Table of Contents

A little pair programming for us all to get our feet wet

Binary trees

Tree

I want to work with trees

Binary trees

$\text{Tree}(a : \text{Type})$

. . . as containers of some type, a

Binary trees

$$\text{Tree}(a : \text{Type}) \;:\; \text{Type}$$

. . . which, I assert, is a type itself

Binary trees

$$\texttt{data}\ \ \mathrm{Tree}(a : \mathrm{Type})\ :\ \mathrm{Type}\ \ \texttt{where}$$

. . . constructed by

Binary trees

```
data Tree(a : Type) : Type where
  Leaf
```

. . . building a leaf

Binary trees

$$\texttt{data}\ \ \mathrm{Tree}(a : \mathrm{Type})\ :\ \mathrm{Type}\ \ \texttt{where}$$
$$\quad \texttt{Leaf}\ :\ a\ \to\ \mathrm{Tree}$$

. . . from some value of type a

Binary trees

$$\texttt{data}\ \ \mathrm{Tree}(a : \mathrm{Type})\ :\ \mathrm{Type}\ \ \texttt{where}$$
$$\quad \texttt{Leaf}\ :\ a\ \to\ \mathrm{Tree}$$
$$\quad \texttt{Succ}$$

. . . or building a branching node

Binary trees

$$\texttt{data}\ \ \mathrm{Tree}(a : \mathrm{Type})\ :\ \mathrm{Type}\ \ \texttt{where}$$
$$\texttt{Leaf}\ :\ a\ \to\ \mathrm{Tree}$$
$$\texttt{Succ}\ :\ \mathrm{Tree}\ \&\ a\ \&\ \mathrm{Tree} \to \mathrm{Tree}$$

. . . using a left subtree, a value of type a, and a right subtree

Any questions so far? (I love questions)

Peano Arithmetic

$\mathbb{N}$

I assert that: the natural numbers

Peano Arithmetic

$$\mathbb{N} \ : \ \text{Type}$$

. . . are a type

Peano Arithmetic

```
data ℕ : Type where
```

. . . introduced as

Peano Arithmetic

```
data ℕ : Type where
  Zero
```

. . . either the special value, zero

Peano Arithmetic

```
data ℕ : Type where
  Zero : ℕ
```

. . . which is automatically a natural

Peano Arithmetic

```
data ℕ : Type where
  Zero : ℕ
  Succ
```

. . . or by applying "successor"

Peano Arithmetic

$$\begin{aligned}&\texttt{data}\ \mathbb{N} : \text{Type}\ \texttt{where}\\ &\quad \texttt{Zero} : \mathbb{N}\\ &\quad \texttt{Succ} : \mathbb{N} \to \mathbb{N}\end{aligned}$$

. . . a function taking a natural number, n *to* $n + 1$

Peano Arithmetic (cont.)

$rec_{\mathbb{N}}$

I assert that: the naturals have recursion

Peano Arithmetic (cont.)

$rec_{\mathbb{N}} \quad :$

. . . which is the type of thing

Peano Arithmetic (cont.)

$rec_{\mathbb{N}}$:

$(P$ $)$

. . . given a property of naturals

Peano Arithmetic (cont.)

$$
\begin{aligned}
&rec_{\mathbb{N}} \quad : \\
&\quad (P : \mathbb{N} \to \mathrm{Type}) \to
\end{aligned}
$$

Peano Arithmetic (cont.)

$$
\begin{aligned}
&rec_{\mathbb{N}} \quad : \\
&\quad (P : \mathbb{N} \to \text{Type}) \to \\
&\quad (ind \qquad\qquad\qquad\qquad)
\end{aligned}
$$

. . . given an induction step

Peano Arithmetic (cont.)

$$
\begin{aligned}
&rec_{\mathbb{N}} \quad : \\
&\quad (P : \mathbb{N} \to \mathrm{Type}) \to \\
&\quad (ind : (\mathtt{Succ}\ n : \mathbb{N}) \to P(n) \to P(\mathtt{Succ}\ n)) \to
\end{aligned}
$$

Peano Arithmetic (cont.)

$$
\begin{aligned}
&rec_{\mathbb{N}} \quad : \\
&\quad (P : \mathbb{N} \to \mathrm{Type}) \to \\
&\quad (ind : (\mathtt{Succ}\ n : \mathbb{N}) \to P(n) \to P(\mathtt{Succ}\ n)) \to \\
&\quad (base \qquad\qquad\quad)
\end{aligned}
$$

. . . and a base case

Peano Arithmetic (cont.)

$$
\begin{aligned}
&rec_{\mathbb{N}} \quad : \\
&\quad (P : \mathbb{N} \to \mathrm{Type}) \to \\
&\quad (ind : (\texttt{Succ}\ n : \mathbb{N}) \to P(n) \to P(\texttt{Succ}\ n)) \to \\
&\quad (base : P(\texttt{Zero})) \to
\end{aligned}
$$

Peano Arithmetic (cont.)

$$
\begin{aligned}
&rec_{\mathbb{N}} \quad : \\
&\quad (P : \mathbb{N} \to \mathrm{Type}) \to \\
&\quad (ind : (\texttt{Succ}\ n : \mathbb{N}) \to P(n) \to P(\texttt{Succ}\ n)) \to \\
&\quad (base : P(\texttt{Zero})) \to \\
&\quad (m \qquad)
\end{aligned}
$$

. . . finds now that, for any natural

Peano Arithmetic (cont.)

$$
\begin{aligned}
&rec_{\mathbb{N}} \quad : \\
&\quad (P : \mathbb{N} \to \mathrm{Type}) \to \\
&\quad (ind : (\mathtt{Succ}\ n : \mathbb{N}) \to P(n) \to P(\mathtt{Succ}\ n)) \to \\
&\quad (base : P(\mathtt{Zero})) \to \\
&\quad (m : \mathbb{N}) \to
\end{aligned}
$$

Peano Arithmetic (cont.)

$$
\begin{aligned}
&rec_{\mathbb{N}} \quad : \\
&\quad (P : \mathbb{N} \to \mathrm{Type}) \to \\
&\quad (ind : (\texttt{Succ}\ n : \mathbb{N}) \to P(n) \to P(\texttt{Succ}\ n)) \to \\
&\quad (base : P(\texttt{Zero})) \to \\
&\quad (m : \mathbb{N}) \to \\
&\quad P(m)
\end{aligned}
$$

. . . that the recursive property holds

Peano Arithmetic (e.g.)

Assuming . . .

$$\begin{aligned}
\text{Even}(n : \mathbb{N}) &: \text{Type} \\
\text{Odd}(n : \mathbb{N}) &: \text{Type} \\
\text{zeroIsEven} &: \text{Even}(\texttt{Zero}) \\
\text{succEven} &: \text{Even}(n) \to \text{Odd}(\texttt{Succ}\ n) \\
\text{succOdd} &: \text{Odd}(n) \to \text{Even}(\texttt{Succ}\ n)
\end{aligned}$$

Peano Arithmetic (e.g.)

Assuming . . .

$$\begin{aligned}
\text{Even}(n : \mathbb{N}) &: \text{Type} \\
\text{Odd}(n : \mathbb{N}) &: \text{Type} \\
\text{zeroIsEven} &: \text{Even}(\texttt{Zero}) \\
\text{succEven} &: \text{Even}(n) \to \text{Odd}(\texttt{Succ}\ n) \\
\text{succOdd} &: \text{Odd}(n) \to \text{Even}(\texttt{Succ}\ n)
\end{aligned}$$

Can we *grasp* how to show

Peano Arithmetic (e.g.)

Assuming . . .

$$\begin{aligned}
\text{Even}(n : \mathbb{N}) &: \text{Type} \\
\text{Odd}(n : \mathbb{N}) &: \text{Type} \\
\text{zeroIsEven} &: \text{Even}(\texttt{Zero}) \\
\text{succEven} &: \text{Even}(n) \to \text{Odd}(\texttt{Succ}\ n) \\
\text{succOdd} &: \text{Odd}(n) \to \text{Even}(\texttt{Succ}\ n)
\end{aligned}$$

Can we *grasp* how to show

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

So, in order to construct a function of this type

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) =$$

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

. . . we'll recurse on the argument

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

base :

base =

. . . with a particular choice of base case

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

base :

base =

ind :

$$\text{ind}(n, x) =$$

. . . and inductive step

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

$$\text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero})$$

$$\text{base} =$$

$$\text{ind} :$$

$$\text{ind}(n, x) =$$

. . . the base case must demonstrate the parity of zero

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

$$\text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero})$$

$$\text{base} = \text{Inl}(\qquad\qquad)$$

$$\text{ind} :$$

$$\text{ind}(n, x) =$$

. . . which we immediately know to be "even"

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

$$\text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero})$$

$$\text{base} = \text{Inl}(\text{zeroIsEven})$$

$$\text{ind} :$$

$$\text{ind}(n, x) =$$

. . . reflected in the meaning of this previous construction

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

$$\text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero})$$

$$\text{base} = \text{Inl}(\text{zeroIsEven})$$

$$\text{ind} : \qquad \to$$

$$\qquad\qquad \to$$

$$\text{ind}(n, x) =$$

. . . for the inductive case, we know there are two arguments

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

$$\text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero})$$

$$\text{base} = \text{Inl}(\text{zeroIsEven})$$

$$\text{ind} : \qquad \to$$

$$\text{Even}(n) \ \vee \ \text{Odd}(n) \to$$

$$\text{ind}(n, x) =$$

. . . the second of which is a "prior" instance of our goal

Peano Arithmetic, decidability of parity

$$\begin{aligned}
&\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n) \\
&\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m) \\
&\quad \texttt{where} \\
&\qquad \text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero}) \\
&\qquad \text{base} = \text{Inl}(\text{zeroIsEven}) \\
&\qquad \text{ind} : (\texttt{Succ}\ n : \mathbb{N}) \to \\
&\qquad\quad \text{Even}(n) \ \vee \ \text{Odd}(n) \to \\
&\qquad \text{ind}(n, x) =
\end{aligned}$$

. . . which, combined with a ***new*** *number,*

Peano Arithmetic, decidability of parity

$$\begin{array}{l}
\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n) \\
\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m) \\
\quad \texttt{where} \\
\qquad \text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero}) \\
\qquad \text{base} = \text{Inl}(\text{zeroIsEven}) \\
\qquad \text{ind} : (\texttt{Succ}\ n : \mathbb{N}) \to \\
\qquad\quad \text{Even}(n) \ \vee \ \text{Odd}(n) \to \\
\qquad\quad \text{Even}(\texttt{Succ}\ n) \ \vee \ \text{Odd}(\texttt{Succ}\ n) \\
\qquad \text{ind}(n, x) =
\end{array}$$

. . . must give us an updated goal

Peano Arithmetic, decidability of parity

$$\begin{aligned}
&\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n) \\
&\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m) \\
&\quad \texttt{where} \\
&\qquad \text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero}) \\
&\qquad \text{base} = \text{Inl}(\text{zeroIsEven}) \\
&\qquad \text{ind} : (\texttt{Succ}\ n : \mathbb{N}) \to \\
&\qquad\quad \text{Even}(n) \ \vee \ \text{Odd}(n) \to \\
&\qquad\quad \text{Even}(\texttt{Succ}\ n) \ \vee \ \text{Odd}(\texttt{Succ}\ n) \\
&\qquad \text{ind}(n, x) = \qquad\quad x
\end{aligned}$$

. . . we'll take a look at the prior instance of the goal

Peano Arithmetic, decidability of parity

$$
\begin{aligned}
&\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n) \\
&\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m) \\
&\quad \texttt{where} \\
&\qquad \text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero}) \\
&\qquad \text{base} = \text{Inl}(\text{zeroIsEven}) \\
&\qquad \text{ind} : (\texttt{Succ}\ n : \mathbb{N}) \to \\
&\qquad\quad \text{Even}(n) \ \vee \ \text{Odd}(n) \to \\
&\qquad\quad \text{Even}(\texttt{Succ}\ n) \ \vee \ \text{Odd}(\texttt{Succ}\ n) \\
&\qquad \text{ind}(n, x) = \ \texttt{case}\ x\ \texttt{of}
\end{aligned}
$$

. . . examining its structure

Peano Arithmetic, decidability of parity

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m)$$

`where`

$$\text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero})$$

$$\text{base} = \text{Inl}(\text{zeroIsEven})$$

$$\text{ind} : (\texttt{Succ}\ n : \mathbb{N}) \to$$

$$\text{Even}(n) \ \vee \ \text{Odd}(n) \to$$

$$\text{Even}(\texttt{Succ}\ n) \ \vee \ \text{Odd}(\texttt{Succ}\ n)$$

$$\text{ind}(n, x) = \ \texttt{case}\ x\ \texttt{of}$$

$$\text{Inl}(evn) \to$$

$$\text{Inr}(odn) \to$$

. . . which is either a proof on the left of evenness, or on the right of oddness

