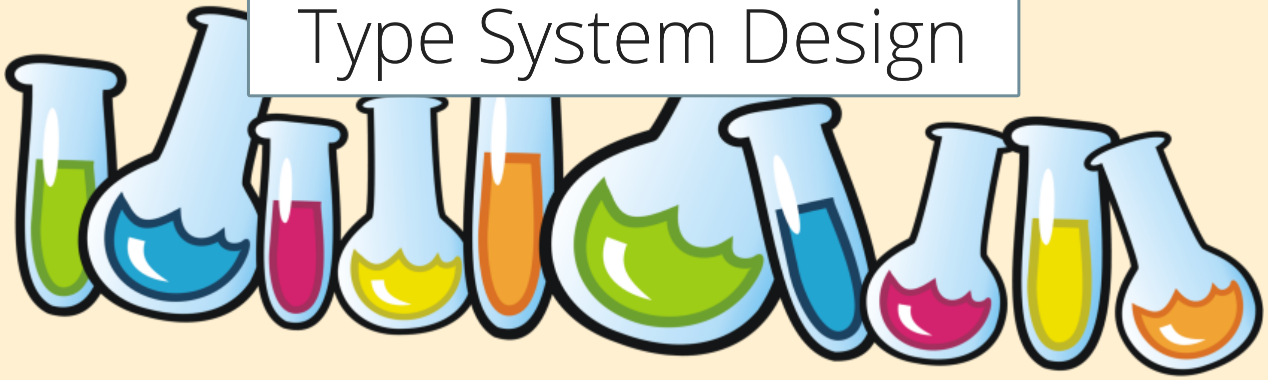
Toward a Science of Type System Design

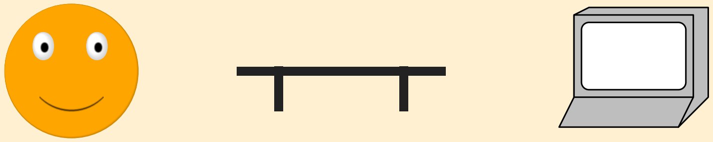


Ben Greenman

2024-08-26

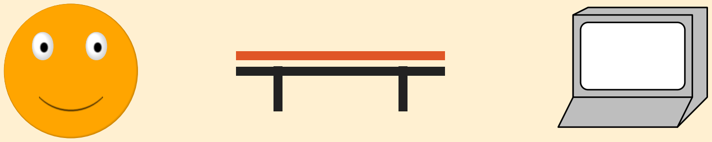
What is a **programming language**?

What is a **programming language**?



A **bridge** from ideas to computations

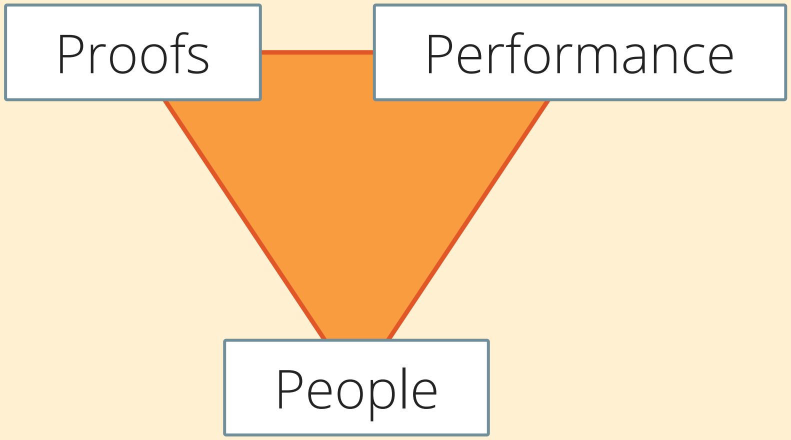
Types are guardrails



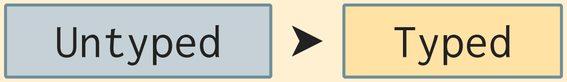
What is science?

What is science?

A search for truth …
via **conjectures** and **refutations**.

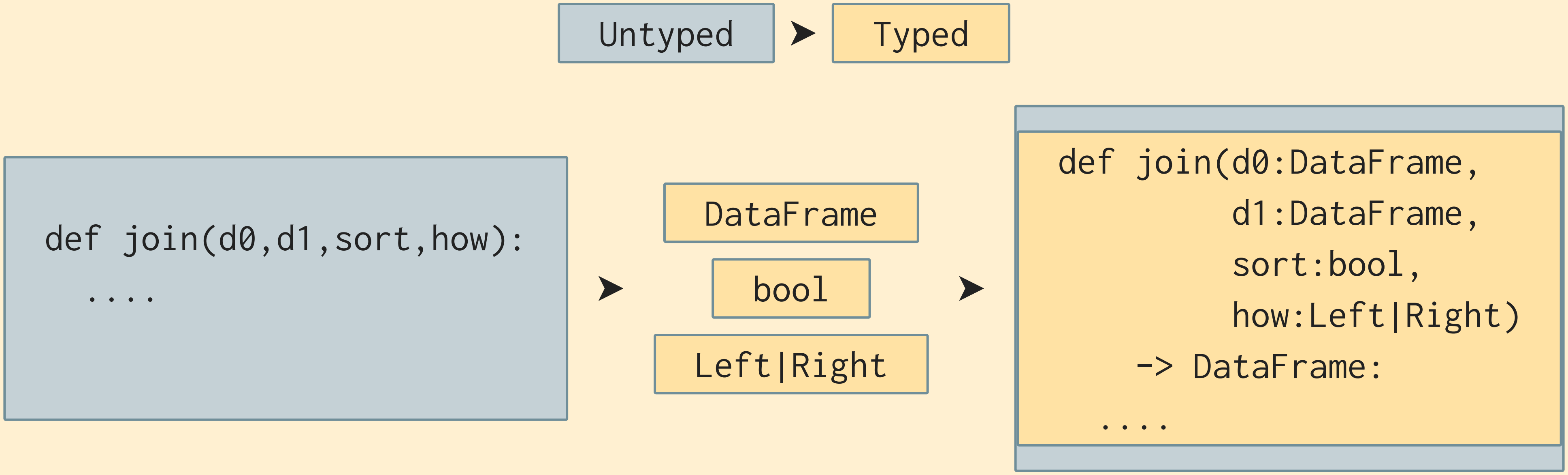

Proofs
Performance
People

Gradual Typing



Untyped ➤ Typed

Gradual Typing



Untyped ➤ Typed

```
def join(d0,d1,sort,how):
  ....
```

➤

DataFrame

bool

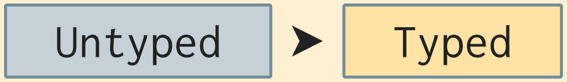
Left|Right

➤

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

Types where useful ... and nowhere else!

Gradual Typing



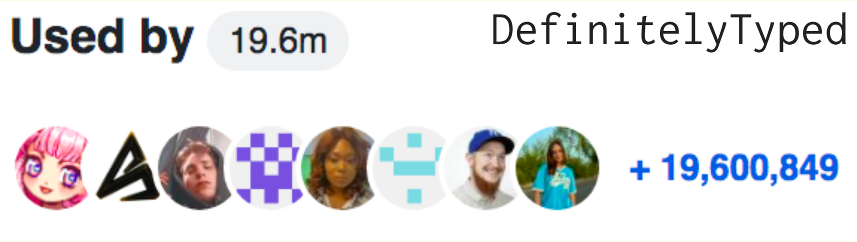
Untyped ➤ Typed

Gradual Typing

Untyped ➤ Typed

JS ➤ TS

TypeScript is **JavaScript with syntax for types.**



Used by 19.6m

DefinitelyTyped

+ 19,600,849

Problem: what do types mean?

Problem: what do types mean?

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

```
join("hello", ...)
```

Is **d0** really a data frame?

Problem: what do types mean?

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

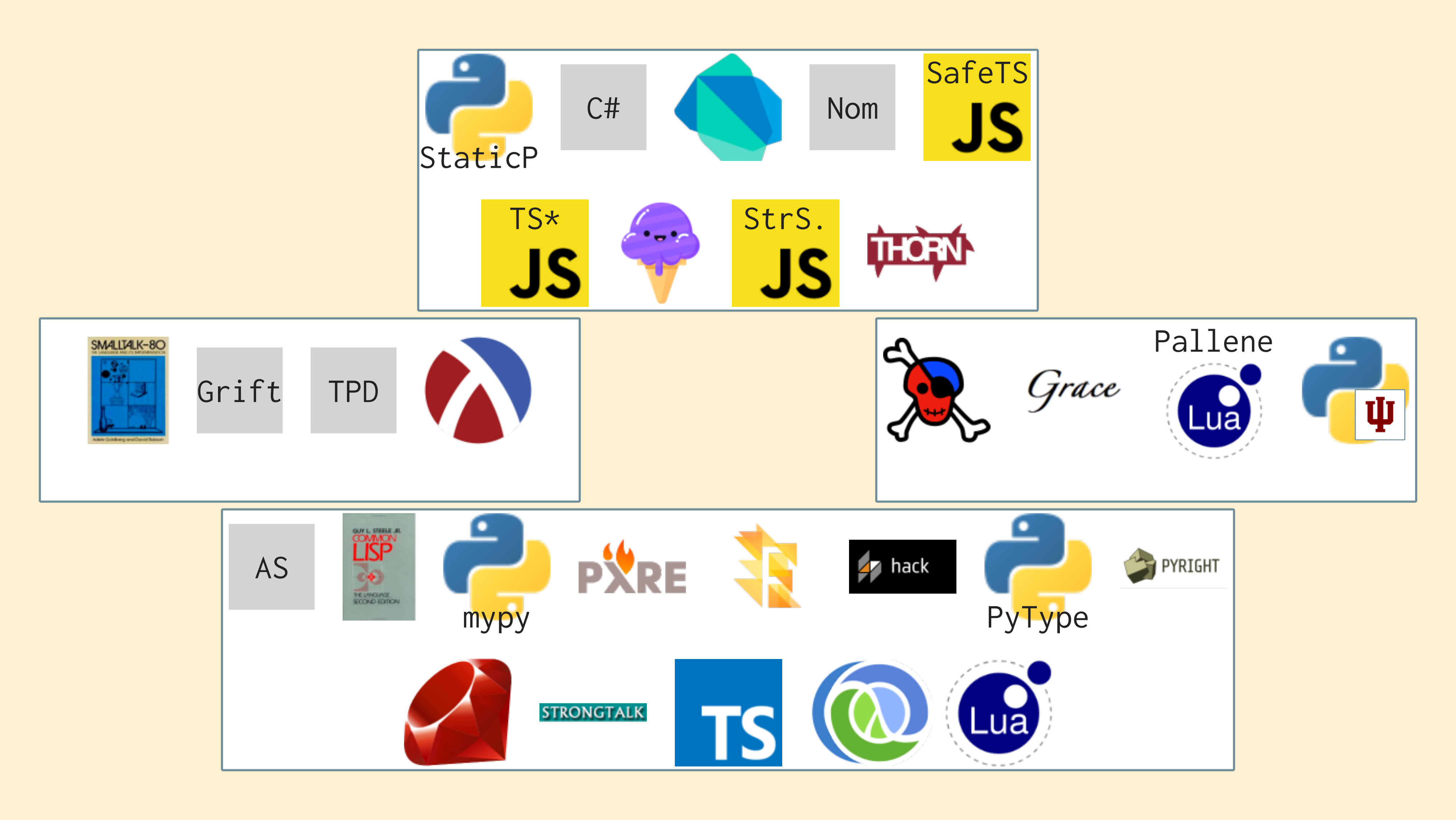
```
join("hello", ...)
```

Is **d0** really a data frame?

TS No.

