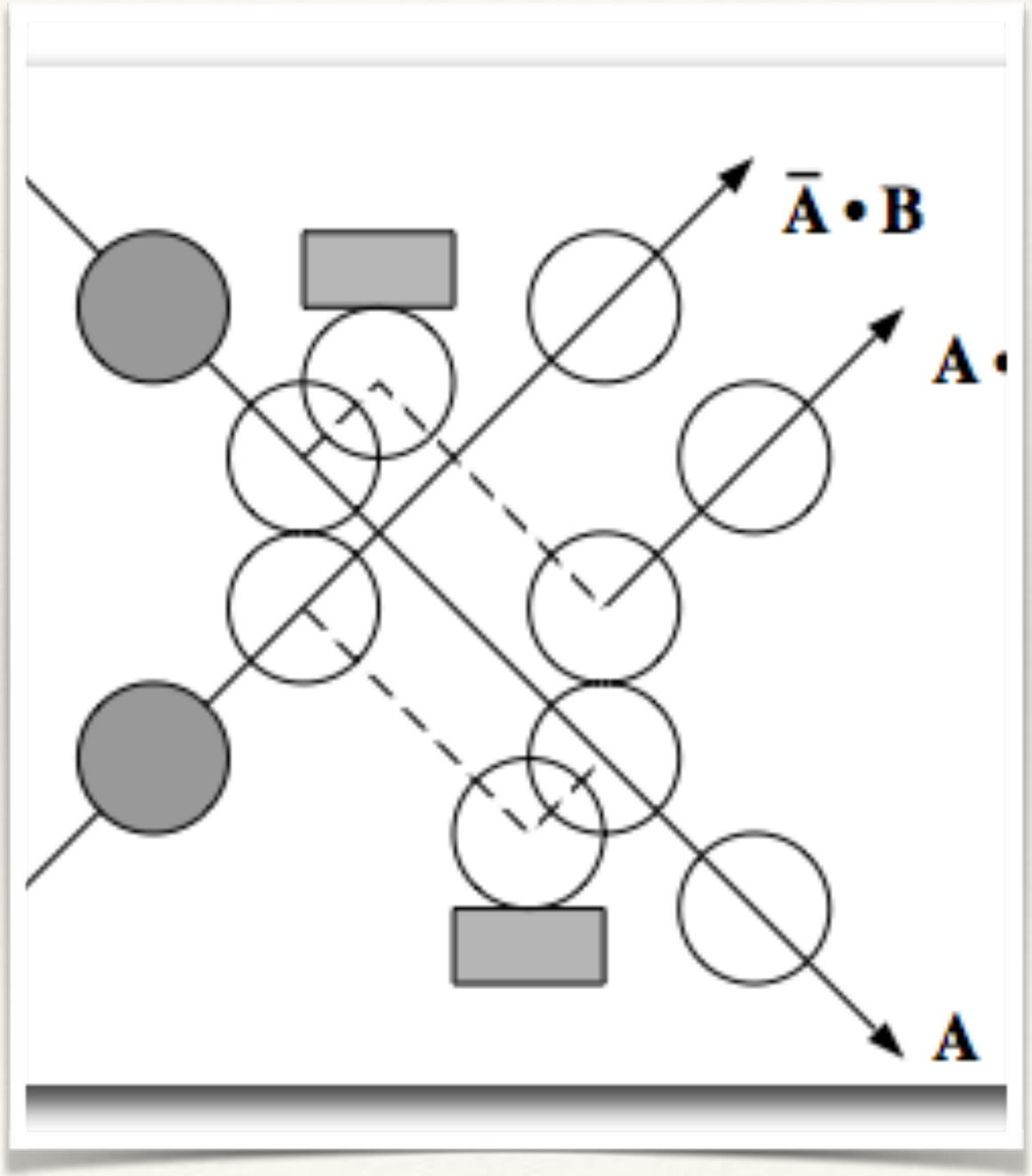


$c = true$

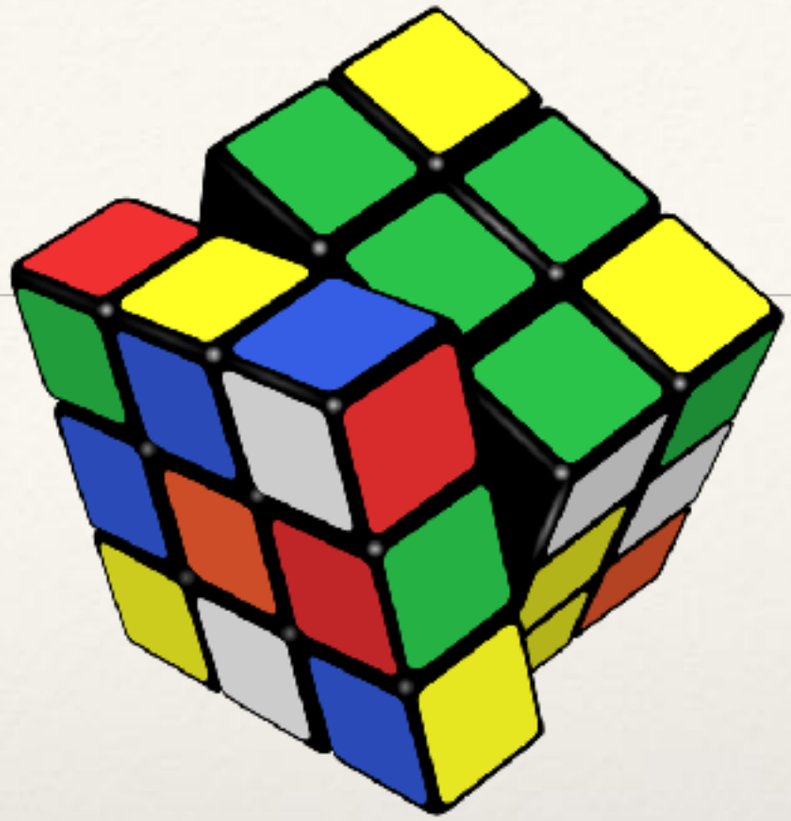
If $y = false$, then $Y = c$ AND x
If $x = false$, $y = true$, then $y =$ NOT c
If $y = true$, then $X = c$ OR x

Conservation of Information

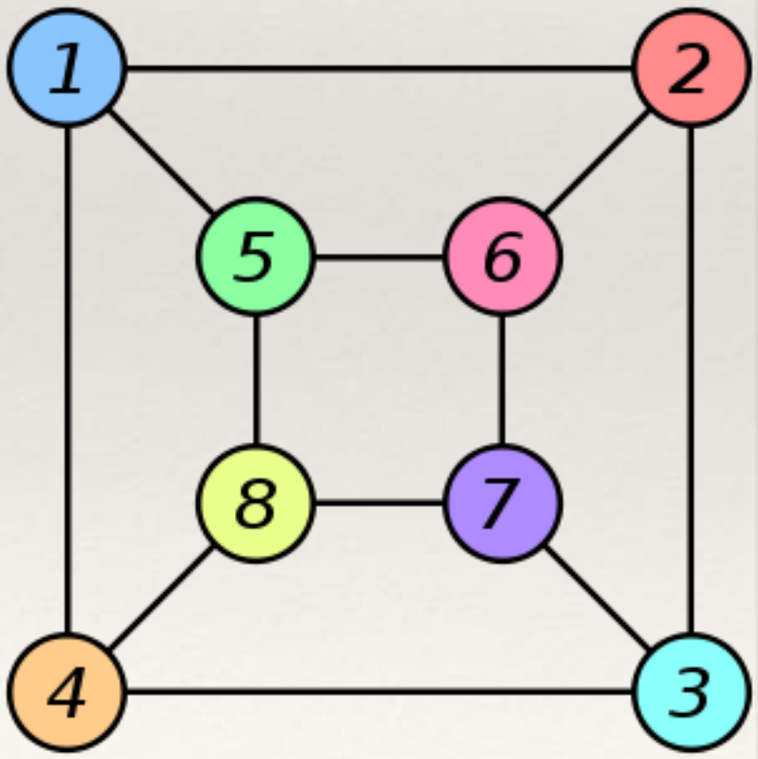
- Is it just a low-level concern, a curiosity, an esoteric model, or is it really foundational?
- We argue that, if taken seriously, it revolutionizes the theory and practice of computation, and even its logical foundations.
- This perspective has far reaching implications in many high-level applications

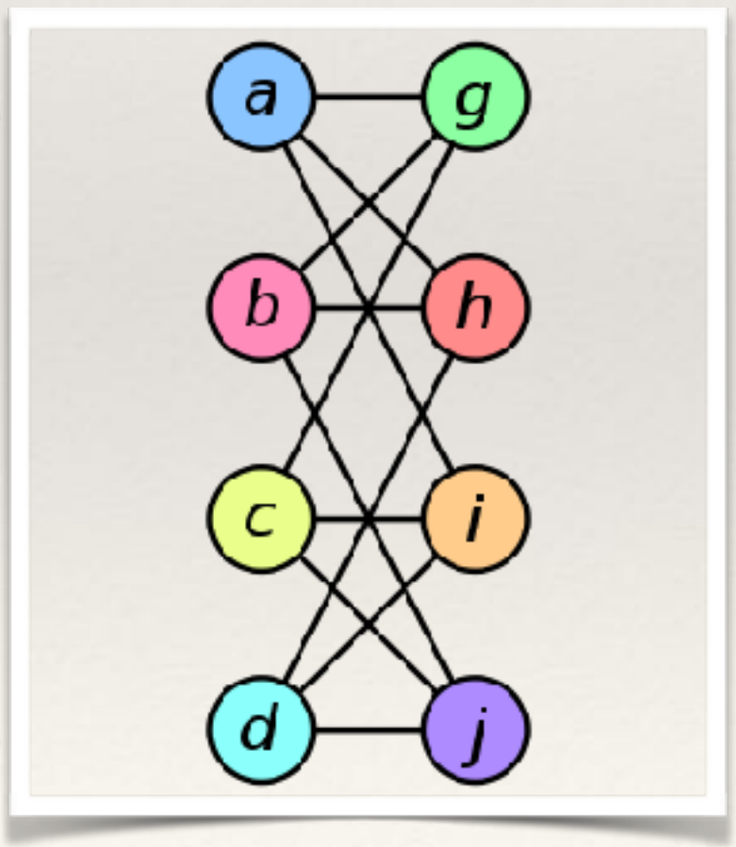


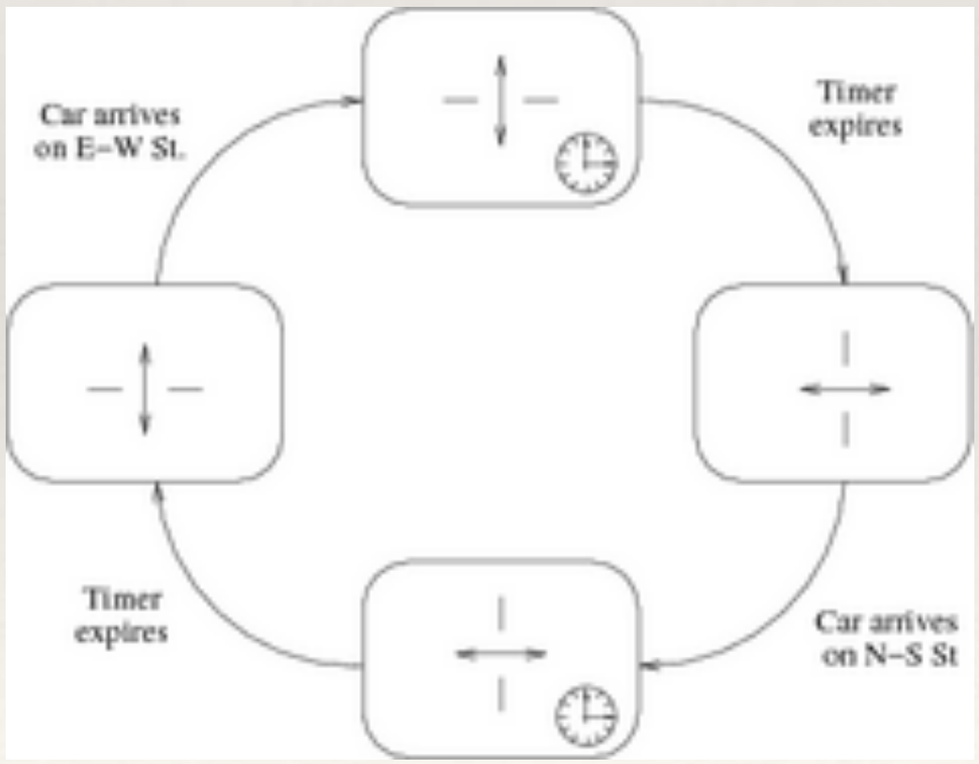
Key Idea



- All programs / proofs / deductions are isomorphisms / equivalences

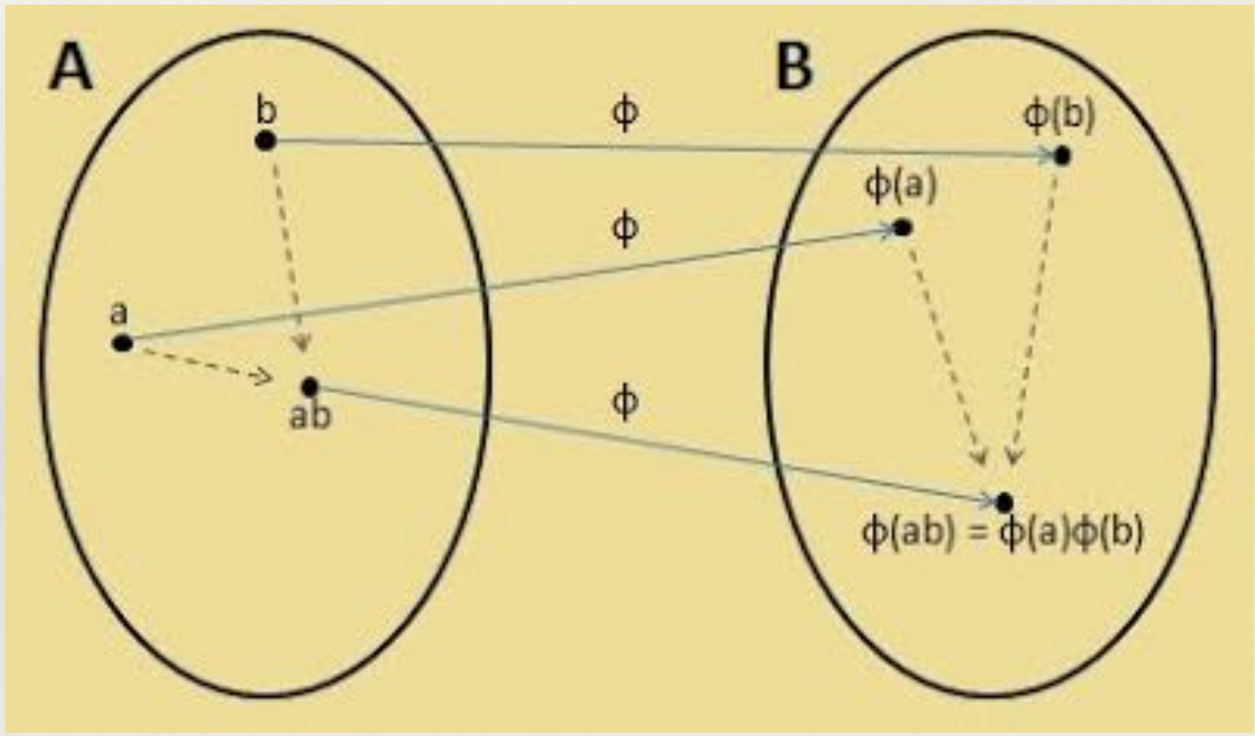






Type Isomorphisms

- Some problems are naturally reversible.
- Those that are not can be embedded in reversible problems.
- When it comes to writing programs to solve these problems, all is needed is a language for expressing equivalences.
- Technically a language that is sound and complete with respect to isomorphisms between types.



