Statistical properties and evaluation of random λ-terms – a survey

Maciej Bendkowski[1]

[1]Theoretical Computer Science Departament
Jagiellonian University in Kraków

EUTypes meeting
Kraków 23-24 February 2019

Why consider random λ-terms?

- Random generation of structures containing *scoped variables* with applications to software testing. For instance, the following *random* λ-term[1] exhibited a bug in GHC's strictness analizer:

$$(\lambda a.\, \mathsf{seq}\; a\; (\mathsf{seq}\; a\; \perp)\; \mathsf{tail})(\lambda a.\, \mathsf{seq} \perp (+1))$$

were

$$\perp \;\mathsf{seq}\; a = \perp$$
$$a \;\mathsf{seq}\; b = b.$$

[1] M. Pałka. *Random Structured Test Data Generation for Black-Box Testing*. PhD thesis. 2014.

Why consider random λ-terms?

- Random generation of structures containing *scoped variables* with applications to software testing. For instance, the following *random* λ-term[1] exhibited a bug in GHC's strictness analizer:

$$(\lambda a.\, \mathsf{seq}\; a\; (\mathsf{seq}\; a \perp)\; \mathsf{tail})(\lambda a.\, \mathsf{seq} \perp (+1))$$

were

$$\perp\ \mathsf{seq}\ \ a = \perp$$
$$a\ \mathsf{seq}\ \ b = b.$$

- Such structures are difficult to analyse and inspire the development of new combinatorial techniques.

[1] M. Pałka. *Random Structured Test Data Generation for Black-Box Testing*. PhD thesis. 2014.

THEORETICAL COMPUTER SCIENCE
Jagiellonian University

How to represent λ-terms?

Let us focus on λ-terms with de Bruijn indices.

$T ::= \lambda T \mid T\ T \mid N$
$N ::= 0 \mid \mathsf{S}\ N.$

$$\lambda xyz.xz(yz) \equiv \lambda\lambda\lambda \underline{2}\underline{0}(\underline{1}\underline{0})$$

Benefits:

[2] Requires some non-trivial extensions of well-known tools.

How to represent λ-terms?

Let us focus on λ-terms with de Bruijn indices.

$T ::= \lambda T \mid T\ T \mid N$
$N ::= 0 \mid S\ N.$

$$\lambda xyz.xz(yz) \equiv \lambda\lambda\lambda \underline{2}\underline{0}(\underline{1}\underline{0})$$

Benefits:

- We do not have to worry about α-equivalence.

[2] Requires some non-trivial extensions of well-known tools.

How to represent λ-terms?

Let us focus on λ-terms with de Bruijn indices.

$T ::= \lambda T \mid T\ T \mid N$
$N ::= 0 \mid \mathsf{S}\ N.$

$$\lambda xyz.xz(yz) \equiv \lambda\lambda\lambda\underline{2}\underline{0}(\underline{1}\underline{0})$$

Benefits:

- We do not have to worry about α-equivalence.
- It is perhaps the simplest model we can start to analyse.

[2] Requires some non-trivial extensions of well-known tools.

THEORETICAL COMPUTER SCIENCE
Jagiellonian University

How to represent λ-terms?

Let us focus on λ-terms with de Bruijn indices.

$T ::= \lambda T \mid T\ T \mid N$
$N ::= 0 \mid S\ N.$

$$\lambda xyz.xz(yz) \equiv \lambda\lambda\lambda \underline{2}\underline{0}(\underline{1}\underline{0})$$

Benefits:

- We do not have to worry about α-equivalence.
- It is perhaps the simplest model we can start to analyse.
- (Some) techniques of *analytic combinatorics* are applicable[2].

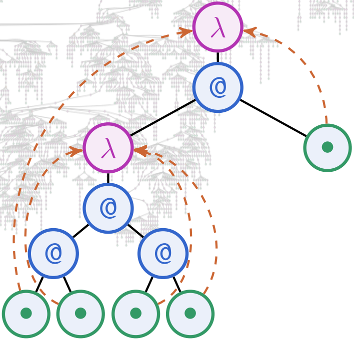
[2] Requires some non-trivial extensions of well-known tools.

THEORETICAL COMPUTER SCIENCE Jagiellonian University

Alternative representations?

What about a representation were only *closed* terms are allowed?
Also, how should we measure the term size?