AS
GUY L. STEELE JR.
COMMON LISP
THE LANGUAGE SECOND EDITION
mypy
hack
PYRE
PyType
PYRIGHT
STRONGTALK
TS
Lua
SMALLTALK-80
Grift
TPD
Grace
Lua
StaticP
C#
Nom
SafeTS
JS
TS*
JS
StrS.
JS
THORN

Many languages,
many meanings for types!



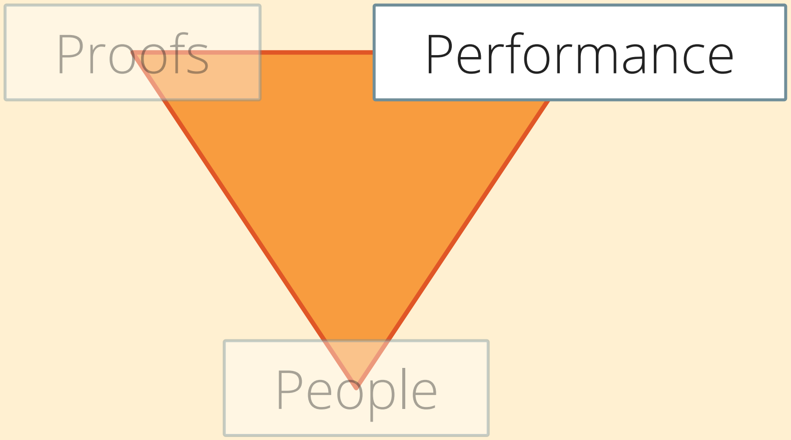
StaticP C# Nom SafeTS TS* StrS.

Grift TPD

Pallene

AS mypy PyType

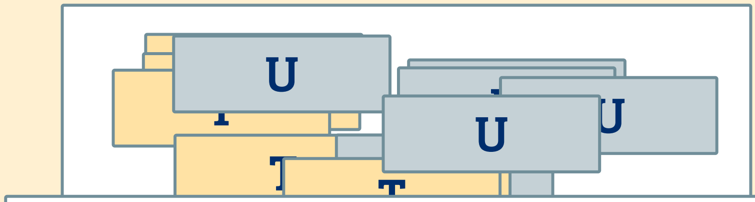
Where's the science?



Proofs

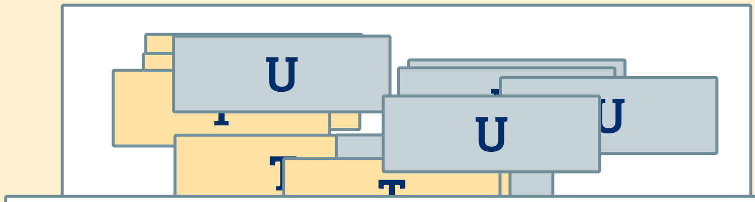
Performance

People



What do **reliable types** cost?

Typed Racket



What do **reliable types** cost?

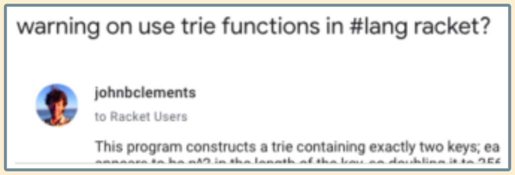
Typed Racket

Initial worst-case: 1.4x
ecoop'15

30x

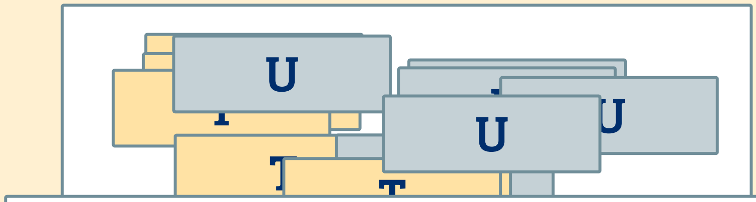
12,000x

(1ms to 12sec)


warning on use trie functions in #lang racket?

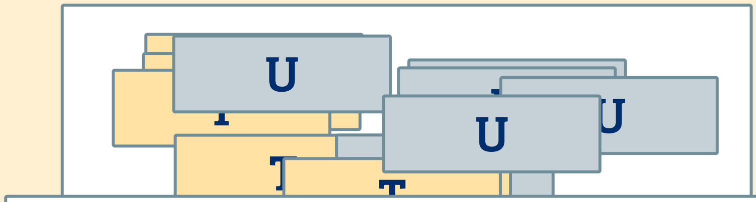
johnbclements
to Racket Users

This program constructs a trie containing exactly two keys; ea
appears to be n^2 in the length of the key, so doubling it to 256



What do **reliable types** cost?

Typed Racket



What do **reliable types** cost?

Typed Racket

Need a way to measure!

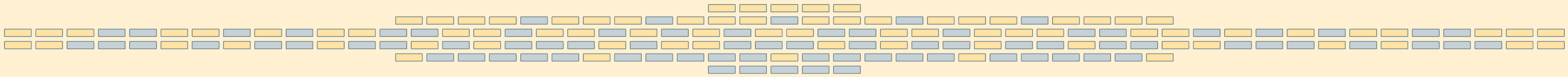
1. Start with a program

```
def join(d0,d1,sort,how):
  ....
```

2. Add full types

```
def join(d0:DataFrame,
         d1:DataFrame,
         sort:bool,
         how:Left|Right)
    -> DataFrame:
  ....
```

3. Explore all configurations



Current status: 21 benchmarks, +40k configurations

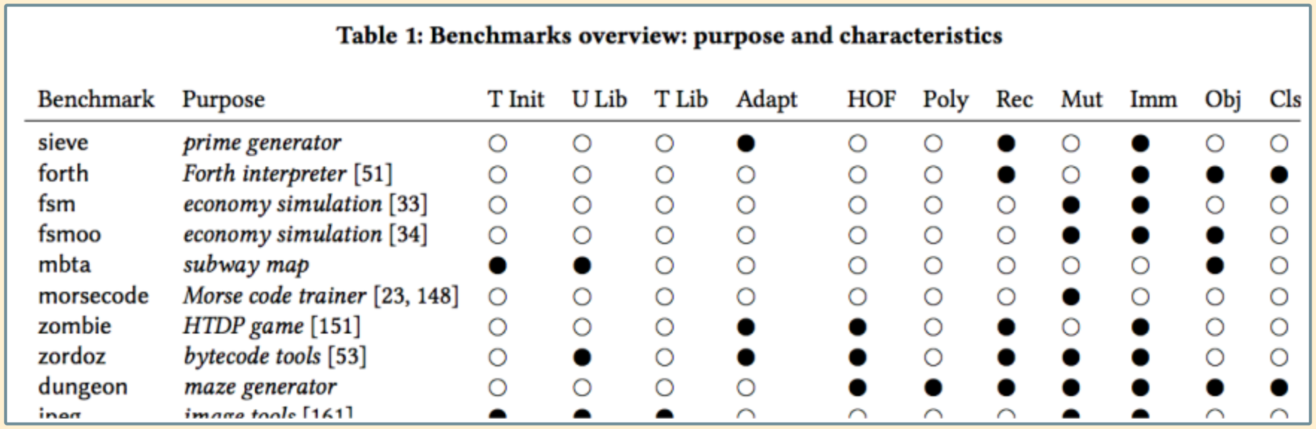
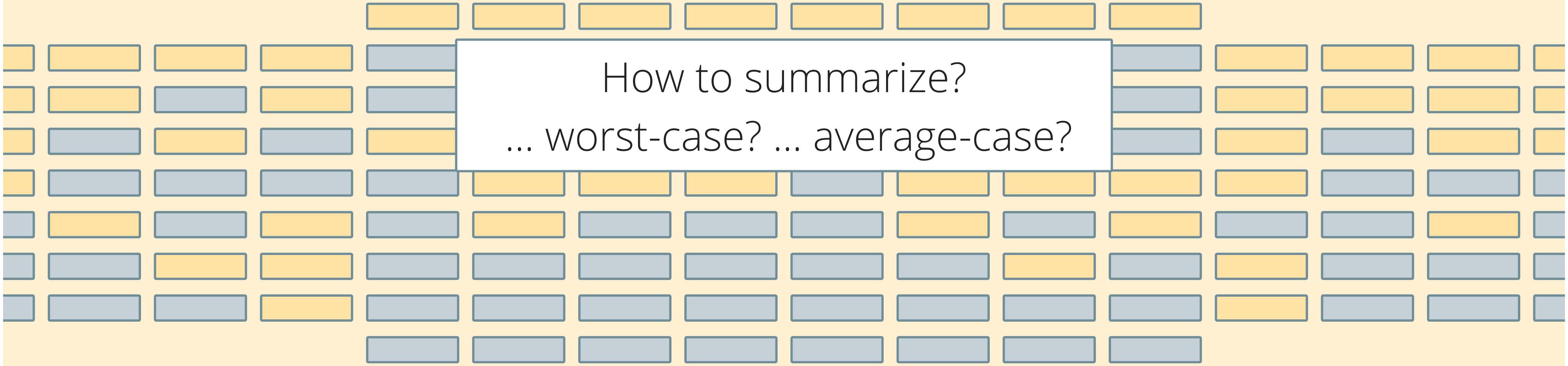


Table 1: Benchmarks overview: purpose and characteristics

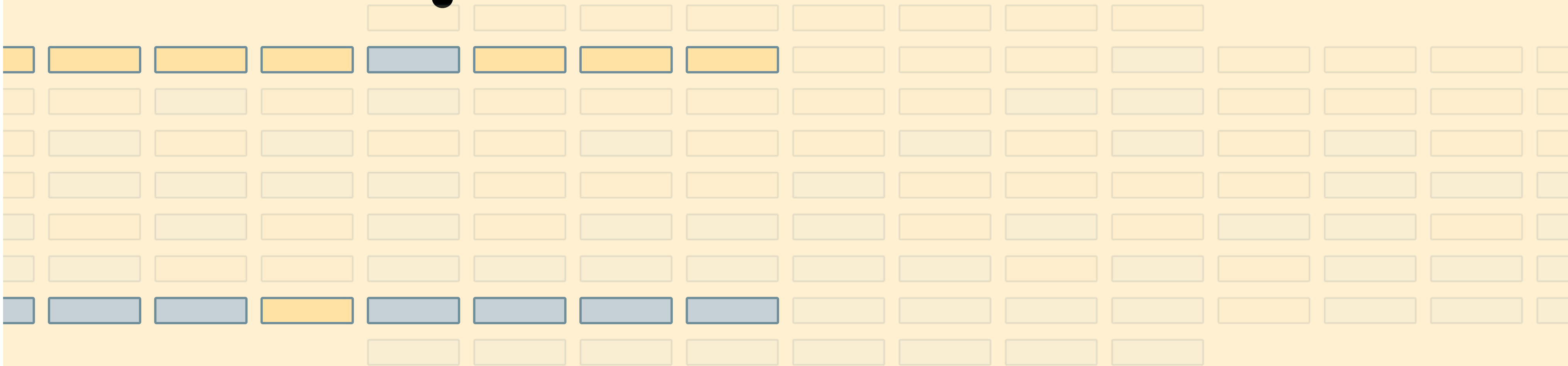
Benchmark	Purpose	T Init	U Lib	T Lib	Adapt	HOF	Poly	Rec	Mut	Imm	Obj	Cls
sieve	*prime generator*	○	○	○	●	○	○	●	○	●	○	○
forth	*Forth interpreter* [51]	○	○	○	○	○	○	●	○	●	●	●
fsm	*economy simulation* [33]	○	○	○	○	○	○	○	●	●	○	○
fsmoo	*economy simulation* [34]	○	○	○	○	○	○	○	●	●	●	○
mbta	*subway map*	●	●	○	○	○	○	○	○	○	●	○
morsecode	*Morse code trainer* [23, 148]	○	○	○	○	○	○	○	●	○	○	○
zombie	*HTDP game* [151]	○	○	○	●	●	○	●	○	●	○	○
zordoz	*bytecode tools* [53]	○	●	○	●	●	○	●	●	●	○	○
dungeon	*maze generator*	○	○	○	○	●	●	●	●	●	●	●
jpeg	*image tools* [161]	●	●	●	○	○	○	○	●	●	○	○