From Irreversible to Reversible

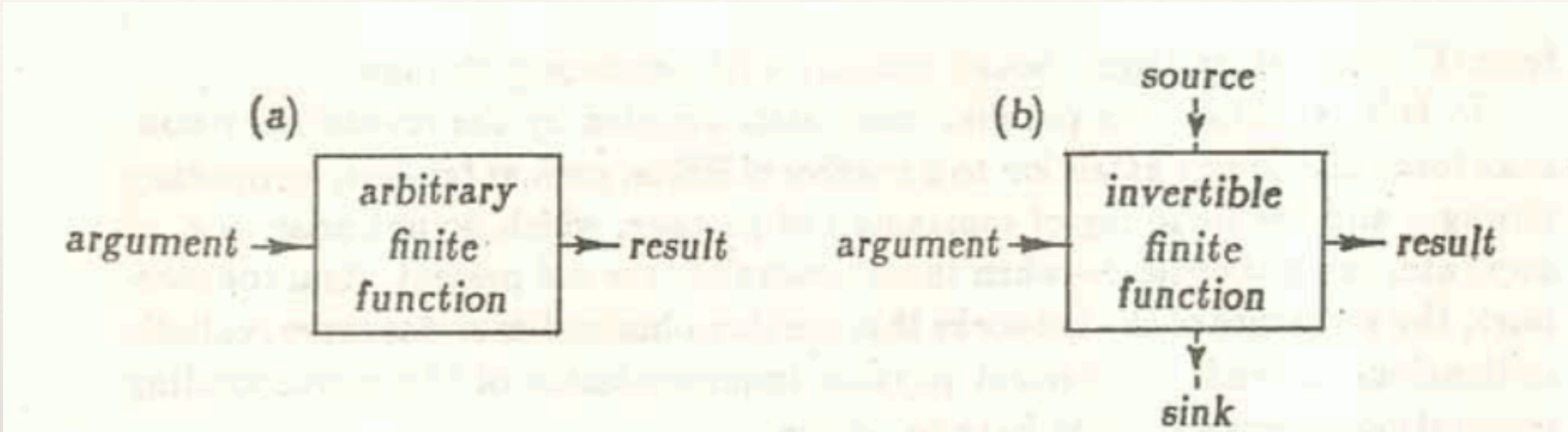


FIG. 4.2 *Any finite function (a) can be realized as an invertible finite function (b) having a number of auxiliary input lines which are fed with constants and a number of auxiliary output lines whose values are disregarded.*

(Toffoli 1980)

Sound and Complete Type Isomorphisms (for finite types)

$$
\begin{array}{rcll}
0 + b & \rightleftharpoons & b & \textit{identity for } + \\
b_1 + b_2 & \rightleftharpoons & b_2 + b_1 & \textit{commutativity for } + \\
b_1 + (b_2 + b_3) & \rightleftharpoons & (b_1 + b_2) + b_3 & \textit{associativity for } +
\end{array}
$$

$$
\begin{array}{rcll}
1 \times b & \rightleftharpoons & b & \textit{identity for } \times \\
b_1 \times b_2 & \rightleftharpoons & b_2 \times b_1 & \textit{commutativity for } \times \\
b_1 \times (b_2 \times b_3) & \rightleftharpoons & (b_1 \times b_2) \times b_3 & \textit{associativity for } \times
\end{array}
$$

$$
\begin{array}{rcll}
0 \times b & \rightleftharpoons & 0 & \textit{distribute over } 0 \\
(b_1 + b_2) \times b_3 & \rightleftharpoons & (b_1 \times b_3) + (b_2 \times b_3) & \textit{distribute over } +
\end{array}
$$

$$
\frac{}{b_1 \rightleftharpoons b_1} \quad \frac{b_1 \rightleftharpoons b_2}{b_2 \rightleftharpoons b_1} \quad \frac{b_1 \rightleftharpoons b_2 \quad b_2 \rightleftharpoons b_3}{b_1 \rightleftharpoons b_3}
$$

$$
\frac{b_1 \rightleftharpoons b_3 \quad b_2 \rightleftharpoons b_4}{(b_1 + b_2) \rightleftharpoons (b_3 + b_4)} \quad \frac{b_1 \rightleftharpoons b_3 \quad b_2 \rightleftharpoons b_4}{(b_1 \times b_2) \rightleftharpoons (b_3 \times b_4)}
$$

Name the Isomorphisms

$$
\begin{array}{rrcll}
\mathit{zeroe}: & 0 + b & \rightleftharpoons & b & : \mathit{zeroi} \\
\mathit{swap}^{+}: & b_1 + b_2 & \rightleftharpoons & b_2 + b_1 & : \mathit{swap}^{+} \\
\mathit{assocl}^{+}: & b_1 + (b_2 + b_3) & \rightleftharpoons & (b_1 + b_2) + b_3 & : \mathit{assocr}^{+} \\
 & & & & \\
\mathit{unite}: & 1 \times b & \rightleftharpoons & b & : \mathit{uniti} \\
\mathit{swap}^{\times}: & b_1 \times b_2 & \rightleftharpoons & b_2 \times b_1 & : \mathit{swap}^{\times} \\
\mathit{assocl}^{\times}: & b_1 \times (b_2 \times b_3) & \rightleftharpoons & (b_1 \times b_2) \times b_3 & : \mathit{assocr}^{\times} \\
 & & & & \\
\mathit{distrib}_0: & 0 \times b & \rightleftharpoons & 0 & : \mathit{factor}_0 \\
\mathit{distrib}: & (b_1 + b_2) \times b_3 & \rightleftharpoons & (b_1 \times b_3) + (b_2 \times b_3) & : \mathit{factor}
\end{array}
$$

$$
\frac{}{\mathit{id} : b \rightleftharpoons b} \qquad \frac{c : b_1 \rightleftharpoons b_2}{\mathit{sym}\, c : b_2 \rightleftharpoons b_1} \qquad \frac{c_1 : b_1 \rightleftharpoons b_2 \quad c_2 : b_2 \rightleftharpoons b_3}{(c_1 \fatsemi c_2) : b_1 \rightleftharpoons b_3}
$$

$$
\frac{c_1 : b_1 \rightleftharpoons b_3 \quad c_2 : b_2 \rightleftharpoons b_4}{(c_1 + c_2) : (b_1 + b_2) \rightleftharpoons (b_3 + b_4)} \qquad \frac{c_1 : b_1 \rightleftharpoons b_3 \quad c_2 : b_2 \rightleftharpoons b_4}{(c_1 \times c_2) : (b_1 \times b_2) \rightleftharpoons (b_3 \times b_4)}
$$

Example

The type isomorphism:

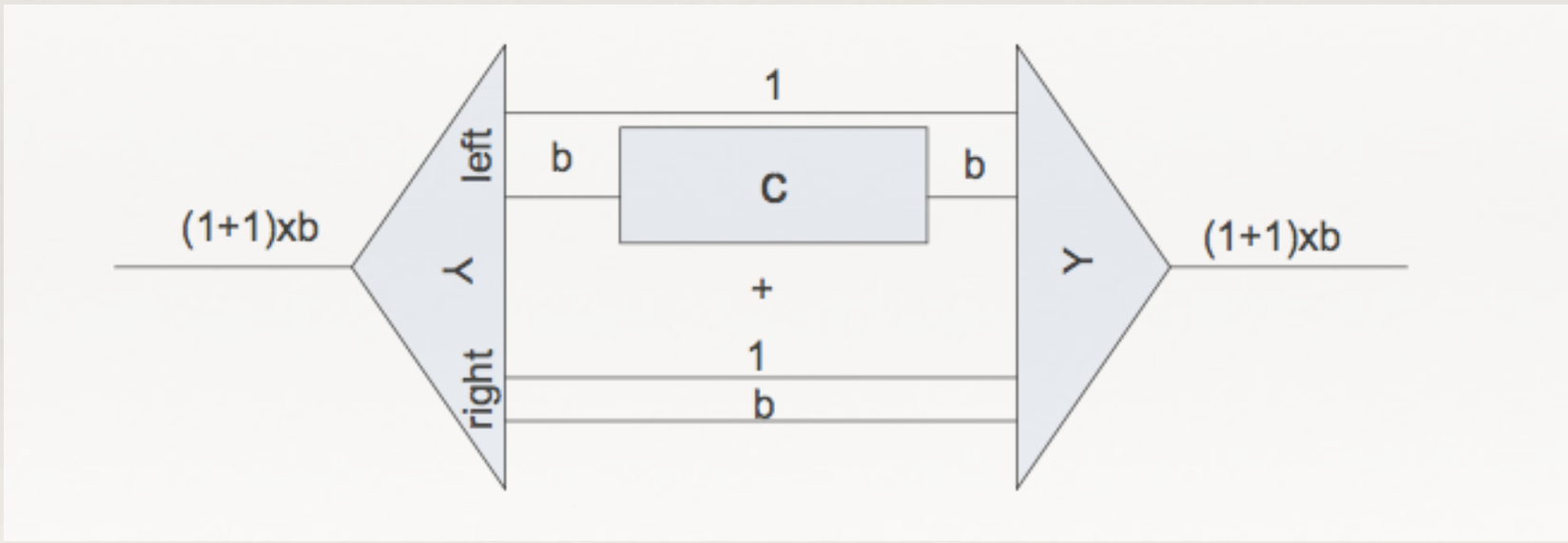
$$
\begin{aligned}
& (1+1)\times((1+1)\times b) \\
= \quad & (1+1)\times((1\times b)+(1\times b)) \\
= \quad & (1+1)\times(b+b) \\
= \quad & (1\times(b+b))+(1\times(b+b)) \\
= \quad & (b+b)+(b+b)
\end{aligned}
$$

corresponds to the program:

$$(id \times (distrib \fatsemi (unite \times unite))) \fatsemi (distrib \fatsemi (unite \times unite))$$

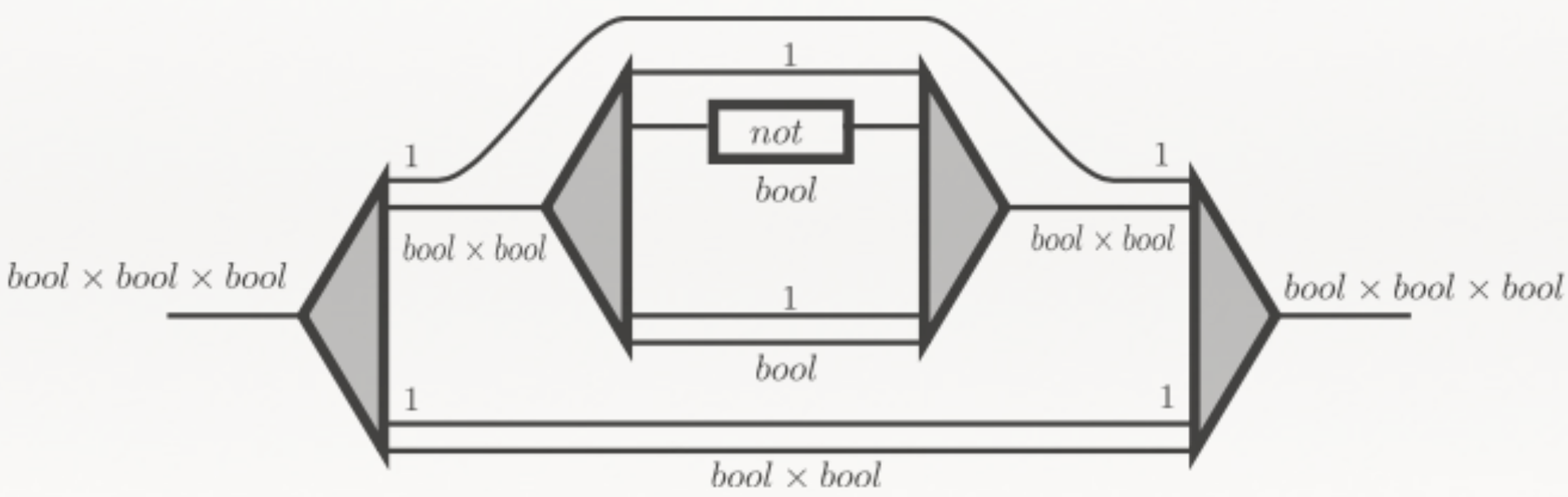
Conditionals

$$\mathbf{if}_c : bool \times b \rightleftharpoons bool \times b$$
$$\mathbf{if}_c = distrib \fatsemi ((id \times c) + id) \fatsemi factor$$



Toffoli and Fredkin Gates

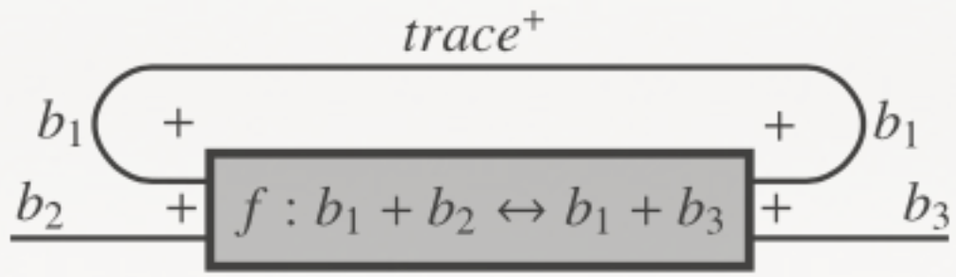
- Fredkin gate: $\mathbf{if}_{swap^{\times}}$
- Toffoli gate: $\mathbf{if}_{\mathbf{if}_{swap^{+}}}$



Recursion

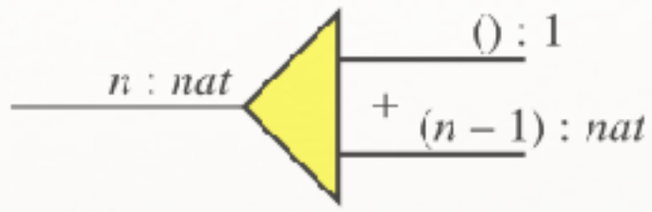
- Add recursive types (so that we can now define natural numbers, lists, trees, etc.)

