Language Composition

Laurence Tratt
http://tratt.net/laurie/

Software Development Team, King’s College London

2013-02-27
The programming language status quo limits us.
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.
The programming language status quo limits us.
Language composition might offer a way forward.
We’re not very good at it yet.
Possible future routes.
What’s the problem with the status quo?
Normal programmers brain
Languages conceptual size

Python
Languages conceptual size
Programming languages’ speed of light

\[ \gamma \approx \frac{v}{c} \]

\[ \text{Speed} \]

\[ \gamma \]

L. Tratt  http://tratt.net/laurie/  
Language Composition  
2013-02-27
Programming languages’ speed of light

\[ S \]

Programmer productivity

\[ c \]

L. Tratt  http://tratt.net/laurie/  Language Composition  2013-02-27  5 / 39
Programming languages’ speed of light

- Machine code

Programmer productivity

\( S \)
Programming languages’ speed of light

\[ S \]

Programmer productivity

Machine code

Assembly

L. Tratt http://tratt.net/laurie/ Language Composition 2013-02-27
Programming languages’ speed of light

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

![Diagram showing the speed of light for various programming languages.](image)

- **Assembly**
- **Machine code**
- **K & R C**
- **Python**

The diagram illustrates the programmer productivity (on the y-axis) and the speed of light (on the x-axis) for different programming languages.
Programming languages’ speed of light

![Diagram showing the speed of light in different programming languages]

- Assembly
- Machine code
- K & R C
- Python
- Scala

Programmer productivity

$L. Tratt$ http://tratt.net/laurie/
Programming languages’ speed of light

![Graph showing the speed of light in programming languages](image)

- **Assembly**
- **Machine code**
- **K & R C**
- **Python**
- **Scala**

**Diminishing Returns**

**Programmer productivity**

L. Tratt [http://tratt.net/laurie/](http://tratt.net/laurie/)

Language Composition 2013-02-27 5 / 39
Can S become too big?
Can S become too big?

C++
Wiggle room

Basic functionality
Wiggle room
Even worse
Even worse

[Diagram showing layers labeled 'Basic functionality' and 'Goodies']
Even worse

Basic functionality

Goodies
Even worse
Is this about DSLs?
Is this about DSLs?

Haskell / Ruby / Scala
Is this about DSLs?

Haskell / Ruby / Scala

DSL
DSL
DSL
DSL
Is this about DSLs?

Haskell /
Ruby /
Scala

L. Tratt  http://tratt.net/laurie/  Language Composition  2013-02-27  9 / 39
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 $\triangleq \text{compiler} + \text{runtime}$
What does composition mean?

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

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

Compiler $\triangleq$ parser + 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

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
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))}}}
```
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 $\triangleq$ Python + macros
Converge ≜
Python + compile-time meta-programming
Compile-time meta-programming

Code (as trees, not text) is programmatically generated.
Compile-time meta-programming

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

<table>
<thead>
<tr>
<th>Expression</th>
<th>2 + 3</th>
<th>evaluates to 5.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Splice</td>
<td>$&lt;x&gt;$</td>
<td>evaluates $x$ at compile-time; the AST returned overwrites the splice.</td>
</tr>
<tr>
<td>Insertion</td>
<td>[</td>
<td>2 + ${x}$</td>
</tr>
</tbody>
</table>
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.

**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.

**DSL Block** \( \$«x»: \ldots \) passes the text ‘...’ to the function \( x \) at compile-time.
An example
Where it falls apart

- Parser composition: a mess.
Where it falls apart

- 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.
Where it falls apart (2)

- Parser composition: a mess.
- Extension languages second-class citizens.
- Text only.
```c
func custom_prescription(Patient p) : Medicine
{
    if (p.penicillin_allergy())
       return NULL;
    Medicine m =
        \includegraphics[width=0.5\textwidth]{penicillin.png}
    candidate = generate(P, m);
    if (!check_with_doctor(candidate))
       return NULL;
    m.set_variable(R, candidate);
    return m;
}
```
Example (4)

```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,')
               > 0:
            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?
This research graciously funded by Oracle.
Boil down to the JVM
Boil down to the JVM

Meta-tracing to the rescue
RPython translation

Interpreter

RPython translator

L. Tratt  http://tratt.net/laurie/  Language Composition  2013-02-27  28 / 39
RPython translation

Interpreter

Optimised Interpreter

JIT

RPython translator
pc := 0
while 1:

    instr := load_next_instruction(pc)
    if instr == POP:
        stack.pop()
        pc += 1
    elif instr == BRANCH:
        off = load_branch_jump(pc)
        pc += off
    elif ...:
        ...

Observation: interpreters are big loops.
pc := 0
while 1:
    jit_merge_point(pc)
    instr := load_next_instruction(pc)
    if instr == POP:
        stack.pop()
        pc += 1
    elseif instr == BRANCH:
        off = load_branch_jump(pc)
        if off < 0: can_enter_jit(pc)
        pc += off
    elseif ...
    ...

