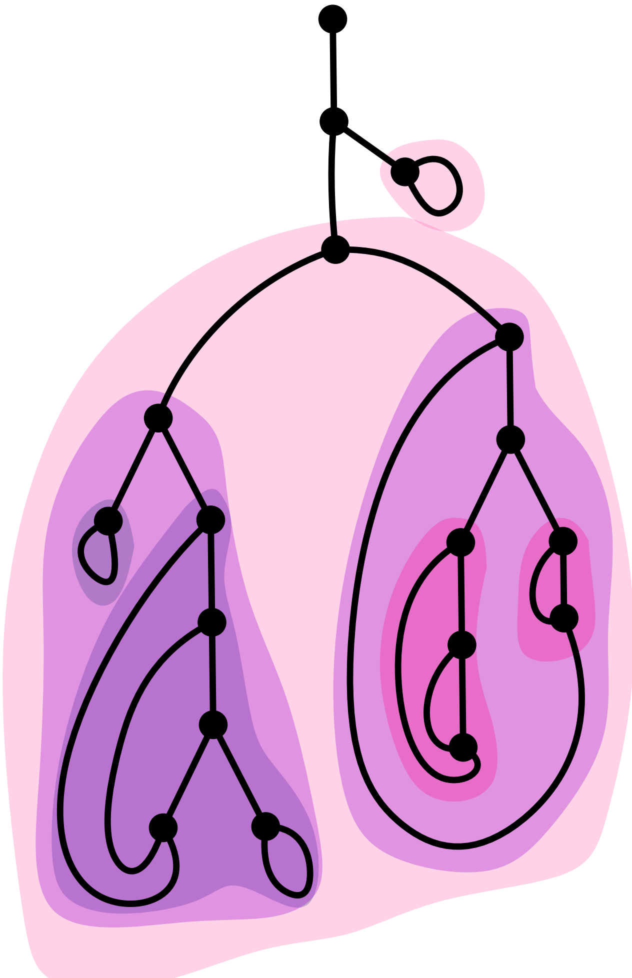
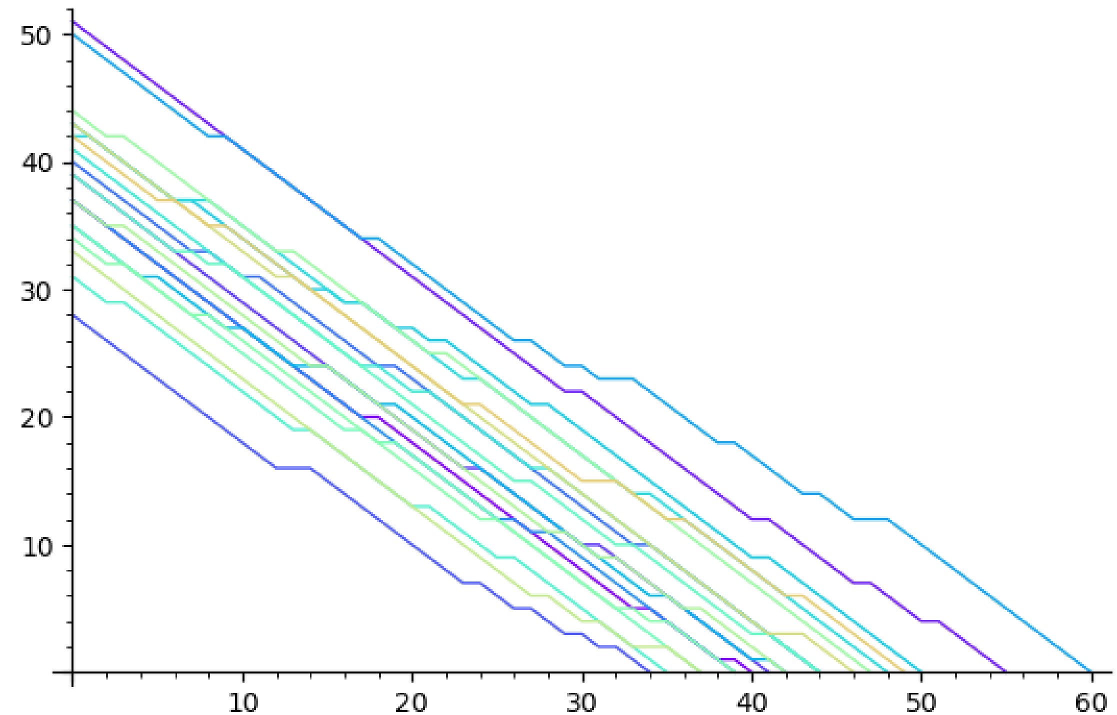
On the number of β-redices in random closed linear λ-terms





43rd Australasian Combinatorics Conference, 16 December 2021

Olivier Bodini (LIPN, Paris 13)

Alexandros Singh (LIPN, Paris 13)

Noam Zeilberger (LIX, Polytechnique)

What is the λ-calculus?