- Add categorical trace (the categorical abstraction of feedback, looping, and recursion)

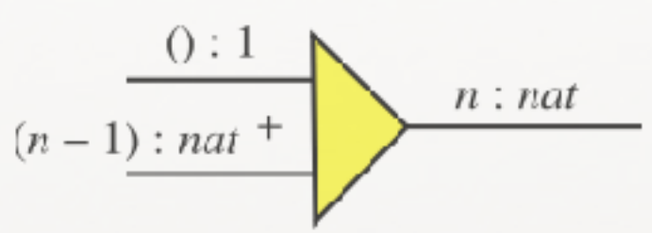


Numbers

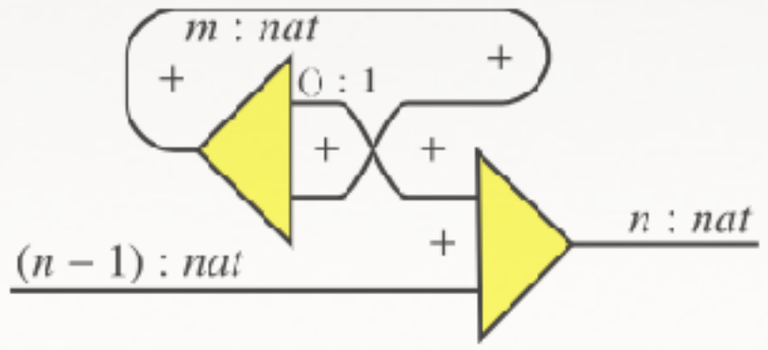
- Unfolding a natural number:



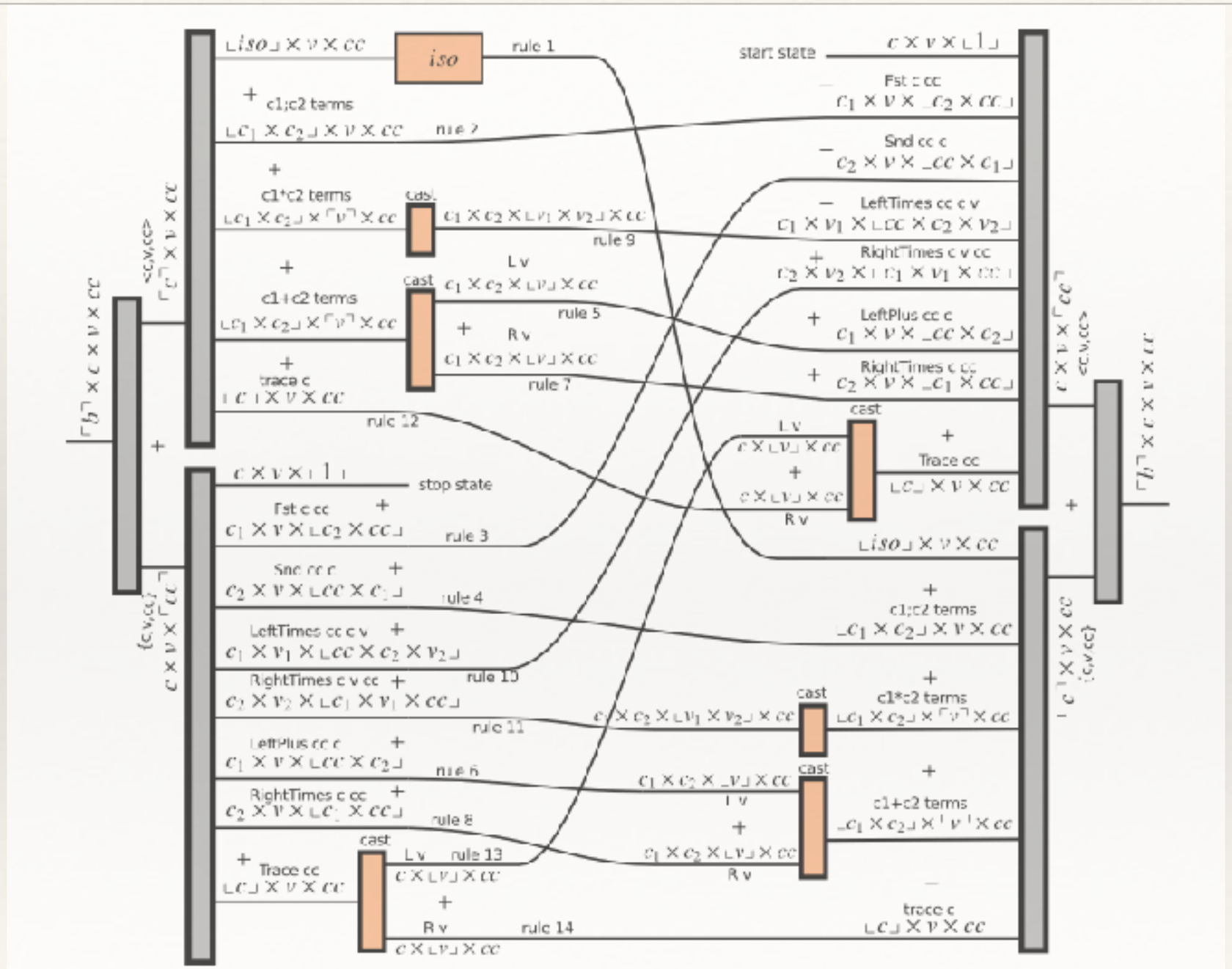
- Folding a natural number:



- add1/sub1 (partial isomorphisms)



A meta-circular interpreter



Status

- Fragment with finite types universal for reversible combinational circuits
- Curry-Howard correspondence gives a variant of linear logic: a superstructural reversible logic
- Extension with recursion Turing-complete reversible language
- Extension with “exclusive disjunctions” corresponds to quantum computing over a finite field
- Extension with isomorphisms between isomorphisms gives a reversible language for program transformations (optimizations)

Reversible Program Transformations

Π level 2

Let $c_1 : t_1 \leftrightarrow t_2$, $c_2 : t_2 \leftrightarrow t_3$, and $c_3 : t_3 \leftrightarrow t_4$:

$$c_1 \odot (c_2 \odot c_3) \Leftrightarrow (c_1 \odot c_2) \odot c_3$$

$$(c_1 \oplus (c_2 \oplus c_3)) \odot assocl_+ \Leftrightarrow assocl_+ \odot ((c_1 \oplus c_2) \oplus c_3)$$

$$(c_1 \otimes (c_2 \otimes c_3)) \odot assocl_* \Leftrightarrow assocl_* \odot ((c_1 \otimes c_2) \otimes c_3)$$

$$((c_1 \oplus c_2) \oplus c_3) \odot assocr_+ \Leftrightarrow assocr_+ \odot (c_1 \oplus (c_2 \oplus c_3))$$

$$((c_1 \otimes c_2) \otimes c_3) \odot assocr_* \Leftrightarrow assocr_* \odot (c_1 \otimes (c_2 \otimes c_3))$$

$$assocr_+ \odot assocr_+ \Leftrightarrow ((assocr_+ \oplus id) \odot assocr_+) \odot (id \oplus assocr_+)$$

$$assocr_* \odot assocr_* \Leftrightarrow ((assocr_* \otimes id) \odot assocr_*) \odot (id \otimes assocr_*)$$

$$((a \oplus b) \otimes c) \odot dist \Leftrightarrow dist \odot ((a \otimes c) \oplus (b \otimes c))$$

$$(a \otimes (b \oplus c)) \odot distl \Leftrightarrow distl \odot ((a \otimes b) \oplus (a \otimes c))$$

$$((a \otimes c) \oplus (b \otimes c)) \odot factor \Leftrightarrow factor \odot ((a \oplus b) \otimes c)$$

$$((a \otimes b) \oplus (a \otimes c)) \odot factorl \Leftrightarrow factorl \odot (a \otimes (b \oplus c))$$

Let $c, c_1, c_2, c_3 : t_1 \leftrightarrow t_2$ and $c', c'' : t_3 \leftrightarrow t_4$:

$$id \odot c \Leftrightarrow c \quad c \odot id \Leftrightarrow c \quad c \odot !c \Leftrightarrow id \quad !c \odot c \Leftrightarrow id$$

$$c \Leftrightarrow c \qquad \frac{c_1 \Leftrightarrow c_2 \quad c_2 \Leftrightarrow c_3}{c_1 \Leftrightarrow c_3} \qquad \frac{c_1 \Leftrightarrow c' \quad c_2 \Leftrightarrow c''}{c_1 \odot c_2 \Leftrightarrow c' \odot c''}$$

Let $c_0 : 0 \leftrightarrow 0$, $c_1 : 1 \leftrightarrow 1$, and $c : t_1 \leftrightarrow t_2$:

$$identl_+ \odot c \Leftrightarrow (c_0 \oplus c) \odot identl_+ \qquad identr_+ \odot (c_0 \oplus c) \Leftrightarrow c \odot identr_+$$

$$unite_+r \odot c \Leftrightarrow (c \oplus c_0) \odot unite_+r \qquad uniti_+r \odot (c \oplus c_0) \Leftrightarrow c \odot uniti_+r$$

$$identl_* \odot c \Leftrightarrow (c_1 \otimes c) \odot identl_* \qquad identr_* \odot (c_1 \otimes c) \Leftrightarrow c \odot identr_+$$

$$unite_*r \odot c \Leftrightarrow (c \otimes c_1) \odot unite_*r \qquad uniti_*r \odot (c \otimes c_1) \Leftrightarrow c \odot uniti_*r$$

$$identl_* \Leftrightarrow distl \odot (identl_* \oplus identl_*)$$

$$identl_+ \Leftrightarrow swap_+ \odot uniti_+r \qquad identl_* \Leftrightarrow swap_* \odot uniti_*r$$

$$(id \otimes swap_+) \odot distl \Leftrightarrow distl \odot swap_+$$

$$dist \odot (swap_* \oplus swap_*) \Leftrightarrow swap_* \odot distl$$

Let $c_1 : t_1 \leftrightarrow t_2$ and $c_2 : t_3 \leftrightarrow t_4$:

$$swap_+ \odot (c_1 \oplus c_2) \Leftrightarrow (c_2 \oplus c_1) \odot swap_+ \quad swap_* \odot (c_1 \otimes c_2) \Leftrightarrow (c_2 \otimes c_1) \odot swap_*$$

$$(assocr_+ \odot swap_+) \odot assocr_+ \Leftrightarrow ((swap_+ \oplus id) \odot assocr_+) \odot (id \oplus swap_+)$$

$$(assocl_+ \odot swap_+) \odot assocl_+ \Leftrightarrow ((id \oplus swap_+) \odot assocl_+) \odot (swap_+ \oplus id)$$

$$(assocr_* \odot swap_*) \odot assocr_* \Leftrightarrow ((swap_* \otimes id) \odot assocr_*) \odot (id \otimes swap_*)$$

$$(assocl_* \odot swap_*) \odot assocl_* \Leftrightarrow ((id \otimes swap_*) \odot assocl_*) \odot (swap_* \otimes id)$$

Let $c_1 : t_1 \leftrightarrow t_2$, $c_2 : t_3 \leftrightarrow t_4$, $c_3 : t_1 \leftrightarrow t_2$, and $c_4 : t_3 \leftrightarrow t_4$.
Let $a_1 : t_5 \leftrightarrow t_1$, $a_2 : t_6 \leftrightarrow t_2$, $a_3 : t_1 \leftrightarrow t_3$, and $a_4 : t_2 \leftrightarrow t_4$.

$$\frac{c_1 \Leftrightarrow c_3 \quad c_2 \Leftrightarrow c_4}{c_1 \oplus c_2 \Leftrightarrow c_3 \oplus c_4} \qquad \frac{c_1 \Leftrightarrow c_3 \quad c_2 \Leftrightarrow c_4}{c_1 \otimes c_2 \Leftrightarrow c_3 \otimes c_4}$$

$$id \oplus id \Leftrightarrow id \qquad id \otimes id \Leftrightarrow id$$

$$(a_1 \odot a_3) \oplus (a_2 \odot a_4) \Leftrightarrow (a_1 \oplus a_2) \odot (a_3 \oplus a_4)$$

$$(a_1 \odot a_3) \otimes (a_2 \odot a_4) \Leftrightarrow (a_1 \otimes a_2) \odot (a_3 \otimes a_4)$$

$$unite_+r \oplus id \Leftrightarrow assocr_+ \odot (id \oplus identl_+)$$

$$unite_*r \otimes id \Leftrightarrow assocr_* \odot (id \otimes identl_*)$$

Let $c : t_1 \leftrightarrow t_2$:

$$(c \otimes id) \odot absorbl \Leftrightarrow absorbl \odot id \quad (id \otimes c) \odot absorbr \Leftrightarrow absorbr \odot id$$

$$id \odot factorl_0 \Leftrightarrow factorl_0 \odot (id \otimes c) \quad id \odot factorzr \Leftrightarrow factorzr \odot (c \otimes id)$$

$$absorbr \Leftrightarrow absorbl$$

$$absorbr \Leftrightarrow (distl \odot (absorbr \oplus absorbr)) \odot identl_+$$

$$unite_*r \Leftrightarrow absorbr \qquad absorbl \Leftrightarrow swap_* \odot absorbr$$

$$absorbr \Leftrightarrow (assocl_* \odot (absorbr \otimes id)) \odot absorbr$$

$$(id \otimes absorbr) \odot absorbl \Leftrightarrow (assocl_* \odot (absorbl \otimes id)) \odot absorbr$$