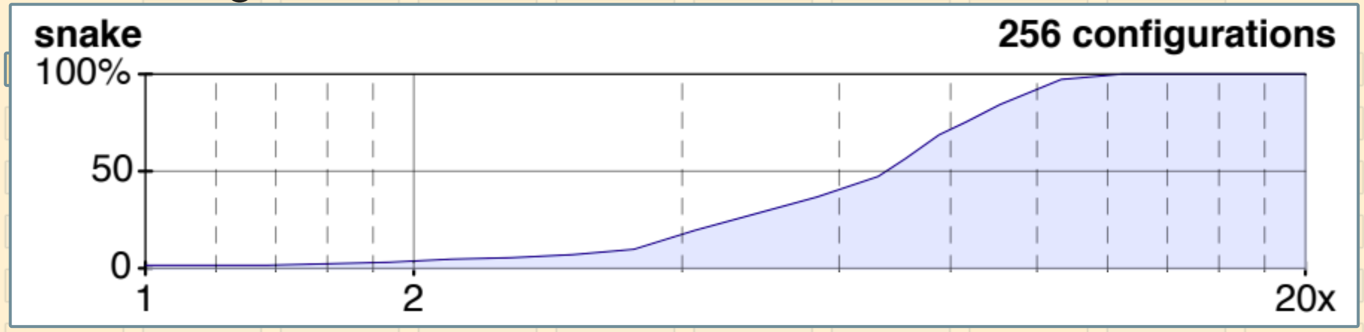
How to summarize?

… worst-case? … average-case?

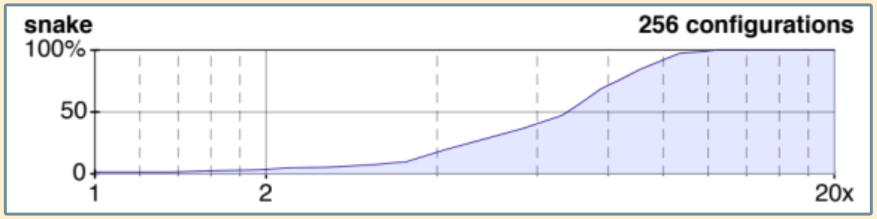
Key insight: **too slow** = useless!

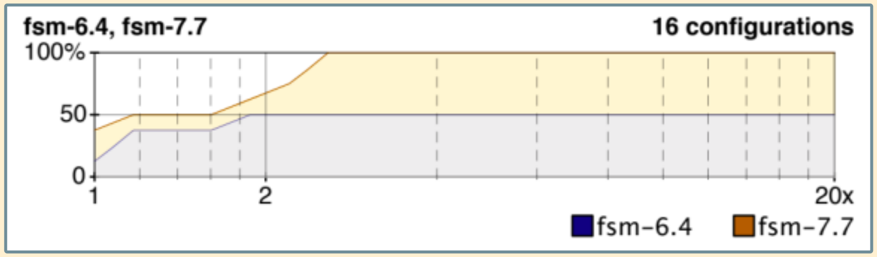


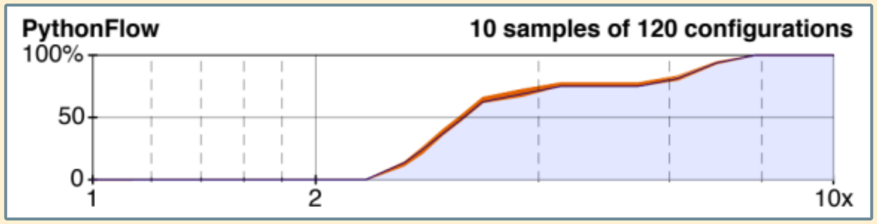
Key insight: **too slow** = useless!



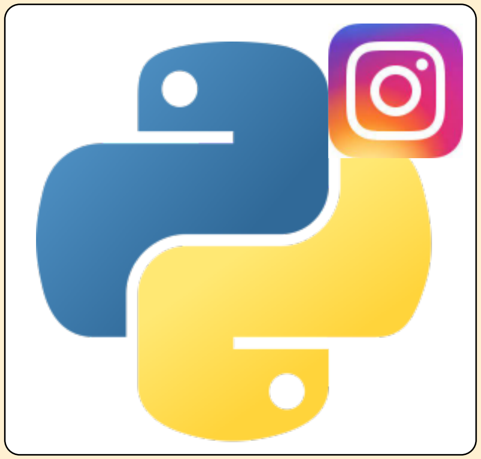
x-axis = limit for **too slow** vs. untyped code (log scale)
y-axis = % useful configs.


snake
256 configurations
100%
50
0
1
2
20x


fsm-6.4, fsm-7.7
16 configurations
100%
50
0
1
2
20x
fsm-6.4
fsm-7.7


PythonFlow
10 samples of 120 configurations
100%
50
0
1
2
10x

Static Python



Reliable types

Speeding up Instagram

Fast in general?

Challenge: 3 levels of types

```
lst
```

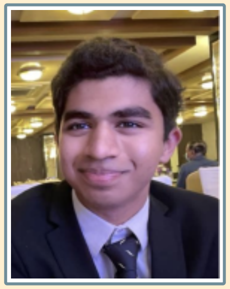
```
lst: List
```

```
lst: List[int]
```

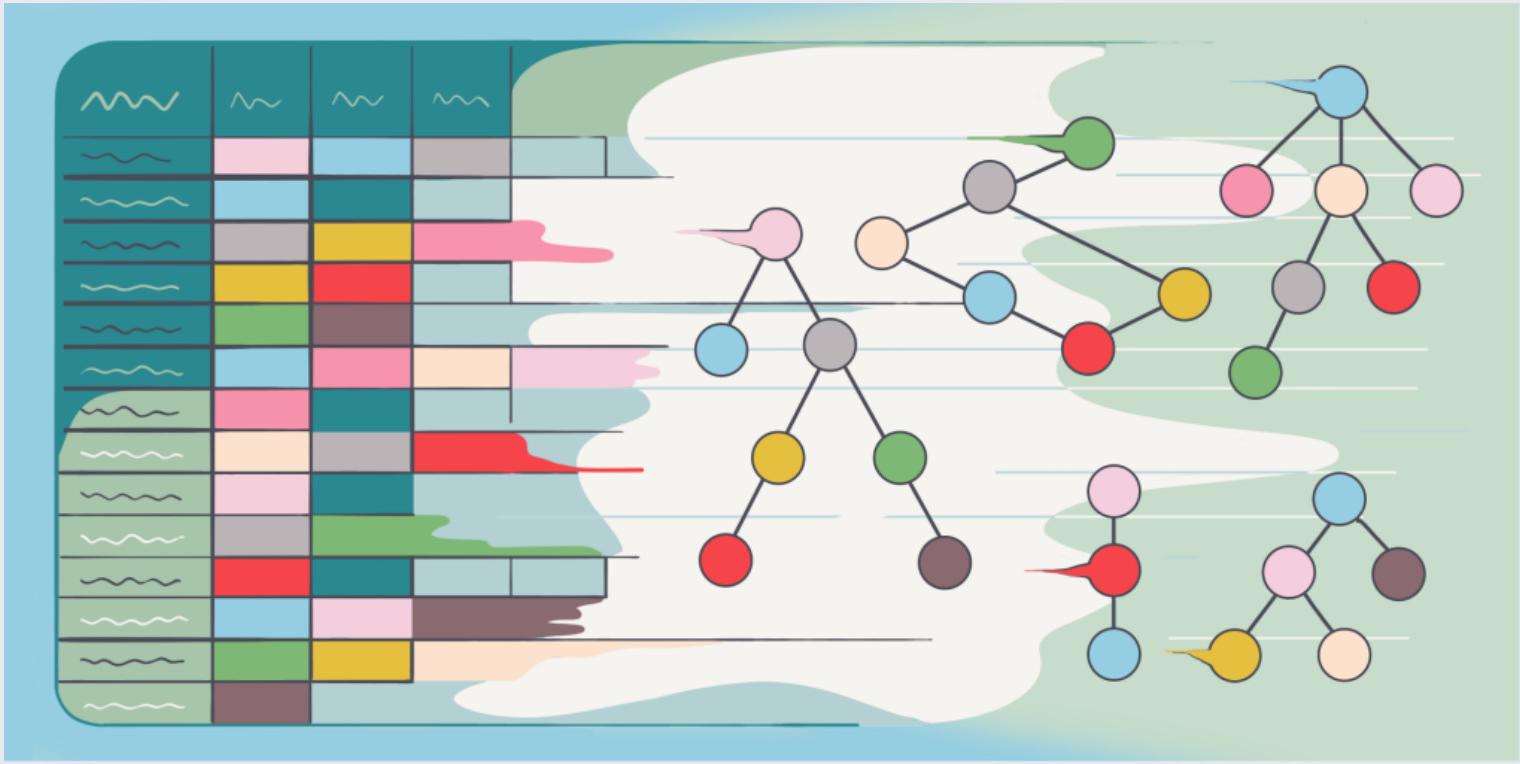
10 types ==> 49k combos!!

How to measure?

Mrigank Pawagi
IISc

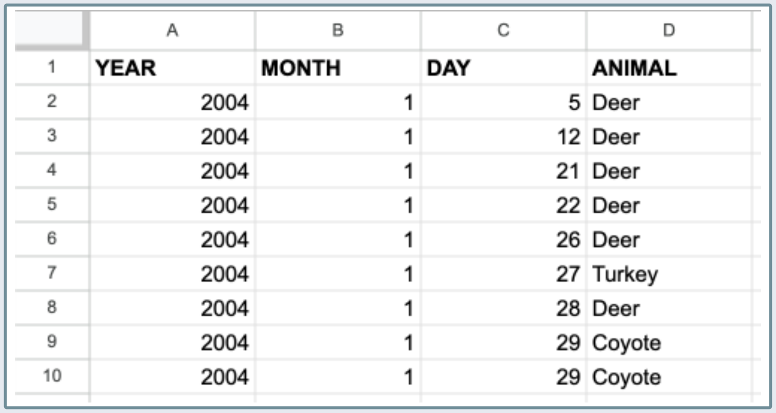


Vivaan Rajesh
Georgia Tech



Programming for tables

What about types?



	A	B	C	D
1	YEAR	MONTH	DAY	ANIMAL
2	2004	1	5	Deer
3	2004	1	12	Deer
4	2004	1	21	Deer
5	2004	1	22	Deer
6	2004	1	26	Deer
7	2004	1	27	Turkey
8	2004	1	28	Deer
9	2004	1	29	Coyote
10	2004	1	29	Coyote

Goal: describe table "shapes" to catch buggy programs

✗ `T.anml`

✗ `T.year + T.day`

Decades of prior work ...