What is the λ-calculus?

- A **universal** system of computation

What is the λ-calculus?

- A **universal** system of computation
- Its terms are formed using the following grammar

$$x \mid \lambda x.t \mid (s\ t)$$

variable

What is the λ-calculus?

- A **universal** system of computation
- Its terms are formed using the following grammar

$$x \mid \lambda x.t \mid (s\ t)$$

variable

abstraction
represents an *anonymous* function

What is the λ-calculus?

- A **universal** system of computation
- Its terms are formed using the following grammar

$$x \mid \lambda x.t \mid (s\ t)$$

variable

abstraction
represents an *anonymous* function

application
feeding an argument t to a function s

What is the λ-calculus?

- A **universal** system of computation
- Its terms are formed using the following grammar

$$x \mid \lambda x.t \mid (s\ t)$$

variable

abstraction
represents an *anonymous* function

application
feeding an argument t to a function s

- We're interested in terms up to α-equivalence:

$$(\lambda x.xx)(\lambda x.xx) \overset{\alpha}{=} (\lambda y.yy)(\lambda x.xx) \overset{\alpha}{\neq} (\lambda y.ya)(\lambda x.xx)$$

Computing with the λ-calculus

- Substitution rule:

$$T_1[v := T_2]$$

"replace free occurences of v in T_1 with T_2"

(α-converting T if necessary, to avoid capturing variables of T_2)

Computing with the λ-calculus

- Substitution rule:

$$T_1[v := T_2]$$

"replace free occurences of v in T_1 with T_2"
(α-converting T if necessary, to avoid capturing variables of T_2)

- Examples of substitutions
  - $(\lambda x.(x\ y))[y := x] \neq (\lambda x.(x\ x))$
  - $(\lambda x.(x\ y))[y := x] \stackrel{\alpha}{=} (\lambda z.(z\ y))[y := x] = (\lambda z.(z\ x))$

Computing with the λ-calculus

- Substitution rule:

$$T_1[\nu := T_2]$$

"replace free occurences of ν in T_1 with T_2"

(α-converting T if necessary, to avoid capturing variables of T_2)

- Examples of substitutions
  - $(\lambda x.(x\ y))[y := x] \neq (\lambda x.(x\ x))$
  - $(\lambda x.(x\ y))[y := x] \stackrel{\alpha}{=} (\lambda z.(z\ y))[y := x] = (\lambda z.(z\ x))$

- Dynamics of the λ-calculus: β-reductions

(λ-terms together with β-reduction are enough to encode any computation!)

$$((\lambda x.t_1)\ t_2) \xrightarrow{\beta} t_1[x := t_2]$$

Computing with the λ-calculus

- Substitution rule:

$$T_1[v := T_2]$$

"replace free occurences of v in T_1 with T_2"
(α-converting T if necessary, to avoid capturing variables of T_2)

- Examples of substitutions
  - $(\lambda x.(x\ y))[y := x] \neq (\lambda x.(x\ x))$
  - $(\lambda x.(x\ y))[y := x] \overset{\alpha}{=} (\lambda z.(z\ y))[y := x] = (\lambda z.(z\ x))$

- Dynamics of the λ-calculus: β-reductions
(λ-terms together with β-reduction are enough to encode any computation!)

$$((\lambda x.t_1)\ t_2) \xrightarrow{\beta} t_1[x := t_2]$$

- Examples of reductions

Computing with the λ-calculus

- Substitution rule:

$$T_1[v := T_2]$$

"replace free occurences of v in T_1 with T_2"
(α-converting T if necessary, to avoid capturing variables of T_2)

- Examples of substitutions
  - $(\lambda x.(x\ y))[y := x] \neq (\lambda x.(x\ x))$
  - $(\lambda x.(x\ y))[y := x] \stackrel{\alpha}{=} (\lambda z.(z\ y))[y := x] = (\lambda z.(z\ x))$

- Dynamics of the λ-calculus: β-reductions
(λ-terms together with β-reduction are enough to encode any computation!)

$$((\lambda x.t_1)\ t_2) \stackrel{\beta}{\to} t_1[x := t_2]$$

- Examples of reductions
  - $((\lambda x.x)\ y) \stackrel{\beta}{\to} x[x := y] = y$

Computing with the λ-calculus

- Substitution rule:

$$T_1[v := T_2]$$

"replace free occurences of v in T_1 with T_2"
(α-converting T if necessary, to avoid capturing variables of T_2)

- Examples of substitutions
  - $(\lambda x.(x\ y))[y := x] \neq (\lambda x.(x\ x))$
  - $(\lambda x.(x\ y))[y := x] \overset{\alpha}{=} (\lambda z.(z\ y))[y := x] = (\lambda z.(z\ x))$