Assume that abstractions, applications and variables contribute some *weight* to the term size.



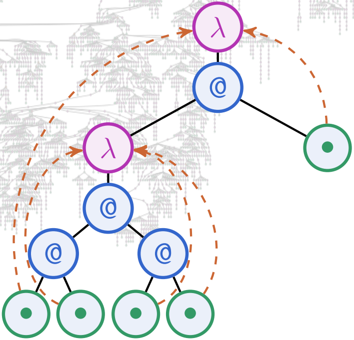
Variable weight options:

- No weight.

Alternative representations?

What about a representation were only *closed* terms are allowed?
Also, how should we measure the term size?

Assume that abstractions, applications and variables contribute some *weight* to the term size.



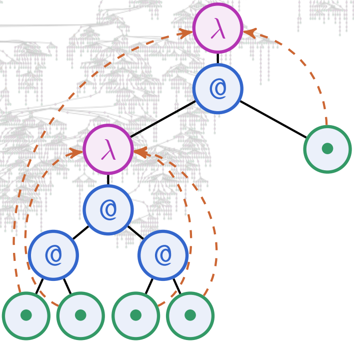
Variable weight options:

- No weight.
- Constant weight.

Alternative representations?

What about a representation were only *closed* terms are allowed?
Also, how should we measure the term size?

Assume that abstractions, applications and variables contribute some *weight* to the term size.



Variable weight options:

- No weight.
- Constant weight.
- Weight proportional to the binder distance (cf. de Bruijn indices).

Counting λ-terms – how hard can it be?

Let's start with the following *counting problem*. Given n, what is the number Λ_n of λ-terms of size n? How does Λ_n change with $n \to \infty$?

THEORETICAL COMPUTER SCIENCE
Jagiellonian University

Counting λ-terms – how hard can it be?

Let's start with the following *counting problem*. Given n, what is the number Λ_n of λ-terms of size n? How does Λ_n change with $n \to \infty$?

Bodini, Gardy, Gittenberger, Jacquot '13

Assume that variables contribute constant weight one. For special cases, such as linear or affine terms, the counting problem admits asymptotic estimates. In general, the problem remains <u>unsolved</u>.

Counting λ-terms – how hard can it be?

Let's start with the following *counting problem*. Given n, what is the number Λ_n of λ-terms of size n? How does Λ_n change with $n \to \infty$?

Bodini, Gardy, Gittenberger, Jacquot '13

Assume that variables contribute constant weight one. For special cases, such as linear or affine terms, the counting problem admits asymptotic estimates. In general, the problem remains unsolved.

David, Grygiel, Kozik, Raffalli, Theyssier, Zaionc '13

Assume that variables contribute no weight. The counting problem is still unsolved, however asymptotically almost all λ-terms are shown to be strongly normalizing.

Back to the de Bruijn representation

What if we assume that $\underline{n} \equiv S^{(n)}0$ has *natural* weight $n + 1$?
Recall that in the de Bruijn representation we have

$$T ::= \lambda T \mid T\ T \mid N$$
$$N ::= 0 \mid \mathrm{S}\ N.$$

Back to the de Bruijn representation

What if we assume that $\underline{n} \equiv S^{(n)}0$ has *natural* weight $n + 1$?
Recall that in the de Bruijn representation we have

$$T ::= \lambda T \mid T\ T \mid N$$
$$N ::= 0 \mid \mathrm{S}\ N.$$

B., Grygiel, Lescanne, Zaionc '16

In the de Bruijn representation, the counting problem admits asymptotic estimates. More importantly, asymptotically almost all λ-terms are <u>not strongly normalising</u>.

THEORETICAL COMPUTER SCIENCE
Jagiellonian University

What about closed λ-terms?

Let T_m denote the class of *m-open λ-terms*, i.e. terms which after prepending m head abstractions become closed. Then, the set T_0 of closed terms satisfies the following *infinite* specification:

$$
\begin{cases}
T_0 ::= \lambda T_1 \mid T_0 T_0 \\
T_1 ::= \lambda T_2 \mid T_1 T_1 \mid \underline{0} \\
T_2 ::= \lambda T_3 \mid T_2 T_2 \mid \underline{0} \mid \underline{1} \\
\qquad \ldots \\
T_m ::= \lambda T_{m+1} \mid T_m T_m \mid \underline{0} \mid \underline{1} \mid \cdots \mid \underline{\text{m-1}} \\
\qquad \ldots
\end{cases}
$$

Bodini, Gittenberger, Gołębiewski '18

The counting problem for closed λ-terms admits asymptotic estimates. Moreover, it is possible to *sample* (unformly) random, closed λ-terms of large, target size n.