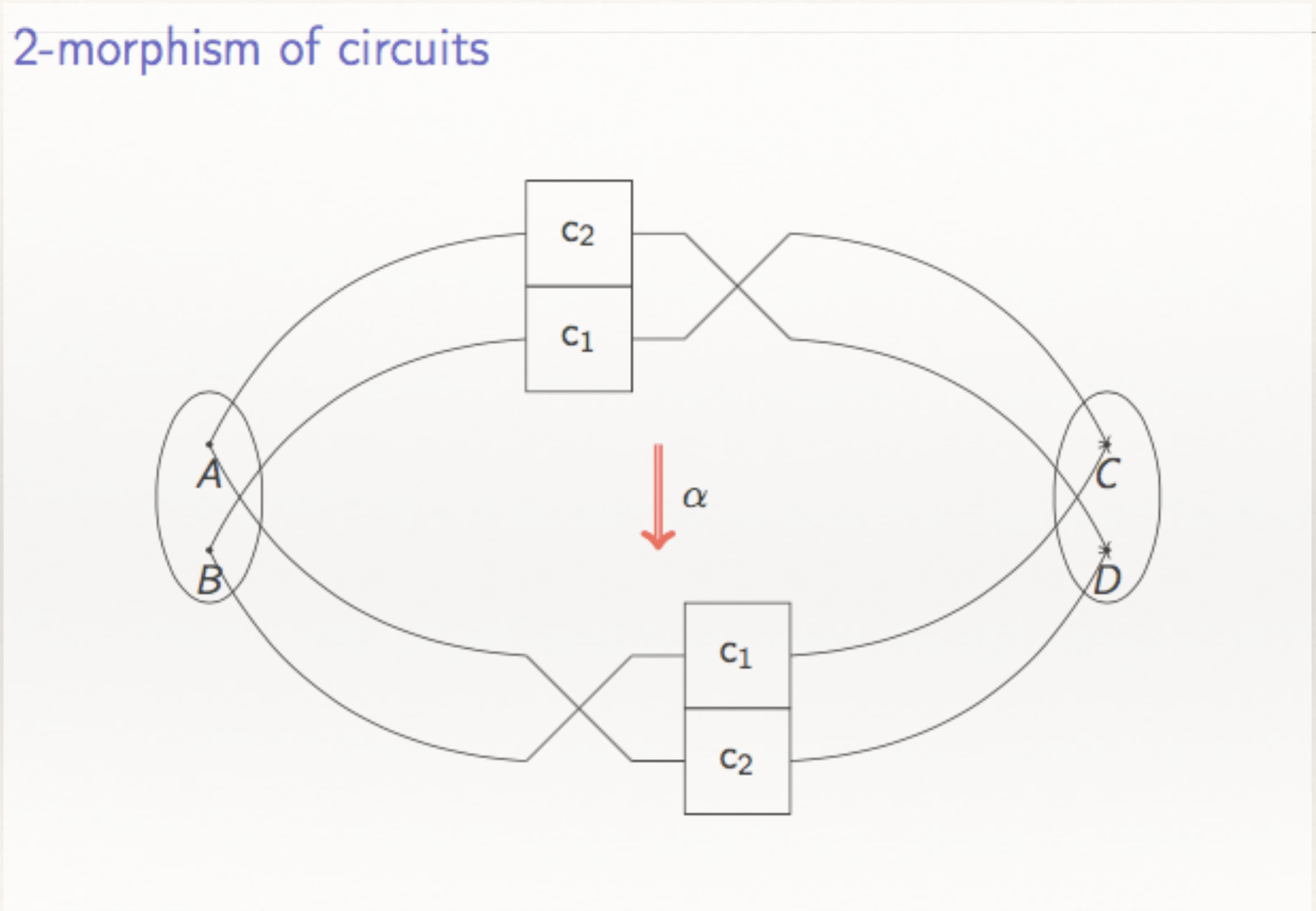
$$id \otimes identl_+ \Leftrightarrow (distl \odot (absorbl \oplus id)) \odot identl_+$$

$$((assocl_+ \otimes id) \odot dist) \odot (dist \oplus id) \Leftrightarrow (dist \odot (id \oplus dist)) \odot assocl_+$$

$$assocl_* \odot distl \Leftrightarrow ((id \otimes distl) \odot distl) \odot (assocl_* \oplus assocl_*)$$

$$(distl \odot (dist \oplus dist)) \odot assocl_+ \Leftrightarrow dist \odot (distl \oplus distl) \odot assocl_+ \odot (assocr_+ \oplus id) \odot ((id \oplus swap_+) \oplus id) \odot (assocl_+ \oplus id)$$

Example of isomorphism between isomorphisms



Results

- A reversible programming model
- with its reversible logic
- and its reversible optimizer

Homotopy Type Theory

Is Math Immune to this Physics Perspective?

200 years ago

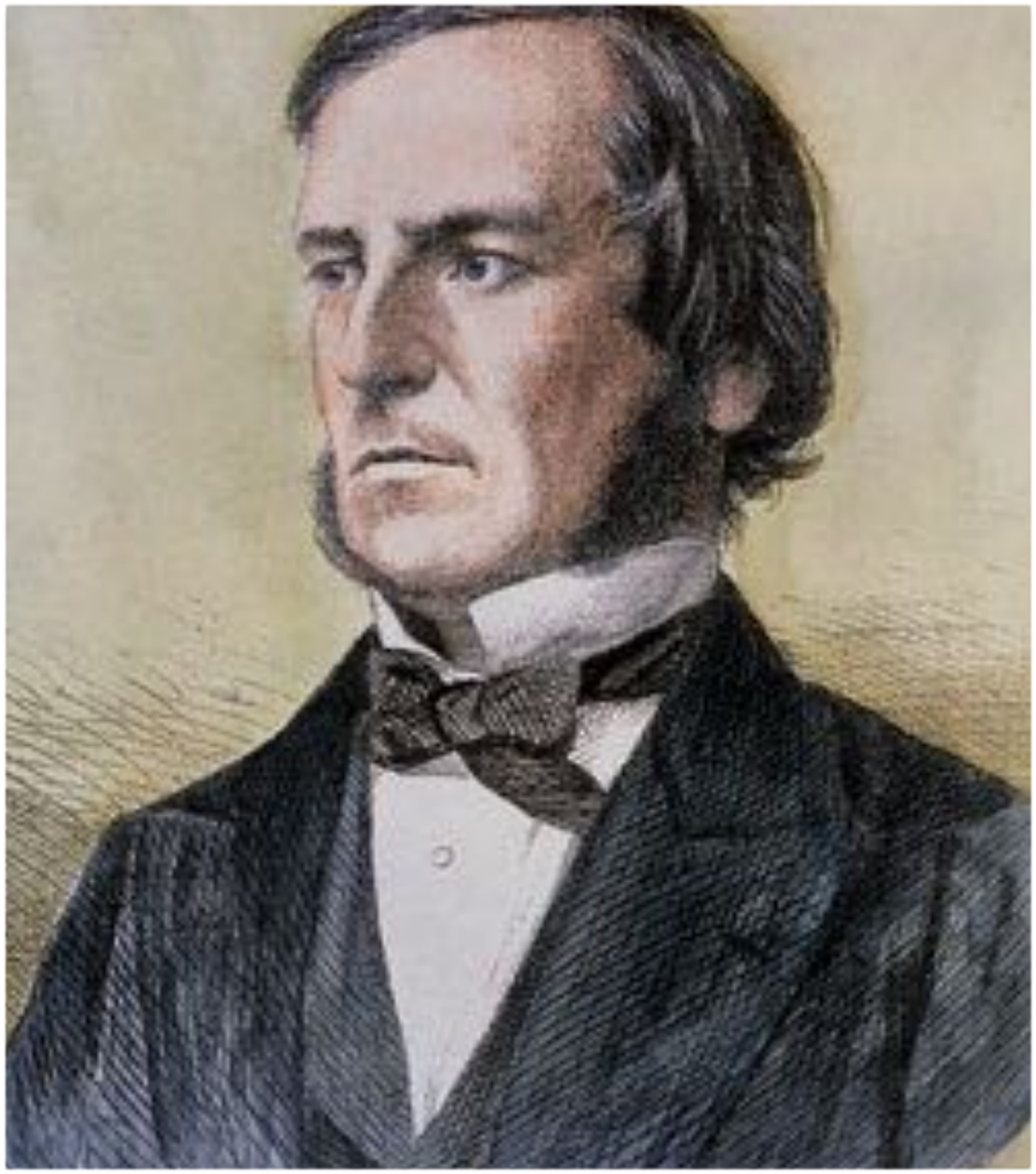
An Investigation of the Laws of Thought, on which are Founded the Mathematical Theories of Logic and Probabilities, G. Boole, 1853.

Opening sentence of Chapter 1:

The design of the following treatise is to investigate the fundamental laws of those operations of the mind by which reasoning is performed; . . .

A few chapters later:

Proposition IV. That axiom of metaphysicians which is termed the principle of contradiction, and which affirms that it is impossible for any being to possess a quality, and at the same time not to possess it, is a consequence of the fundamental law of thought, whose expression is $x^2 = x$.



200 years ago

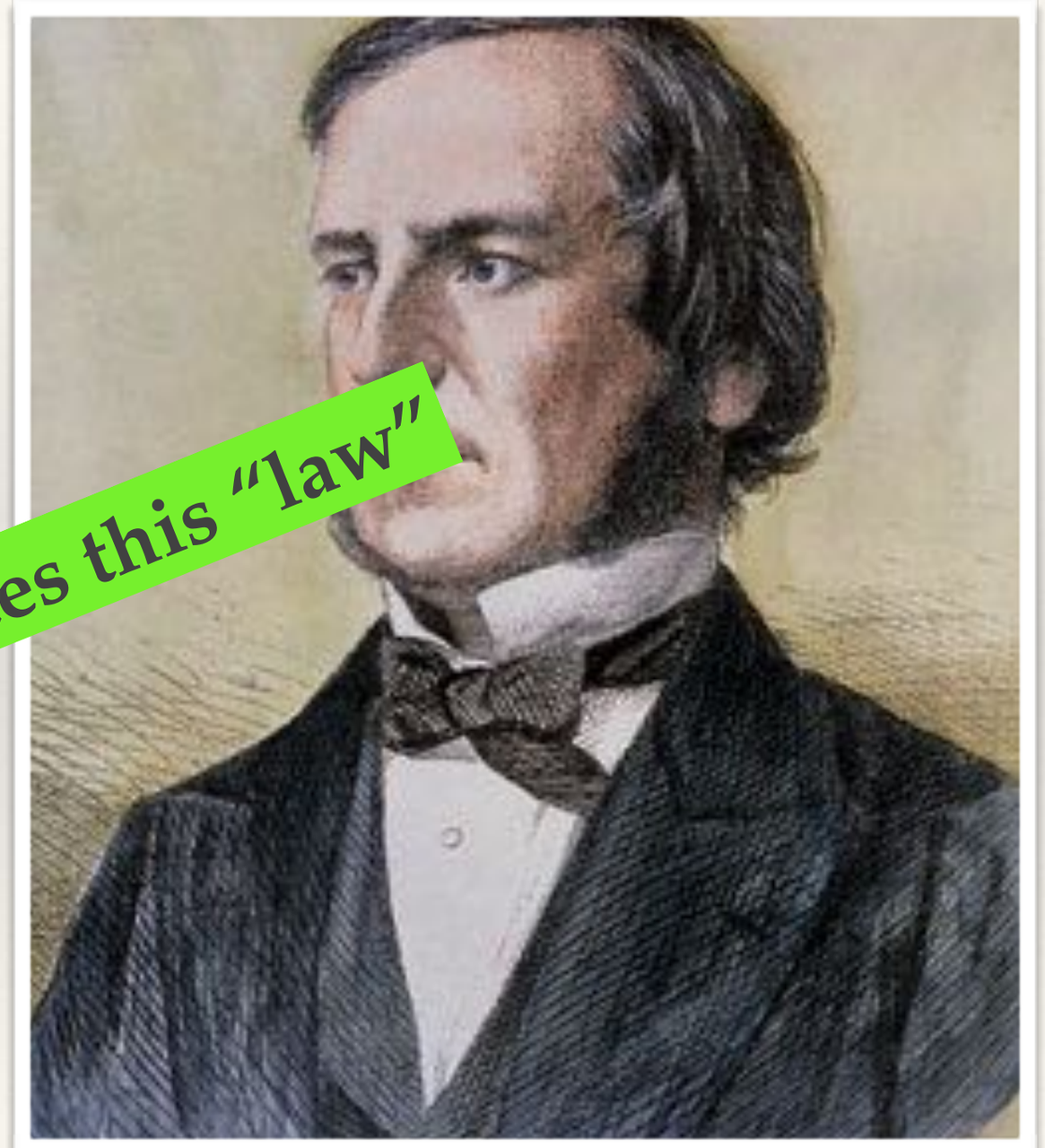
An Investigation of the Laws of Thought, on which are Founded the Mathematical Theories of Logic and Probabilities, G. Boole, 1853.

Opening sentence of Chapter 1:

The design of the following treatise is to investigate the fundamental laws of those operations of the mind by which reasoning is performed; . . .

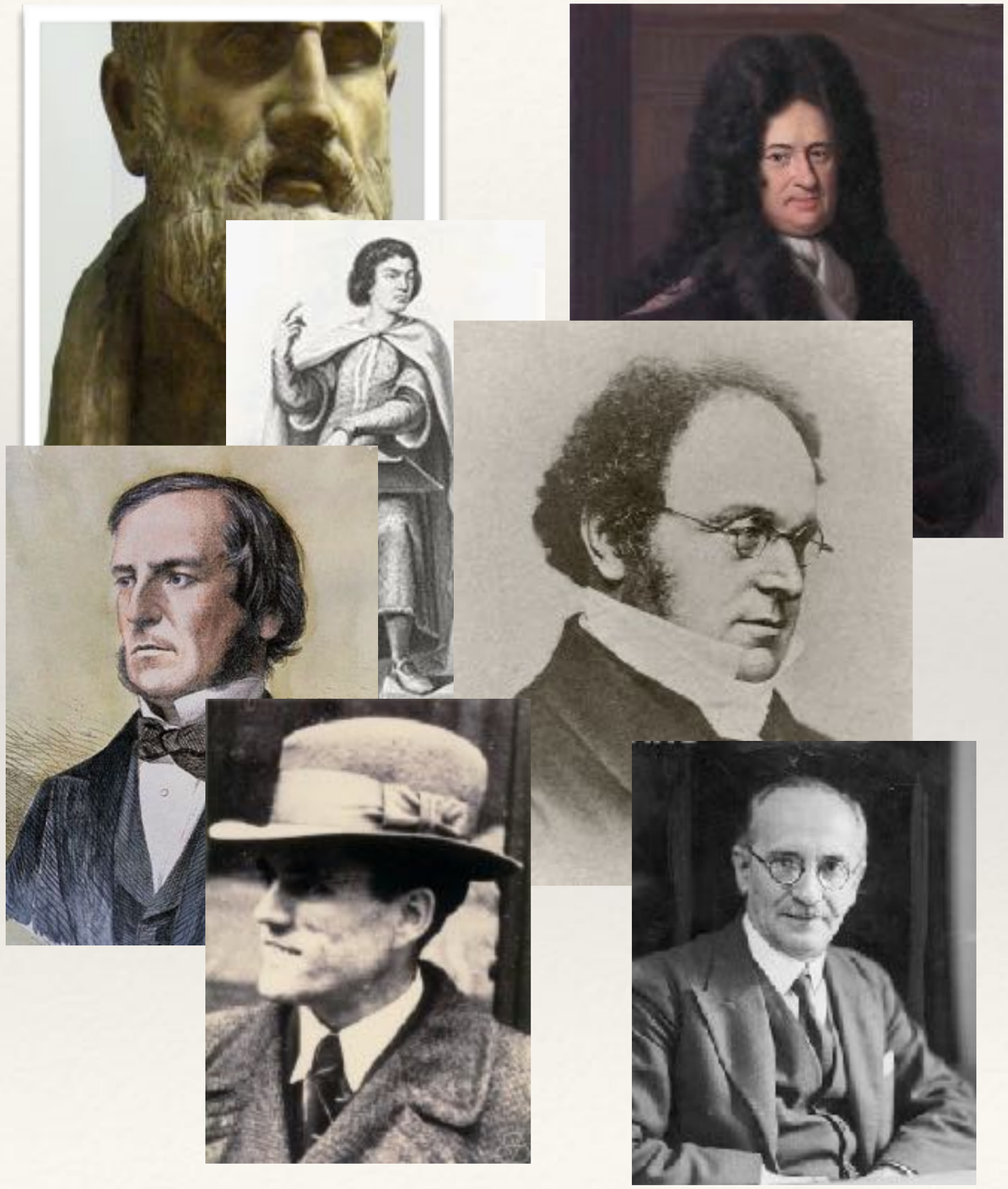
A few chapters later:

Proposition IV. That axiom of [illegible]ians which is termed the pri[illegible]ontradiction, and which affir[illegible] impossible for any being to [illegible]quality, and at the same ti[illegible]possess it, is a consequence of the fu[illegible]amental law of thought, whose expression is $x^2 = x$.



