On Uniquely Closable and Uniquely Typable Skeletons of Lambda Terms

Olivier Bodini and Paul Tarau

Laboratoire d'Informatique de Paris-Nord
UMR CNRS 7030
olivier.bodini@lipn.univ-paris13.fr

Department of Computer Science and Engineering
University of North Texas
paul.tarau@unt.edu

LOPSTR'2017

Research supported by NSF grant **1423324**

Overview

- uniquely closable skeletons of lambda terms are Motzkin-trees that predetermine the unique closed lambda term that can be obtained by labeling their leaves with de Bruijn indices
- uniquely typable skeletons of closed lambda terms predetermine the unique simply-typed lambda term that can be obtained by labeling their leaves with de Bruijn indices
- we derive, through a sequence of logic program transformations, efficient code for their combinatorial generation
- we obtain context-free grammars describing closable and uniquely closable skeletons of lambda terms
- $\Rightarrow$ we study them in depth with tools from analytic combinatorics
- for the difficult case of (uniquely) typable terms: empirical study $\Rightarrow$ open problems about density and asymptotic behavior
- we establish a connection between the two classes of terms: uniquely typable closed lambda term skeletons of size $3n+1$ are in a bijection with binary trees of size n

Motivation

- the study of lambda terms has theoretical ramifications:
- connection to proofs in intuitionistic logic via the Curry-Howard correspondence - the class of simply typed lambda terms
- lambda terms are used in the internal representations of compilers for functional programming languages and proof assistants
- generation of large random lambda terms helps with automated testing
- combinatorial properties need to be well understood for random generation of large terms (e.g. via Boltzmann samplers)

Our focus:

- binary-unary trees that are obtained from lambda terms in de Bruijn form, represented as trees, by erasing the de Bruijn indices labeling variables at their leaves
- a surprising fact: such *skeletons* of the lambda terms turn out to predetermine non-trivial properties the lambda terms they host, e.g., if such terms are closed or simply-typed
- $\Rightarrow$ what are the cases when unique such terms exist?
- our declarative meta-language is Prolog
  - easy combinatorial generation via backtracking over the set of all answers
  - a Definite Clause Grammar (DCG) enforces size constraints
  - we place more complex constraints at points in the code where they ensure the earliest possible pruning of the search space
  - program transformations allow us to derive step-by-step faster and simpler expressions of the underlying combinatorial mechanisms

Closed Lambda Terms and their Motzkin-trees Skeletons

- a *Motzkin tree* (also called binary-unary tree) is a rooted ordered tree built from binary nodes, unary nodes and leaf nodes
- the set of Motzkin trees can be seen as the free algebra generated by the constructors `v/0`, `l/1` and `a/2`
- lambda terms in de Bruijn form: the free algebra generated by the constructors `l/1`, and `a/2` and leaves labeled with natural numbers wrapped with the constructor `v/1`
- lambda term in de Bruijn form is *closed* if for each of its de Bruijn indices it exists a lambda binder to which it points, on the path to the root of the tree representing the term
- they are counted by sequence **A135501** in OEIS
- skeleton of a lambda term: the Motzkin tree obtained by erasing the labels at its leaves

Closable and Unclosable Skeletons

- we call a Motzkin tree *closable* if it is the skeleton of at least one closed lambda term
- the predicate `isClosable/1` tests if it exists a closed lambda term having `X` as its skeleton
- for each lambda binder it increments a count `V` (starting at `0`), and ensures that it is strictly positive for all leaf nodes

```
isClosable(X):-isClosable(X,0).

isClosable(v,V):-V>0.
isClosable(l(A),V):-succ(V,NewV),isClosable(A,NewV).
isClosable(a(A,B),V):-isClosable(A,V),isClosable(B,V).
```

- generators for closable and unclosable skeletons: by filtering the answers of a Motzkin tree generator `motSkel/2`

```
closableSkel(N,X):-motSkel(N,X),isClosable(X).
unClosableSkel(N,X):-motSkel(N,X),not(isClosable(X)).
```