- Dynamics of the λ-calculus: β-reductions
(λ-terms together with β-reduction are enough to encode any computation!)

$$((\lambda x.t_1)\ t_2) \xrightarrow{\beta} t_1[x := t_2]$$

- Examples of reductions
  - $((\lambda x.x)\ y) \xrightarrow{\beta} x[x := y] = y$
  - $(\lambda x.((\lambda y.x\ y)\ u)) \xrightarrow{\beta} (\lambda y.(x\ y))[y := u] = (\lambda x.(x\ u))$

Computing with the λ-calculus

- Substitution rule:

$$T_1[\nu := T_2]$$

"replace free occurences of ν in T_1 with T_2"

(α-converting T if necessary, to avoid capturing variables of T_2)

- Examples of substitutions
  - $(\lambda x.(x\ y))[y := x] \neq (\lambda x.(x\ x))$
  - $(\lambda x.(x\ y))[y := x] \overset{\alpha}{=} (\lambda z.(z\ y))[y := x] = (\lambda z.(z\ x))$

- Dynamics of the λ-calculus: β-reductions

(λ-terms together with β-reduction are enough to encode any computation!)

$$((\lambda x.t_1)\ t_2) \xrightarrow{\beta} t_1[x := t_2]$$

- Examples of reductions
  - $((\lambda x.x)\ y) \xrightarrow{\beta} x[x := y] = y$
  - $(\lambda x.((\lambda y.x\ y)\ u)) \xrightarrow{\beta} (\lambda y.(x\ y))[y := u] = (\lambda x.(x\ u))$
  - $(\lambda x.(x\ x))(\lambda y.(y\ y)) \xrightarrow{\beta} (\lambda y.(y\ y))(\lambda y.(y\ y)) \overset{\alpha}{=} (\lambda x.(x\ x))(\lambda y.(y\ y))$

More on β-reductions

- An occurence of the $((\lambda x.t_1)\ t_2)$ "pattern" is called a β-redex:

$$((\lambda x.((\lambda y.(y\ x))\ x))\ (a\ b))$$

More on β-reductions

- An occurence of the $((\lambda x.t_1)\ t_2)$ "pattern" is called a β-redex:

$$((\lambda x.\underbrace{((\lambda y.(y\ x))\ x)})\ (a\ b))$$

More on β-reductions

- An occurence of the $((\lambda x.t_1)\ t_2)$ "pattern" is called a β-redex:

$$\underbrace{((\lambda x.(\underbrace{((\lambda y.(y\ x))\ x)}))\ (a\ b))}$$

- A term with no $beta$-redices (redexes?) is called a *normal form*

$$((\lambda x.((\lambda y.(y\ x))\ x))\ (a\ b)) \xrightarrow{\beta} (\lambda x.(x\ x))(a\ b) \xrightarrow{\beta} \underset{\text{normal form!}}{(a\ b)(a\ b)}$$

More on β-reductions

- An occurence of the $((\lambda x.t_1)\ t_2)$ "pattern" is called a β-redex:

$$\underbrace{((\lambda x.(\underbrace{((\lambda y.(y\ x))\ x}))\ (a\ b))}$$

- A term with no $beta$-redices (redexes?) is called a *normal form*

$$((\lambda x.((\lambda y.(y\ x))\ x))\ (a\ b)) \xrightarrow{\beta} (\lambda x.(x\ x))(a\ b) \xrightarrow{\beta} \underset{\text{normal form!}}{(a\ b)(a\ b)}$$

- β-reduction is quite complicated:

More on β-reductions

- An occurence of the $((\lambda x.t_1)\ t_2)$ "pattern" is called a β-redex:

$$\underbrace{((\lambda x.(\underbrace{(\lambda y.(y\ x))\ x}))\ (a\ b))}$$

- A term with no $beta$-redices (redexes?) is called a *normal form*

$$((\lambda x.((\lambda y.(y\ x))\ x))\ (a\ b)) \xrightarrow{\beta} (\lambda x.(x\ x))(a\ b) \xrightarrow{\beta} \underset{\text{normal form!}}{(a\ b)(a\ b)}$$

- β-reduction is quite complicated:
  - Reducing a redex can create new redices!

    $$((\lambda x.(x\ z))\ (\lambda y.y)) \xrightarrow{\beta} ((\lambda y.y)\ z)$$

More on β-reductions

- An occurence of the $((\lambda x.t_1)\ t_2)$ "pattern" is called a β-redex:

$$\underbrace{((\lambda x.(\underbrace{((\lambda y.(y\ x))\ x)}))\ (a\ b))}$$

- A term with no $beta$-redices (redexes?) is called a *normal form*

$$((\lambda x.((\lambda y.(y\ x))\ x))\ (a\ b)) \xrightarrow{\beta} (\lambda x.(x\ x))(a\ b) \xrightarrow{\beta} \underset{\text{normal form!}}{(a\ b)(a\ b)}$$

