Talk aims

1. The programming language status quo limits us.

2. Language composition might offer a way forward.

3. We're not very good at it yet.

4. Possible future routes.

L. Tratt  http://tratt.net/laurie/
Talk aims

1. The programming language status quo limits us.
2. Language composition might offer a way forward.
Talk aims

1. The programming language status quo limits us.
2. Language composition might offer a way forward.
3. We’re not very good at it yet.
Talk aims

1. The programming language status quo limits us.
2. Language composition might offer a way forward.
3. We’re not very good at it yet.
4. Possible future routes.
What’s the problem with the status quo?
Languages conceptual size
Languages conceptual size

Normal programmers brain
Languages conceptual size

K&R C
Languages conceptual size

Python
Programming languages’ speed of light

\[ \gamma \]

\[ c \]

\( \gamma \) vs. Speed

L. Tratt  
http://tratt.net/laurie/  
Language Composition  
2013-02-13
Programming languages’ speed of light

![Graph showing the speed of light in programming languages vs. programmer productivity.](http://tratt.net/laurie/)

L. Tratt  
[http://tratt.net/laurie/](http://tratt.net/laurie/)  
Language Composition  
2013-02-13
Programming languages’ speed of light

![Graph showing the speed of light in terms of programmer productivity.]

- Machine code is represented at the point (0, 0).

L. Tratt  http://tratt.net/laurie/  Language Composition  2013-02-13  5 / 31
Programming languages’ speed of light

![Graph showing the relationship between programmer productivity and language speed]

- Machine code
- Assembly

L. Tratt  http://tratt.net/laurie/  Language Composition  2013-02-13  5 / 31
Programming languages’ speed of light

![Graph showing the speed of light for different programming languages. The x-axis represents programmer productivity, and the y-axis represents the speed of light. The graph includes points for Assembly, Machine code, K & R C, and C.](http://tratt.net/laurie/)
Programming languages’ speed of light

![Graph showing the speed of light for different programming languages, with Assembly, Machine code, K & R C, and Python plotted on the graph. The x-axis represents programmer productivity, and the y-axis represents time (in years).](http://tratt.net/laurie/)
Programming languages’ speed of light

Diminishing Returns

0 1 2 3 4 5 6 7 8 9 10

S

Programmer productivity

Assembly

Machine code

K & R C

Python

Scala

L. Tratt http://tratt.net/laurie/  Language Composition  2013-02-13  5 / 31
Can S become too big?
Can S become too big?
Wiggle room
Wiggle room

Basic functionality
Wiggle room

- Basic functionality
- Goodies
Even worse
Even worse
Even worse
Even worse

Basic functionality

Goodies

Bad luck
Is this about DSLs?

Haskell / Ruby / Scala
Is this about DSLs?

Haskell / Ruby / Scala

DSL
DSL
DSL
Is this about DSLs?

Haskell / Ruby / Scala

DSL
DSL
DSL
DSL
Is this about DSLs?

Any language you want
A way forward?
Idea: allow users to compose languages.
What does composition mean?
What does composition mean?

Language $\triangleq$ syntax + semantics
What does composition mean?

Language implementation $\triangleq$ compiler $+$ runtime
What does composition mean?

Language implementation $\triangleq$ compiler + virtual machine
What does composition mean?

Compiler ≜ parser + code generator
What does composition mean?

\[ \text{Compiler} \triangleq \text{parser} + \text{code generator} \]
What does composition mean?