Some Closable and Unclosable Motzkin Skeletons

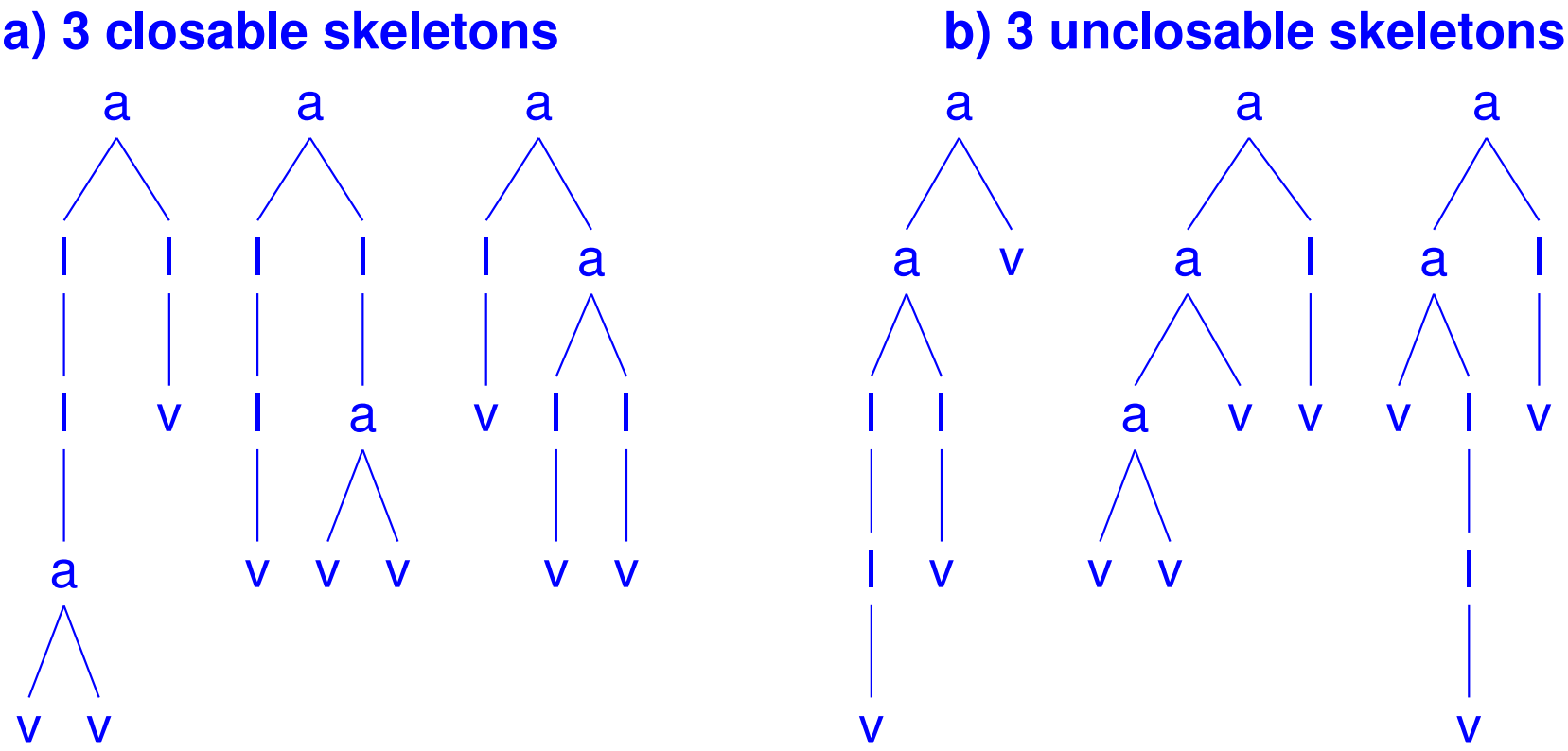


Figure: Closable vs. unclosable skeletons of size 7

Size definition: each constructor costs as much as its arity

Proposition

*The set of terms of size n for the size definition {*application=2, lambda=1, variable=0*} is equal to the set of terms of size* $n+1$ *for the size definition {*application=1, lambda=1, variable=1*}.*

- true, given that the number of leaves in a Motzkin tree is the number of binary nodes + *1*
- the term $l(a(v(0), v(0)))$ will have size $3 = 1+2$ with our definition, which corresponds to size $4 = 1+1+1+1$ using the size definition for sequence **A135501**
- our size definition is implemented as

```
l(SX,X):-succ(X,SX). % for l/1 constructors
a-->l,l.             % for a/2 constructors
```

with Prolog's DCG notation controlling the consumption of size units from `N` to `0`

Deriving a CF Grammar

Proposition

A Motzkin tree is a skeleton of a closed lambda term if and only if it exists at least one lambda binder on each path from the leaf to the root.

- we can derive a grammar generating closable skeletons
- at least one lambda (`l/1` constructor) on each path
- $\Rightarrow$ Motzkin trees below the `l/1` constructor contain only a/2 branches and leaves
- this also runs about 3 times as fast as `closableSkel/2`

```
closable(N,X):-closable(X,N,0).

closable(l(Z))-->l,motSkel(Z). % a/2 and leaves only
closable(a(X,Y))-->a,closable(X),closable(Y).
```