Peano Arithmetic, decidability of parity

$$\begin{aligned}
&\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n) \\
&\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m) \\
&\quad \texttt{where} \\
&\qquad \text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero}) \\
&\qquad \text{base} = \text{Inl}(\text{zeroIsEven}) \\
&\qquad \text{ind} : (\texttt{Succ}\ n : \mathbb{N}) \to \\
&\qquad\quad \text{Even}(n) \ \vee \ \text{Odd}(n) \to \\
&\qquad\quad \text{Even}(\texttt{Succ}\ n) \ \vee \ \text{Odd}(\texttt{Succ}\ n) \\
&\qquad \text{ind}(n, x) = \ \texttt{case}\ x\ \texttt{of} \\
&\qquad\quad \text{Inl}(evn) \to \text{Inr}(\text{succEven}(evn)) \\
&\qquad\quad \text{Inr}(odn) \to
\end{aligned}$$

. . . if it shows evenness, then the successor is odd

Peano Arithmetic, decidability of parity

$$\begin{aligned}
&\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n) \\
&\text{decideParity}(n) = \text{rec}_{\mathbb{N}}(\text{ind}, \text{base}, m) \\
&\quad \texttt{where} \\
&\qquad \text{base} : \text{Even}(\texttt{Zero}) \ \vee \ \text{Odd}(\texttt{Zero}) \\
&\qquad \text{base} = \text{Inl}(\text{zeroIsEven}) \\
&\qquad \text{ind} : (\texttt{Succ}\ n : \mathbb{N}) \to \\
&\qquad\quad \text{Even}(n) \ \vee \ \text{Odd}(n) \to \\
&\qquad\quad \text{Even}(\texttt{Succ}\ n) \ \vee \ \text{Odd}(\texttt{Succ}\ n) \\
&\qquad \text{ind}(n, x) = \ \texttt{case}\ x\ \texttt{of} \\
&\qquad\quad \text{Inl}(evn) \to \text{Inr}(\text{succEven}(evn)) \\
&\qquad\quad \text{Inr}(odn) \to \text{Inl}(\text{succOdd}(odn))
\end{aligned}$$

. . . if it shows oddness, then the successor is even

Whew! That was actually a *lot* of pair programming.

Whew! That was actually a *lot* of pair programming. Or was it mathematics?

Table of Contents

Dr. Martin-Löf

or: How I learned to stop worrying and love computation

Joseph Abrahamson

2014 September 18

Who was Per Martin-Löf?

- Born 1942, Erdös-"died" in 2009
  - (e.g. he retired)

Who was Per Martin-Löf?

- Born 1942, Erdös-"died" in 2009
  - (e.g. he retired)
- Swedish joint chair of Mathematics and Philosophy at Stockholm University

Who was Per Martin-Löf?

- Born 1942, Erdös-"died" in 2009
  - (e.g. he retired)
- Swedish joint chair of Mathematics and Philosophy at Stockholm University
- Major contributions to statistics and mathematical logic

Who was Per Martin-Löf?

- Born 1942, Erdös-"died" in 2009
  - (e.g. he retired)
- Swedish joint chair of Mathematics and Philosophy at Stockholm University
- Major contributions to statistics and mathematical logic
- **"Martin-Löf's Intuitionistic Type Theory"**

Table of Contents

What is MLTT

A culminating effort to express and mechanize the computational aspects of Brouwer's *Intuitionism* program.

What is MLTT

A culminating effort to express and mechanize the computational aspects of Brouwer's *Intuitionism* program.

An open system for constructing *constructions*

What is MLTT

A culminating effort to express and mechanize the computational aspects of Brouwer's *Intuitionism* program.

An open system for constructing *constructions*

An intuitionistic system which favors verifiability

What is MLTT

A culminating effort to express and mechanize the computational aspects of Brouwer's *Intuitionism* program.

- An open system for constructing *constructions*
- An intuitionistic system which favors verifiability
- Exposes the computational nature of mathematical constructions

What is MLTT

A culminating effort to express and mechanize the computational aspects of Brouwer's *Intuitionism* program.

- An open system for constructing *constructions*
- An intuitionistic system which favors verifiability
- Exposes the computational nature of mathematical constructions
- Clearly presents the relationship between types, theorems, programs, and proofs

What is MLTT

A culminating effort to express and mechanize the computational aspects of Brouwer's *Intuitionism* program.

- An open system for constructing *constructions*
- An intuitionistic system which favors verifiability
- Exposes the computational nature of mathematical constructions
- Clearly presents the relationship between types, theorems, programs, and proofs

Inspiration for NuPRL, Coq, Isabelle, Epigram, Agda, Idris, Haskell, OCaml, Rust, Scala, Swift, ...

What is MLTT

Intuitionistic Logic

Computation

Types

What is MLTT

Intuitionistic Logic

Disallow any "funny tricks", everything is clearly verifiable

Computation

Types

What is MLTT

Intuitionistic Logic

Disallow any "funny tricks", everything is clearly verifiable

Computation

Demonstrate how constructions *interact* and lead to computation

Types

What is MLTT

Intuitionistic Logic

Disallow any "funny tricks", everything is clearly verifiable

Computation

Demonstrate how constructions *interact* and lead to computation

Types

Represent the meaning of constructions via types.

What is MLTT

Intuitionistic Logic

Disallow any "funny tricks", everything is clearly verifiable

Computation

Demonstrate how constructions *interact* and lead to computation

Types

Represent the meaning of constructions via types. (There's a whole lot to this story that PML tells, but in the interest of time it's not the focus.)

What is Intuitionism?

Have you ever felt that "proof by contradiction" is a little *dodgy*?

What is Intuitionism?

L. E. J. Brouwer

Have you ever felt that "proof by contradiction" is a little *dodgy*?

This guy did, too

What is Intuitionism?

L. E. J. Brouwer

Have you ever felt that "proof by contradiction" is a little *dodgy*?

This guy did, too

Invented a "new kind of logical foundation" to fight those conerns in the early 20th century (1907-1912)

What is Intuitionism?

- Mathematics is a mental activity—patterns in the mind of the observer

What is Intuitionism?

- Mathematics is a mental activity—patterns in the mind of the observer
- The goal of mathematical communication is to engender some specific kind of patterns in the mind of the reader

What is Intuitionism?

- Mathematics is a mental activity—patterns in the mind of the observer
- The goal of mathematical communication is to engender some specific kind of patterns in the mind of the reader
- Began with the work of "pre-Intuitionists" like Poincaire, Borel, Lesbegue, Weyl, Kronecker

What is Intuitionism?

- Mathematics is a mental activity—patterns in the mind of the observer
- The goal of mathematical communication is to engender some specific kind of patterns in the mind of the reader
- Began with the work of "pre-Intuitionists" like Poincaire, Borel, Lesbegue, Weyl, Kronecker
    - (Borel and Lesbegue show up all the time in measure theory—this is embedded deep in the foundations of modern Probability Theory)

What is Intuitionism?

- Mathematics is a mental activity—patterns in the mind of the observer
- The goal of mathematical communication is to engender some specific kind of patterns in the mind of the reader
- Began with the work of "pre-Intuitionists" like Poincaire, Borel, Lesbegue, Weyl, Kronecker
    - (Borel and Lesbegue show up all the time in measure theory—this is embedded deep in the foundations of modern Probability Theory)
- Got into big fights with the "Formalist Logicians" of the time

What is Intuitionism?

- Mathematics is a mental activity—patterns in the mind of the observer
- The goal of mathematical communication is to engender some specific kind of patterns in the mind of the reader
- Began with the work of "pre-Intuitionists" like Poincaire, Borel, Lesbegue, Weyl, Kronecker
    - (Borel and Lesbegue show up all the time in measure theory—this is embedded deep in the foundations of modern Probability Theory)
- Got into big fights with the "Formalist Logicians" of the time
    - In particular his *mentor*, Hilbert

What is Intuitionism?

One of the most important and hackle-raising things Brouwer did was eliminate the notion of the "Law of Excluded Middle", e.g.

What is Intuitionism?

One of the most important and hackle-raising things Brouwer did was eliminate the notion of the "Law of Excluded Middle", e.g.

A proposition is either true or it is false, there is nothing else.

What is Intuitionism?

One of the most important and hackle-raising things Brouwer did was eliminate the notion of the "Law of Excluded Middle", e.g.

A proposition is either true or it is false, there is nothing else.

Since Brouwer believed mathematics was always a mental activity he accounted for the idea that we may have simply *not yet determined* whether something is true or false.

What is Intuitionism?

One of the most important and hackle-raising things Brouwer did was eliminate the notion of the "Law of Excluded Middle", e.g.

A proposition is either true or it is false, there is nothing else.

Since Brouwer believed mathematics was always a mental activity he accounted for the idea that we may have simply *not yet determined* whether something is true or false.

So, while don't (automatically) believe "P or not P" we instead believe "it's not *not* the case that P or not P...I just don't know which yet!".

Hilbert's Response

This infuriated Hilbert:

'Taking the Principle of the Excluded Middle from the mathematician ... is the same as ... prohibiting the boxer the use of his fists."

What is Formalist Logic?

To see why Hilbert was angry, and to better understand Intuitionism, we need to examine the Formalist Logical Program.

Russell and Whitehead's *Principia Mathematica*

- In the early 20th century Bertrand Russell and Alfred Whitehead were frustrated with how *fast and loose* mathematics was

Russell and Whitehead's *Principia Mathematica*

- In the early 20th century Bertrand Russell and Alfred Whitehead were frustrated with how *fast and loose* mathematics was
- They struggled to find the ultimate *Logical Foundations*

Russell and Whitehead's *Principia Mathematica*

- In the early 20th century Bertrand Russell and Alfred Whitehead were frustrated with how *fast and loose* mathematics was
- They struggled to find the ultimate *Logical Foundations*
- Overcame problems with infinity like Russell's Paradox to write *The Principia Mathematica*

Russell and Whitehead's *Principia Mathematica*

- In the early 20th century Bertrand Russell and Alfred Whitehead were frustrated with how *fast and loose* mathematics was
- They struggled to find the ultimate *Logical Foundations*
- Overcame problems with infinity like Russell's Paradox to write *The Principia Mathematica*
    - A manual for how to prove $1 + 1 = 2$ from the utter basics—in just under 400 pages of dense formal language

Russell and Whitehead's *Principia Mathematica*

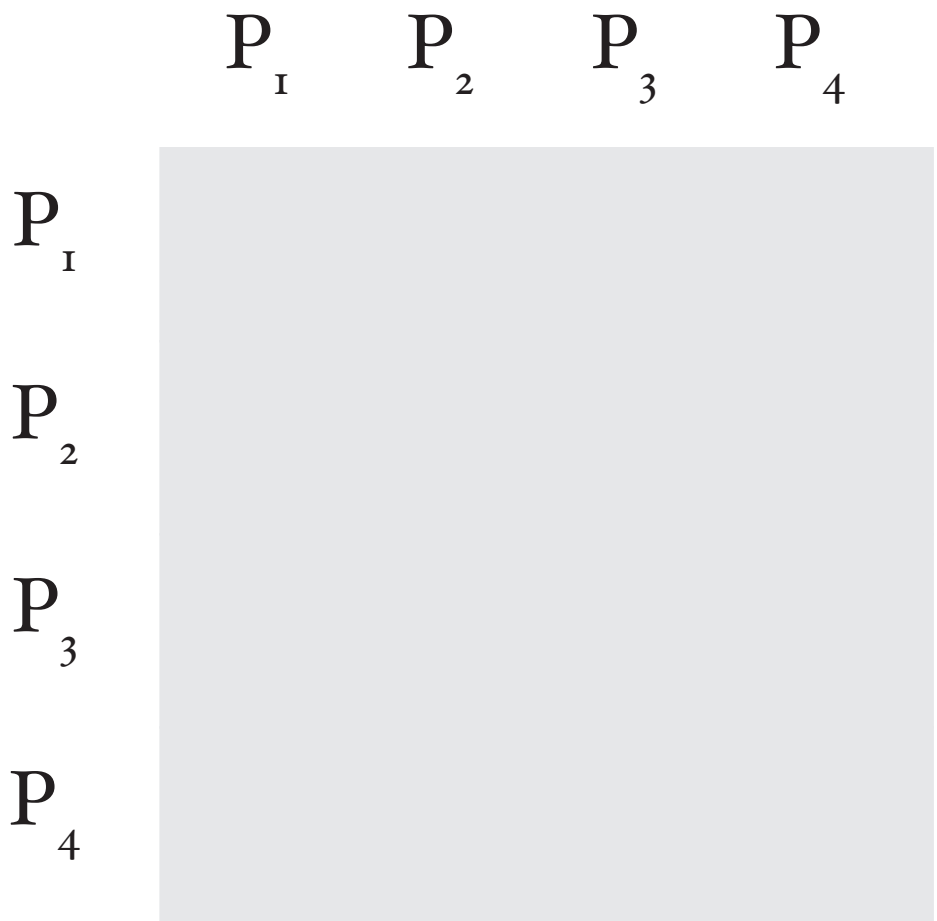
- In the early 20th century Bertrand Russell and Alfred Whitehead were frustrated with how *fast and loose* mathematics was
- They struggled to find the ultimate *Logical Foundations*
- Overcame problems with infinity like Russell's Paradox to write *The Principia Mathematica*
    - A manual for how to prove $1 + 1 = 2$ from the utter basics—in just under 400 pages of dense formal language
    - In other words: *a triumph!*