Observation: interpreters are big loops.
Tracing JITs

User program (lang FL)

```python
if x < 0:
    x = x + 1
else:
    x = x + 2
x = x + 3
```
### User program (lang FL) | Trace when x is set to 6

<table>
<thead>
<tr>
<th>if x &lt; 0:</th>
<th>guard_type(x, int)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>guard_not_less_than(x, 0)</td>
</tr>
<tr>
<td>x = x + 1</td>
<td>guard_type(x, int)</td>
</tr>
<tr>
<td>else:</td>
<td>x = int_add(x, 2)</td>
</tr>
<tr>
<td></td>
<td>guard_type(x, int)</td>
</tr>
<tr>
<td>x = x + 2</td>
<td>x = int_add(x, 3)</td>
</tr>
<tr>
<td>x = x + 3</td>
<td></td>
</tr>
<tr>
<td>User program (lang <em>FL</em>)</td>
<td>Optimised trace</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>if x &lt; 0:</td>
<td>guard_type(x, int)</td>
</tr>
<tr>
<td>x = x + 1</td>
<td>guard_not_less_than(x, 0)</td>
</tr>
<tr>
<td>else:</td>
<td>x = int_add(x, 5)</td>
</tr>
<tr>
<td>x = x + 2</td>
<td></td>
</tr>
<tr>
<td>x = x + 3</td>
<td></td>
</tr>
</tbody>
</table>
## Converge 1 vs. Converge 2 VMs

<table>
<thead>
<tr>
<th></th>
<th>Converge 1</th>
<th>Converge 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (KLoc)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effort (man months)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[9x253]Converge 1 vs. Converge 2 VMs

L. Tratt [http://tratt.net/laurie/](http://tratt.net/laurie/)
## Converge 1 vs. Converge 2 VMs

<table>
<thead>
<tr>
<th></th>
<th>Converge 1</th>
<th>Converge 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (KLoc)</td>
<td>13</td>
<td></td>
</tr>
<tr>
<td>Effort (man months)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Converge 1 vs. Converge 2 VMs

<table>
<thead>
<tr>
<th></th>
<th>Converge 1</th>
<th>Converge 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (KLoc)</td>
<td>13</td>
<td>5.5</td>
</tr>
<tr>
<td>Effort (man months)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Converge 1 vs. Converge 2 VMs

<table>
<thead>
<tr>
<th></th>
<th>Converge 1</th>
<th>Converge 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (KLoc)</td>
<td>13</td>
<td>5.5</td>
</tr>
<tr>
<td>Effort (man months)</td>
<td>18</td>
<td></td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Converge 1 vs. Converge 2 VMs

<table>
<thead>
<tr>
<th></th>
<th>Converge 1</th>
<th>Converge 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (KLoc)</td>
<td>13</td>
<td>5.5</td>
</tr>
<tr>
<td>Effort (man months)</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>Performance</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Converge 1 vs. Converge 2 VMs