Analytic Combinatorics $\Rightarrow$ Density of Closable Skeletons

- $M(z) = \sum m_n z^n$ the ordinary generating function for Motzkin trees (m_n is the number of Motzkin trees of size n)
- $M(z)$ follows the algebraic functional equation $M = z + zM + zM^2$
- $M(z) = \frac{1-z-\sqrt{-3z^2-2z+1}}{2z}$
- classical result: asymptotically, the number m_n of Motzkin trees of size n is equivalent to $\frac{\sqrt{3}}{2\sqrt{\pi}} 3^n n^{-3/2}$
- from our grammar definition: the ordinary generating function $C(z)$ for closable lambda terms is $C(z) = zC(z)^2 + zM(z)$.
- consequently, $C(z) = \frac{1-\sqrt{2z\sqrt{-3z^2-2z+1}+2z^2-2z+1}}{2z}$
- Flajolet-Odlysko transfer theorems $\Rightarrow$
- $c_n \sim \frac{\sqrt{15}}{10\sqrt{\pi}} 3^n n^{-3/2}$.

$\Rightarrow$ when n tends to the infinity, the proportion of closable lambda term skeletons tends to $\frac{1}{\sqrt{5}} \doteq 44.7\%$.

An Efficient Recurrence Relation

- we can calculate very efficiently the coefficients c_n
- from the $C(z) = zC(z)^2 + zM(z)$, it follows that $C(z)$ satisfies $z^2C(z)^4 - 2zC(z)^3 + (-z^2 + z + 1)C(z)^2 + (z-1)C(z) + z^2$
- we derive, after several computation steps:

$$
\begin{aligned}
&\left(1200n^5 + 18480n^4 + 90816n^3 + 161088n^2 + 87552n\right) c_n + \\
&\left(800n^5 + 13520n^4 + 79024n^3 + 202312n^2 + 231768n + 95760\right) c_{n+1} + \\
&\left(-100n^5 - 1840n^4 - 12848n^3 - 38792n^2 - 44100n - 9576\right) c_{n+2} + \\
&\left(-100n^5 - 1990n^4 - 14648n^3 - 48254n^2 - 66276n - 23940\right) c_{n+3} + \\
&\left(-225n^5 - 4815n^4 - 38883n^3 - 147519n^2 - 260286n - 167580\right) c_{n+4} + \\
&\left(150n^5 + 3435n^4 + 29817n^3 + 120441n^2 + 218739n + 131670\right) c_{n+5} + \\
&\left(-25n^5 - 610n^4 - 5642n^3 - 24128n^2 - 45405n - 26334\right) c_{n+6} = 0
\end{aligned}
$$

with the initial conditions $c_0 = 0, c_1 = 0, c_2 = 1, c_3 = 1, c_4 = 2, c_5 = 5$

Uniquely Closable Skeletons

We call a skeleton *uniquely closable* if it exists exactly one closed lambda term having it as its skeleton.

Proposition

A skeleton is uniquely closable if and only if exactly one lambda binder is available above each of its leaf nodes.

Proof.

Note that if more than one were available for any leaf `v`, one could choose more then one de Bruijn index at the corresponding leaf `v/1` of a lambda term, resulting in more than one possible lambda terms having the given skeleton. □

A Grammar for Uniquely Closable Skeletons

By specializing with respect to having or not having a lambda binder above, we obtain the predicate `uniquelyClosable/2` which mimics a context-free grammar generating all uniquely closable skeletons of a given size.

```
uniquelyClosable(N,X):-uniquelyClosable(X,N,0).

uniquelyClosable(l(A))-->l,closedAbove(A).
uniquelyClosable(a(A,B))-->a,
  uniquelyClosable(A),
  uniquelyClosable(B).

closedAbove(v)-->[].
closedAbove(a(A,B))-->a,closedAbove(A),closedAbove(B).
```

Our program transformations result in code running an order of magnitude faster than the original specification, with all counts up to size `30` obtained in less than a minute.