Remy POPL 1989

Harper++ POPL 1991

Slepak++ ESOP 2014

Kazerounian++ PLDI 2019

Wand I&C 1991

Buneman++ TDS 1996

Vazou++ ICFP 2015

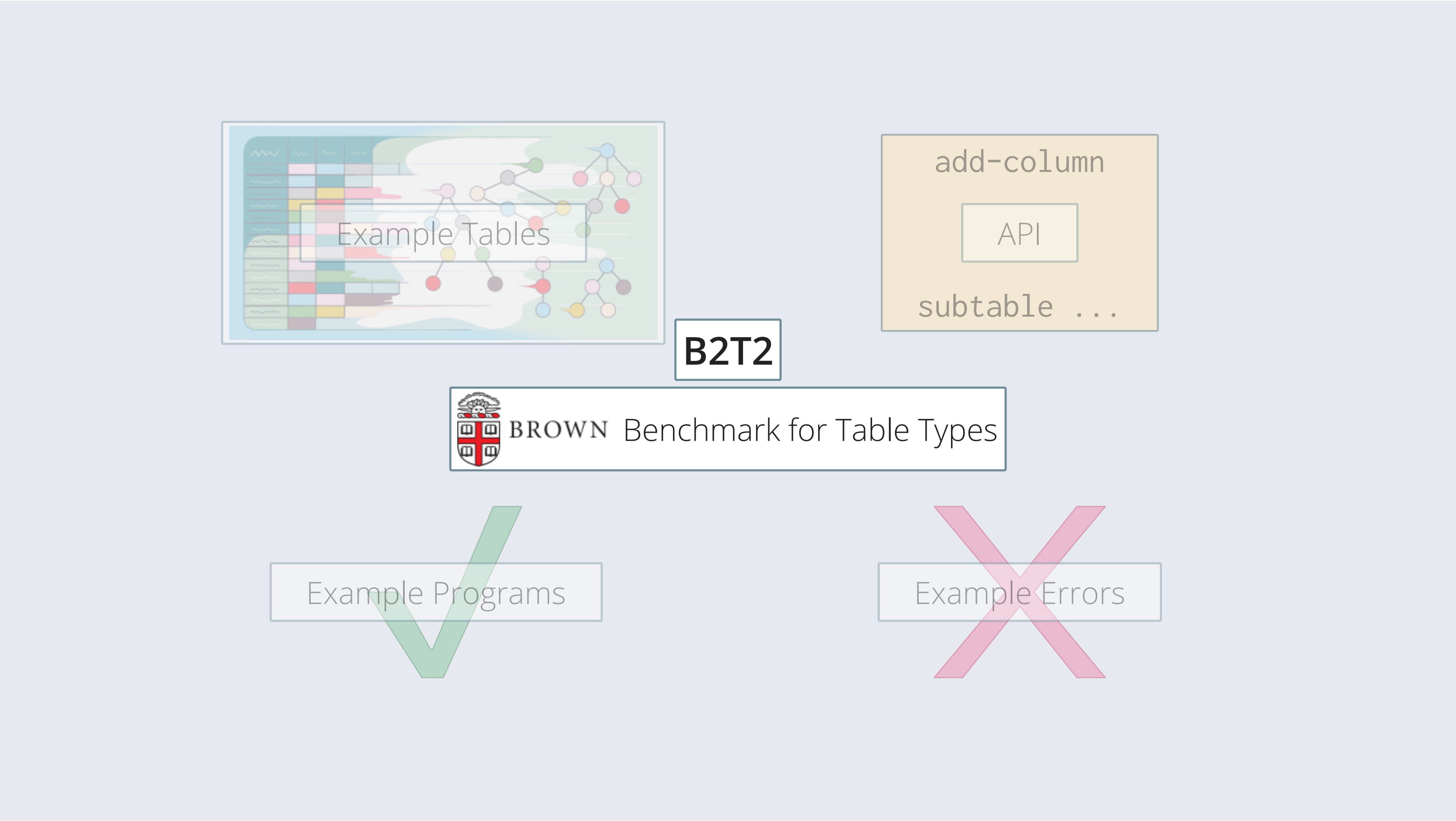
Morris++ POPL 2019

Gaster 1998

Ohori++ ICFP 2011

Petricek ECOOP 2017

...



Example Tables

add-column

API

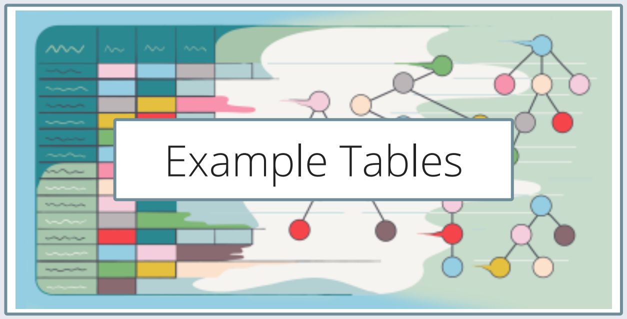
subtable ...

B2T2

BROWN Benchmark for Table Types

✓ Example Programs

✗ Example Errors



Example Tables

add-column

API

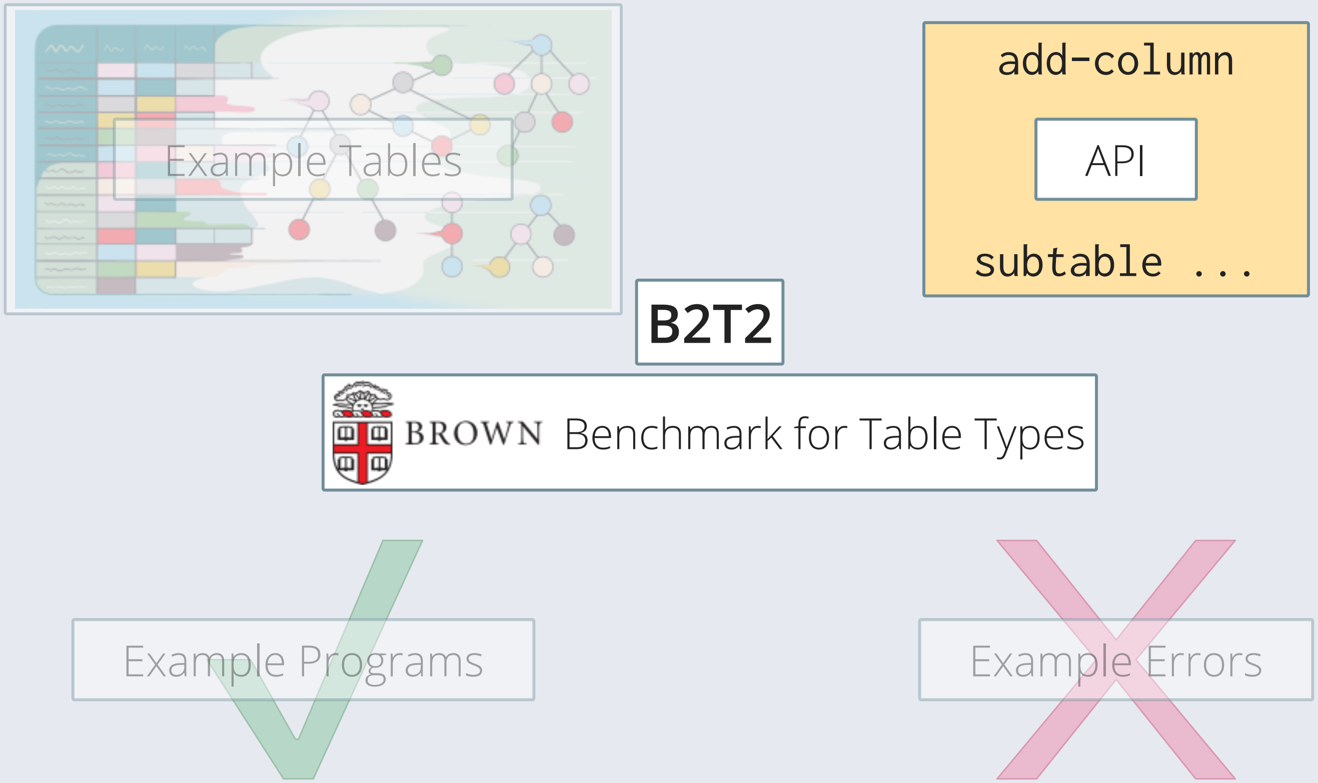
subtable ...

B2T2

BROWN Benchmark for Table Types

✓ Example Programs

✗ Example Errors



Example Tables

add-column

API

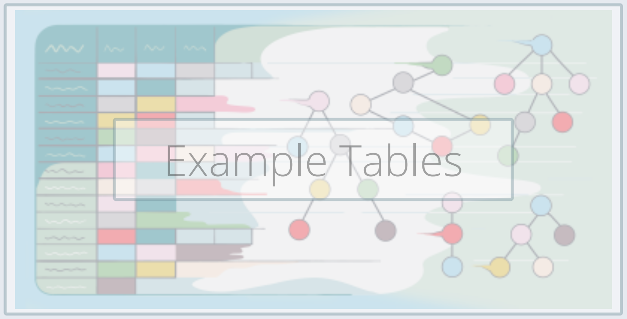
subtable ...

B2T2

BROWN Benchmark for Table Types

Example Programs

Example Errors


Example Tables

```
add-column
```

API

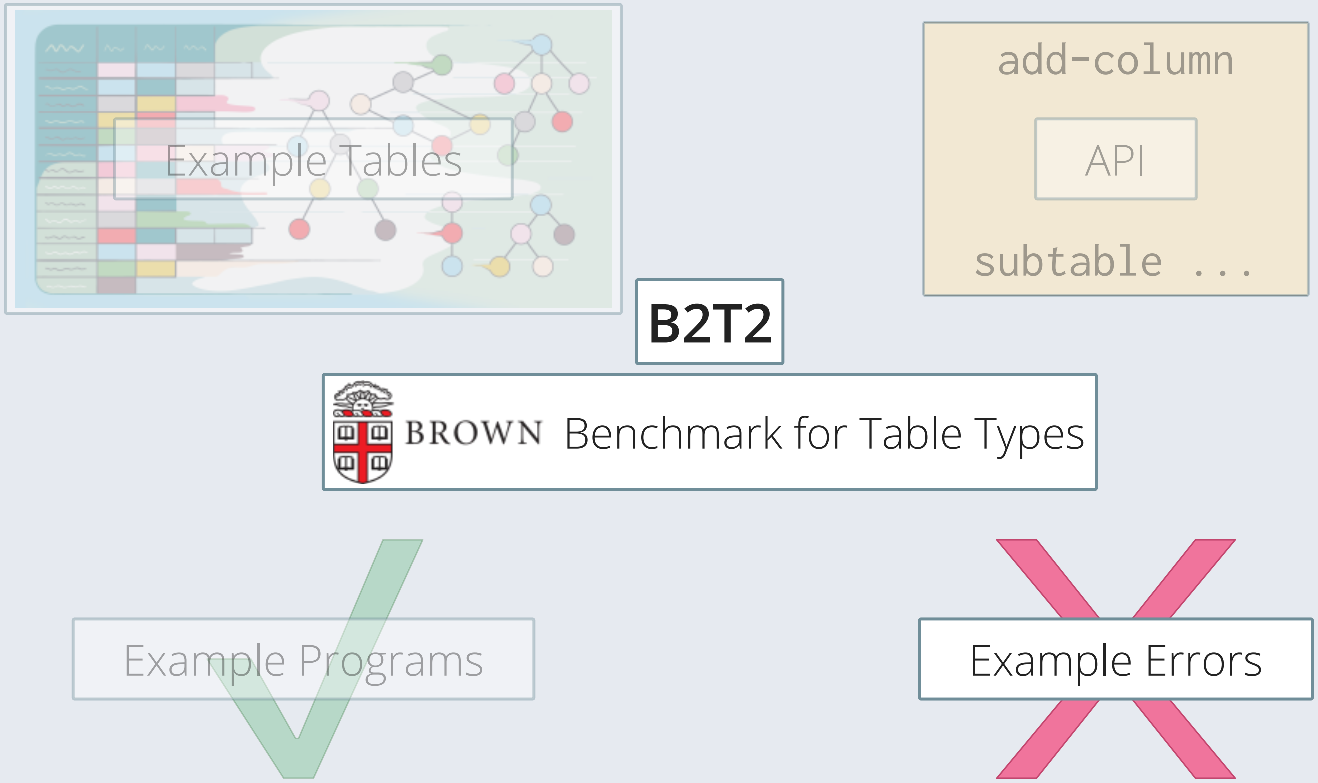
```
subtable ...
```

B2T2

BROWN Benchmark for Table Types

✓ Example Programs

✗ Example Errors



Example Tables

add-column

API

subtable ...

