GTP Benchmarks for Gradual Typing Performance

Ben Greenman

BROWN

THE UNIVERSITY OF UTAH
Benchmarks + Experiments are important.
Benchmarks + Experiments are important.

Must be:
Relevant
Rigorous
Reproducible
Benchmarks + Experiments are important. Must be: Relevant, Rigorous, Reproducible.
Benchmarks + Experiments are important.

Must be:
Relevant
Rigorous
Reproducible
Benchmarks + Experiments are important.

Must be:
- Relevant
- Rigorous
- Reproducible
How to encourage *domain-specific* benchmarks?
How to encourage domain-specific benchmarks?

Main takeaway: think like a practitioner
GTP = Gradual Typing Performance
Gradual Typing

Untyped ➤ Typed
def join(d0, d1, sort, how):
    ....

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

TypeScript is JavaScript with syntax for types.

Used by 19.6m

DefinitelyTyped

+ 19,600,849
Gradual Typing Performance?

Untyped ➤ Typed
Gradual Typing Performance?

Untyped ➞ Typed

Run-time cost of sound types
Gradual Typing Performance?

Untyped ➤ Typed

**Run-time cost** of sound types

def join(d0:DataFrame,
        d1:DataFrame,
        sort:bool,
        how:Left|Right)
    -> DataFrame:
        ....
Gradual Typing Performance?

Untyped ➞ Typed

Run-time cost of sound types

?? join(x,y,z) How to validate?

```python
def join(d0: DataFrame, d1: DataFrame, sort: bool, how: Left|Right) -> DataFrame:
    ...
```
Gradual Typing Performance?

Untyped ➤ Typed

Run-time cost of sound types

?? join(x, y, z) How to validate?

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

(Typescript does not validate)
Gradual Typing Performance?

Untyped ➤ Typed

Run-time cost of sound types

join(x,y,z) How to validate?

def join(d0:DataFrame, d1:DataFrame, sort:bool, how:Left|Right) -> DataFrame:
    ....
Gradual Typing Performance?

Untyped ➞ Typed

**Run-time cost** of sound types

?? \texttt{join(x,y,z)} How to validate?

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

Many interactions, Maybe high costs
What does soundness cost?
What does soundness cost?

Typed Racket
+ object types, function types, ...
+ type-driven optimizer

Worst-case slowdown: 1.4x
ecoop '15
What does soundness cost?

Typed Racket
+ object types, function types, ...
+ type-driven optimizer

Worst-case slowdown: 1.4x
eecoop '15
What does soundness cost?

Typed Racket
+ object types, function types, ...
+ type-driven optimizer

Worst-case slowdown: 1.4x
eecoop '15

2x
30x
12,000x

(1ms to 12sec)
What does soundness cost?

**Typed Racket**
+ object types, function types, ...
+ type-driven optimizer

Worst-case slowdown: **1.4x**

*ecoop '15*
What does soundness cost?

Need a way to measure!
GTP Benchmarks

What to measure? Cost of sound types
GTP Benchmarks

What to measure?  Cost of sound types
Which programs?  ... Any
GTP Benchmarks

What to measure? Cost of sound types
Which programs? ... Any
How fast is good enough? ???
GTP Benchmarks

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
</tr>
</thead>
<tbody>
<tr>
<td>What to measure?</td>
<td>Cost of sound types</td>
</tr>
<tr>
<td>Which programs?</td>
<td>Any</td>
</tr>
<tr>
<td>How fast is good enough?</td>
<td>???</td>
</tr>
<tr>
<td><strong>What is a benchmark?</strong></td>
<td>???</td>
</tr>
</tbody>
</table>
GTP Benchmarks

What to measure?  Cost of sound types
Which programs?  ... Any
How fast is good enough?  ???

What is a benchmark?  ???

Think like a practitioner
What is a gradual typing benchmark?

Untyped code?

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

Not enough.

Typed code?

```python
def join(d0:DataFrame, ...):
```

Not enough.
What is a gradual typing benchmark?