Growth of Counts of Uniquely Closable Skeletons

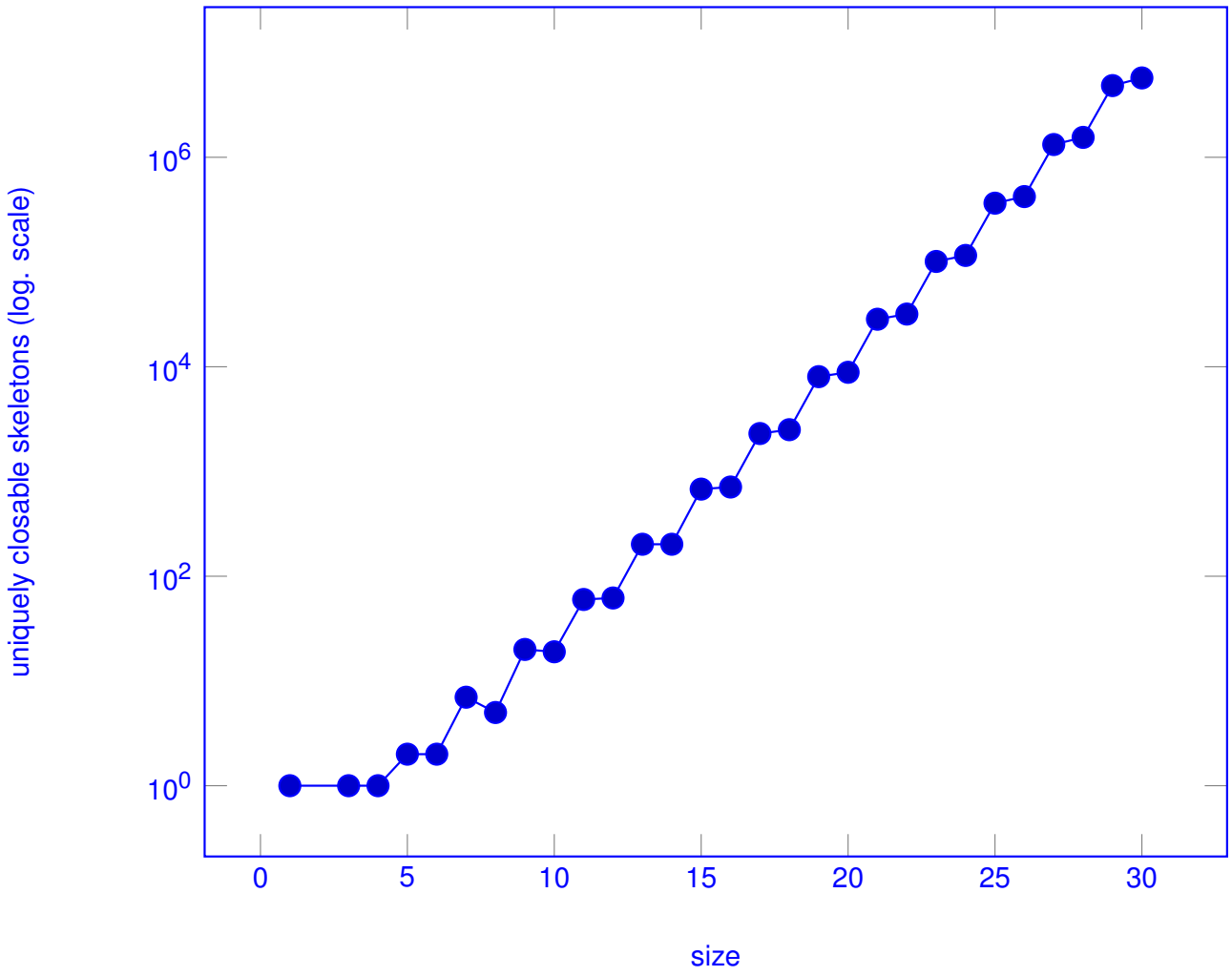


Figure: Uniquely closable skeletons by increasing sizes

The Asymptotics of Uniquely Closable Skeletons

- $B(z)$: the ordinary generating function for binary trees
- the series $B(z)$ follows the algebraic functional equation $B = z + zM^2$ and consequently $B(z) = \frac{1-\sqrt{-4z^2+1}}{2z}$.
- the ordinary generating function $U(z)$ for uniquely closable lambda terms satisfies $U(z) = zU(z)^2 + zB(z)$.
- consequently, $U(z) = \frac{1-\sqrt{2z\sqrt{-4z^2+1}-2z+1}}{2z}$.
- we use the Flajolet-Odlysko transfer theorems
- the asymptotics of the number u_n of uniquely closable skeletons: $u_n \sim \frac{2^{1/4+n}}{4Gamma(3/4)n^{5/4}}$.

An Efficient Recurrence Relation

- like for $C(z)$ we can calculate quickly the coefficients u_n
- $U(z)$: $z^2U(z)^4 - 2zU(z)^3 + (z+1)U(z)^2 - U(z) + z^2 = 0$
- we deduce a linear differential equation:

$$0 = -128z^5 - 40z^4 + 52z^3 + 18z^2 - 6z + (16z^5 + 56z^4 - 20z^3 - 20z^2 + 8z - 2)U(z) +$$
$$(512z^8 - 512z^7 - 320z^6 + 96z^5 + 144z^4 + 16z^3 - 24z^2 - 6z + 2)(\tfrac{\mathrm{d}}{\mathrm{d}z}U(z)) +$$
$$(256z^9 - 128z^8 - 128z^7 - 32z^6 + 64z^5 + 24z^4 - 16z^3 - 2z^2 + z)(\tfrac{\mathrm{d}^2}{\mathrm{d}z^2}U(z))$$

with the initial condition $U(0) = 0$.