2300 years ago

- Stoics invent the propositional calculus
- Re-invented in the 12th century
- Re-re-invented by Leibniz in the 17-18th century
- Re-re-re-invented by Boole and de-Morgan in the 19th century
- Improved by Gentzen and Łukasiewicz in the 20th century



Modus Ponens

$$\frac{P \longrightarrow Q \qquad P}{Q}$$

X is a man implies X is mortal
Socrates is a man
Hence Socrates is moral.

Obvious logical inference rule. Isn't it?

Does it have anything to do with physics?

The Curry-Howard Correspondence

The proposition $P \longrightarrow Q$ is isomorphic to a function that takes a value of type P as an argument and returns a value of type Q as a result.

$$\frac{P \longrightarrow Q \qquad P}{Q}$$

Modus Ponens is *function application*. In programming terms *this is a computational physical process!*

Logic can be revisited

In other terms, what is so good in logic that quantum physics should obey? Can't we imagine that our conceptions about logic are wrong, so wrong that they are unable to cope with the quantum miracle? […] Instead of teaching logic to nature, it is more reasonable to learn from her. Instead of interpreting quantum into logic, we shall interpret logic into quantum.

(Girard 2007)

Linear Logic

When a function $P \longrightarrow Q$ is applied to a value to type P, it consumes the value of type P.

The value of type P cannot be used again; it is gone!

SodaMachine : *Money* $\longrightarrow$ *Candy*

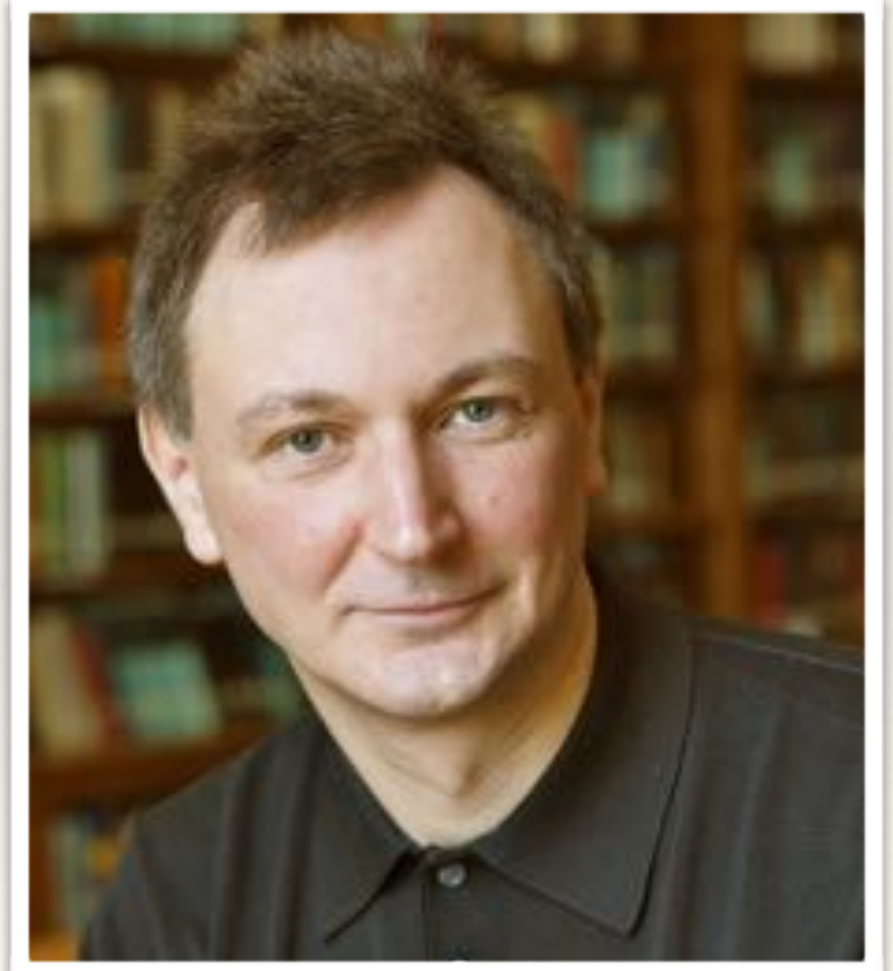
After we say *SodaMachine($1.50)* our money has been used; it is gone.

Is a Proof Correct?

The Origins and Motivations of Univalent Foundations

Professor Voevodsky's Personal Mission to Develop Computer Proof Verification to Avoid Mathematical Mistakes

The world of mathematics is becoming very large, the complexity of mathematics is becoming very high, and there is a danger of an accumulation of mistakes



Is a Proof Correct?

ulative and lacking firm foundation. The groundbreaking 1986 paper "Algebraic Cycles and Higher K-theory" by Spencer Bloch was soon after publication found by Andrei Suslin to contain a mistake in the proof of Lemma 1.1. The proof could not be fixed, and almost all of the claims of the paper were left unsubstantiated.

A new proof, which replaced one paragraph from the original paper by thirty pages of complex arguments, was not made public until 1993, and it took many more years for it to be accepted as correct. Interestingly, this new proof was based on an older result of Mark Spivakovsky, who, at about the same time, announced a proof of the resolution of singularities conjecture. Spivakovsky's proof of resolution of singularities was believed to be correct for several years before being found to contain a mistake. The conjecture remains open.

I DIDN'T HAVE THE
THE AREAS WHE
LEADING ME AND
CONSIDERED TO B

Is a Proof Correct?

ulative and lacking firm foundation. The groundbreaking 1986 p "Algebraic Cycles and Higher K-theory" by Spencer Bloch was soon af on found by Andrei Suslin to contain a mistake in the proof of L of could not be fixed, and almost all of the claims of the ostantiated.

A new proof, which replaced one original paper by thirty pages of complex arguments, until 1993, and it took many more years for it to be ac terestingly, this new proof was based on an older result o , who, at about the same tim of the resolution of singularit ivakovsky's proof of reso- lution of s was believed to be correct for several years before being found to contain a mistake. The conjecture remains open.

This story got me scared.

I DIDN'T HAVE THE
THE AREAS WHE
LEADING ME AND
CONSIDERED TO B

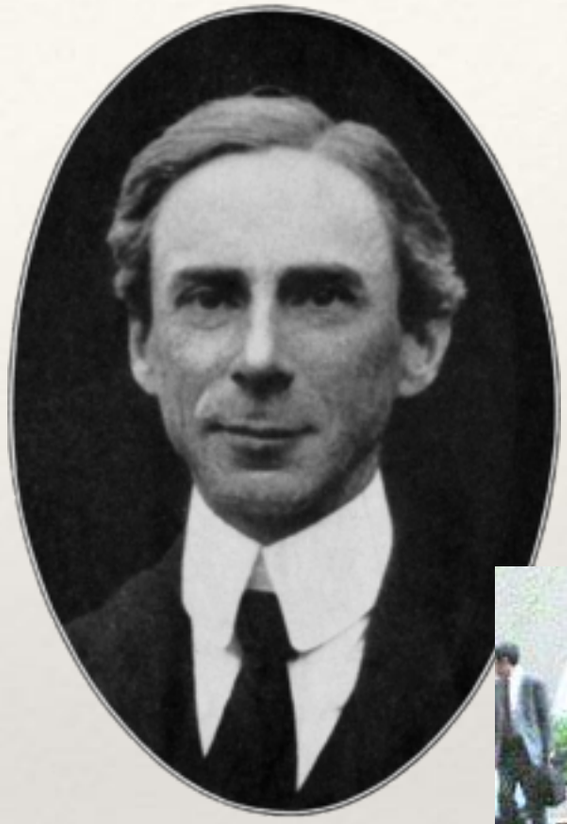
Computer Assistance and Foundations of Mathematics

:o what I could do to create such tools. And it soon long-term solution was somehow to make it possible for me to use computers to verify my abstract, logical, and mathematical constructions. The software for doing this has been in development since the sixties.

The primary challenge that needed to be addressed was that the foundations of mathematics were unprepared for the requirements of the task. Formulating mathematical reasoning in a language precise enough for a computer to follow meant using a foundational system of mathematics not as a standard of consistency to establish a few fundamental theorems, but as a tool that can be employed in

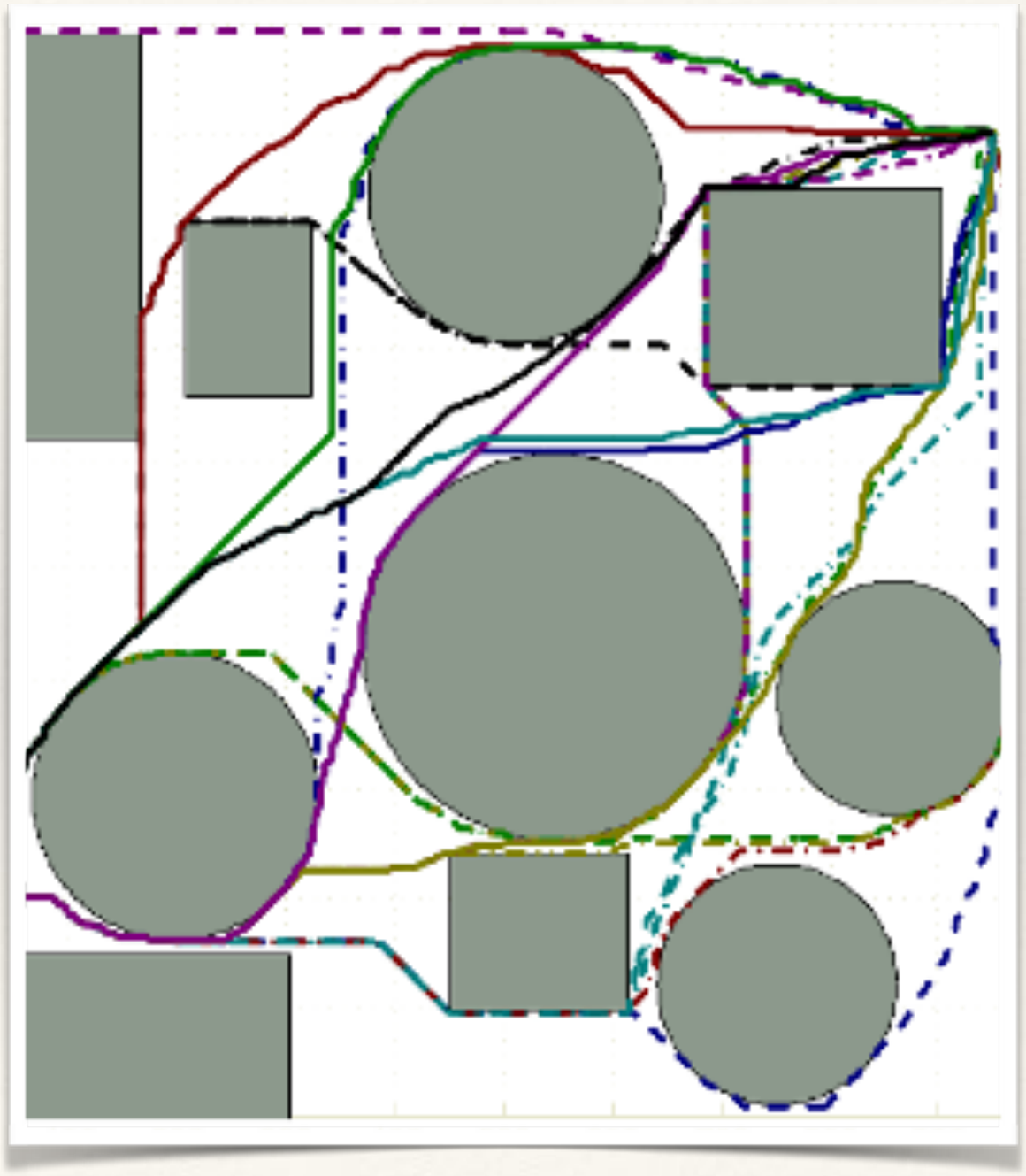
Type Theory

- Avoids paradoxes of set theory
- Amenable to computer verification
- Most importantly: natural for *higher-order* mathematics —- infinity groupoids
- Amazing coincidence: models of type theory are consistent with homotopy theory



HoTT: Equality, Equivalence, and Paths

- A proof of $A = B$ is interpreted as:
- an equivalence (isomorphism) of spaces A and B
- and as a path (smooth transformation) from space A to space B
- Can naturally talk about paths between paths between paths between spaces