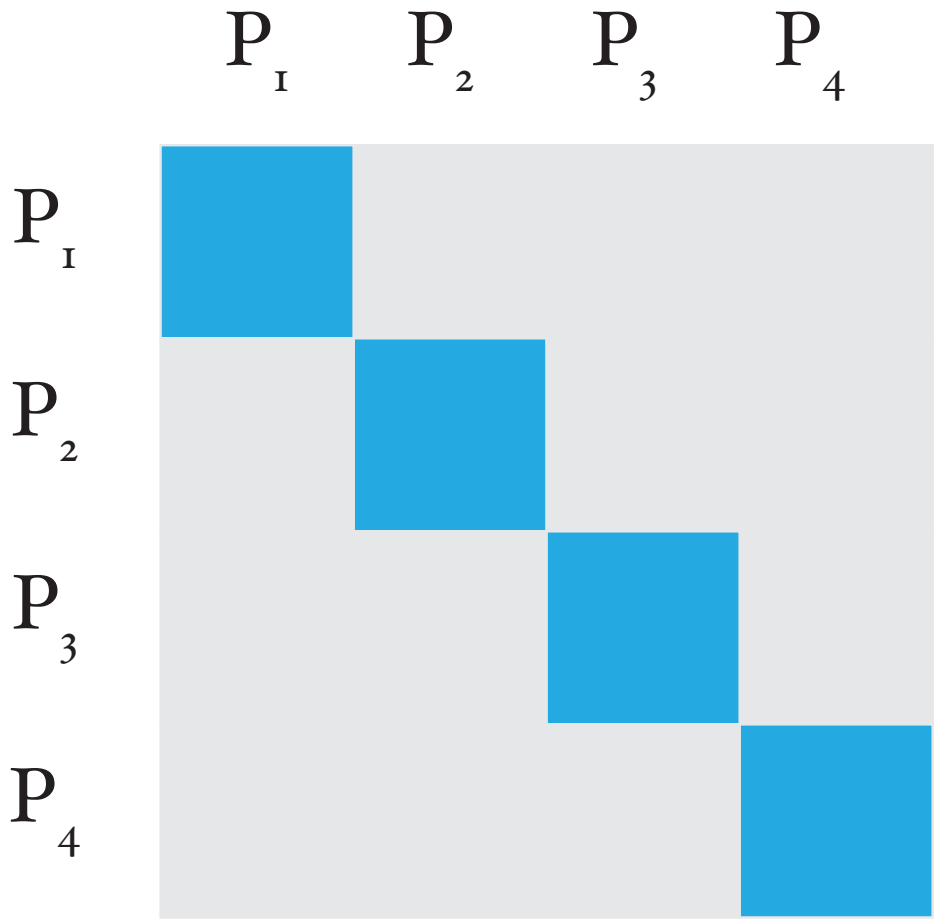
Hilbert's Program

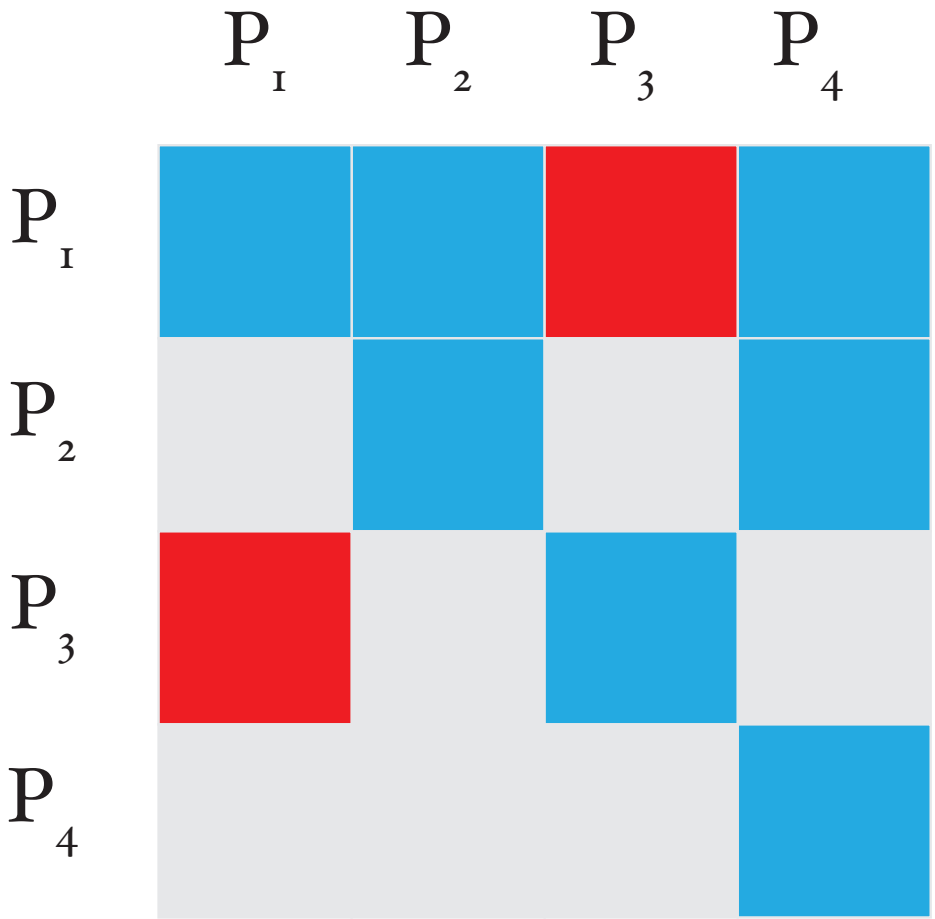
- Hilbert built on this accomplishment in setting the direction of mathematics in the 20th century:
- He believed we would use *formal logic* to answer all of the problems of mathematics
- "We must know—*we will know!*"

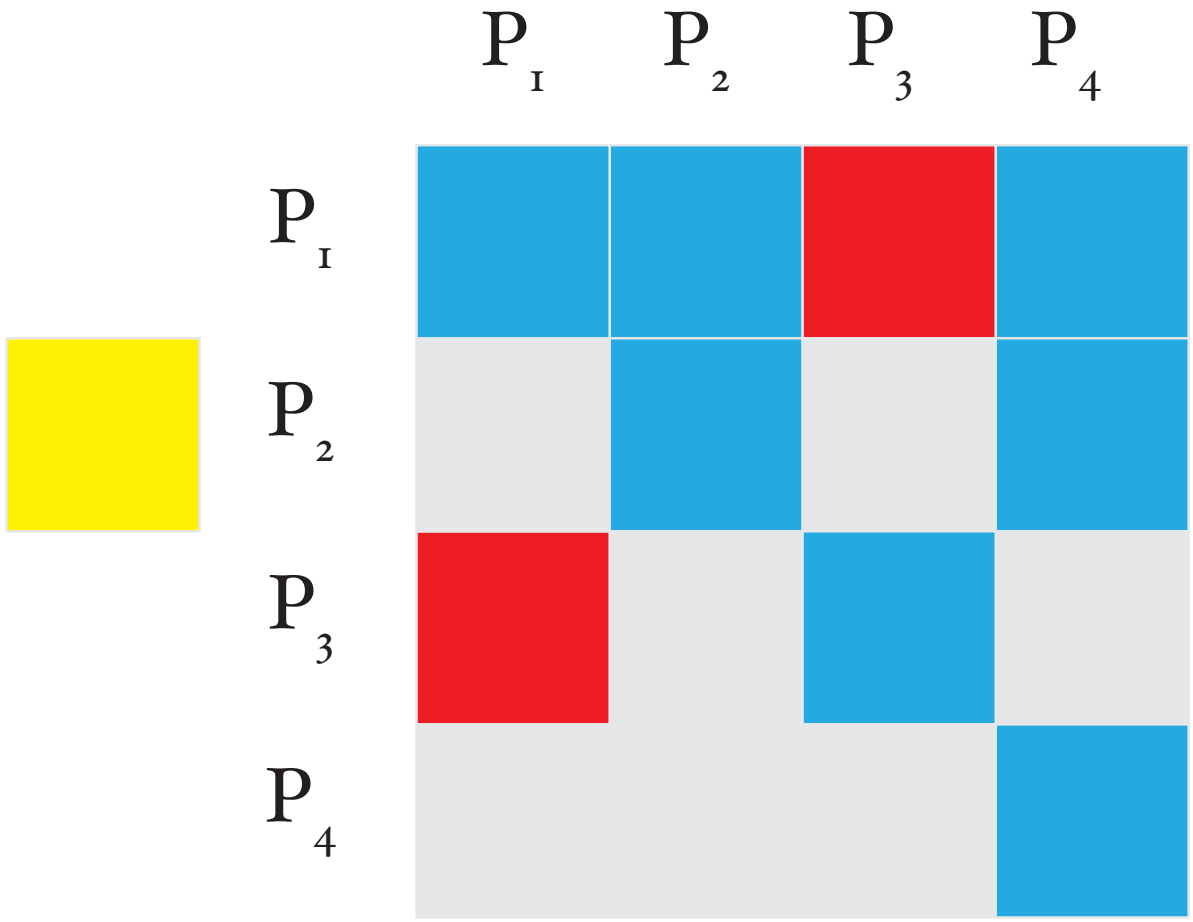
How does formal logic work?