- we efficiently compute u_n using the P-recurrence:

$$\left(256n^2 + 256n\right)u_n + \left(-128n^2 - 640n - 512\right)u_{n+1} +$$
$$\left(-128n^2 - 704n - 880\right)u_{n+2} + \left(-32n^2 - 64n + 152\right)u_{n+3} +$$
$$\left(64n^2 + 592n + 1324\right)u_{n+4} + \left(24n^2 + 232n + 540\right)u_{n+5} +$$
$$\left(-16n^2 - 200n - 616\right)u_{n+6} +$$
$$\left(-2n^2 - 32n - 128\right)u_{n+7} + \left(n^2 + 17n + 72\right)u_{n+8} = 0$$

with the initial conditions $u_0 = 0, u_1 = 0, u_2 = 1, u_3 = 0, u_4 = 1, u_5 = 1, u_6 = 2, u_7 = 2$

Typable and Untypable Closable Skeletons

- we call a Motzkin skeleton *typable* if it exists at least one simply-typed closed lambda term having it as its skeleton
- an *untypable* skeleton is a closable skeleton for which no such term exists
- we design efficient the Prolog code by *interleaving*:
  - term generation,
  - checking for closedness
  - type inference steps, via unification with occurs check
- we design a *two stage program*:
  - the first stage generates code
  - the second, executes it, via Prolog's metacall
  - the second stage also ensures that the terms generated are closed

The Two Stage Code Interleaving Generation and Type Inference

- `genSkelEqs/4` generates type unification equations
- if satisfied by a closed lambda term, they ensure that the term is simply-typable

```
genSkelEqs(N,X,T,Eqs):-genSkelEqs(X,T,[],Eqs,true,N,0).

genSkelEqs(v,V,Vs,(el(V,Vs),Es),Es)-->{Vs=[_|_]}.
genSkelEqs(l(A),(S->T),Vs,Es1,Es2)-->l,genSkelEqs(A,T,[S|Vs],Es1,Es
genSkelEqs(a(A,B),T,Vs,Es1,Es3)-->a,genSkelEqs(A,(S->T),Vs,Es1,Es2)
  genSkelEqs(B,S,Vs,Es2,Es3).

el(V,Vs):-member(V0,Vs),unify_with_occurs_check(V0,V).
```

- each lambda binder adds a new type variable to the list (starting empty at the root) on the way down to a leaf node
- a term is then closed if the list of those variables `Vs` is not empty at each leaf node

The Code, continued

- to generate the typable terms, one simply *executes the equations* `Eqs`:

```
typableClosedTerm(N,Term):-genSkelEqs(N,Term,_,Eqs),Eqs.
```

- `typableSkel/2` generates skeletons that are typable by running the same equations `Eqs` and ensuring they have *at least one solution*, using the Prolog built-in `once/1`
- `untypableSkel/2` succeeds, when the negation of these equations succeeds, indicating that no simply-typed lambda term exists having the given skeleton
- this is much faster than naively generating all the closed lambda terms and then finding their distinct skeletons

```
typableSkel(N,Skel):-genSkelEqs(N,Skel,_,Eqs),once(Eqs).
untypableSkel(N,Skel):-genSkelEqs(N,Skel,_,Eqs),not(Eqs).
```

Examples of Typeable and Untypable Skeletons

In Fig. 3 we show 3 typable and 3 untypable Motzkin skeletons.

