Language Composition

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

Software Development Team, King’s College London

2013-03-21
The programming language status quo limits us.
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.
The status quo
Languages conceptual size
Languages conceptual size

Normal programmers brain
Languages conceptual size

![Diagram showing the conceptual size of K&R C](image)
Languages conceptual size

Python
Languages conceptual size
Programming languages’ speed of light

\[
\begin{align*}
\gamma & \approx 0.999 \text{ for } v = 0.999c \\
\gamma & \approx 1.058 \text{ for } v = 0.9999c
\end{align*}
\]
Programming languages’ speed of light

![Graph showing the speed of light versus programmer productivity.](graph.png)
Programming languages’ speed of light

Machine code

$\mathcal{N}$

Programmer productivity

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

![Graph showing the speed of light in programming languages vs programmer productivity.](#)
Programming languages’ speed of light

![Graph showing the speed of light for different programming languages.](http://tratt.net/laurie/)

- Machine code
- K & R
- C
- Assembly
Programming languages’ speed of light

![Graph showing the speed of light for different programming languages]

Legend:
- Assembly
- Machine code
- K & R
- C
- Python

The graph plots programmer productivity against programming languages, illustrating their relative speeds of light.
Programming languages’ speed of light

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

Programmer productivity

S

0

C

0

0

Machine code

Scala

Python

K & R C

Assembly
Programming languages’ speed of light

![Graph showing the speed of light for various programming languages.]

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

Diminishing Returns

S

Programmer productivity

C

L. Tratt  http://tratt.net/laurie/  Language Composition  2013-03-21  5 / 39
Can S become too big?
Can S become too big?

C++
Wiggle room

Basic functionality

Goodies
Even worse
Even worse
Even worse

Basic functionality

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

Haskell / Ruby / Scala

DSL

DSL

DSL

L. Tratt
http://tratt.net/laurie/
Is this about DSLs?

Haskell / Ruby / Scala

DSL
DSL
DSL

L. Tratt
http://tratt.net/laurie/
Is this about DSLs?

Any language you want

L. Tratt http://tratt.net/laurie/
Part II

A way forward?
Idea: allow users to *compose* languages.
What does composition mean?
Language $\triangleq$ syntax + semantics
Language implementation \( \triangleq \) compiler + runtime
What does composition mean?

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

Compiler $\triangleq$ parser + code generator
What does composition mean?

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

Minimally compose:
- parsers
- virtual machines
Example (1)

```sql
for (pid : SELECT pid FROM personnel WHERE salary > 100000) {
    if (!is_worth_it(pid))
        UPDATE personnel SET salary=0 WHERE pid=pid;
}
```
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))
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))
    }
  }
}
Why aren’t we (me?) very good at it yet?
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.

**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** \$$\langle x \rangle: \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.
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.
func custom_prescription(Patient p) : Medicine
{
    if (p.penicillin_allergy())
        return NULL;

    Medicine m =

    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
... 

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.
Adding a JIT to an RPython interpreter

... pc := 0
while 1:
    jit_merge_point(pc)
    instr := load_next_instruction(pc)
    if instr == POP:
        stack.pop()
        pc += 1
    elif instr == BRANCH:
        off = load_branch_jump(pc)
        if off < 0: can_enter_jit(pc)
        pc += off
    elif ...
        ...

Observation: interpreters are big loops.
User program (lang FL)

```python
if x < 0:
    x = x + 1
else:
    x = x + 2
x = x + 3
```
Tracing JITs