Compose:
- parsers
- virtual machines
Example (1)

```java
for (pid : SELECT pid FROM personnel WHERE salary > 100000) {
    if (!is_worth_it(pid))
        UPDATE personnel SET salary=0 WHERE pid=pid;
}
```
Example (1)

SQL and Java

```java
for (pid : SELECT pid FROM personnel WHERE salary > 100000) {
    if (!is_worth_it(pid))
        UPDATE personnel SET salary=0 WHERE pid=pid;
}
```
Example (2)

Tax code

2010-2011

allowance

age < 65: £6,475
age >= 65 and age <= 74: £9,490
age > 74: £9,640

reduction: if income > £100,000 then

max(0, allowance - ((income - £100,000) / 2))
Tax code

income tax {
  2010-2011 {
    allowance {
      age < 65: £6,475
      age >= 65 and age <= 74: £9,490
      age > 74: £9,640

      reduction: if income > £100,000 then
        max(0, allowance - ((income - £100,000) / 2))
    }
  }
}
Why aren’t we (me?) very good at it yet?
Converge

Converge
Converge $\triangleq$ Python + macros
Converge \triangleq Python + compile-time meta-programming
Compile-time meta-programming

Code (as trees, not text) is programmatically generated.
Code (as trees, not text) is programmatically generated.

**Expression**  \( 2 + 3 \) evaluates to 5.

**Splice** \( $<x> \) evaluates \( x \) at compile-time; the AST returned overwrites the splice.

**Quasi-quote** \[ | 2 + 3 | \] evaluates to a *hygienic* AST representing \( 2 + 3 \).

**Insertion** \[ | 2 + ${x} | \] ‘inserts’ the AST \( x \) into the AST being created by the quasi-quotes.
Compile-time meta-programming

Code (as trees, not text) is programmatically generated.

| Expression | 2 + 3 | evaluates to 5. |
| Splice     | $<x>$ | evaluates $x$ at compile-time; the AST returned overwrites the splice. |
| Insertion  | [ | 2 + $\{x\}$ | ] | ‘inserts’ the AST $x$ into the AST being created by the quasi-quotes. |
| DSL Block  | $<x>$: ... | passes the text ‘...’ to the function $x$ at compile-time. |
An example
Where it falls apart

- Parser composition: a mess.
Parser composition: a mess.

Extension languages second-class citizens.
Parsing composition

Should be easy
Parsing composition

- LR
- Earley
- PEG
- **LR** composition undefined (in general).
- **Earley**
- **PEG**
- **LR** composition undefined (in general).
- **Earley** composition ambiguous (in general).
- **PEG**
- **LR** composition undefined (in general).
- **Earley** composition ambiguous (in general).
- **PEG** composition can shadow (in general).
Where it falls apart (2)

- Parser composition: a mess.
- Extension languages second-class citizens.
Parser composition: a mess.

Extension languages second-class citizens.

Text only.
Example (3)
Example (3)

```c
func custom_prescription(Patient p) : Medicine {
    if (p.penicillin_allergy())
        return NULL;

    Medicine m = [chemical_structure_image]
    candidate = generate(P, m);
    if (!check_with_doctor(candidate))
        return NULL;
    m.set_variable(R, candidate);

    return m;
}
```
```python
func check_all_suitable(trial_id):
    for patient_id in SELECT pid FROM trial WHERE id=${trial_id}:
        if SELECT * FROM prescribed
            WHERE contains('drug',
                warn("Patient ${patient_id} currently prescribed a "
                "penicillin derived anti-biotic and must be "
                "seen by a specialist before trial begins.")
```

What are our options?
Abandon parsing...
Abandon parsing...

...for SDE?
Boil down to the JVM
Boil down to the JVM

Meta-tracing to the rescue
Compose:
- parsers
- virtual machines
Compose:

- parsers *Incremental parsing*
- virtual machines
Compose:

- parsers *Incremental parsing*
- virtual machines *Meta-tracing*
Summary

The status quo needn't be so.
Language composition might offer a way forward.
We're not very good at it yet.
Incremental parsing and meta-tracing might save us.

L. Tratt
http://tratt.net/laurie/
The status quo needn’t be so.
The status quo needn’t be so.
Language composition might offer a way forward.
The status quo needn’t be so.
Language composition might offer a way forward.
We’re not very good at it yet.
The status quo needn’t be so.

Language composition might offer a way forward.

We’re not very good at it yet.

Incremental parsing and meta-tracing *might* save us.
Further reading

- *Parsing: the solved problem that isn’t*, Tratt
Thank you for listening