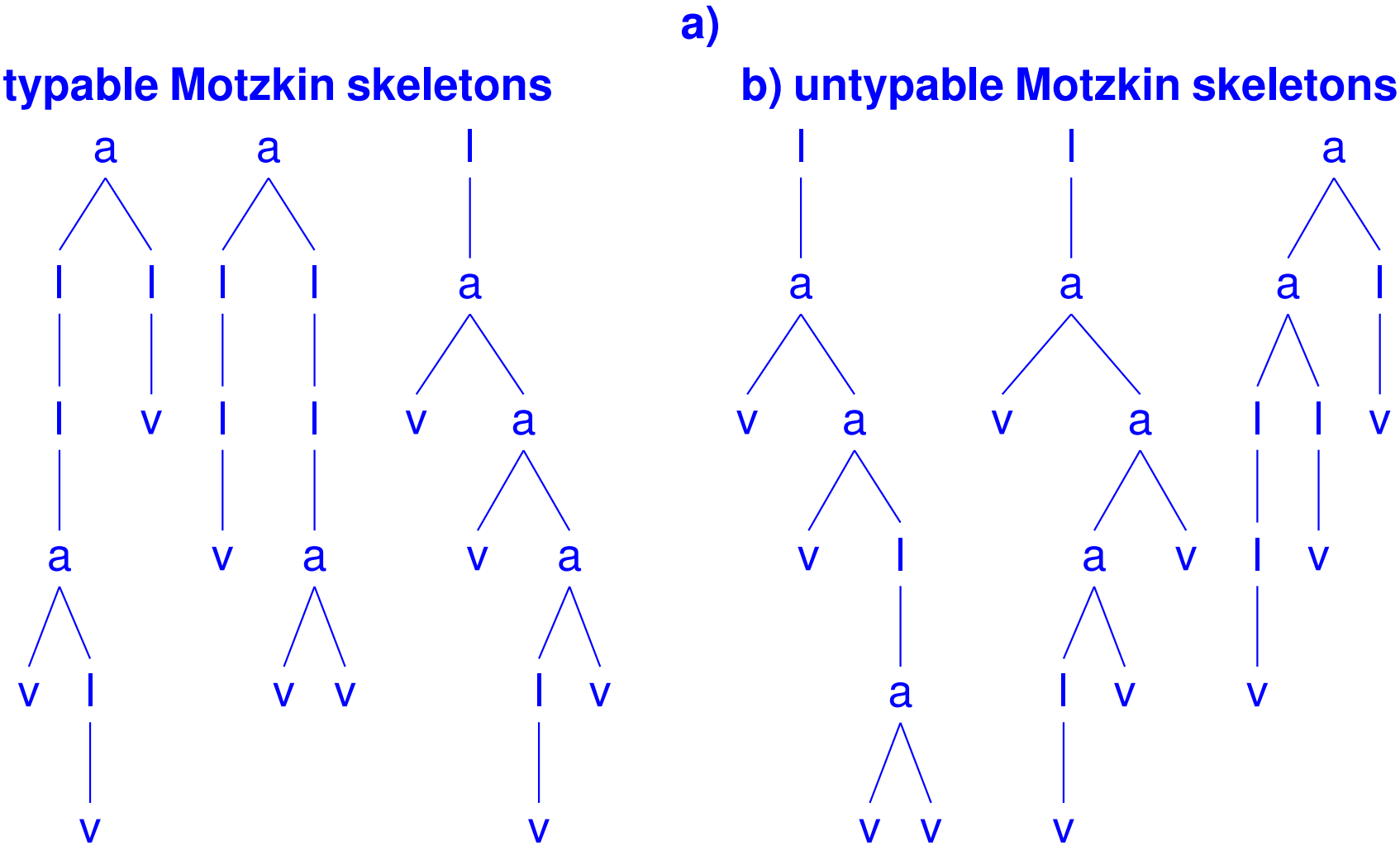


Figure: Typable vs. untypable skeletons of size 8

An open problem: what is the exact asymptotic behavior in this case?

Fig. 4 compares the growth of typable and untypable skeletons.