Univalence

$(A \equiv B) \simeq (A \simeq B)$

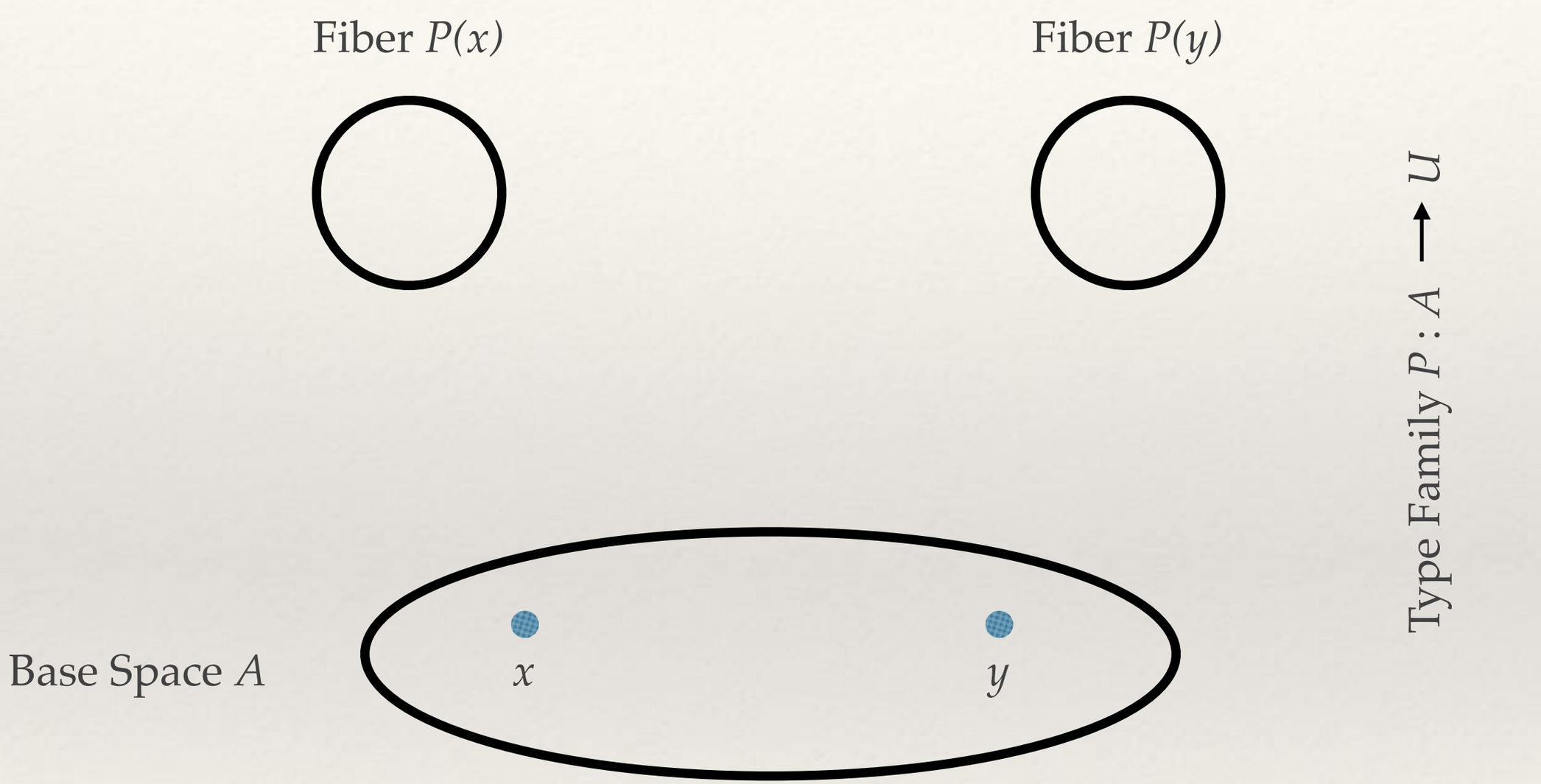
where $A \equiv B$ is the type of paths

and $A \simeq B$ is the type of equivalences

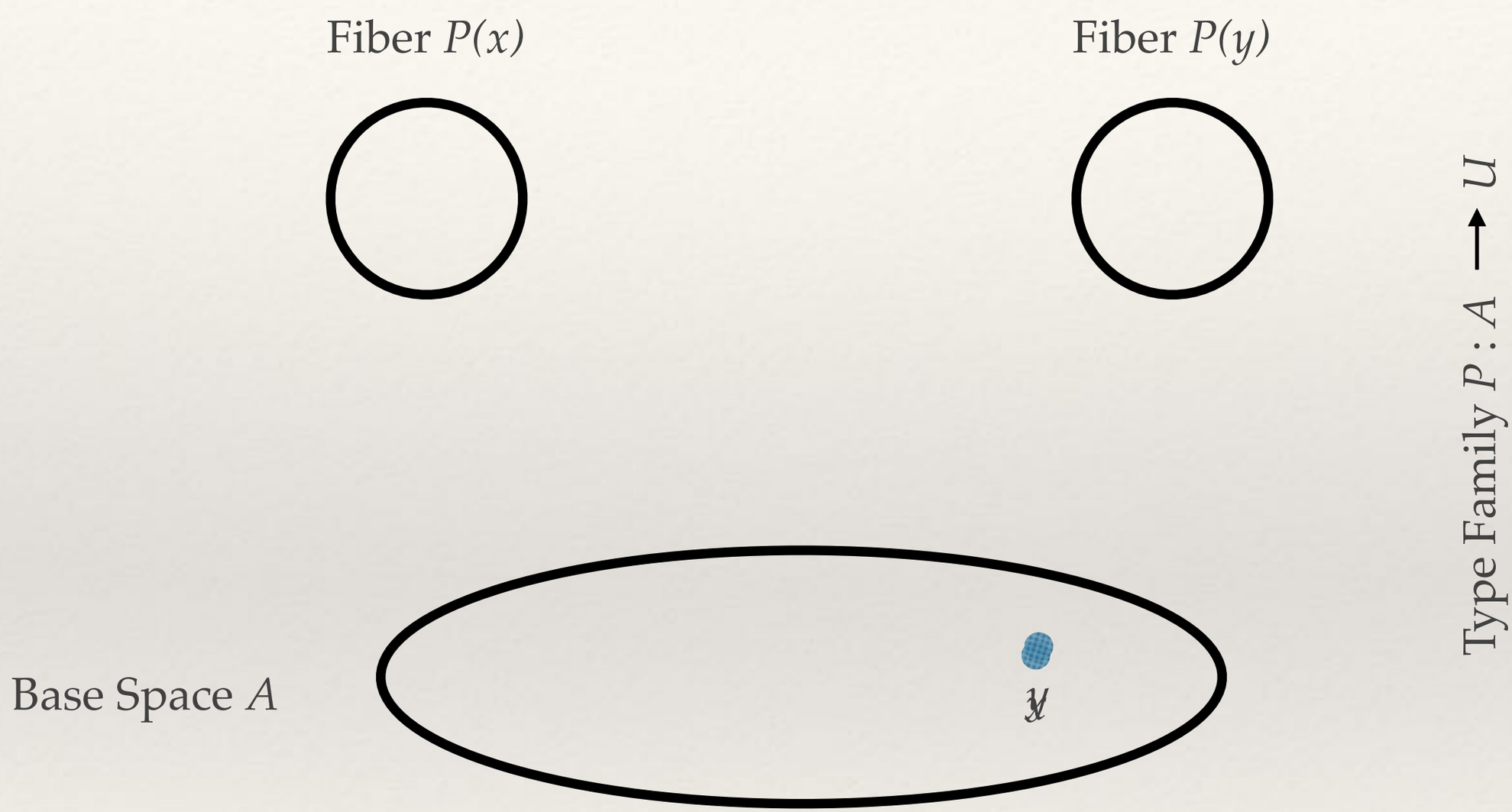
A postulate / axiom in "book HoTT"