User program (lang $FL$) | Trace when $x$ is set to 6
--- | ---
if $x < 0$: | guard$_{type}(x, \text{int})$
\quad $x = x + 1$ | guard$_{not\_less\_than}(x, 0)$
else: | guard$_{type}(x, \text{int})$
\quad $x = x + 2$ | $x = \text{int\_add}(x, 2)$
$x = x + 3$ | guard$_{type}(x, \text{int})$
\quad $x = \text{int\_add}(x, 3)$
### User program (lang FL) | Optimised trace
---|---
if $x < 0$:
  $x = x + 1$
else:
  $x = x + 2$
$x = x + 3$
guard\_type($x$, \text{int})$
guard\_not\_less\_than($x$, 0)$
x = \text{int\_add}($x$, 5)$
## 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>
## 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>
<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></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>
<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></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>2-150x</td>
</tr>
<tr>
<td></td>
<td>50000</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>---------</td>
<td>-----------</td>
</tr>
<tr>
<td>C (GCC 4.6.3)</td>
<td>0.004</td>
<td>± 0.002</td>
</tr>
<tr>
<td>HotSpot (1.7.0_09)</td>
<td>0.107</td>
<td>± 0.006</td>
</tr>
<tr>
<td>Converge1 (git #68c795d2be)</td>
<td>2.053</td>
<td>± 0.029</td>
</tr>
<tr>
<td>Converge2 (2.0)</td>
<td>0.118</td>
<td>± 0.004</td>
</tr>
<tr>
<td>Lua (5.2.1)</td>
<td>0.201</td>
<td>± 0.008</td>
</tr>
<tr>
<td>LuaJIT2 (2.0.0)</td>
<td>0.014</td>
<td>± 0.006</td>
</tr>
<tr>
<td>CPython (2.7.3)</td>
<td>0.368</td>
<td>± 0.010</td>
</tr>
<tr>
<td>Jython (2.5.3)</td>
<td>1.820</td>
<td>± 0.029</td>
</tr>
<tr>
<td>PyPy–nonopt (1.9*)</td>
<td>0.127</td>
<td>± 0.006</td>
</tr>
<tr>
<td>PyPy (1.9)</td>
<td>0.069</td>
<td>± 0.008</td>
</tr>
<tr>
<td>Ruby (1.9.3-p327)</td>
<td>0.312</td>
<td>± 0.008</td>
</tr>
<tr>
<td>JRuby (1.7.1)</td>
<td>2.050</td>
<td>± 0.039</td>
</tr>
</tbody>
</table>
## Dhrystone benchmark

<table>
<thead>
<tr>
<th></th>
<th>50000</th>
<th>5000000</th>
</tr>
</thead>
<tbody>
<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</td>
<td>10</td>
<td>100</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>C (GCC 4.6.3)</td>
<td>0.012 ± 0.006</td>
<td>0.079 ± 0.006</td>
</tr>
<tr>
<td>HotSpot (1.7.0_09)</td>
<td>0.109 ± 0.010</td>
<td>0.169 ± 0.014</td>
</tr>
<tr>
<td>Converge1 (git #68c795d2be)</td>
<td>9.931 ± 0.102</td>
<td>100.216 ± 1.356</td>
</tr>
<tr>
<td>Converge2 (2.0)</td>
<td>0.637 ± 0.006</td>
<td>2.850 ± 0.014</td>
</tr>
<tr>
<td>Lua (5.2.1)</td>
<td>0.665 ± 0.024</td>
<td>6.574 ± 0.139</td>
</tr>
<tr>
<td>LuaJIT2 (2.0.0)</td>
<td>0.085 ± 0.006</td>
<td>0.763 ± 0.010</td>
</tr>
<tr>
<td>CPython (2.7.3)</td>
<td>1.585 ± 0.022</td>
<td>15.698 ± 0.227</td>
</tr>
<tr>
<td>Jython (2.5.3)</td>
<td>2.820 ± 0.069</td>
<td>13.870 ± 0.345</td>
</tr>
<tr>
<td>PyPy–nonopt (1.9*)</td>
<td>0.515 ± 0.010</td>
<td>2.839 ± 0.016</td>
</tr>
<tr>
<td>PyPy (1.9)</td>
<td>0.267 ± 0.006</td>
<td>0.544 ± 0.008</td>
</tr>
<tr>
<td>Ruby (1.9.3-p327)</td>
<td>0.793 ± 0.018</td>
<td>7.159 ± 0.061</td>
</tr>
<tr>
<td>JRuby (1.7.1)</td>
<td>2.130 ± 0.025</td>
<td>3.640 ± 0.053</td>
</tr>
</tbody>
</table>
### Richards benchmark