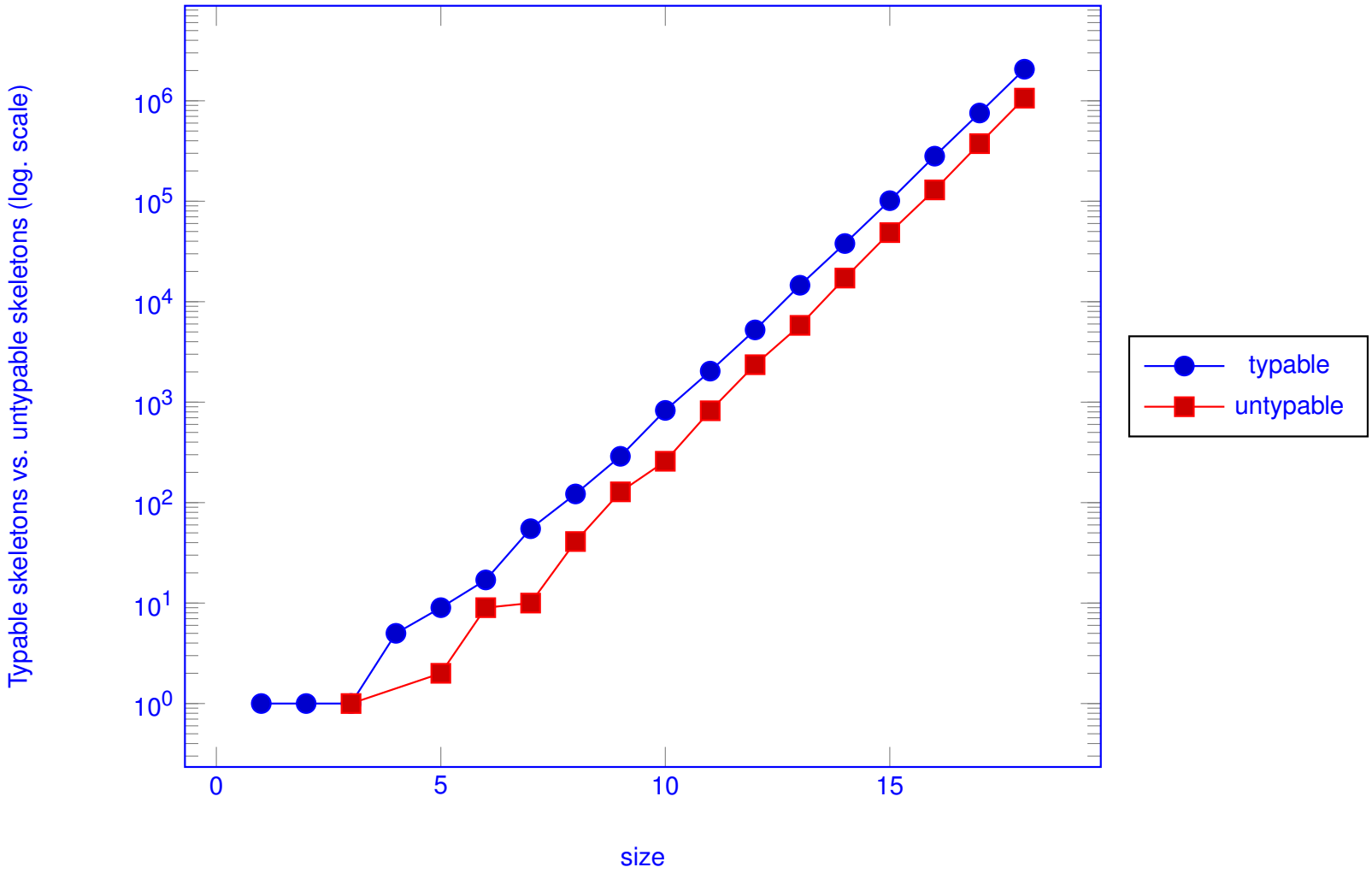


Figure: Typable skeletons vs. untypable skeletons by increasing sizes

Uniquely Typable Skeletons and their Relation to Uniquely Closable Skeletons

A *uniquely typable skeleton* is one for which it exists exactly one simply-typed closed lambda term having it as a skeleton.