<table>
<thead>
<tr>
<th></th>
<th>Converge 1</th>
<th>Converge 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (KLoc)</td>
<td>13</td>
<td>5.5</td>
</tr>
<tr>
<td>Effort (man months)</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>Performance</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Converge 1</td>
<td>Converge 2</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>Size (KLoc)</td>
<td>13</td>
<td>5.5</td>
</tr>
<tr>
<td>Effort (man months)</td>
<td>18</td>
<td>3</td>
</tr>
<tr>
<td>Performance</td>
<td>x</td>
<td>2-150x</td>
</tr>
<tr>
<td></td>
<td>50000</td>
<td>5000000</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------</td>
<td>---------</td>
</tr>
<tr>
<td>C (GCC 4.6.3)</td>
<td>0.004 ± 0.002</td>
<td>0.179 ± 0.010</td>
</tr>
<tr>
<td>HotSpot (1.7.0_09)</td>
<td>0.107 ± 0.006</td>
<td>0.240 ± 0.010</td>
</tr>
<tr>
<td>Converge1 (git #68c795d2be)</td>
<td>2.053 ± 0.029</td>
<td>207.274 ± 3.048</td>
</tr>
<tr>
<td>Converge2 (2.0)</td>
<td>0.118 ± 0.004</td>
<td>1.914 ± 0.022</td>
</tr>
<tr>
<td>Lua (5.2.1)</td>
<td>0.201 ± 0.008</td>
<td>19.417 ± 0.474</td>
</tr>
<tr>
<td>LuaJIT2 (2.0.0)</td>
<td>0.014 ± 0.006</td>
<td>0.879 ± 0.016</td>
</tr>
<tr>
<td>CPython (2.7.3)</td>
<td>0.368 ± 0.010</td>
<td>35.072 ± 0.537</td>
</tr>
<tr>
<td>Jython (2.5.3)</td>
<td>1.820 ± 0.029</td>
<td>28.432 ± 0.466</td>
</tr>
<tr>
<td>PyPy–nonopt (1.9*)</td>
<td>0.127 ± 0.006</td>
<td>5.898 ± 0.071</td>
</tr>
<tr>
<td>PyPy (1.9)</td>
<td>0.069 ± 0.008</td>
<td>1.085 ± 0.014</td>
</tr>
<tr>
<td>Ruby (1.9.3-p327)</td>
<td>0.312 ± 0.008</td>
<td>29.819 ± 0.257</td>
</tr>
<tr>
<td>JRuby (1.7.1)</td>
<td>2.050 ± 0.039</td>
<td>10.576 ± 0.304</td>
</tr>
<tr>
<td>Language/C Compiler</td>
<td>10</td>
<td>±</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------</td>
<td>-------</td>
</tr>
<tr>
<td>C (GCC 4.6.3)</td>
<td>0.012</td>
<td>± 0.006</td>
</tr>
<tr>
<td>HotSpot (1.7.0_09)</td>
<td>0.109</td>
<td>± 0.010</td>
</tr>
<tr>
<td>Converge1 (git #68c795d2be)</td>
<td>9.931</td>
<td>± 0.102</td>
</tr>
<tr>
<td>Converge2 (2.0)</td>
<td>0.637</td>
<td>± 0.006</td>
</tr>
<tr>
<td>Lua (5.2.1)</td>
<td>0.665</td>
<td>± 0.024</td>
</tr>
<tr>
<td>LuaJIT2 (2.0.0)</td>
<td>0.085</td>
<td>± 0.006</td>
</tr>
<tr>
<td>CPython (2.7.3)</td>
<td>1.585</td>
<td>± 0.022</td>
</tr>
<tr>
<td>Jython (2.5.3)</td>
<td>2.820</td>
<td>± 0.069</td>
</tr>
<tr>
<td>PyPy–nonopt (1.9*)</td>
<td>0.515</td>
<td>± 0.010</td>
</tr>
<tr>
<td>PyPy (1.9)</td>
<td>0.267</td>
<td>± 0.006</td>
</tr>
<tr>
<td>Ruby (1.9.3-p327)</td>
<td>0.793</td>
<td>± 0.018</td>
</tr>
<tr>
<td>JRuby (1.7.1)</td>
<td>2.130</td>
<td>± 0.025</td>
</tr>
</tbody>
</table>
## Fannkuch-redux benchmark

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (GCC 4.6.3)</td>
<td>0.163 ± 0.006</td>
<td>1.992 ± 0.010</td>
</tr>
<tr>
<td>HotSpot (1.7.0_09)</td>
<td>0.350 ± 0.008</td>
<td>3.448 ± 0.029</td>
</tr>
<tr>
<td>Converge1 (git #68c795d2be)</td>
<td>†</td>
<td>†</td>
</tr>
<tr>
<td>Converge2 (2.0)</td>
<td>2.658 ± 0.041</td>
<td>33.484 ± 0.517</td>
</tr>
<tr>
<td>Lua (5.2.1)</td>
<td>7.683 ± 0.321</td>
<td>100.536 ± 2.475</td>
</tr>
<tr>
<td>LuaJIT2 (2.0.0)</td>
<td>0.339 ± 0.008</td>
<td>4.180 ± 0.010</td>
</tr>
<tr>
<td>CPython (2.7.3)</td>
<td>9.167 ± 0.237</td>
<td>114.001 ± 2.189</td>
</tr>
<tr>
<td>Jython (2.5.3)</td>
<td>7.776 ± 0.419</td>
<td>76.069 ± 4.753</td>
</tr>
<tr>
<td>PyPy–nonopt (1.9*)</td>
<td>1.402 ± 0.022</td>
<td>16.989 ± 0.220</td>
</tr>
<tr>
<td>PyPy (1.9)</td>
<td>1.256 ± 0.024</td>
<td>15.239 ± 0.223</td>
</tr>
<tr>
<td>Ruby (1.9.3-p327)</td>
<td>13.152 ± 0.200</td>
<td>172.098 ± 2.168</td>
</tr>
<tr>
<td>JRuby (1.7.1)</td>
<td>6.313 ± 0.127</td>
<td>61.934 ± 1.513</td>
</tr>
</tbody>
</table>
Composition of interpreters is feasible.
Composition of interpreters is feasible.

Challenges:
Composition of interpreters is feasible.

Challenges:
1. Isolation.

EPSRC 'Cooler' project starting June 2013.

L. Tratt http://tratt.net/laurie/
Composition of interpreters is feasible.

Challenges:
1. Isolation.
2. Communication.

EPSRC 'Cooler' project starting June 2013.
Composition of interpreters is feasible.

Challenges:
1. Isolation.
2. Communication.
Composition of interpreters is feasible.

Challenges:

1. Isolation.
2. Communication.

EPSRC ‘Cooler’ project starting June 2013.
Compose:
- parsers
- virtual machines
Compose:

- parsers *Incremental parsing*
- virtual machines
Compose:

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

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
- The impact of meta-tracing on VM design and implementation, Bolz, Tratt
- Converge: http://convergepl.org/
Thank you for listening