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (GCC 4.6.3)</td>
<td>0.012 ± 0.006</td>
<td>0.079 ± 0.006</td>
</tr>
<tr>
<td>HotSpot (1.7.0_09)</td>
<td><strong>0.109 ± 0.010</strong></td>
<td><strong>0.169 ± 0.014</strong></td>
</tr>
<tr>
<td>Converge1 (git #68c795d2be)</td>
<td>9.931 ± 0.102</td>
<td>100.216 ± 1.356</td>
</tr>
<tr>
<td>Converge2 (2.0)</td>
<td>0.637 ± 0.006</td>
<td>2.850 ± 0.014</td>
</tr>
<tr>
<td>Lua (5.2.1)</td>
<td>0.665 ± 0.024</td>
<td>6.574 ± 0.139</td>
</tr>
<tr>
<td>LuaJIT2 (2.0.0)</td>
<td>0.085 ± 0.006</td>
<td>0.763 ± 0.010</td>
</tr>
<tr>
<td>CPython (2.7.3)</td>
<td>1.585 ± 0.022</td>
<td>15.698 ± 0.227</td>
</tr>
<tr>
<td>Jython (2.5.3)</td>
<td>2.820 ± 0.069</td>
<td>13.870 ± 0.345</td>
</tr>
<tr>
<td>PyPy–nonopt (1.9*)</td>
<td>0.515 ± 0.010</td>
<td>2.839 ± 0.016</td>
</tr>
<tr>
<td>PyPy (1.9)</td>
<td><strong>0.267 ± 0.006</strong></td>
<td><strong>0.544 ± 0.008</strong></td>
</tr>
<tr>
<td>Ruby (1.9.3-p327)</td>
<td>0.793 ± 0.018</td>
<td>7.159 ± 0.061</td>
</tr>
<tr>
<td>JRuby (1.7.1)</td>
<td>2.130 ± 0.025</td>
<td>3.640 ± 0.053</td>
</tr>
</tbody>
</table>
### Richards benchmark

<table>
<thead>
<tr>
<th></th>
<th>10</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (GCC 4.6.3)</td>
<td>0.012 ± 0.006</td>
<td>0.079 ± 0.006</td>
</tr>
<tr>
<td>HotSpot (1.7.0_09)</td>
<td><strong>0.109 ± 0.010</strong></td>
<td><strong>0.169 ± 0.014</strong></td>
</tr>
<tr>
<td>Converge1 (git #68c795d2be)</td>
<td>9.931 ± 0.102</td>
<td>100.216 ± 1.356</td>
</tr>
<tr>
<td>Converge2 (2.0)</td>
<td>0.637 ± 0.006</td>
<td>2.850 ± 0.014</td>
</tr>
<tr>
<td>Lua (5.2.1)</td>
<td>0.665 ± 0.024</td>
<td>6.574 ± 0.139</td>
</tr>
<tr>
<td>LuaJIT2 (2.0.0)</td>
<td>0.085 ± 0.006</td>
<td>0.763 ± 0.010</td>
</tr>
<tr>
<td>CPython (2.7.3)</td>
<td>1.585 ± 0.022</td>
<td>15.698 ± 0.227</td>
</tr>
<tr>
<td>Jython (2.5.3)</td>
<td>2.820 ± 0.069</td>
<td>13.870 ± 0.345</td>
</tr>
<tr>
<td>PyPy–nonopt (1.9*)</td>
<td>0.515 ± 0.010</td>
<td>2.839 ± 0.016</td>
</tr>
<tr>
<td>PyPy (1.9)</td>
<td><strong>0.267 ± 0.006</strong></td>
<td><strong>0.544 ± 0.008</strong></td>
</tr>
<tr>
<td>Ruby (1.9.3-p327)</td>
<td>0.793 ± 0.018</td>
<td>7.159 ± 0.061</td>
</tr>
<tr>
<td>JRuby (1.7.1)</td>
<td>2.130 ± 0.025</td>
<td>3.640 ± 0.053</td>
</tr>
<tr>
<td>Language</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----</td>
<td>-----</td>
</tr>
<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.
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.
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
Thank you for listening