Open problem: their asymptotic behavior

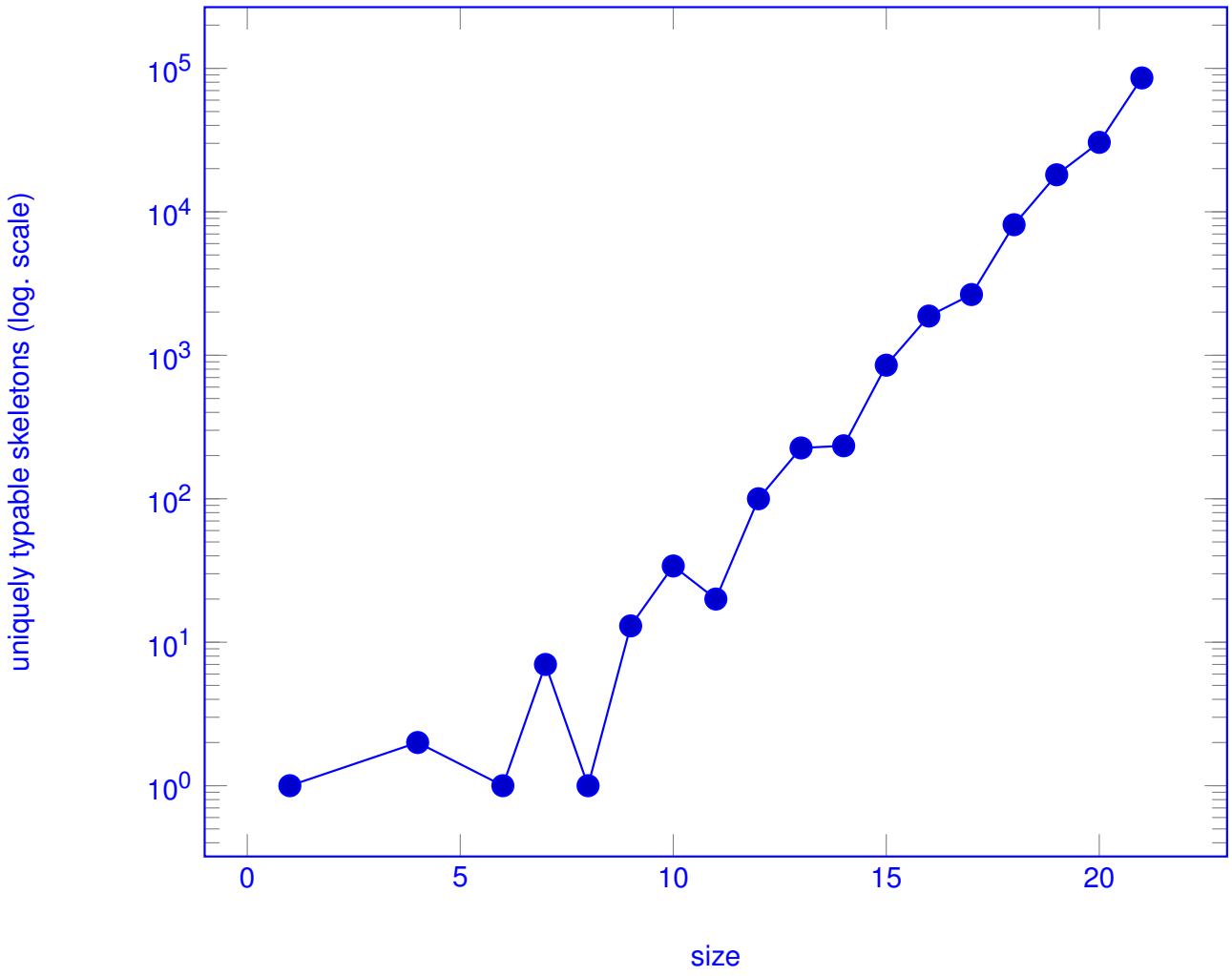


Figure: Uniquely typable skeletons by increasing sizes

An Interesting Easy Case

Proposition

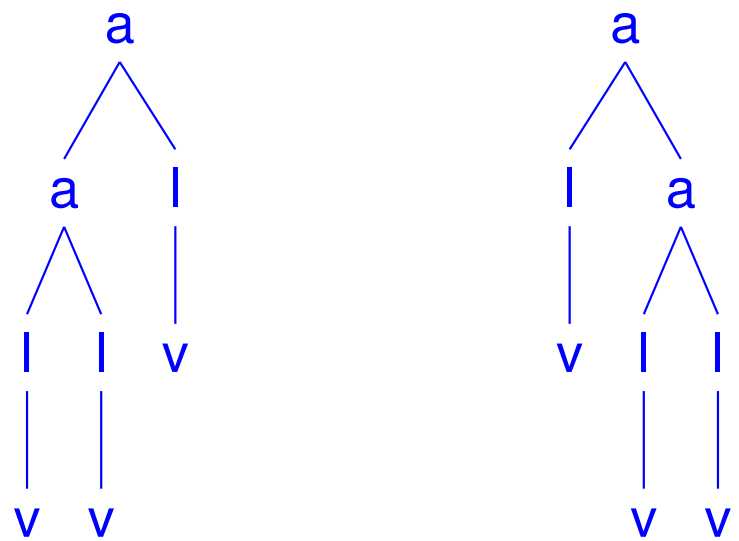
Uniquely closable typable skeletons of size $3n+1$ *are in bijection with Catalan objects (binary trees) of size* n*.*

Proof.

We will exhibit a simple bijection to binary trees. We want to show that terminal subtrees must be of the form l(v(0)). As there's a unique lambda above each leaf, closing it, the leaf should be (in de Bruijn notation), v(0) pointing to the first and only lambda above it. Assume a terminal node of the form a(v(0),v(0)). Then the two leaves must share a lambda binder resulting in a circular term when unifying their types (i.e., as in the case of the well-known term ω=l(a(v(0),v(0)))) and thus it could not be typable. □

Example of the Bijection to Binary Trees

- two trees illustrate the shape of the skeletons and their bijection to binary trees
- the skeleton is mapped into a binary tree simply by replacing its terminal subtrees of the form `l(v)` with a leaf node `v`



In this case, terms of size $3n+1 = 7 = 2+2+1+1+1+0+0+0$ *are mapped to binary trees of size* $n = 2 = 1+1+0+0+0$ *(with* `a/2` `nodes` *there counted as 1 and v/0 nodes as 0) after replacing* `l(v)` *nodes with* `v` *nodes.*

No Interesting Uniquely Closable Terms that are Typable

- As a consequence, *each uniquely closable term that is typable is uniquely typable*, as identity functions of the form `l(v(0))` would correspond to the end of each path from the root to a leaf in a lambda term having this skeleton.
- This tells us that there no “interesting” uniquely closable terms that are typable.
- However, as there are normalizable terms that are not simply typed, an interesting *open problem* is to find out if closable terms, other than those ending with `l(v(0))` are (weakly) normalizable.

Conclusions

- the problem: can the shape of the underlying binary-unary tree predetermine essential properties of lambda terms?
- positive answer with exact asymptotics for the case of closed terms
- positive answer but exact asymptotics still an open problem for simply typed terms
- when simple CF grammars can be derived from the definitions of our term classes, powerful tools from analytic combinatorics become available
- the tools used: a language as simple as (mostly) Horn Clause Prolog can handle elegantly combinatorial generation problems when the synergy between sound unification, backtracking and DCGs is put at work!

the paper is organized as a literate Prolog program - our code is available at:
`http://www.cse.unt.edu/~tarau/research/2017/uct.pro`

Questions?