Untyped code? `def join(d0,d1,sort,how):` Not enough.
Typed code? `def join(d0:DataFrame, ...):` Not enough.

GT promise: can mix typed + untyped code
Need to measure all configurations
What is a gradual typing benchmark?

1. Start with a program

```python
def join(d0,d1,sort,how):
    ....
```
What is a gradual typing benchmark?

1. Start with a program
   ```python
   def join(d0, d1, sort, how):
       ....
   ```

2. Add types
   ```python
   def join(d0: DataFrame, 
             d1: DataFrame, 
             sort: bool, 
             how: Left | Right) 
       -> DataFrame:
       ....
   ```
What is a gradual typing benchmark?

1. Start with a program
   ```python
def join(d0, d1, sort, how):
    ....
   ```

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

3. Explore all configurations
What is a gradual typing benchmark?

5 modules, 32 configurations
What is a gradual typing benchmark?

Explore by module

5 modules, 32 configurations

2 modules, 4 configurations
What is a gradual typing benchmark?

Explore by module

5 modules, 32 configurations

2 modules, 4 configurations

3 modules, 8 configurations
Where to find benchmarks?
Where to find benchmarks?

Wherever people share code
Where to find benchmarks?

Wherever people share code

Current status: 21 benchmarks, +40k configurations
How to analyze the data?
How to analyze the data?

How to summarize?

How to compare?

How to scale?
How to analyze the data?

How to summarize?
How to analyze the data?

How to summarize?

Some ideas:
worst-case?  average?  median?
How to analyze the data?

How to summarize?

worst-case?   average?   median?
How to analyze the data?
How to analyze the data?

Too slow = useless!
How to analyze the data?

Too slow = useless!

x-axis = limit for "too slow" vs. untyped code (log scale)
y-axis = % usable configs.
How to compare
How to compare
How to scale
How to scale

Linear-size random samples
Software for Measurement
Software for Measurement
Software for Measurement
GTP measure

by Ben Greenman

(require gtp-measure)  package: gtp-measure

For benchmarking.

1 Command-line: raco gtp-measure

The gtp-measure raco command is a tool for measuring the performance of a set of gtp-
Software for Measurement

Interruptible! Space-Efficient. Configurable.

GTP measure
by Ben Greenman

(require gtp-measure) package: gtp-measure

For benchmarking.

1 Command-line: raco gtp-measure

The gtp-measure raco command is a tool for measuring the performance of a set of gtp-
Software for Measurement

GTP measure
by Ben Greenman

(require gtp-measure)

For benchmarking.

1 Command-line: raco gtp-measure

The gt-measure raco command is a tool for measuring

- key:bin = "*/Users/ben/code/racket/fork/racket/bin/*"
- key:iterations = 8
- key:jit-warmup = 1
- key:num-samples = 10
- key:sample-factor = 10
- key:cutoff = 9
- key:entry-point = "main.rkt"
- key:start-time = 0
- key:time-limit = #f

Interruptible! Space-Efficient.
Software for Measurement
Software for Measurement

Tiny DSL for experiments

```racket
#lang gtp-measure/manifest

#:config #hash(
  (bin . "/home/gtp/racket-8.8/bin/")
  (cutoff . 6)
  (num-samples . 10))

/home/gtp/benchmarks/morsecode
/home/gtp/benchmarks/take5
```
Software for Measurement
# Software for Measurement

**DSL for data**

```gtp-measure/output/typed-untyped
("00000" ("cpu time: 566 real time: 567 gc time: 62" ....))
("00001" ("cpu time: 820 real time: 822 gc time: 46" ....))
("00010" ("cpu time: 561 real time: 562 gc time: 46" ....))
("00011" ("cpu time: 805 real time: 807 gc time: 47" ....))
....
```
5.3 Output Data: Typed-Untyped Target

Output data for a `gtp typed-untyped` target.