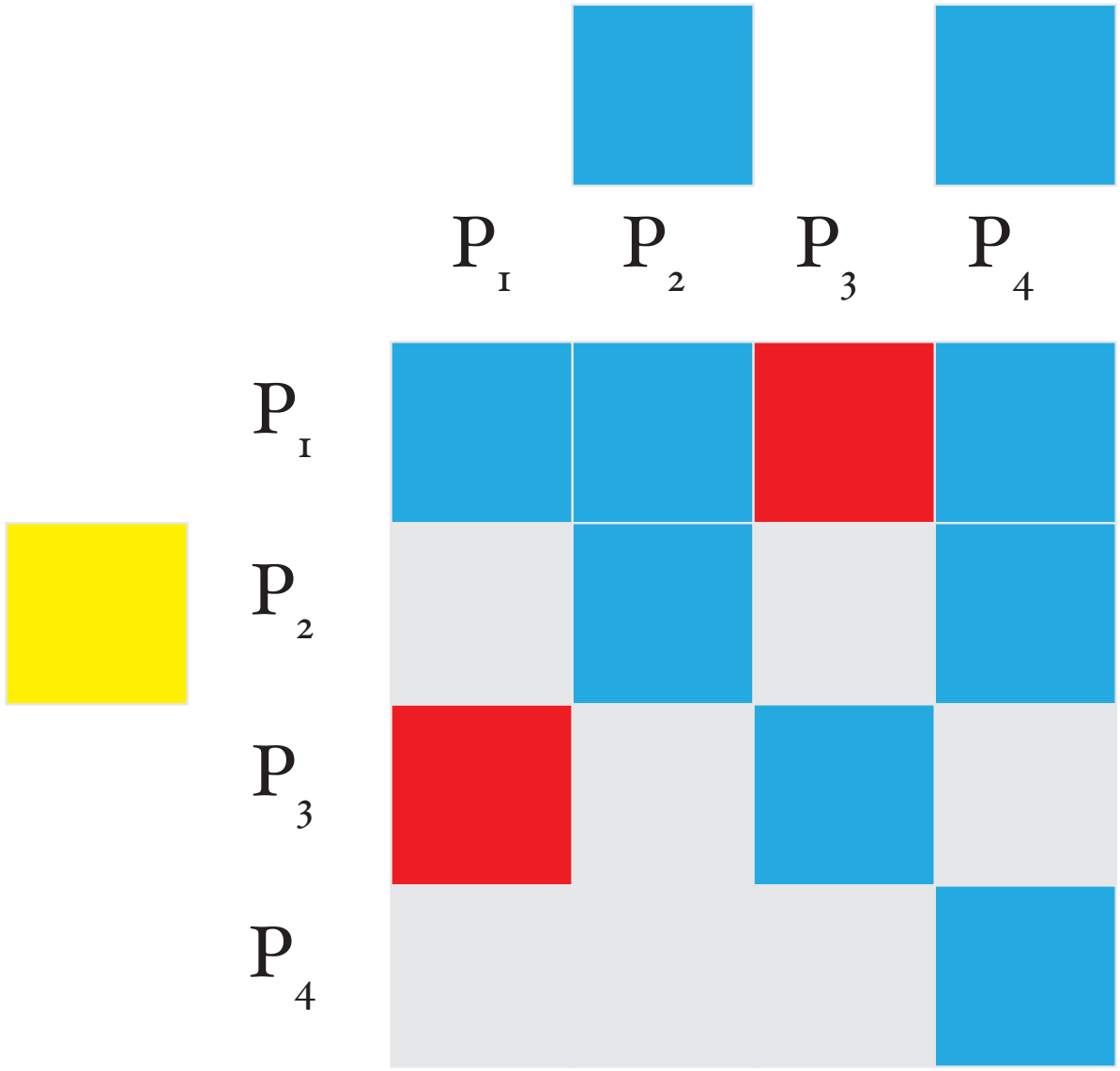
P_1 P_2 P_3 P_4

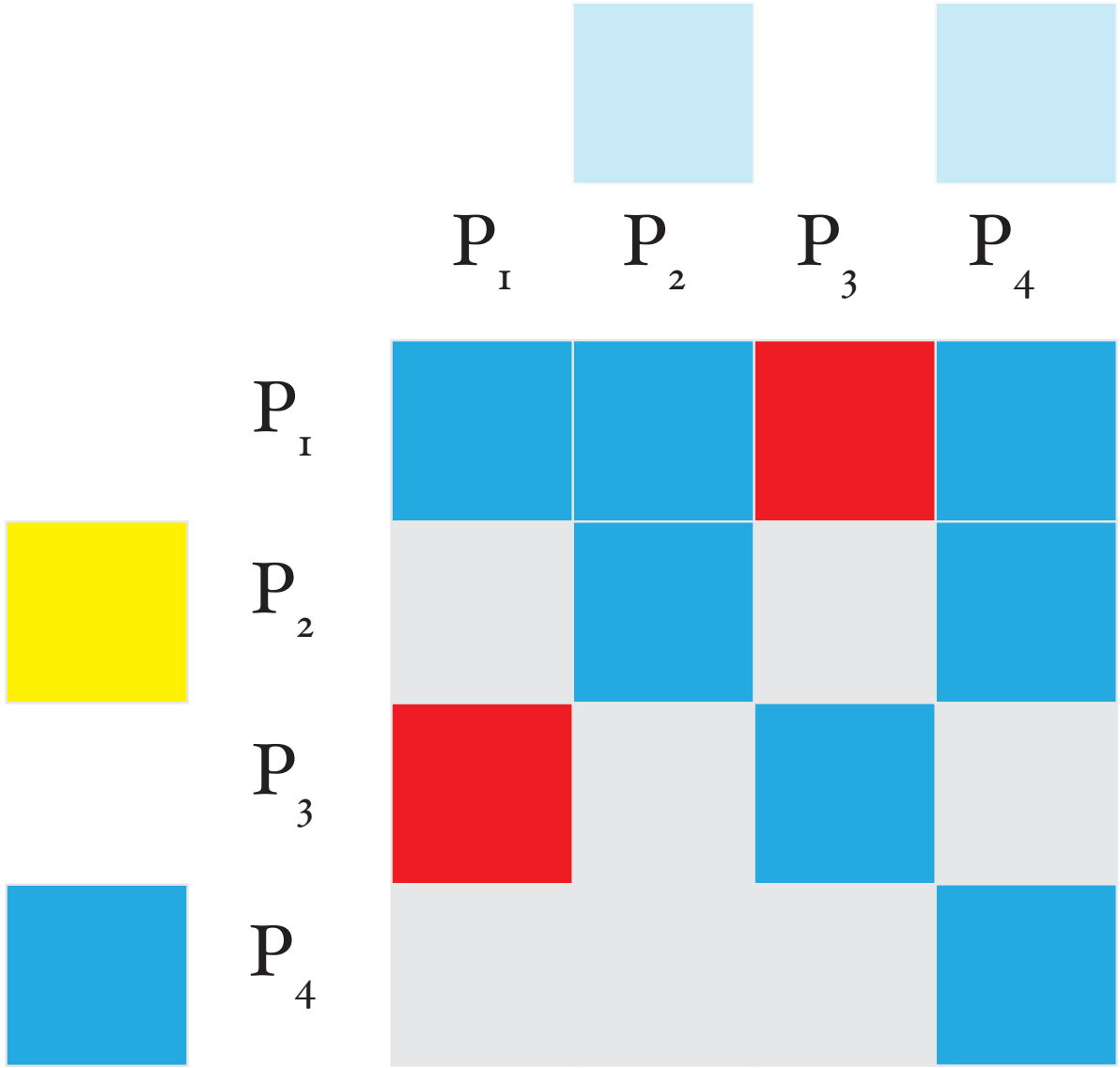

P1 P2 P3 P4
P1
P2
P3
P4

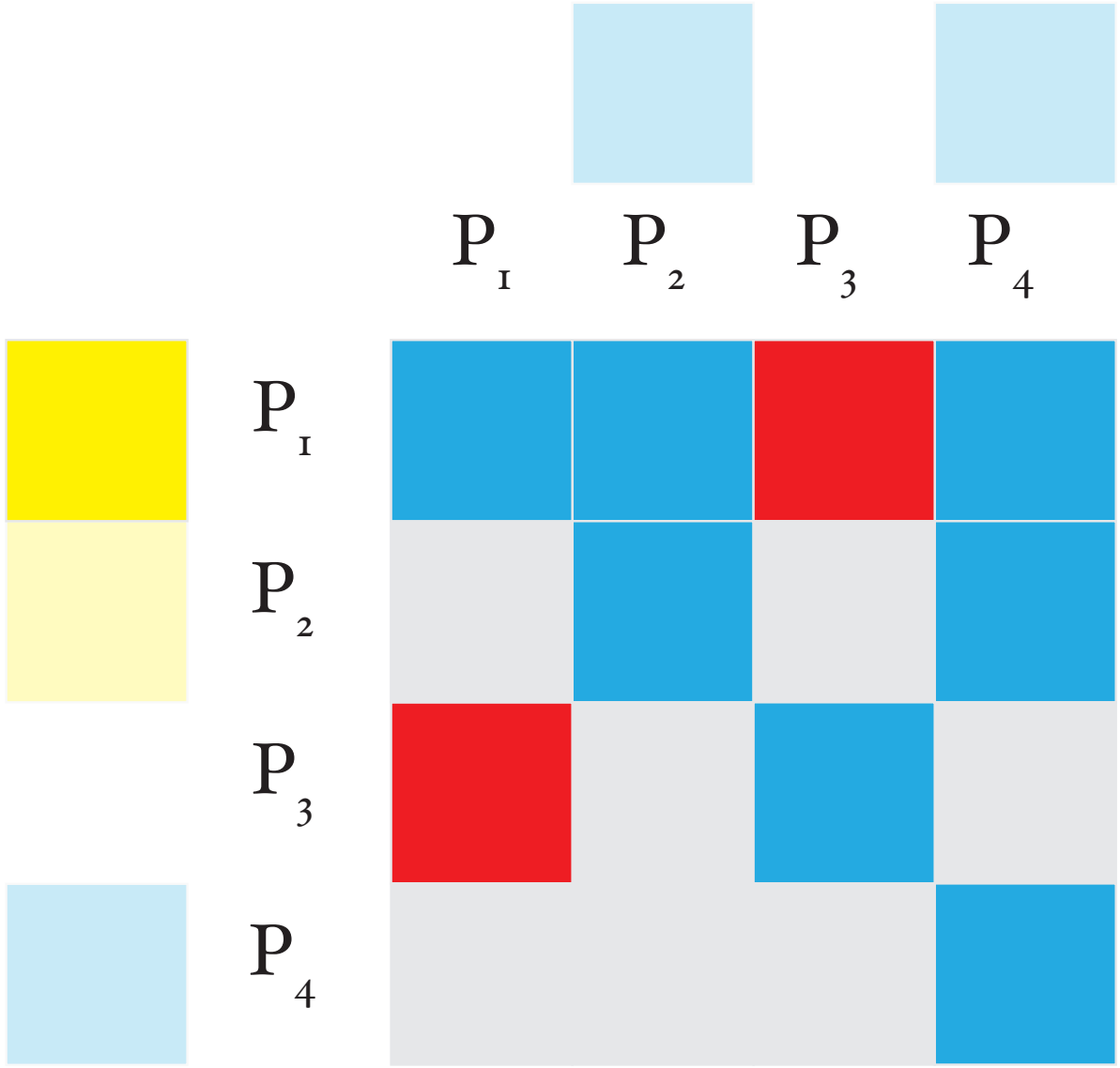

P1
P2
P3
P4
P1
P2
P3
P4

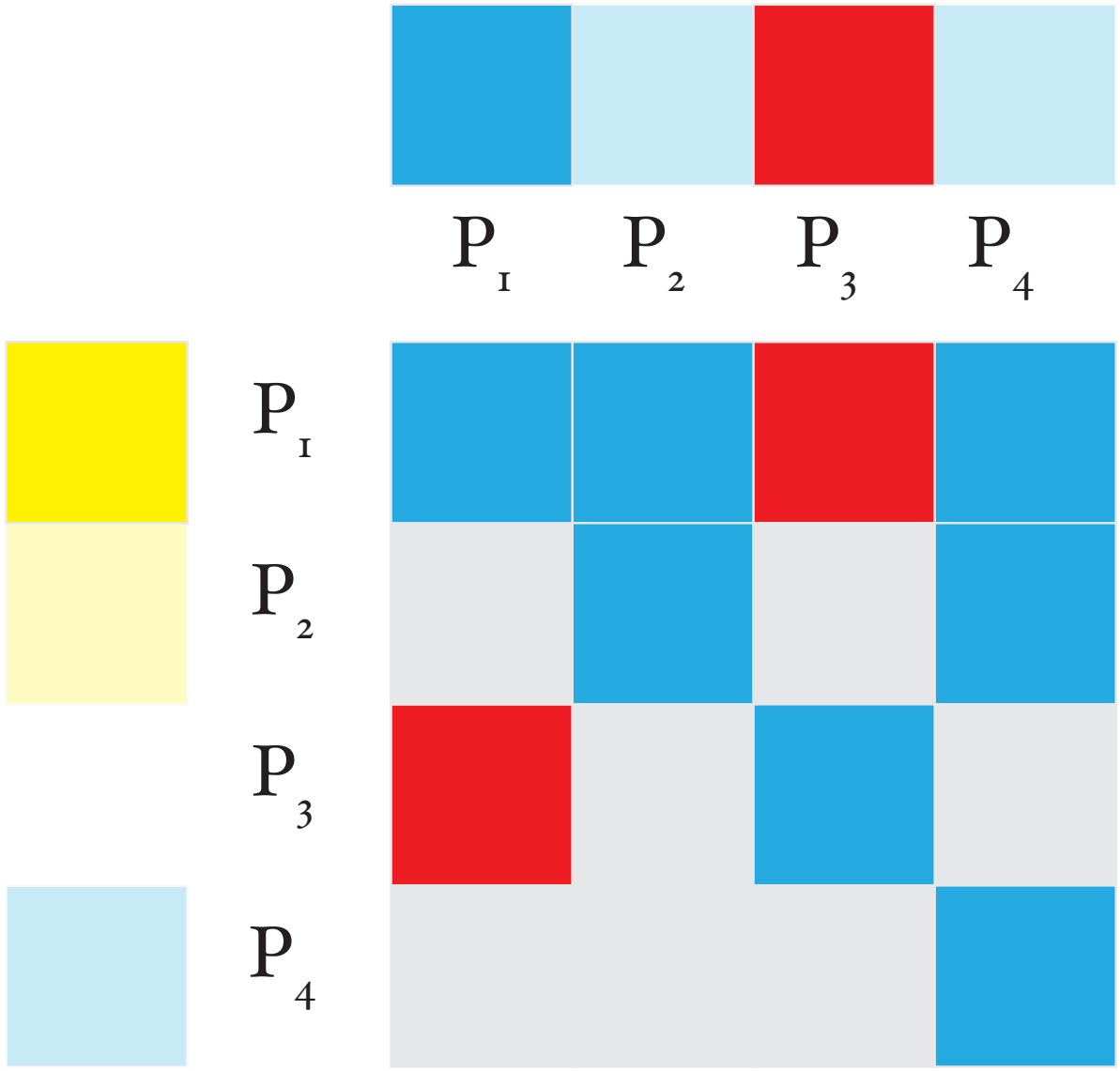

P1
P2
P3
P4
P1
P2
P3
P4

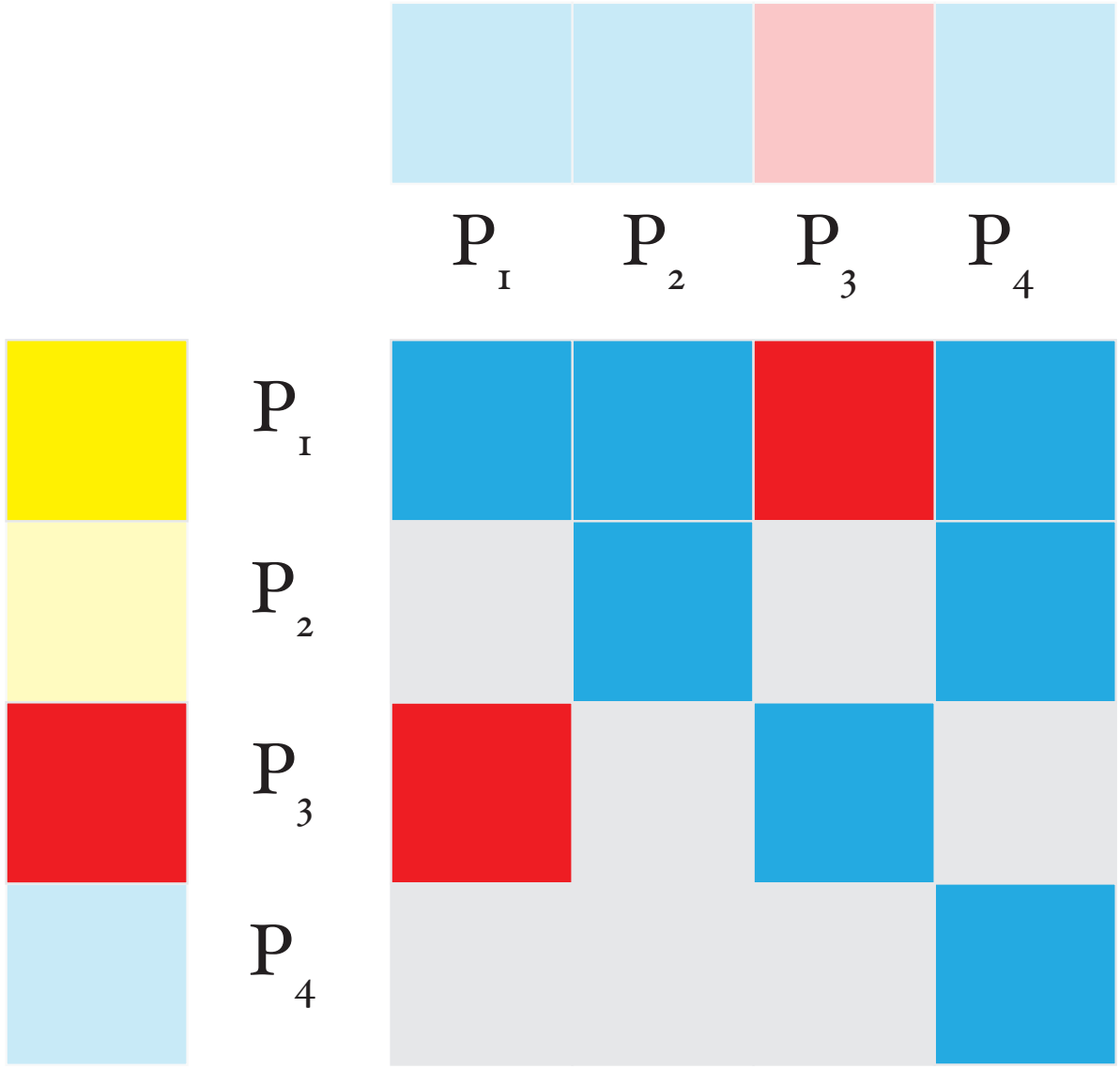

P1
P2
P3
P4
P1
P2
P3
P4

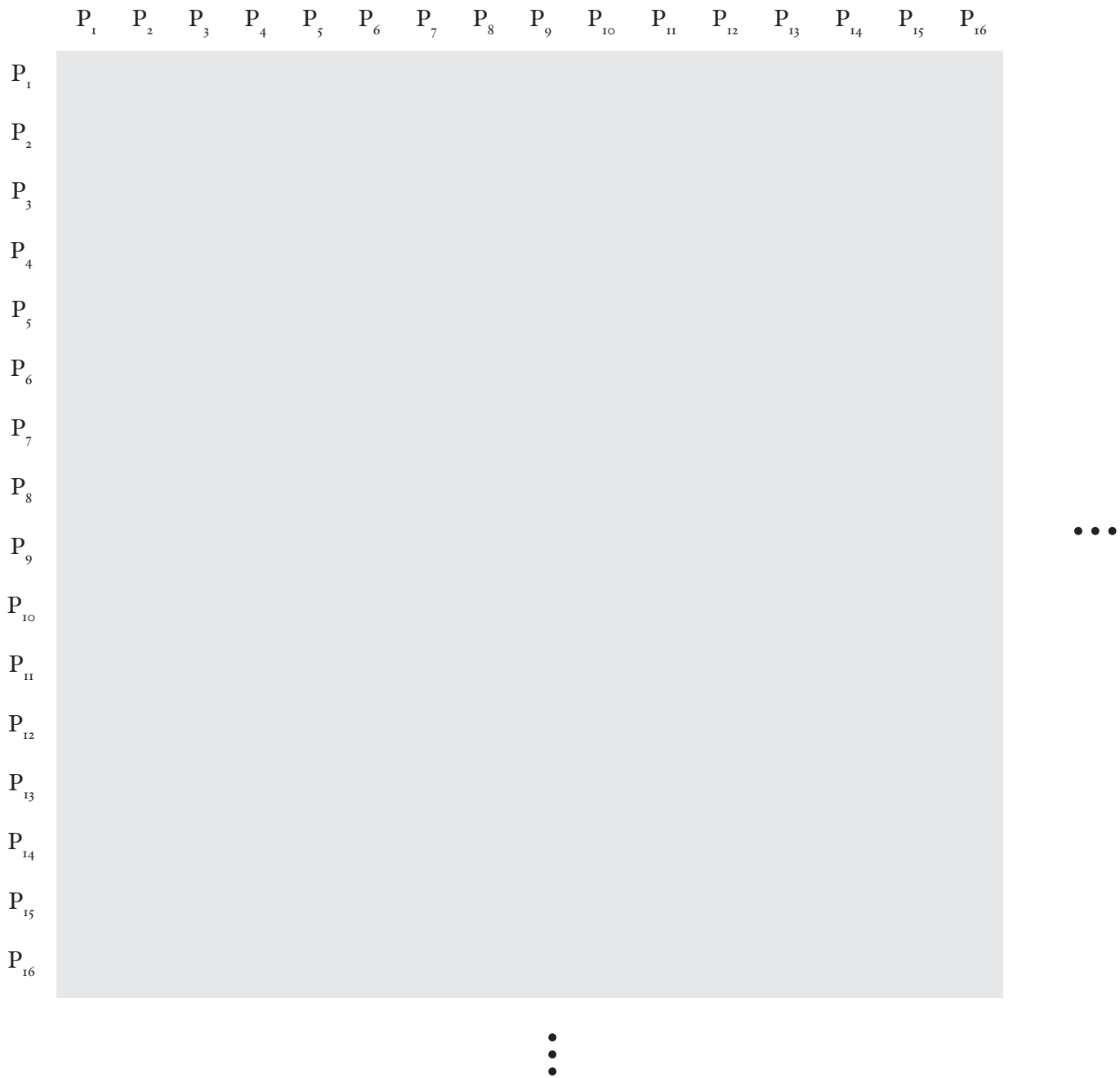

P1
P2
P3
P4
P1
P2
P3
P4

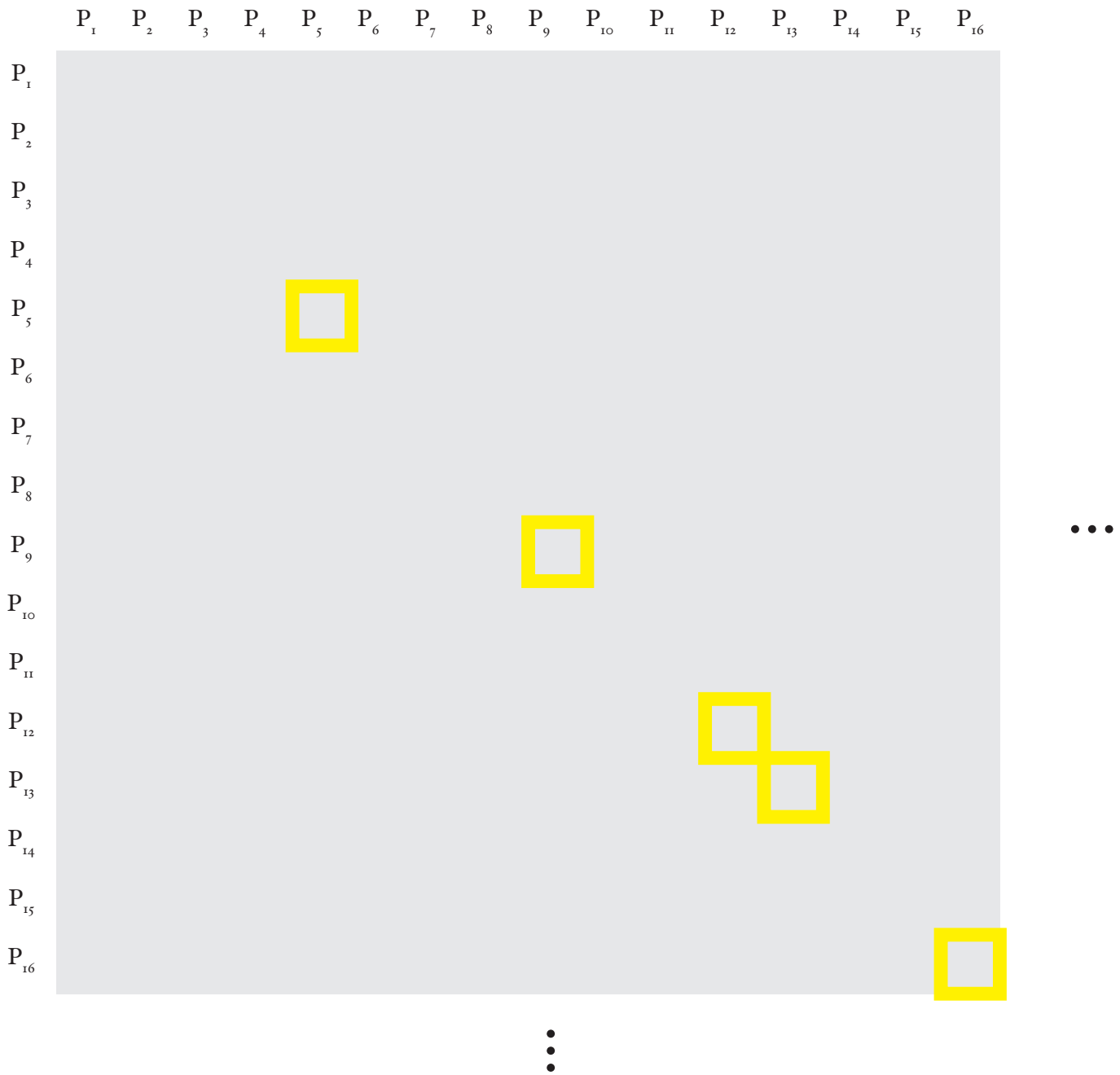

P1
P2
P3
P4
P1
P2
P3
P4

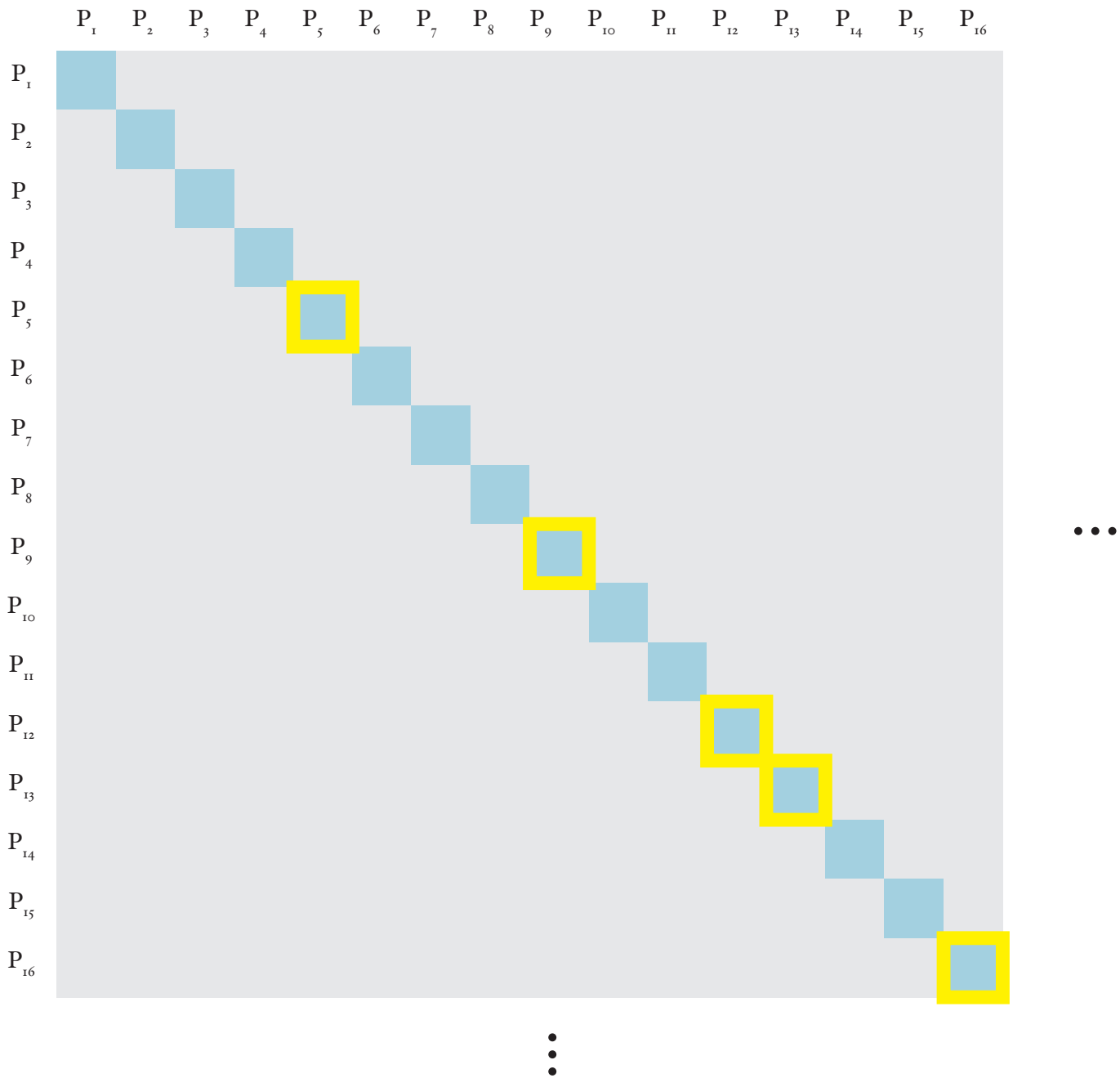

P1
P2
P3
P4
P1
P2
P3
P4

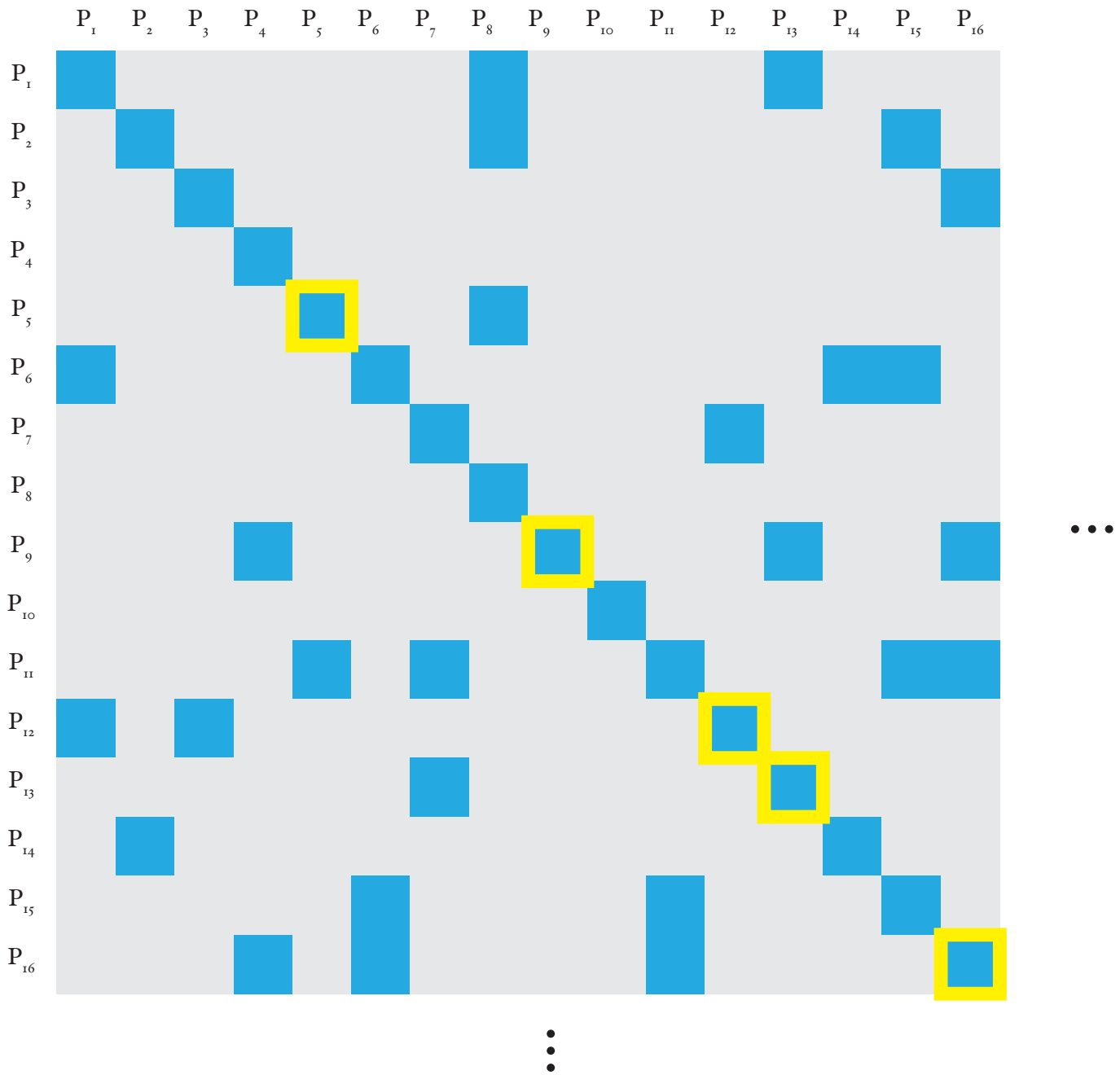

P1
P2
P3
P4
P1
P2
P3
P4

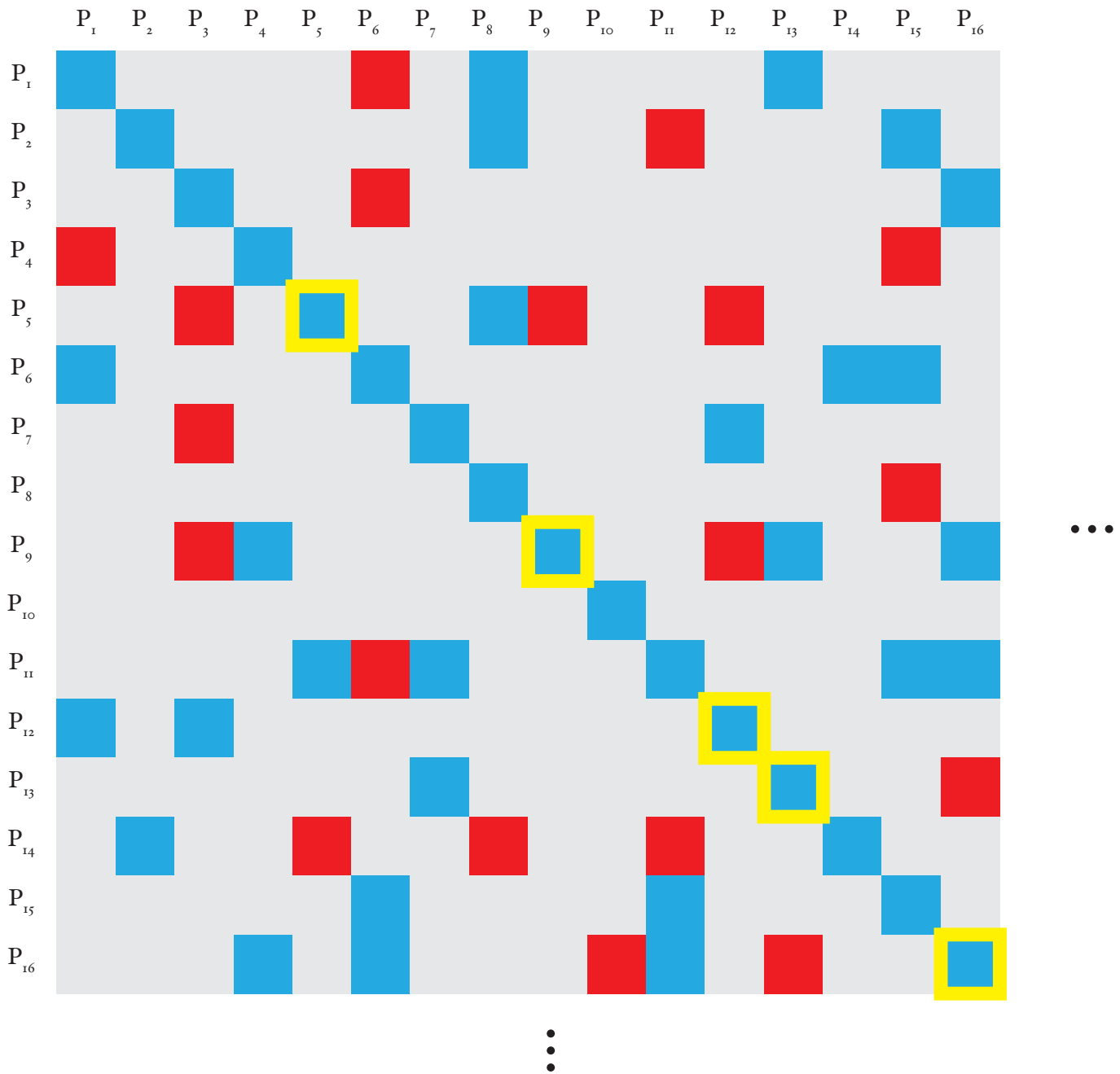

P1
P2
P3
P4
P1
P2
P3
P4

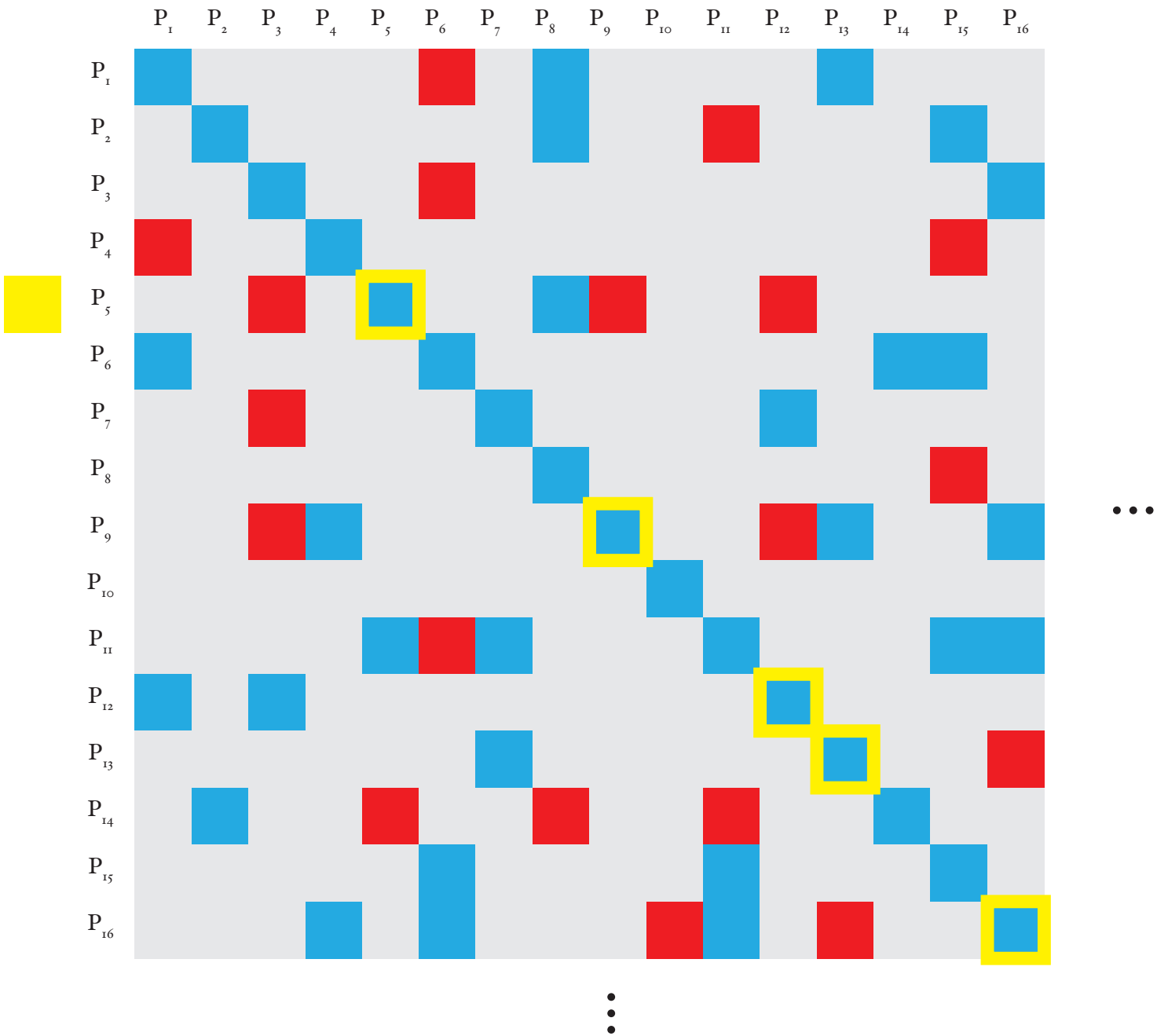

P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16

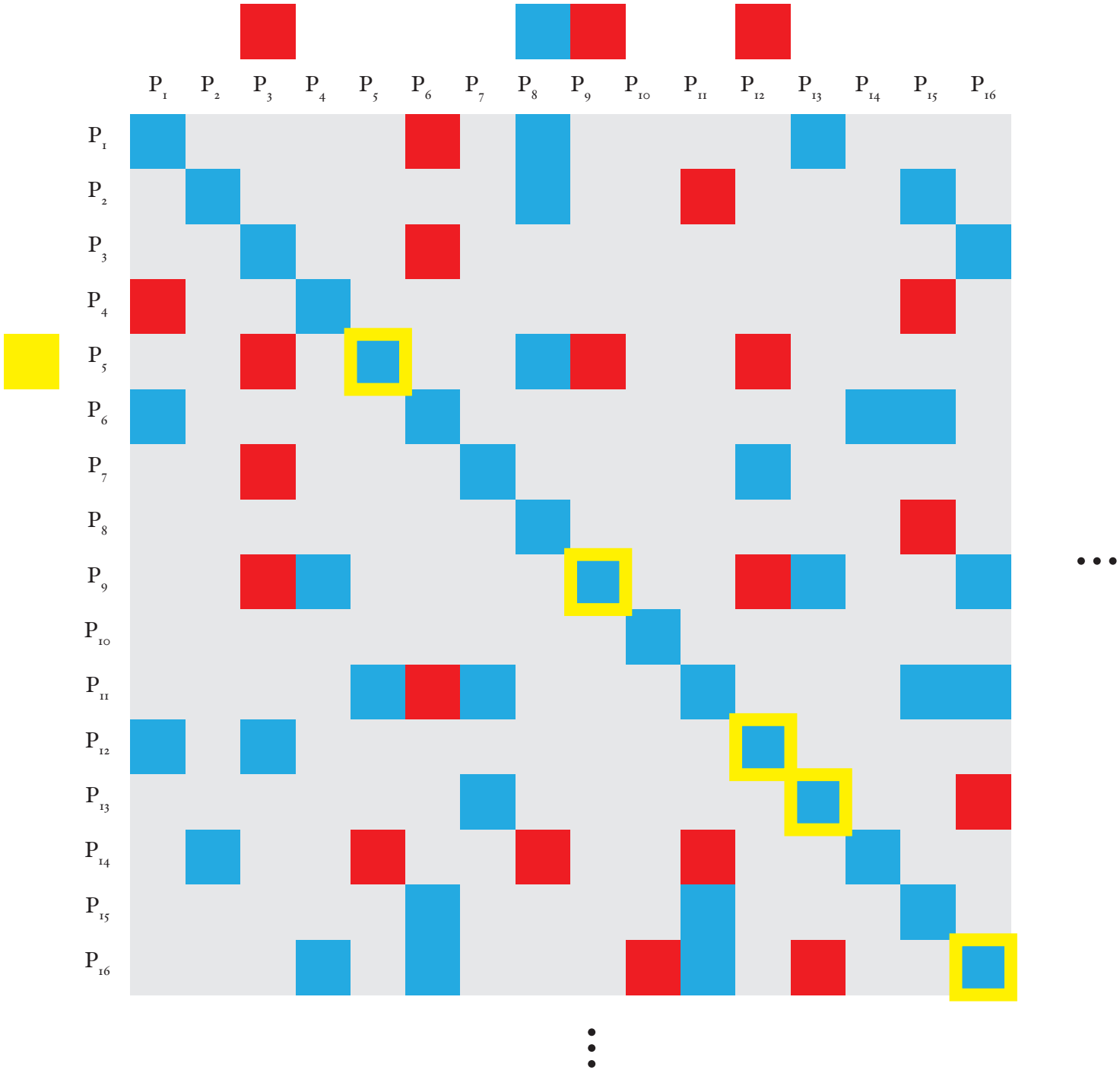

P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16

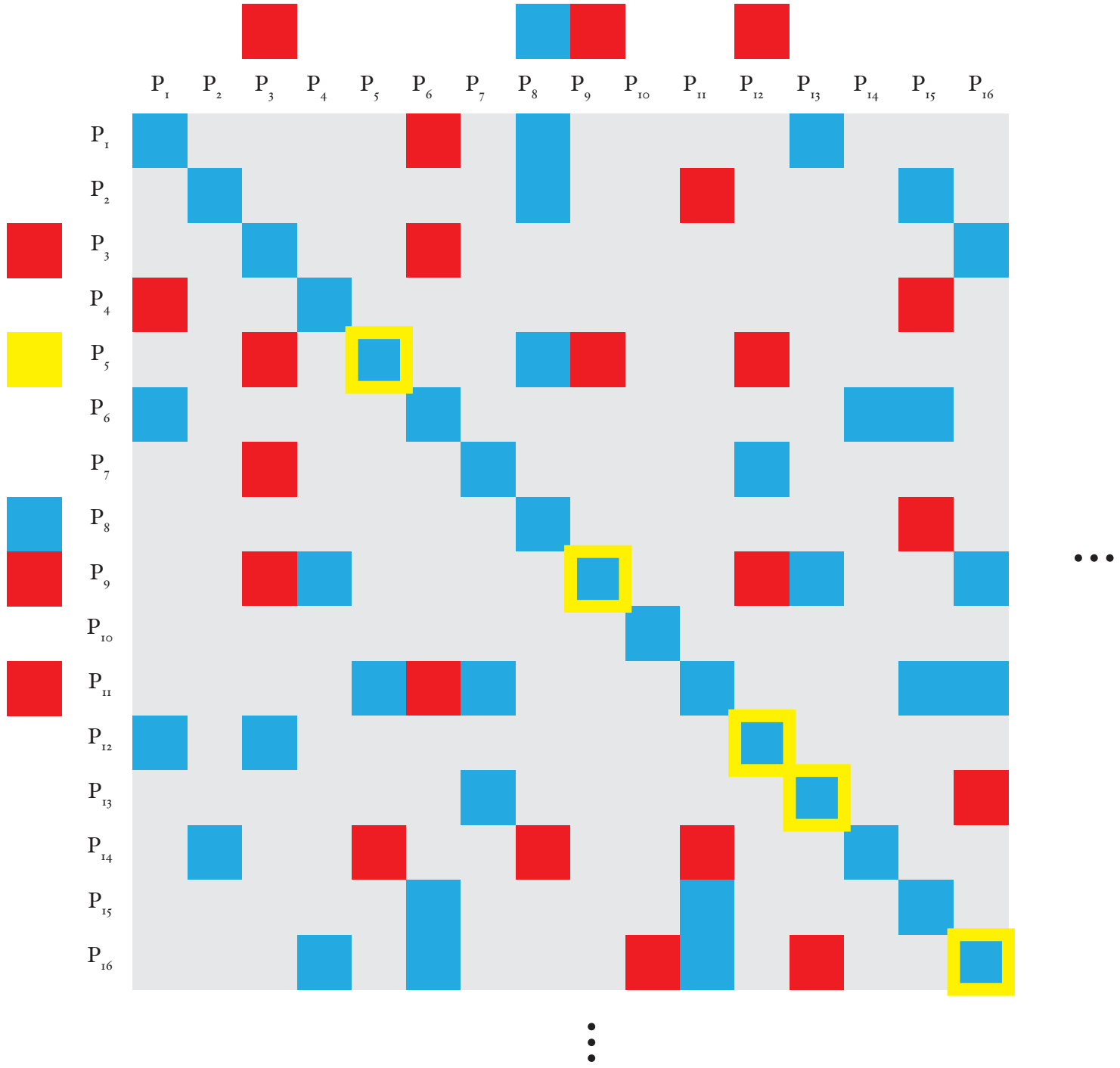

P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16

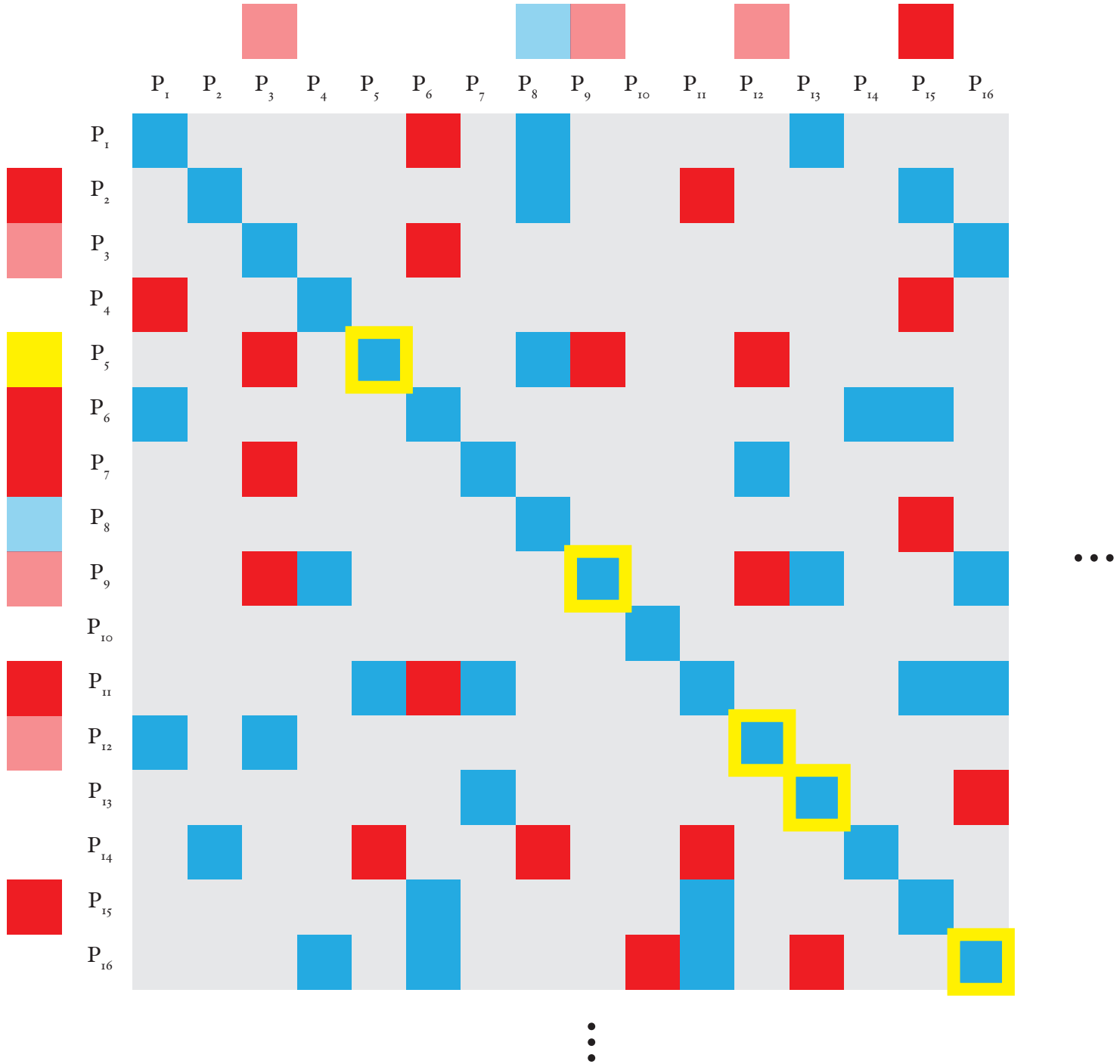

P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16

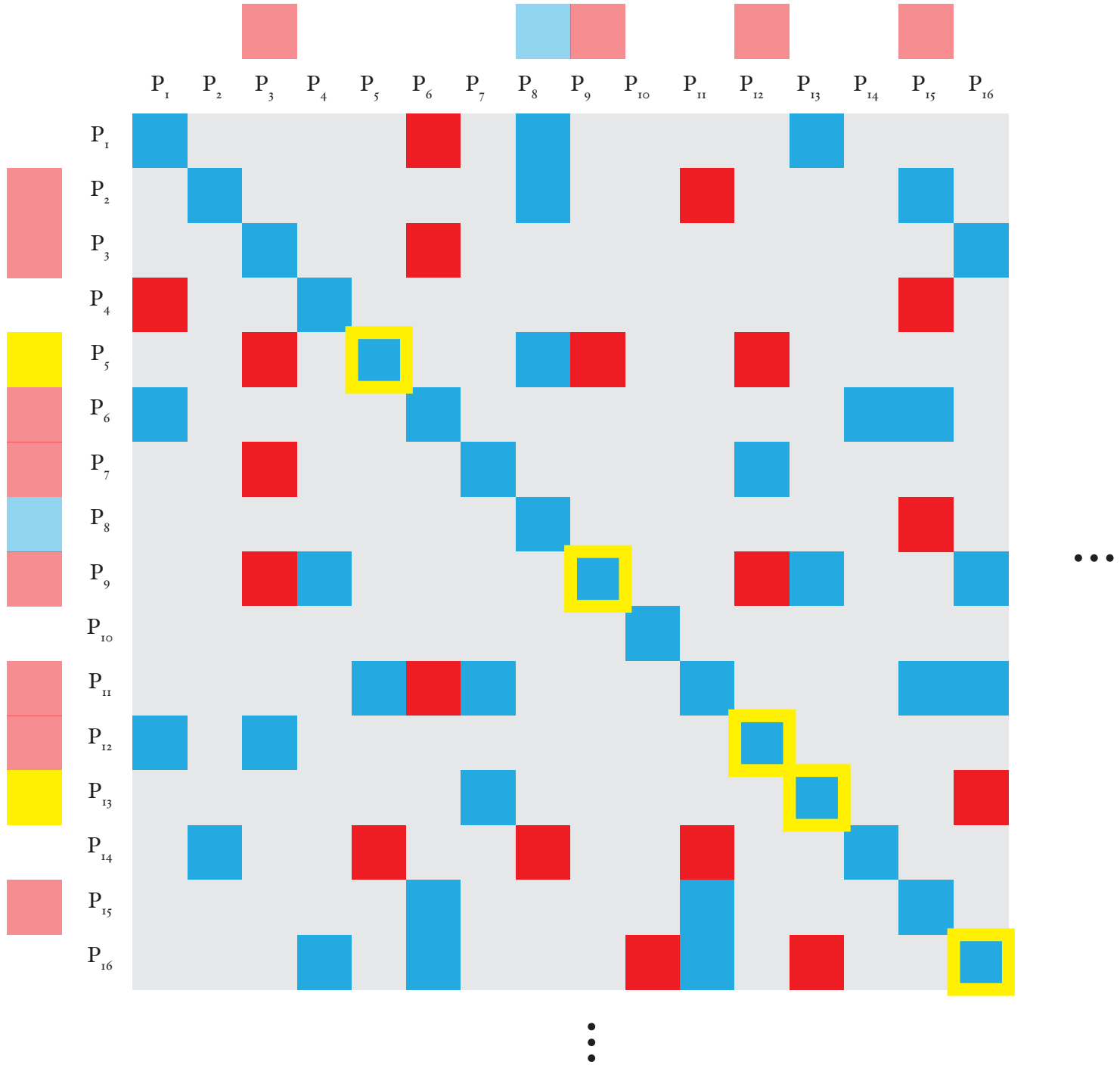

P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16

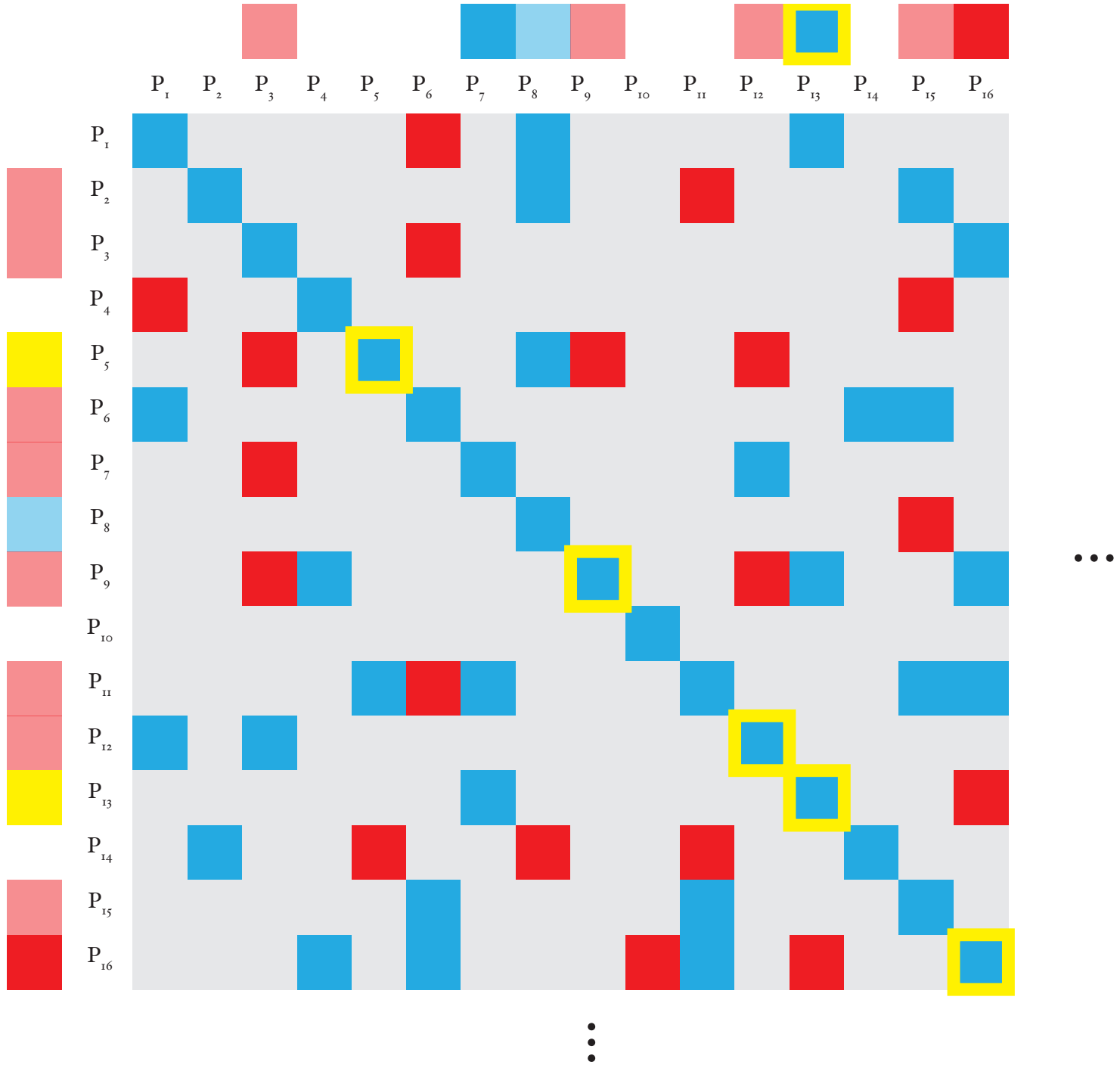

P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1
P2
P3
P4
P5
P6
P7
P8
P9
P10
P11
P12
P13
P14
P15
P16


P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16