Univalent Universes and Reversible Programming Languages

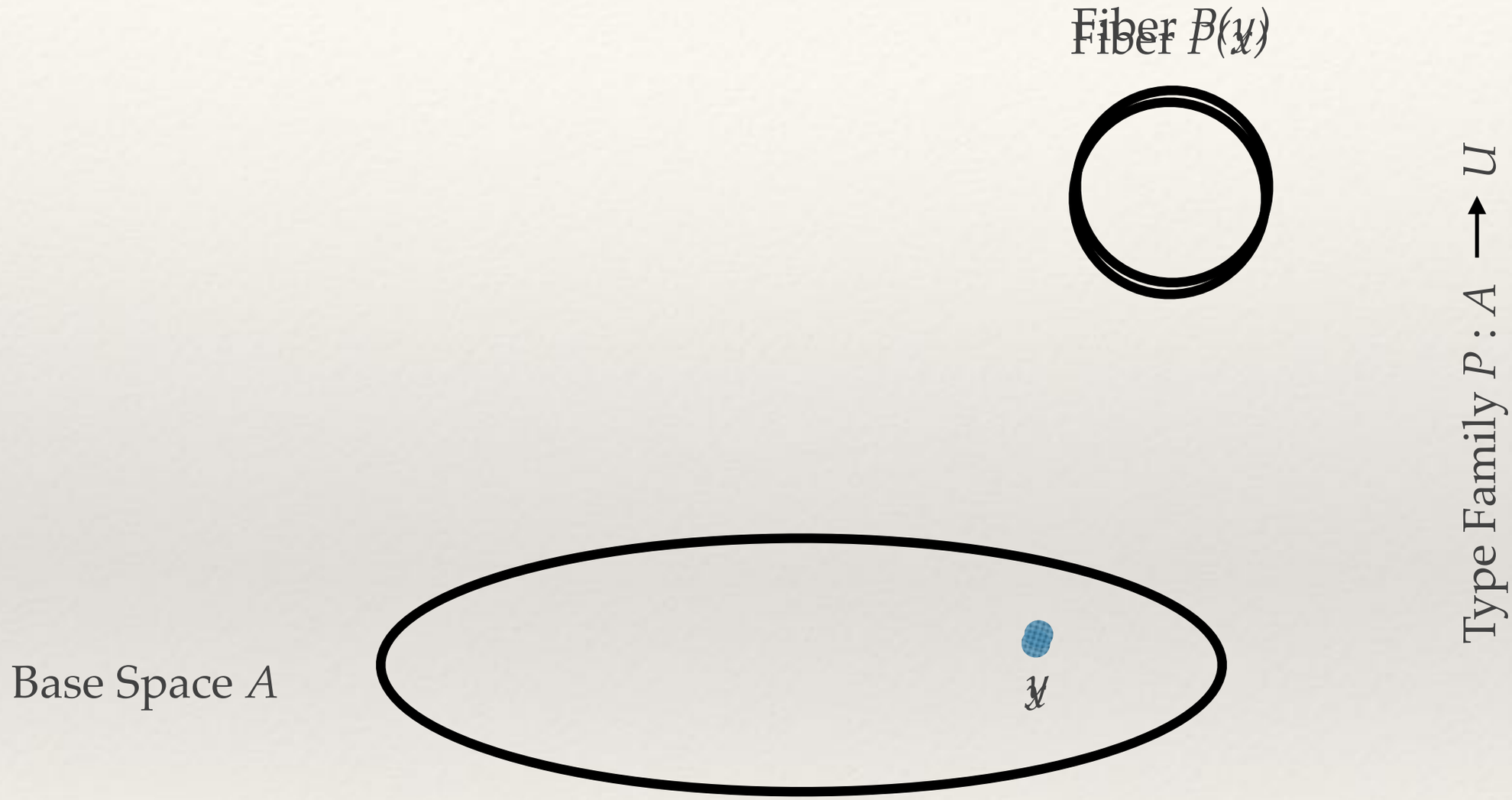
Fibrations



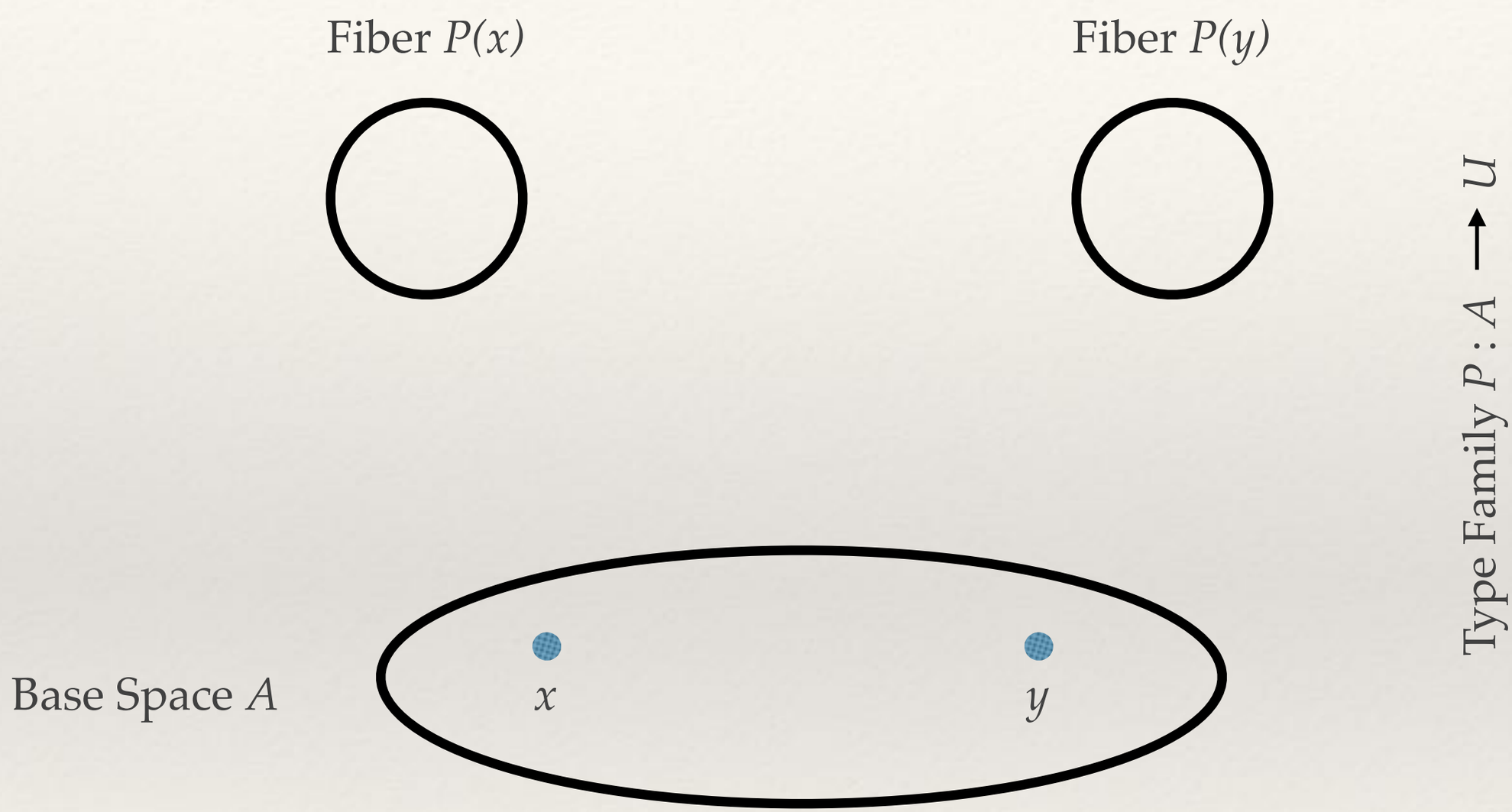
Fibrations



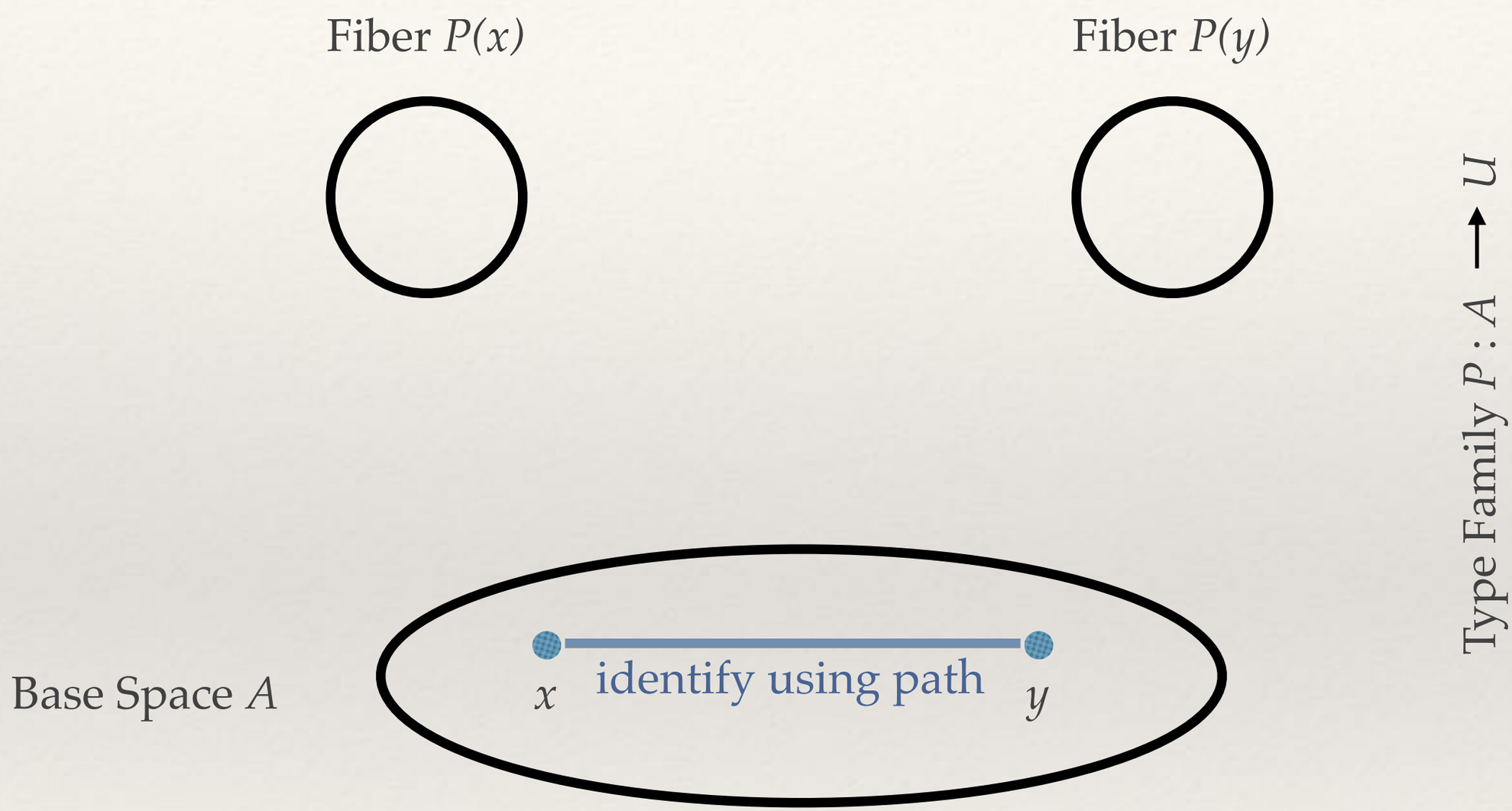
Fibrations



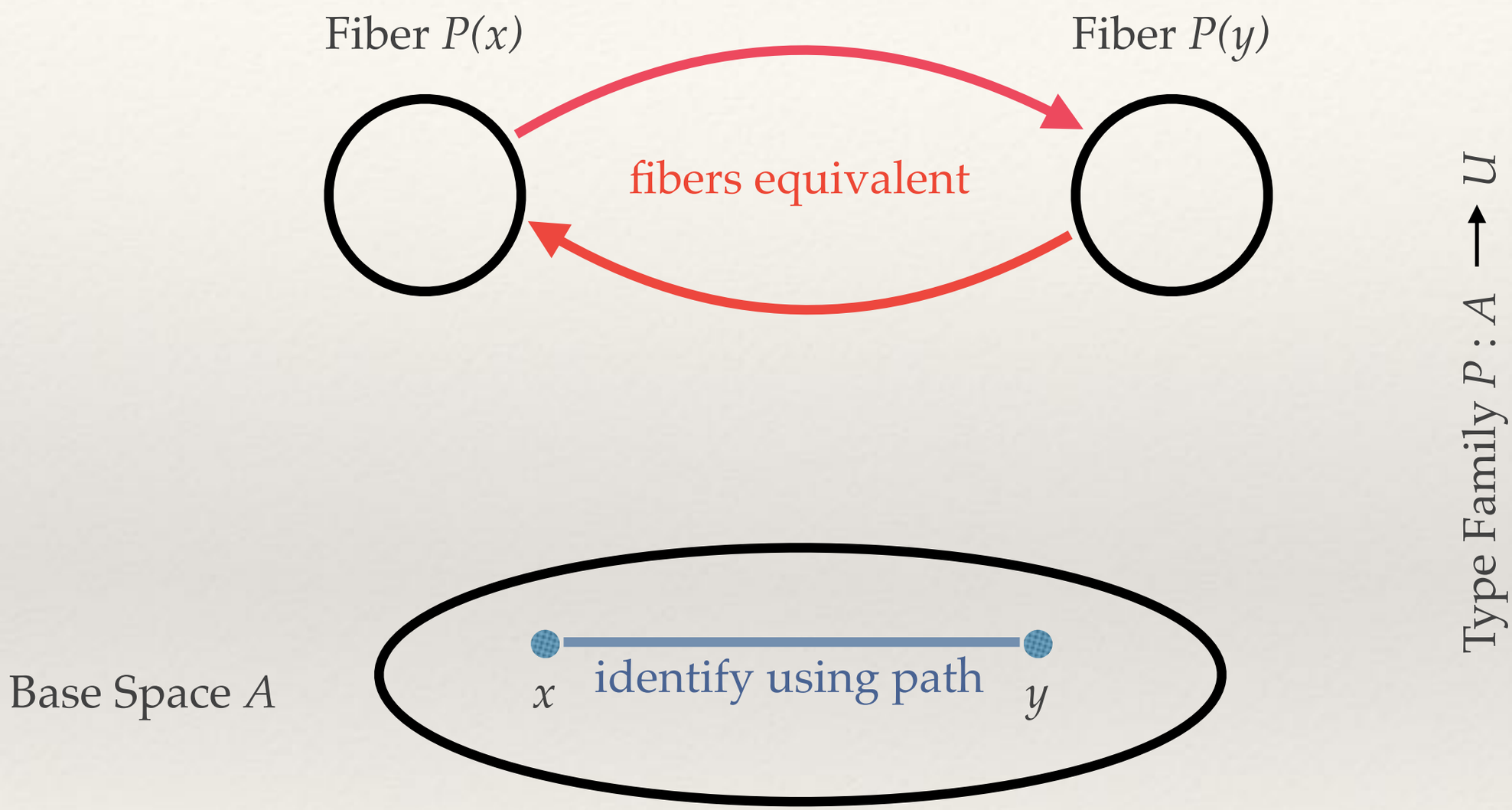
Fibrations



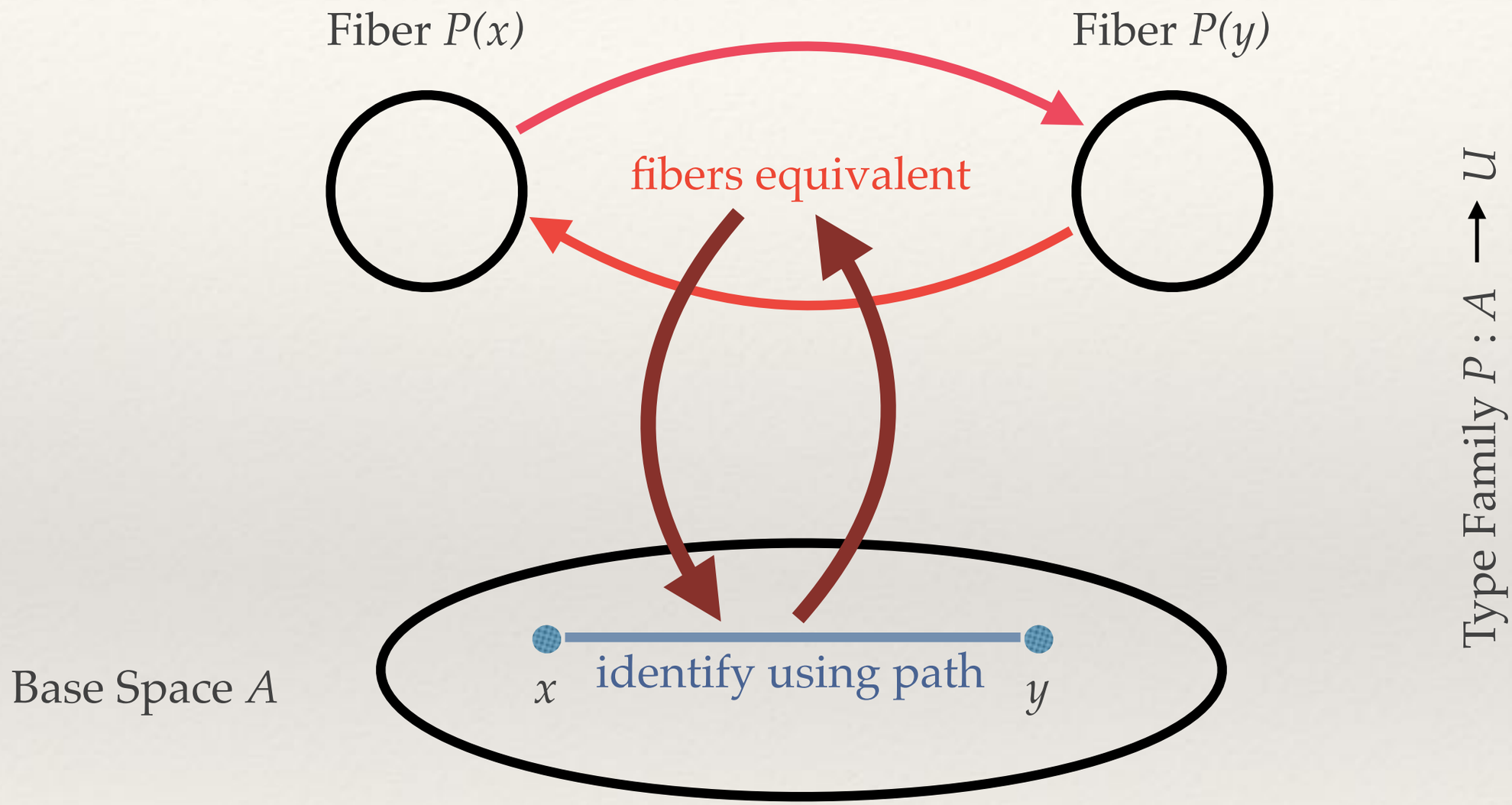
Fibrations


Fiber $P(x)$
Fiber $P(y)$
Type Family $P : A \to U$
Base Space A
identify using path
x
y

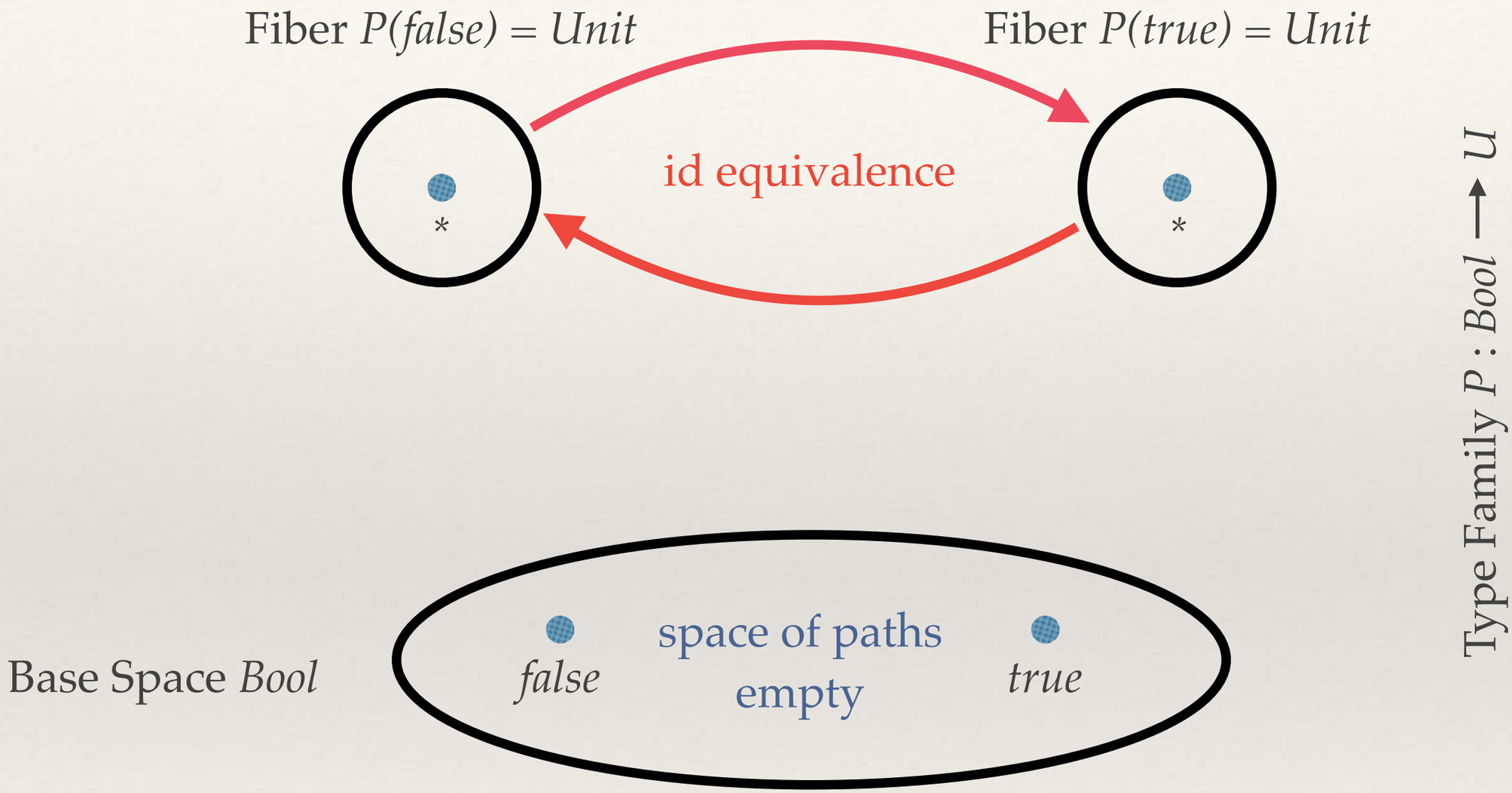
Fibrations


Fiber $P(x)$
Fiber $P(y)$
fibers equivalent
Type Family $P : A \to U$
Base Space A
x
identify using path
y