Each line is the result for one configuration. The first element is the name of the configuration, followed by the measured values:

```haskell
#lang gtp-measure/output/typed-untyped

("00000" ("cpu time: 566 real time: 567 gc time: 62" ....))
("00001" ("cpu time: 820 real time: 822 gc time: 46" ....))
("00010" ("cpu time: 561 real time: 562 gc time: 46" ....))
("00011" ("cpu time: 805 real time: 807 gc time: 47" ....))
...
```
Software for Measurement

#lang gtp-measure/output/typed-untyped

(`"00000" "cpu time: 566 real time: 567 gc time: 62" ....)

(`"00001" "cpu time: 820 real time: 822 gc time: 46" ....)

(`"00010" "cpu time: 561 real time: 562 gc time: 46" ....)

(`"00011" "cpu time: 805 real time: 807 gc time: 47" ....)

....

Running an output file prints a summary:

```
$ racket jpeg-2020-08-17.rktd
dataset info:
- num configs: 32
- num timings: 256
- min time: 110 ms
- max time: 8453 ms
- total time: 968537 ms
```
Software for Visualization
Software for Visualization

![Graph showing mbta typed/baseline ratio: 2.28x with 16 configurations]
Software for Visualization

```
(parameterize ((*OVERHEAD-SHOW-RATIO* #f))
  (overhead-plot (list mbta (typed-racket-info%best-typed-path mbta 2))))
```

![Graph showing mbta, mbta+2 types over 16 configurations]
Software for Visualization

![Chart showing performance metrics for snake-7.8.0.5 and snake-transient]
Software for Visualization
Software for Visualization

Given a performance-info structure, shows the overhead of every configuration in a lattice. Given a number, render an unlabeled lattice.

```
(procedure (performance-lattice pi) \rightarrow pict?
  pi : (or/c performance-info? natural?)

(parameterize ([*FONT-SIZE* 14]
  [*LATTICE-UNIT-WIDTH* 16]
  [*LATTICE-UNIT-HEIGHT* 12]
  [*LATTICE-CONFIG-X-MARGIN* 10]
  [*LATTICE-CONFIG-Y-MARGIN* 25]
  [*LATTICE-LINES* #true]
  [*LATTICE-LINE-ALPHA* 0.5])

(ht-append 4
  (performance-lattice mbta)
  (performance-lattice 3))
```
Continuous Testing
benchmarks  measurement  visualization
All 3 important ... but not to everyone
All 3 important ... but not to everyone

Lesson 2: loose coupling helps adoption
Still ... low adoption
Still ... low adoption

2014: few experiments, ~2 gradual configurations
Still ... low adoption

2014: few experiments, ~2 gradual configurations

Lately: few experiments, but thorough

Ok?

Is Sound Gradual Typing Dead?

Asuna Takizawa, Daniel Pulley, Ben Greenman, Max S. New, Jan Vitek, Matthias Felleisen
Northeastern University, Boston, MA

Abstract
Programmers have come to embrace dynamically-typed languages for prototyping and delivering large and complex systems. When it comes to maintaining and evolving those systems, the lack of explicit static typing becomes a bottleneck. In response, researchers in many cases, the systems start as innocent prototypes. Soon enough, though, they grow into complex, multi-module programs at which point the engineers realize that they are facing a maintenance nightmare, mostly due to the lack of reliable type information.

Gradual typing [21, 26] proposes a language-based solution to this problem. By providing a type system that allows developers to gradually introduce static type checking into an existing dynamically-typed language, gradual typing aims to turn complex systems into maintainable ones.
Thank You
Lessons

How to encourage domain-specific benchmarks?
Lessons

How to encourage **domain-specific** benchmarks?

Think like a practitioner
Lessons

How to encourage *domain-specific* benchmarks?

Think like a practitioner

Separate benchmarks from analysis tools
Lessons

How to encourage **domain-specific** benchmarks?

- Think like a practitioner
- Separate benchmarks from analysis tools
- Borrow nodes
https://github.com/utahplt/gtp-benchmarks

https://github.com/utahplt/gtp-measure

https://github.com/utahplt/gtp-plot