P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16


P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16


P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16


P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16
P1 P2 P3 P4 P5 P6 P7 P8 P9 P10 P11 P12 P13 P14 P15 P16

Incompleteness!

- Despite Hilbert's optimism, the formal program was doomed

Incompleteness!

- Despite Hilbert's optimism, the formal program was doomed
- Church and Turing showed that some statements are undecidable (1936)

Incompleteness!

- Despite Hilbert's optimism, the formal program was doomed
- Church and Turing showed that some statements are undecidable (1936)
  - e.g. "will this program halt?"
- Gödel (1931) showed that our logics are always incomplete and must thus always be prepared to grow

Incompleteness!

- Despite Hilbert's optimism, the formal program was doomed
- Church and Turing showed that some statements are undecidable (1936)
  - e.g. "will this program halt?"
- Gödel (1931) showed that our logics are always incomplete and must thus always be prepared to grow
  - Not very easy to do when you assume you have enumerated all of the laws, all of the axioms!

How can we escape from the ruins of formal logic?

Well, we did it at the very beginning today.

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

Well, we did it at the very beginning today.

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \text{Odd}(n)$$

$$\text{decideParity} : (n : \mathbb{N}) \to \text{Even}(n) \ \vee \ \neg\text{Even}(n)$$

- Intuitionism never said that no properties are decidable

- Intuitionism never said that no properties are decidablethey're just not *automatically* decidable
- we must construct a mechanism for deciding a property

- Intuitionism never said that no properties are decidablethey're just not *automatically* decidable
- we must construct a mechanism for deciding a property
    - . . . an algorithm!

The big realization

- Brouwer's Intuitionism posits that mathematics must be *conceived of*, or *realized*
- Programs of a suitable form can be realizers of this logic
  - The types of these programs are propositions
  - The programs themselves are proofs
  - The act of construction a program is an *effort* of judgement
  - The act of checking its type is an *effort* of verification

Table of Contents

Martin-Löf's Type Theory

MLTT marries Intuitionistic construction to programming.

Martin-Löf's Type Theory

MLTT marries Intuitionistic construction to programming. It provides a toolbox for adding new realizations in a standard way.

- Program construction is proof construction
- Type definition (like Tree and $\mathbb{N}$) is "axiom introduction"