B2T2

BROWN Benchmark for Table Types

Example Programs

Example Errors

Example Program:

Find columns that start with "quiz", compute their average.

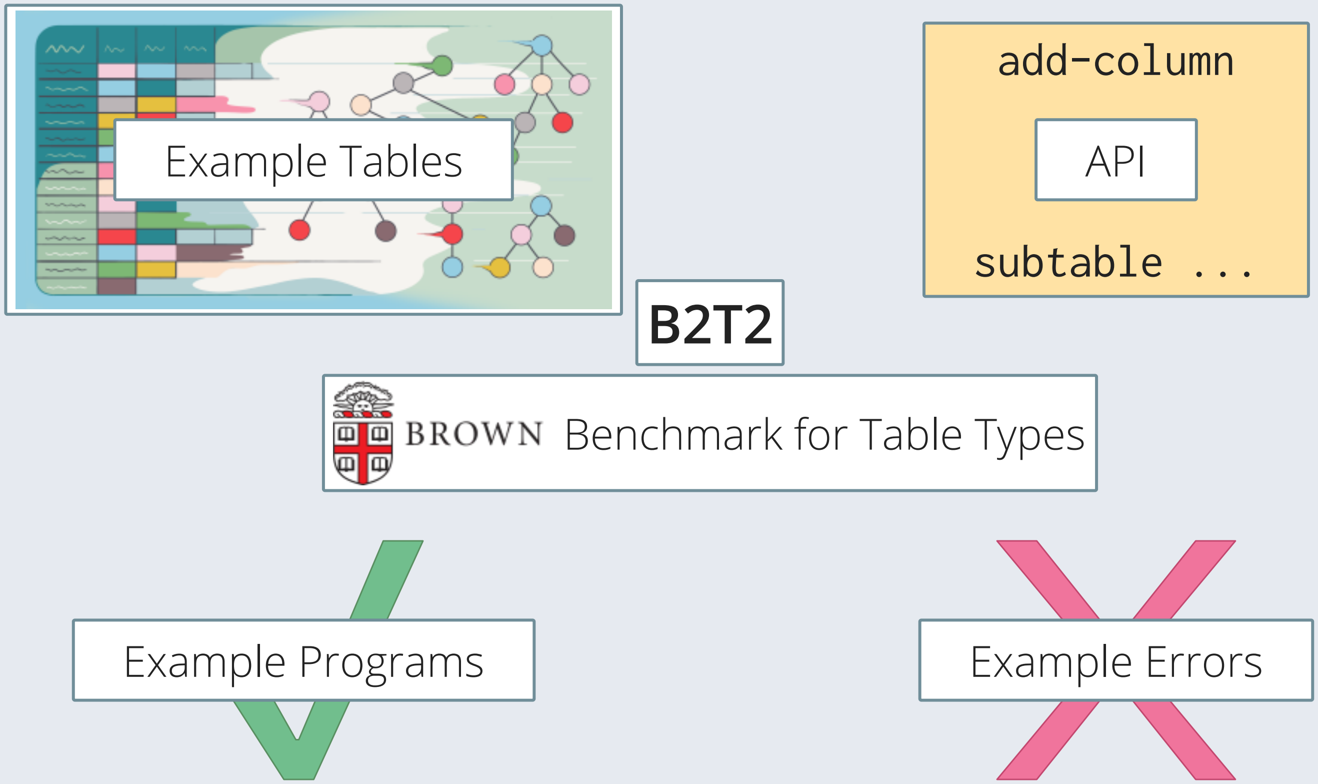
```
buildColumn(gradebook, "average-quiz",
  function(row):
    let quizColnames =
      filter(header(row),
        function(c):
          startsWith(c, "quiz"))
    let scores = map(quizColnames,
      function(c):
        getValue(row, c))
    sum(scores) / length(scores))
```

Example Error:

Task: find participants who ate black and white jellybeans

```
filter(jellybeanTable,
  function(r):
    getValue(r, "black and white") == true)
```

Error: no column `"black and white"`



Example Tables

add-column

API

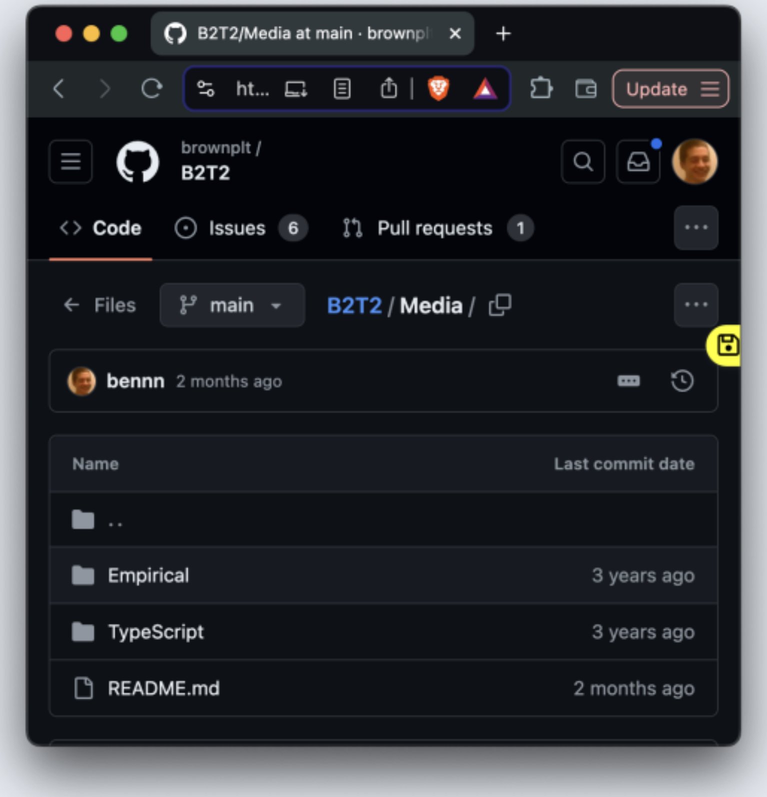
subtable ...

B2T2

BROWN Benchmark for Table Types

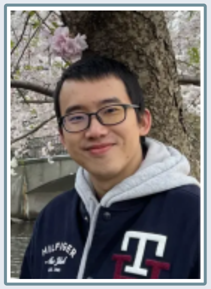
Example Programs

Example Errors



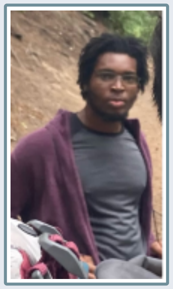
bennn 2 months ago

Name	Last commit date
..	
Empirical	3 years ago
TypeScript	3 years ago
README.md	2 months ago



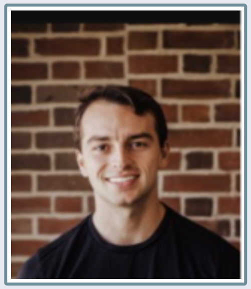
Kuang-Chen Lu

BROWN



Dibri Nsofor

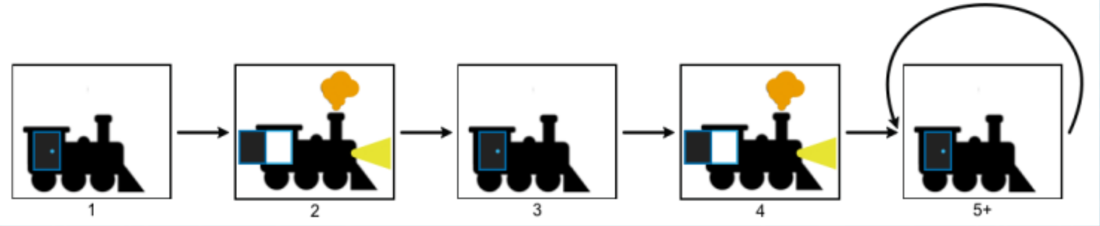
THE UNIVERSITY OF UTAH



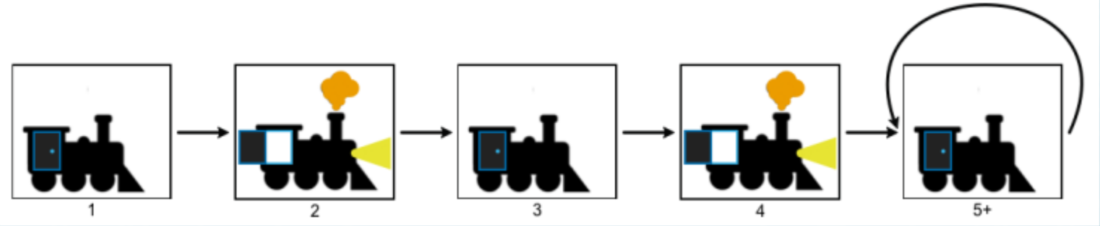
Rob Durst

Spring Health

LTL = Linear Temporal Logic



LTL = Linear Temporal Logic



For systems that change over time

Prop. Logic

`and or ==> not`

+

Temporal Operators

`always eventually next until`

✓ Expressive

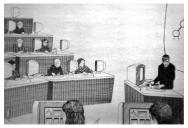
✓ Small

✓ Supports decision procedures

…. and easy to learn?

In what ways is LTL difficult to use?

qualtrics



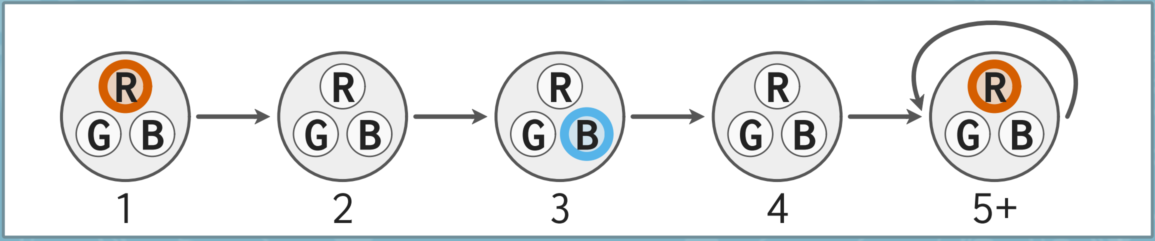
Quiz time!