THEORETICAL COMPUTER SCIENCE Jagiellonian University

Sampling closed, simply-typed λ-terms?

For some applications, sampling closed terms can be insufficient. Sometimes, we need stronger guarantees about the *properties* of generated terms, such as *termination*.

B., Grygiel, Tarau '17

Combining samplers for closed λ-terms and a careful use of rejection, it is possible design practical samplers for closed, simply-typed λ-terms for *moderate* target sizes.

Benchmark term sizes:[3]

[3] Using a standard PC as a reference point.

Sampling closed, simply-typed λ-terms?

For some applications, sampling closed terms can be insufficient. Sometimes, we need stronger guarantees about the *properties* of generated terms, such as *termination*.

B., Grygiel, Tarau '17

Combining samplers for closed λ-terms and a careful use of rejection, it is possible design practical samplers for closed, simply-typed λ-terms for *moderate* target sizes.

Benchmark term sizes:[3]

- Closed λ-terms: $n \leq 1\,000\,000\,000$.

[3] Using a standard PC as a reference point.

THEORETICAL COMPUTER SCIENCE
Jagiellonian University

Sampling closed, simply-typed λ-terms?

For some applications, sampling closed terms can be insufficient. Sometimes, we need stronger guarantees about the *properties* of generated terms, such as *termination*.

B., Grygiel, Tarau '17

Combining samplers for closed λ-terms and a careful use of rejection, it is possible design practical samplers for closed, simply-typed λ-terms for *moderate* target sizes.

Benchmark term sizes:[3]

- Closed λ-terms: $n \leq 1\,000\,000\,000$.
- Closed, simply-typed λ-terms: $n \leq 250$.

[3] Using a standard PC as a reference point.

THEORETICAL COMPUTER SCIENCE Jagiellonian University

Typical properties of λ-terms

B., Bodini, Dovgal '18

Suppose that we sample a large, random (unrestricted) λ-term of size n. What shape and properties should we expect?

Parameter	**Mean, $\sim$**		**Distribution**	
	plain	**closed**	**plain**	**closed**
Variables	$0.307n$		Normal	
Abstractions	$0.258n$		Normal	
Successors	$0.129n$		Normal	
Redexes	$0.091n$		Normal	
Index value	0.420		Geometric	
Head abstractions	0.420	1.447	Geometric	Discrete
m-openness	2.019	0	Discrete	trivial
Free variables	5.722	0	Discrete	trivial

THEORETICAL COMPUTER SCIENCE
Jagiellonian University

Even better control over generated λ-terms?

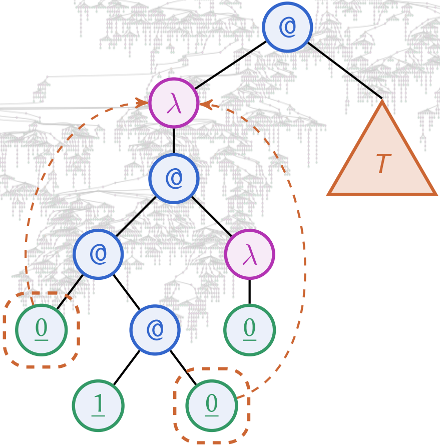
B., Bodini, Dovgal ’18

Using quite general tools[4], it is possible to *skew* the uniform distribution of generated λ-terms, and gain additional control over some of their parameters. For instance, request more abstractions or favour larger de Bruijn indices.

[4] See also https://github.com/maciej-bendkowski/boltzmann-brain

Typical properties of λ-terms (II)

What about evaluation of large, random terms? For instance, how long does it take *on average* to find the leftmost-outermost redex in a random term?



It is always a constant time overhead:

- Plain terms ≈ 6.222,
- Closed terms ≈ 6.054.

What can be said about the typical cost of substitution?

Substitution resolution and explicit substitutions

It is possible to carry out substitutions in various ways, e.g. substitutions can be carried out strictly or suspended until truely required (i.e. resolved non-strictly).