On the Meanings and the Justifications

So, finally, this is what the paper is about:

ON THE MEANINGS OF THE LOGICAL CONSTANTS AND THE JUSTIFICATIONS OF THE LOGICAL LAWS

PML is giving the rules and reasons by which one can make "logical" constructions—propositions and their proofs, types and their programs. He's building a programming language for intuitionistic logic.

On the Meanings and the Justifications

The next year PML publishes a manuscript documenting the very basics of a powerful dependent type theory that captures much of mathematics.

References I

Wadler (2014) "Propositions as Types"
http://homepages.inf.ed.ac.uk/wadler/papers/propositions-as-types/propositions-as-types.pdf

Per Martin-Lf (1971) "An intuitionistic theory of types"
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.131.926&rep=rep1&type=pdf

Per Martin-Lf (1982) "Constructive mathematics and computer programming" http://intuitionistic.files.wordpress.com/2010/07/martin-lof-computer.pdf

Per Martin-Lf (1984) "Intuitionistic Type Theory"
http://intuitionistic.files.wordpress.com/2010/07/martin-lof-tt.pdf

Per (1983) "On the Meanings of the Logical Constants and the Justifications of the Logical Laws"
http://www.pps.univ-paris-diderot.fr/~saurin/Enseignement/LMFI/articles/Martin-Lof83.pdf

References II