Q. Do the traces below satisfy this formula?

{eventually Red} and {eventually Green}

Q. Do the traces below satisfy this formula?

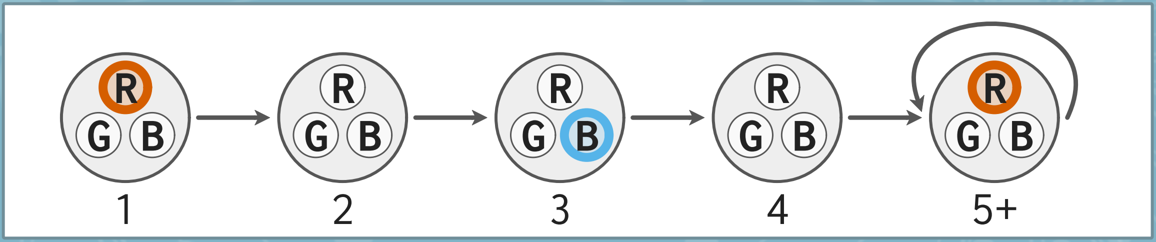
```
{eventually Red} and {eventually Green}
```



Q. Do the traces below satisfy this formula?

`{eventually Red} and {eventually Green}`

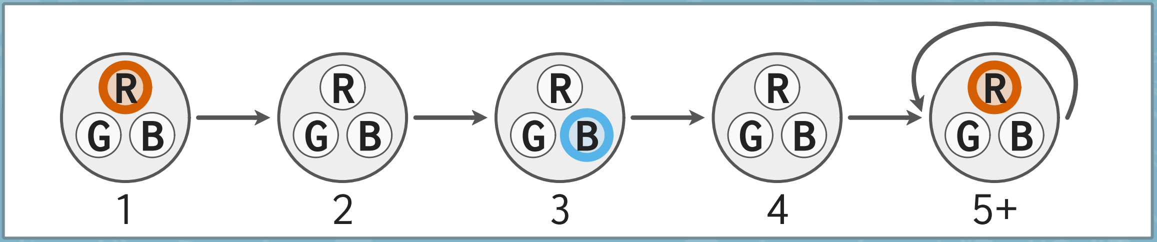
✗

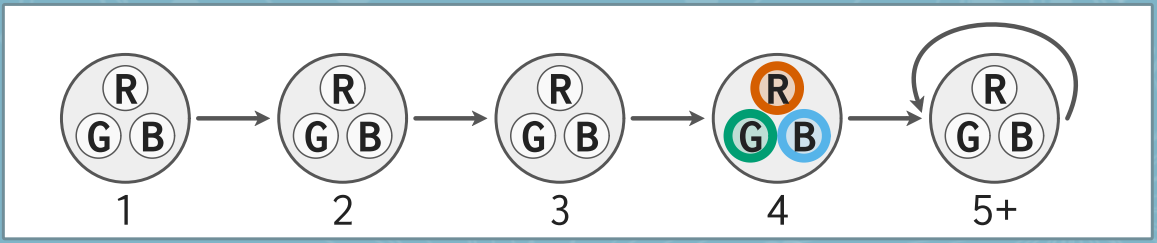


Q. Do the traces below satisfy this formula?

`{eventually Red} and {eventually Green}`

✗

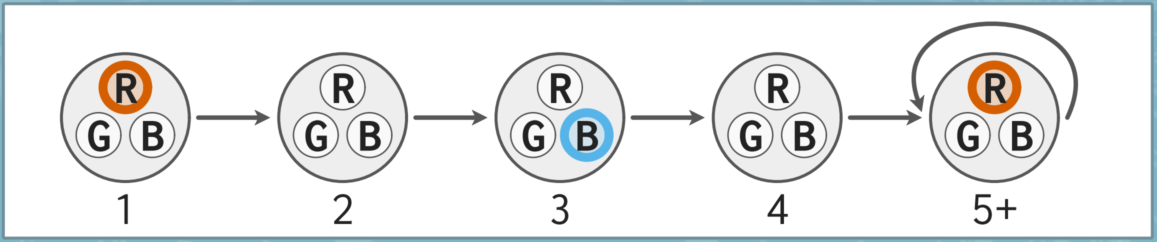




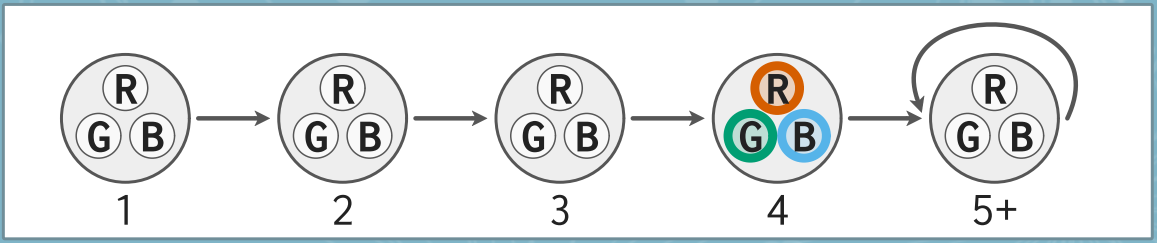
Q. Do the traces below satisfy this formula?

`{eventually Red} and {eventually Green}`

✗

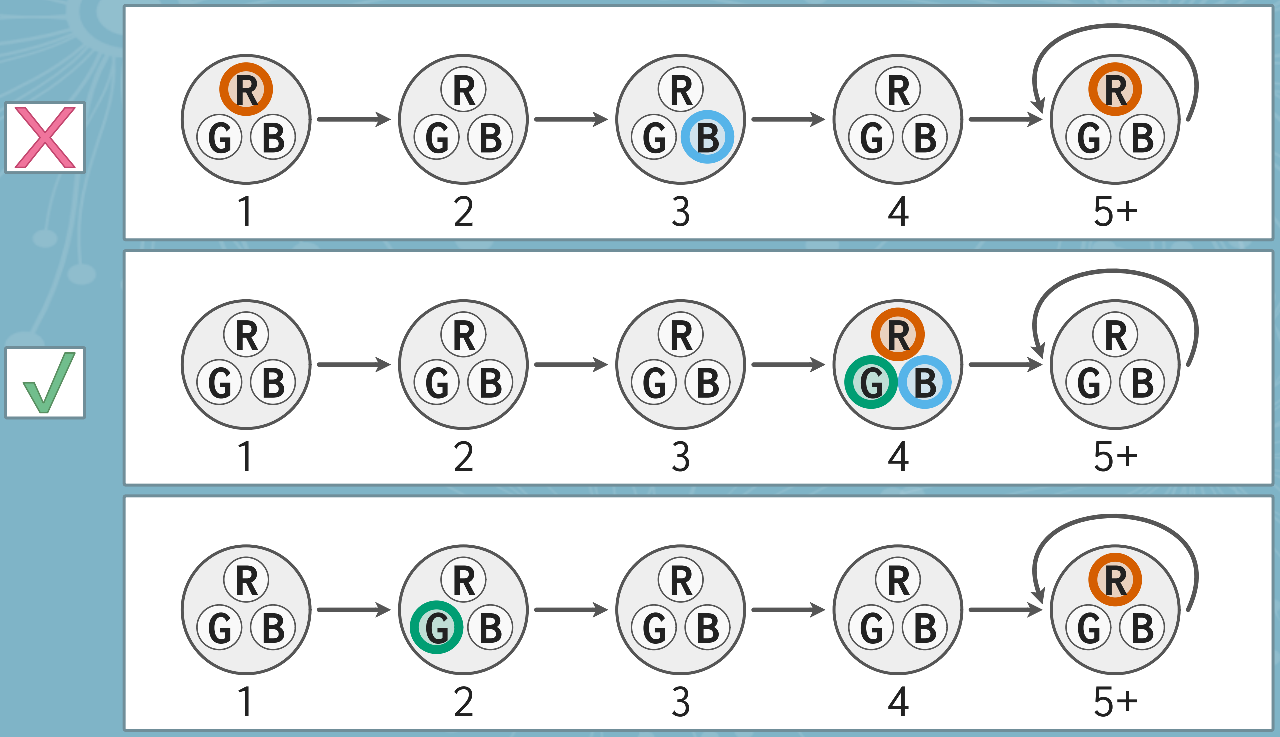


✓



Q. Do the traces below satisfy this formula?

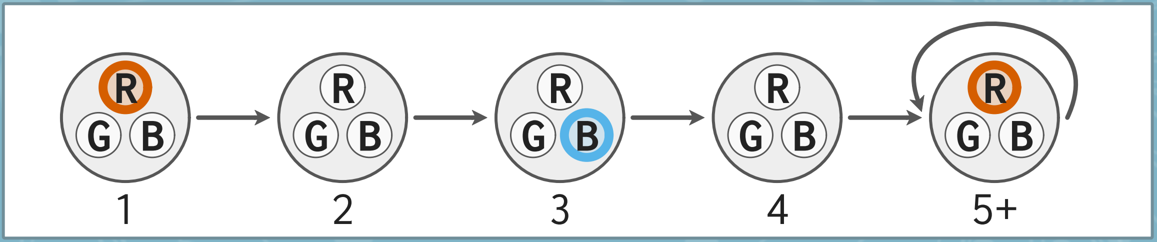
`{eventually Red} and {eventually Green}`



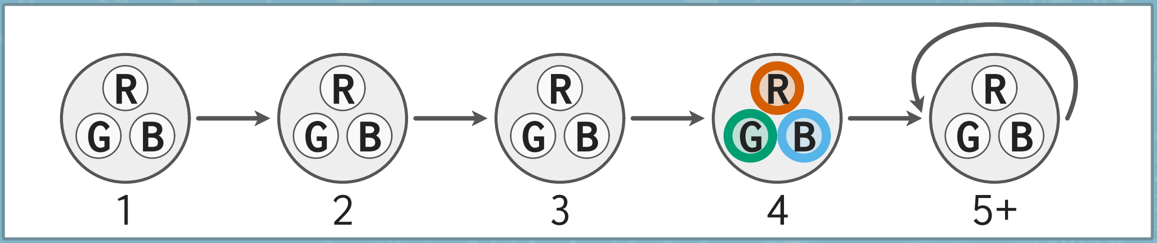
Q. Do the traces below satisfy this formula?

`{eventually Red} and {eventually Green}`

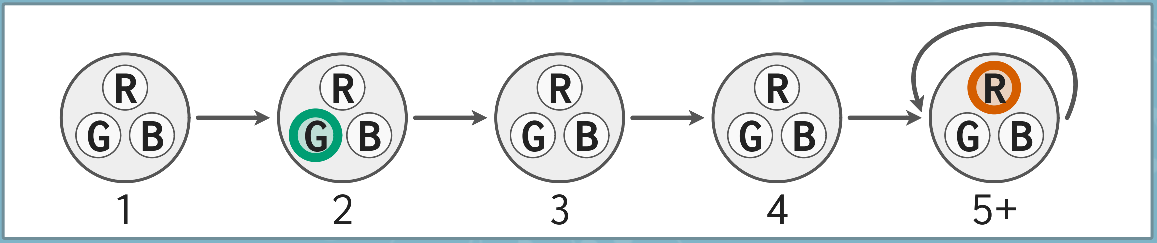
✗



✓



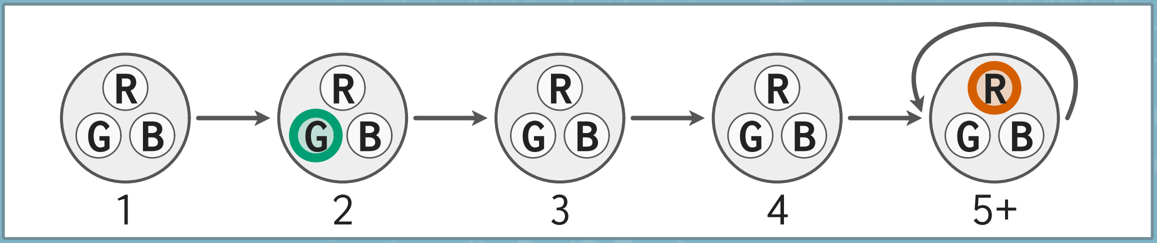
✓



Q. Do the traces below satisfy this formula?

```
{eventually Red} and {eventually Green}
```

✓



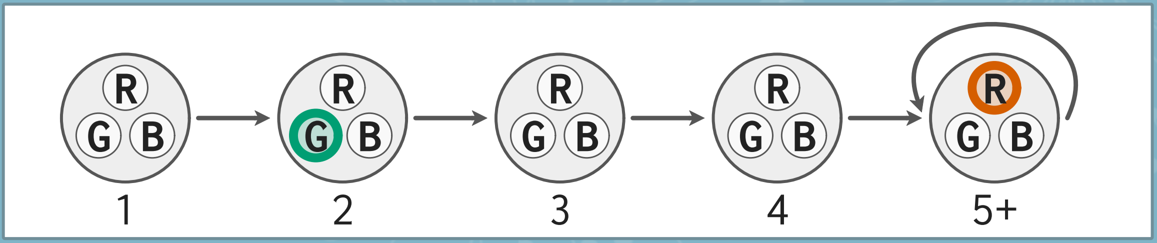
Q. Do the traces below satisfy this formula?