Let us therefore consider a *concrete implementation* of substitution resolution, say $\lambda\upsilon$[5] — a λ-calculus with explicit substitutions — and analyse substitution resolution therin, instead.

$$
\begin{aligned}
t &::= \underline{n} \mid \lambda t \mid tt \mid t[s] \\
s &::= t/ \mid \Uparrow(s) \mid \uparrow \\
\underline{n} &::= \underline{0} \mid S\underline{n}.
\end{aligned}
$$

$$
\begin{aligned}
(\lambda a)b &\to a[b/] && \text{(Beta)} \\
(ab)[s] &\to a[s](b[s]) && \text{(App)} \\
(\lambda a)[s] &\to \lambda(a[\Uparrow(s)]) && \text{(Lambda)} \\
\underline{0}[a/] &\to a && \text{(FVar)} \\
(S\underline{n})[a/] &\to \underline{n} && \text{(RVar)} \\
\underline{0}[\Uparrow(s)] &\to \underline{0} && \text{(FVarLift)} \\
(S\underline{n})[\Uparrow(s)] &\to \underline{n}[s][\uparrow] && \text{(RVarLift)} \\
\underline{n}[\uparrow] &\to S\underline{n}. && \text{(VarShift)}
\end{aligned}
$$

[5] P. Lescanne. *From $\lambda\sigma$ to $\lambda\upsilon$ – a journey through calculi of explicit substitutions.* 1994.

υ-reduction grammars

Instead of a single β rewriting rule, $\lambda\upsilon$ consists of a (Beta) rule and seven auxiliary υ rules governing the execution of substitutions. Luckily, compared with classic λ-calculus they are much easier to analyse in quantitative terms.

B. '19

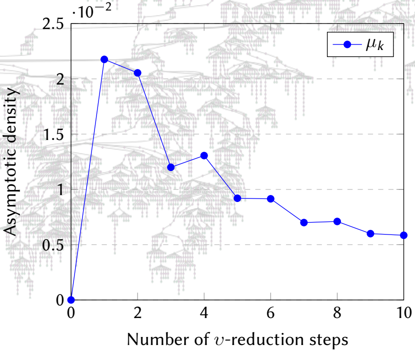
For all $k \geq 0$, the set G_k of $\lambda\upsilon$-terms which reduce to their υ-normal forms (i.e. pure forms without explicit substitutions) in k leftmost-outermost υ rewriting steps forms a *regular tree language.*

$$
\begin{aligned}
G_0 &\to \lambda G_0 \mid G_0 G_0 \mid \underline{\mathrm{n}} \\
G_1 &\to \lambda G_1 \mid G_0 G_1 \mid G_1 G_0 \\
&\quad \mid \underline{0}[(G_0 G_0)/] \mid \underline{0}[\lambda G_0/] \mid \underline{0}[\underline{\mathrm{n}}/] \\
&\quad \mid (S\underline{\mathrm{n}})[t/] \mid \underline{0}[\Uparrow (s)] \mid \underline{\mathrm{n}}[\uparrow]
\end{aligned}
$$

$$\vdots$$

υ-reduction grammars (II)

$(G_k)_k$ admits a neat hierarchical structure and can be analysed using standard techniques of *analytic combinatorics.* In particular, for any fixed $k \geq 0$, the fraction of terms υ-normalising in k steps tends to a computable limit μ_k as the term size tends to infinity.



k	μ_k
0	0.
1	0.02176
2	0.02054
3	0.01200
4	0.01306
5	0.00920
6	0.00915
7	0.00700
8	0.00710
9	0.00600
10	0.00585

Conclusions

- Depending on the representation, typical λ-terms admit contrasting properties, e.g. strong normalisation or lack thereof.

Conclusions

- Depending on the representation, typical λ-terms admit contrasting properties, e.g. strong normalisation or lack thereof.
- Representations using de Bruijn indices in unary notation are now well-understood and admit effective tools for random generation (at least in the untyped universe).

THEORETICAL COMPUTER SCIENCE
Jagiellonian University

Conclusions

- Depending on the representation, typical λ-terms admit contrasting properties, e.g. strong normalisation or lack thereof.
- Representations using de Bruijn indices in unary notation are now well-understood and admit effective tools for random generation (at least in the untyped universe).
- Although the techniques of analytic combinatoric are quite daunting, it is possible to use them in order to analyse the operational costs of substitution in λ-calculus.