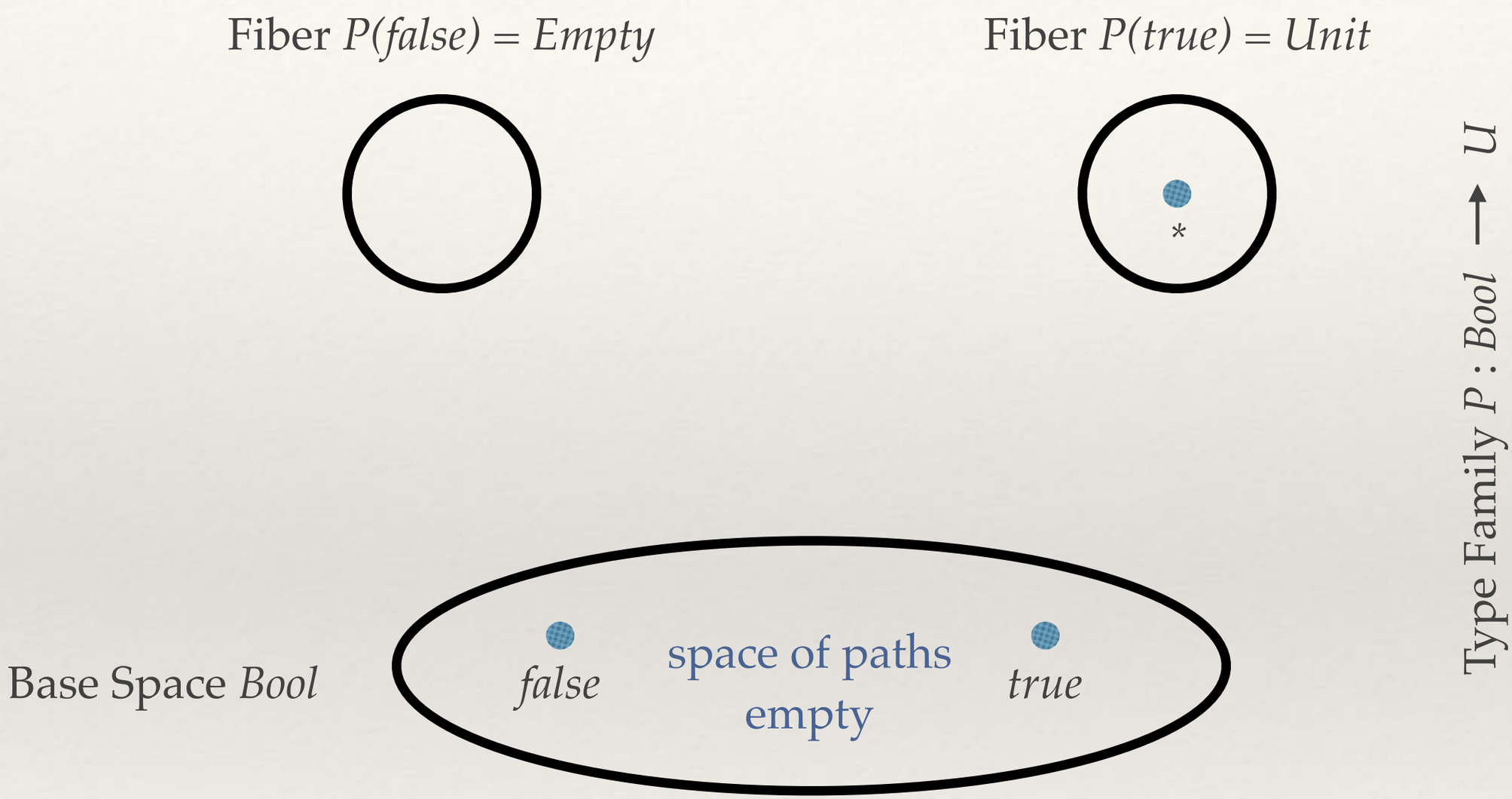
Univalent Fibrations



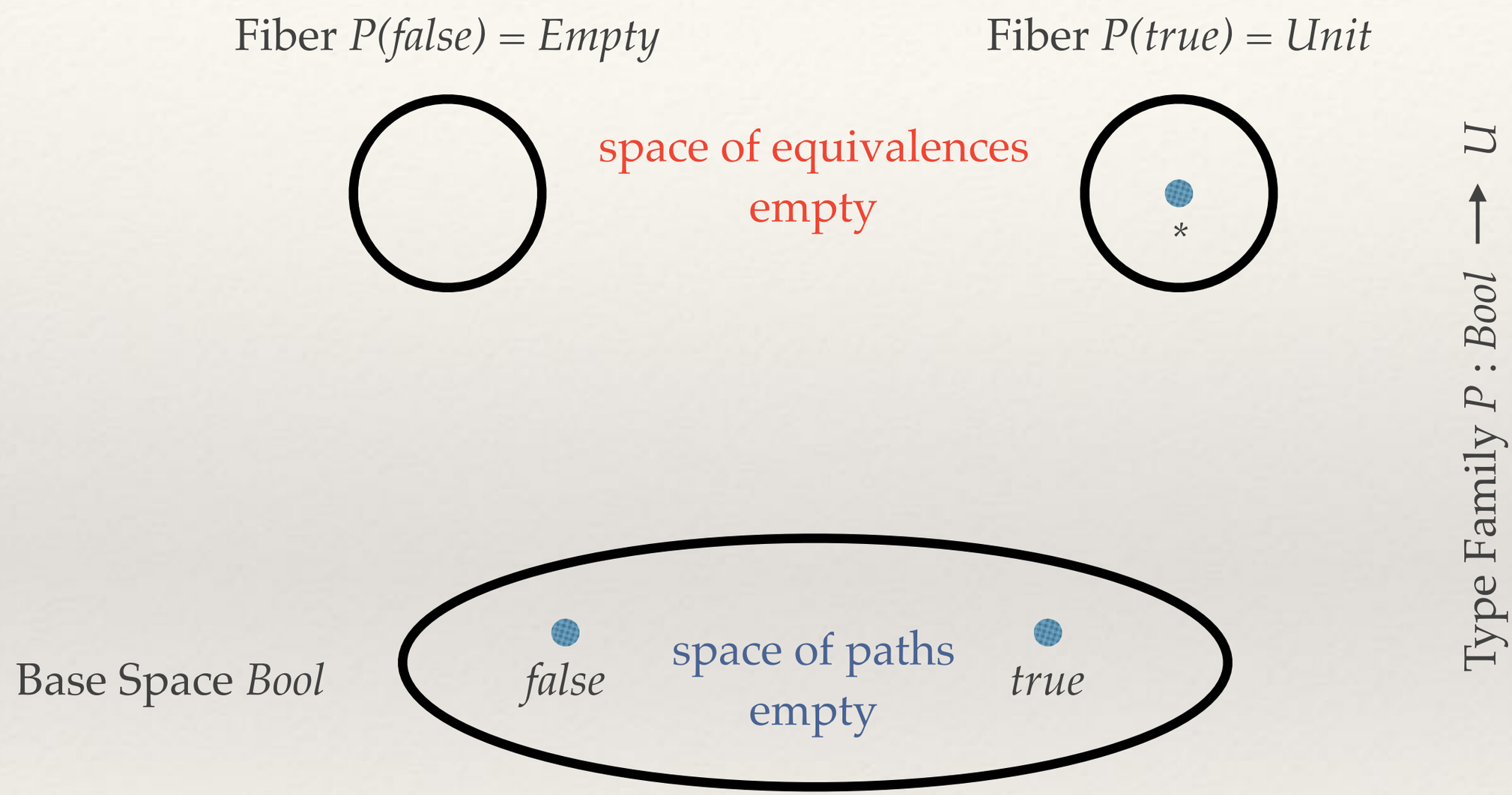
A Non-Univalent Fibration



A Univalent Fibration



A Univalent Fibration



Characterizing Univalent Fibrations

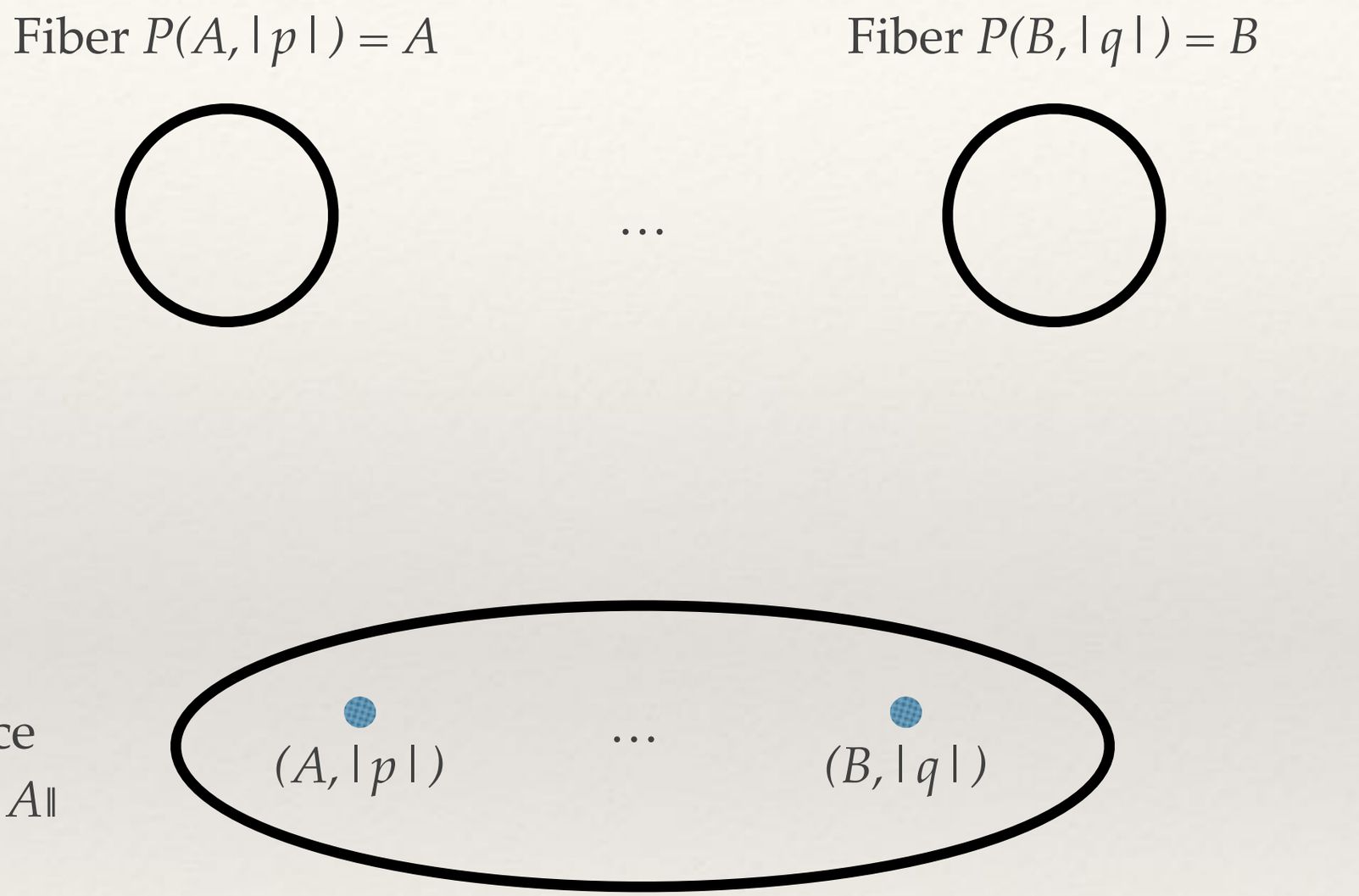
A given type X is <u>rarely</u> the base of a univalent fibration but we have the following theorem.

Theorem. *For any type F, the type:*

$$\Sigma A{:}U \, \| F \equiv A \|$$

is always the base of a univalent fibration.

Choose F = *Bool*



Questions

- What does this mean?
- What does $\Sigma_{(A:U)} \| Bool \equiv A \|$ look like?
- The "normal" thing to do is to explain the ingredients: first Σ-types, then propositional truncations $\|\_\|$, and finally identity types $A \equiv B$
- *That's about four chapters of the HoTT book!*

A Different Answer

- $\Sigma_{(A:\mathcal{U})} \| Bool \equiv A \|$ is the specification of a non-trivial but rather intuitive reversible programming language!
- Paths are reversible programs
- Univalence ensures every equivalence is expressible as a reversible program
- The space has non-trivial 2-paths!!! So we have a layer of reversible programs that manipulate other reversible programs in reversible ways

In Fact…

- $\Sigma_{(A:U)} \| Bool \equiv A \|$ is reversible language from several slides back (restricted to just the type of booleans)

The Restricted Language

- $\Sigma_{(A:U)} \| Bool \equiv A \|$ is a topological space, a higher groupoid
- Points
- Paths
- 2-Paths
- Trivial 3-Paths

$$
\begin{array}{lll}
\tau ::= & 2 & \\
\\
v ::= & \mathit{false} : 2 & \\
| & \mathit{true} : 2 & \\
\\
c ::= & \mathtt{id} : \tau \leftrightarrow \tau & \\
| & \mathtt{swap} : 2 \leftrightarrow 2 & \\
| & ! : (\tau_1 \leftrightarrow \tau_2) \to (\tau_2 \leftrightarrow \tau_1) & \\
| & \circ : (\tau_1 \leftrightarrow \tau_2) \to (\tau_2 \leftrightarrow \tau_3) \to (\tau_1 \leftrightarrow \tau_3) & \\
\\
\alpha ::= & \mathtt{id} : c \Leftrightarrow c & \\
| & \mathit{idl} : \mathtt{id} \circ c \Leftrightarrow c & \\
| & \mathit{idr} : c \circ \mathtt{id} \Leftrightarrow c & \\
| & \mathit{invl} : c \circ ! c \Leftrightarrow \mathtt{id} & \\
| & \mathit{invr} : ! c \circ c \Leftrightarrow \mathtt{id} & \\
| & \rho : \mathtt{swap} \circ \mathtt{swap} \Leftrightarrow \mathtt{id} & \\
| & \mathit{assoc} : (c_1 \circ c_2) \circ c_3 \Leftrightarrow c_1 \circ (c_2 \circ c_3) & \\
| & \odot : (c_1 \Leftrightarrow c_1') \to (c_2 \Leftrightarrow c_2') \to (c_1 \circ c_2 \Leftrightarrow c_1' \circ c_2') & \\
| & !! : (c_1 \Leftrightarrow c_2) \to (c_2 \Leftrightarrow c_1) & \\
| & \bullet : (c_1 \Leftrightarrow c_2) \to (c_2 \Leftrightarrow c_3) \to (c_1 \Leftrightarrow c_3) &
\end{array}
$$

Goals

- Extend this correspondence to richer types
- First, all finite types
- Fractional Types
- Higher-Inductive types

Team