`{eventually Red} and {eventually Green}`

Not satisfied, because Green comes before Red

`Bad Prop` misconception

✓

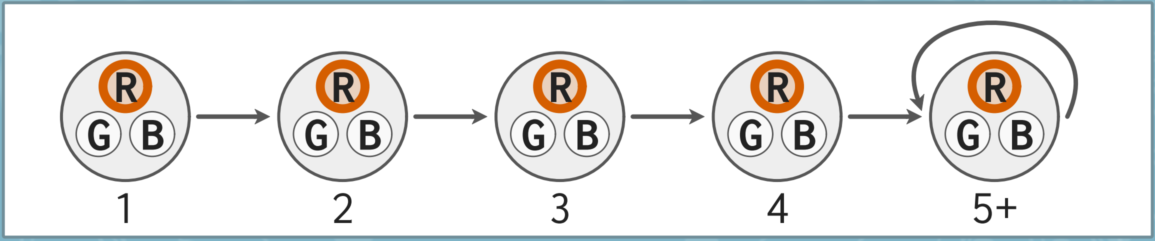


Q. Do the traces below satisfy this formula?

`Red until Blue`

Q. Do the traces below satisfy this formula?

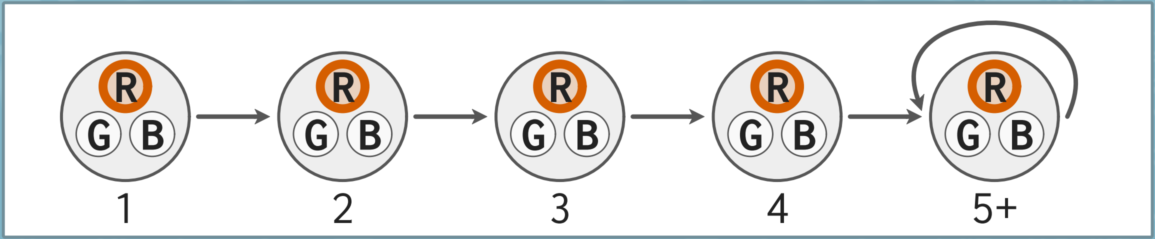
`Red until Blue`



Q. Do the traces below satisfy this formula?

`Red until Blue`

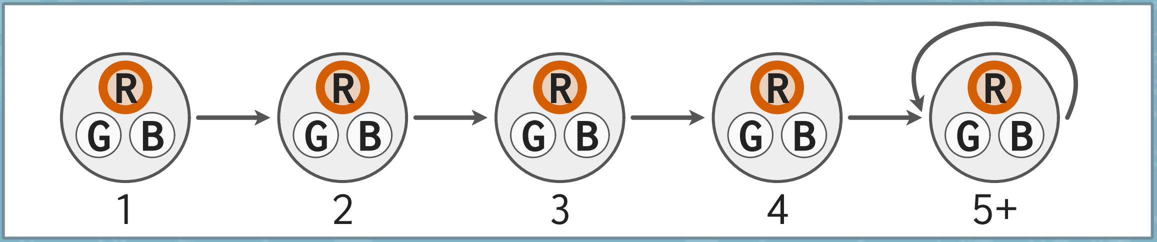
✗

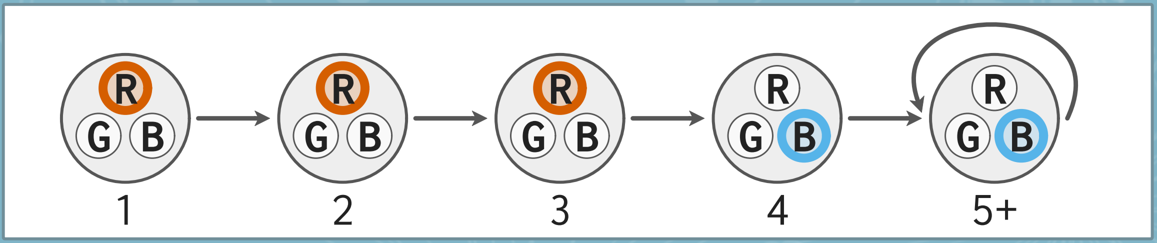


Q. Do the traces below satisfy this formula?

Red until Blue

✗

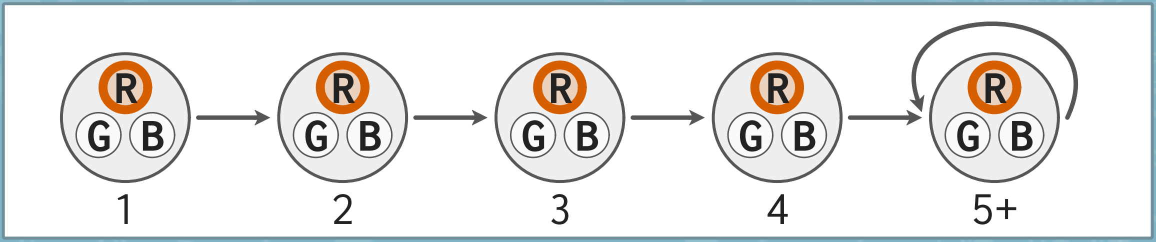




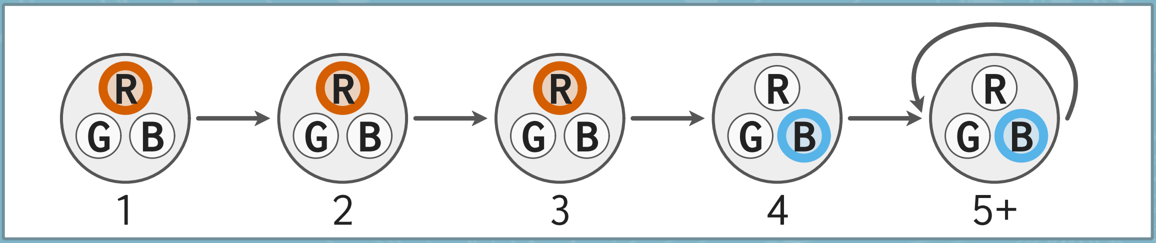
Q. Do the traces below satisfy this formula?

Red until Blue

✗



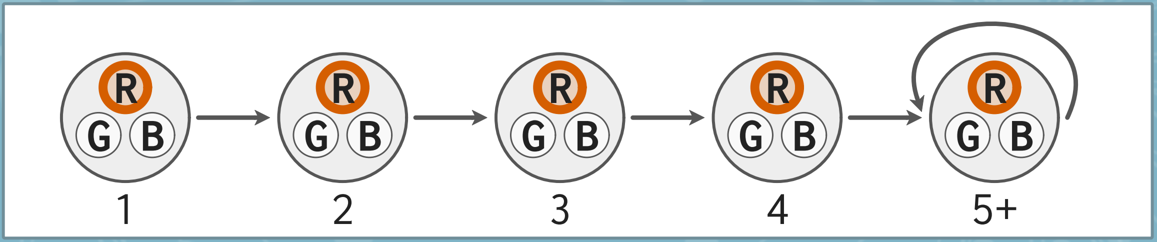
✓



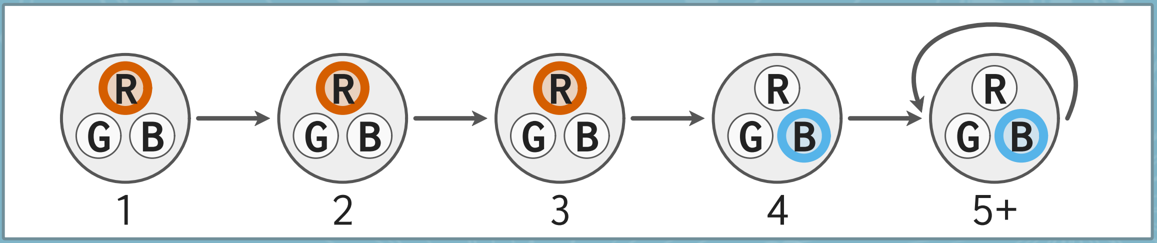
Q. Do the traces below satisfy this formula?

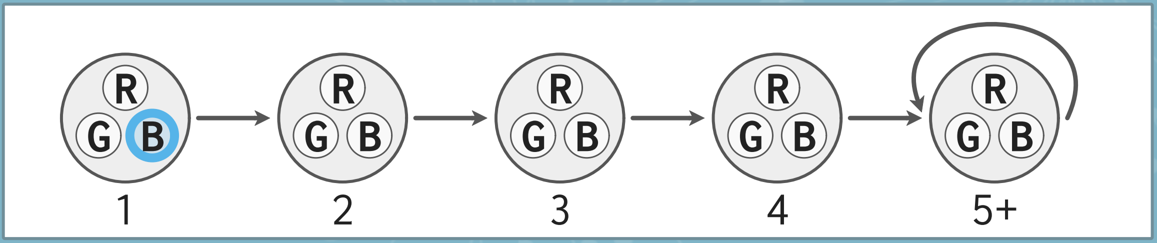
`Red until Blue`

☒



✓

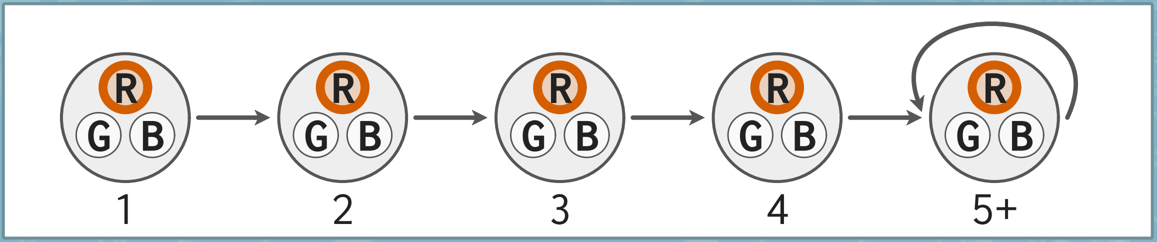




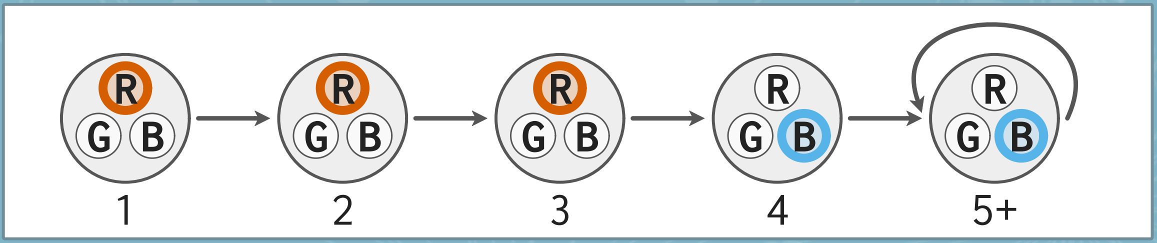
Q. Do the traces below satisfy this formula?