nCatLab "Martin-Lf dependent type theory" http://ncatlab.org/nlab/show/Martin-L%C3%B6f+dependent+type+theory

Stanford Encyclopedia of Philosophy (2014) "Intuitionistic Logic" http://plato.stanford.edu/entries/logic-intuitionistic/

Stanford Encyclopedia of Philosophy (2013) "Intuitionism" http://plato.stanford.edu/entries/intuitionism/

Pfenning and Garcia (2009) "Constructive Logic" (CMU Course Notes) http://www.cs.cmu.edu/~fp/courses/15317-f09/schedule.html

Oleg Kiselyov (2009) "Constructive Law of Excluded Middle" http://okmij.org/ftp/Computation/lem.html

Douglas Hofstadter (1979) "Gdel, Escher, Bach: an Eternal Golden Braid"

References III

Leslie (1999) "Just What is it that Makes Martin-Lfs Type Theory so Different, so Appealing?" http://www.cllc.vuw.ac.nz/talk-papers/whatisit.ps

Andrej Bauer (2008) "Intuitionistic mathematics for physicists" http://math.andrej.com/2008/08/13/intuitionistic-mathematics-for-physics/

Robert J Constable () "The Triumph of Types: Creating a Logic of Computational Reality" http://ncatlab.org/nlab/files/ConstableTriumphOfTypes.pdf

Bengt Nordstrom (2004) "Constructivism. A Computing Science perspective" http://www.cse.chalmers.se/~bengt/papers/vatican.pdf

Robert Harper, blog http://existentialtype.wordpress.com/2012/08/09/churchs-law/

Wikipedia "Paris-Harrington Theorem" http://en.wikipedia.org/wiki/Paris%E2%80%93Harrington_theorem