Red until Blue

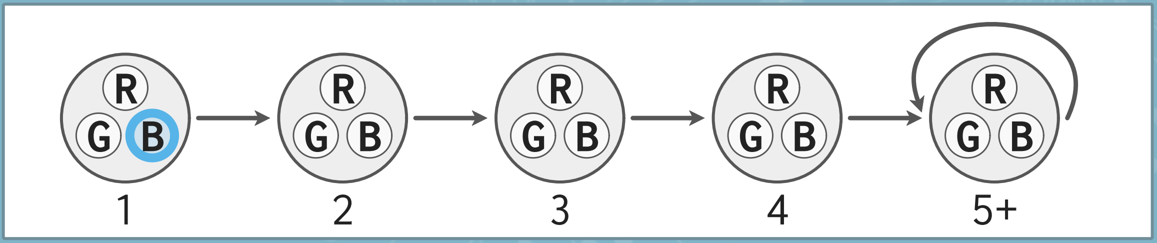
✗



✓



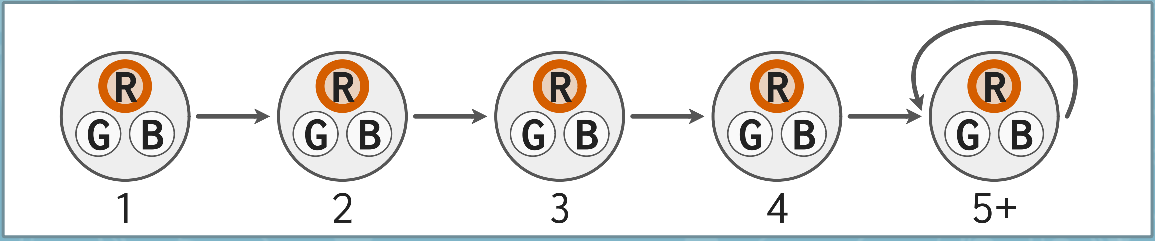
✓



Q. Do the traces below satisfy this formula?

`Red until Blue`

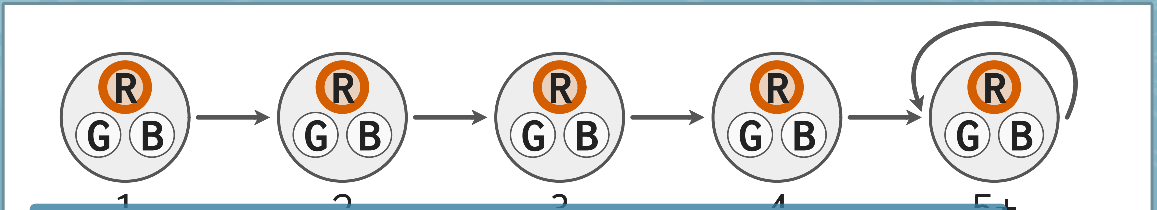
✗



Q. Do the traces below satisfy this formula?

`Red until Blue`

☒



Satisfied because Blue may stay off

Even among researchers!

`Weak U` misconception

Q. Translate to English

`{Red until Blue} and {always Red}`

Q. Translate to English

`{Red until Blue} and {always Red}`

"Red is always on"

Q. Translate to English

{Red until Blue} and {always Red}

☒ "Red is always on"

Q. Translate to English

{Red until Blue} and {always Red}

☒ "Red is always on"

"Red is always on and Blue is eventually on"

Q. Translate to English

```
{Red until Blue} and {always Red}
```

☒ "Red is always on"

☑ "Red is always on and Blue is eventually on"

Q. Translate to English

`{Red until Blue} and {always Red}`

☒ "Red is always on"

☑ "Red is always on and Blue is eventually on"

"This statement can never be satisfied"

Q. Translate to English

`{Red until Blue} and {always Red}`

☒ "Red is always on"

☑ "Red is always on and Blue is eventually on"

☒ "This statement can never be satisfied"

Q. Translate to English

`{Red until Blue} and {always Red}`

☒ "This statement can never be satisfied"

Q. Translate to English

`{Red until Blue} and {always Red}`

When Blue turns on, Red **must** be off

`Exclusive U` misconception

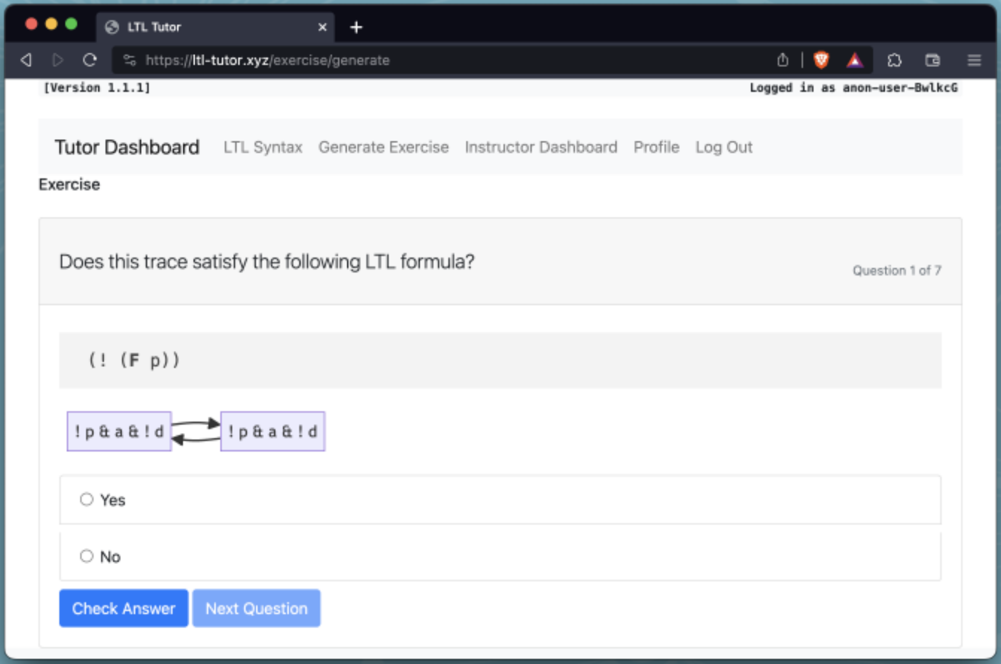
☒ "This statement can never be satisfied"

All done!

Categories of LTL Errors

- Bad Prop
- Bad State Index
- Bad State Quantification
- Cycle G
- Exclusive U
- Implicit F
- Implicit G
- Implicit Prefix
- Other Implicit
- Trace Split U
- Spreading X
- Weak U

https://ltl-tutor.xyz



Exercise

Does this trace satisfy the following LTL formula?

Question 1 of 7

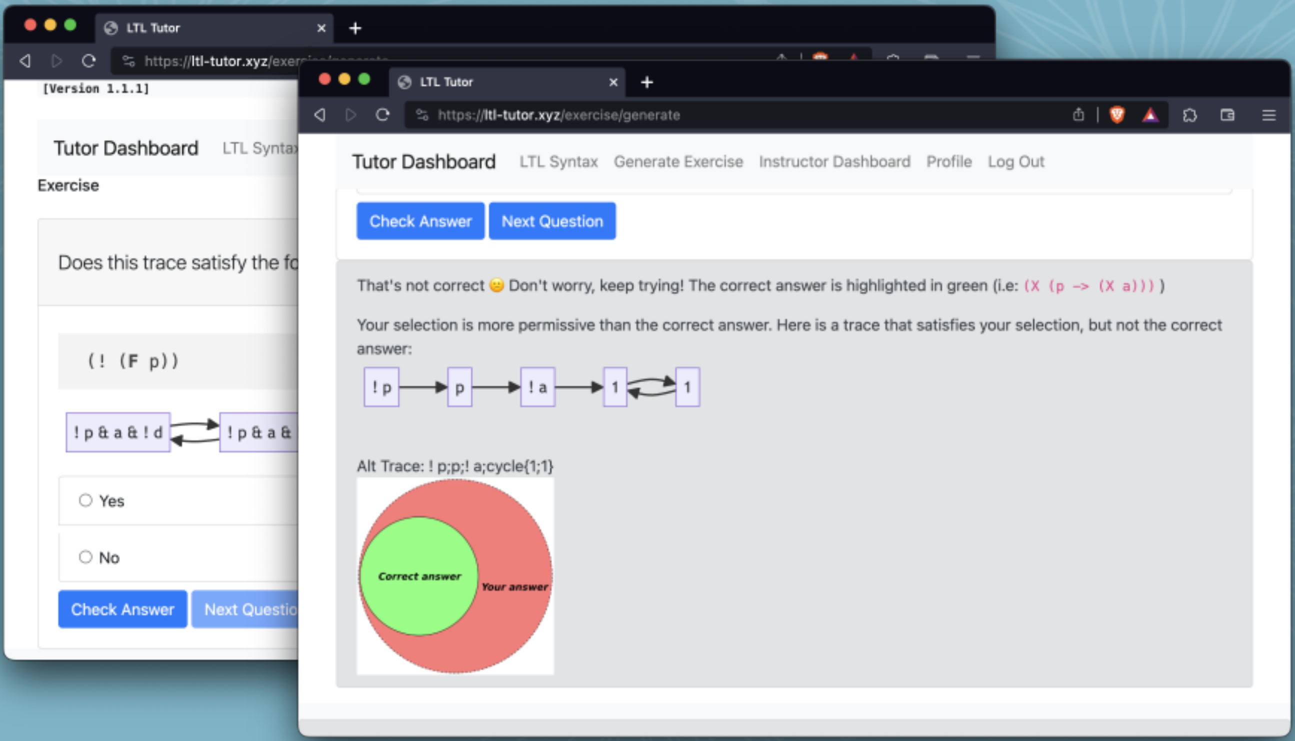
```
(! (F p))
```

!p&a&!d → !p&a&!d

○ Yes

○ No

https://ltl-tutor.xyz



Does this trace satisfy the fo

```
(! (F p))
```

○ Yes

○ No

That's not correct 😕 Don't worry, keep trying! The correct answer is highlighted in green (i.e: `(X (p -> (X a)))`)

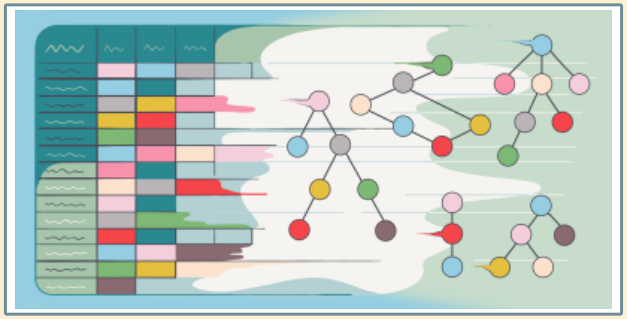
Your selection is more permissive than the correct answer. Here is a trace that satisfies your selection, but not the correct answer:

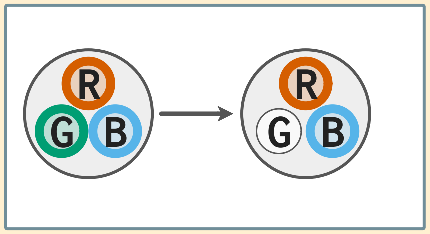
Alt Trace: ! p;p;! a;cycle{1;1}

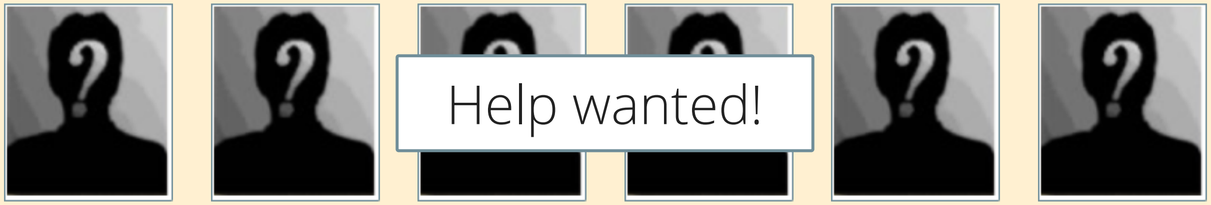
https://ltl-tutor.xyz

Siddhartha Prasad

BROWN



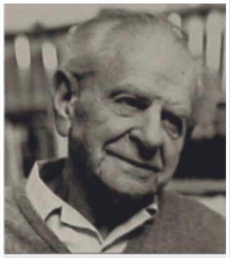




Help wanted!

"A theory which is **not refutable** by any conceivable event is **nonscientific**.

Irrefutability is not a virtue of a theory (as people often think) but a **vice**."



Sir Karl Popper (1902-1994)

Karl Popper