- β-reduction is quite complicated:
  - Reducing a redex can create new redices!

    $$((\lambda x.(x\ z))\ (\lambda y.y)) \xrightarrow{\beta} ((\lambda y.y)\ z)$$

  - Terms may never reach a normal form, their size might even increase!

    $$((\lambda x.(x\ x))(\lambda x.(x\ x\ x))) \xrightarrow{\beta} (\lambda x.(x\ x\ x))(\lambda x.(x\ x\ x))(\lambda x.(x\ x\ x))$$

More on β-reductions

- An occurence of the $((\lambda x.t_1)\ t_2)$ "pattern" is called a β-redex:

$$\underbrace{((\lambda x.(\underbrace{(\lambda y.(y\ x))\ x}))\ (a\ b))}$$

- A term with no $beta$-redices (redexes?) is called a *normal form*

$$((\lambda x.((\lambda y.(y\ x))\ x))\ (a\ b)) \xrightarrow{\beta} (\lambda x.(x\ x))(a\ b) \xrightarrow{\beta} \underset{\text{normal form!}}{(a\ b)(a\ b)}$$

- β-reduction is quite complicated:
  - Reducing a redex can create new redices!

    $$((\lambda x.(x\ z))\ (\lambda y.y)) \xrightarrow{\beta} ((\lambda y.y)\ z)$$

  - Terms may never reach a normal form, their size might even increase!

    $$((\lambda x.(x\ x))(\lambda x.(x\ x\ x))) \xrightarrow{\beta} (\lambda x.(x\ x\ x))(\lambda x.(x\ x\ x))(\lambda x.(x\ x\ x))$$

  - Order in which redices are reduced matters!

$$(\lambda x.z)((\lambda x.(x\ x))(\lambda x.(x\ x))) \begin{cases} \to (\lambda x.z)((x\ x)[x := (\lambda x.(x\ x))]) = \ldots \\ \to z[x := (\lambda x.x\ x)(\lambda x.x\ x)] = z \end{cases}$$

Previous work on the reduction of λ-terms

Previous work on the reduction of λ-terms

- Asymptotically almost all λ-terms are strongly normalizing.
  [DGKRTZ13]

Previous work on the reduction of λ-terms

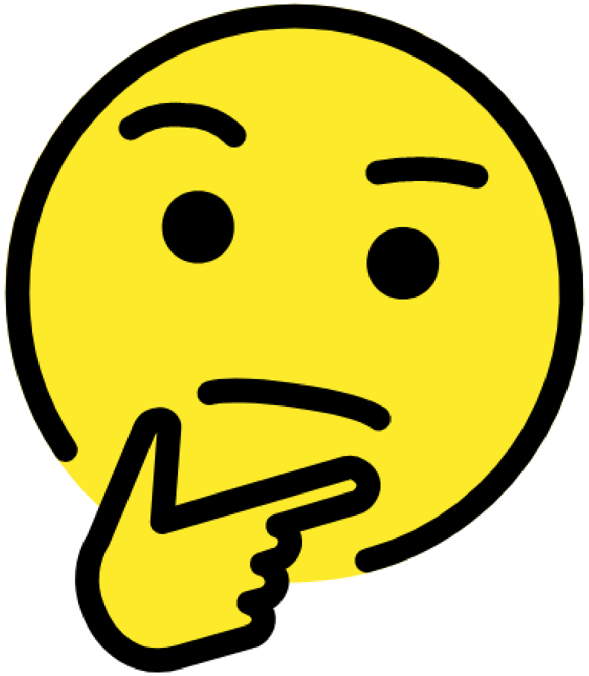
- Asymptotically almost all λ-terms are strongly normalizing.
  [DGKRTZ13]

- Asymptotically almost no λ-term is strongly normalizing.
  [DGKRTZ13,BGLZ16]

Previous work on the reduction of λ-terms

- Asymptotically almost all λ-terms are strongly normalizing.
  [DGKRTZ13]

- Asymptotically almost no λ-term is strongly normalizing.
  [DGKRTZ13,BGLZ16]



Previous work on the reduction of λ-terms

- Asymptotically almost all λ-terms are strongly normalizing. [DGKRTZ13]
  For terms expressed in the previously-presented syntax and size defined recursively as:

$$|x| = 0, \; |(a\ b)| = 1 + |a| + |b|, \; |\lambda x.t| = 1 + |t|$$

- Asymptotically almost no λ-term is strongly normalizing. [DGKRTZ13,BGLZ16]
  For terms expressed using de Bruijn indices or combinators (together with appropriate size functions)

Parameter sensitive to the definition of the syntax and the size of terms!

- Almost every simply-typed λ-term has a long β-reduction sequence [SAKT17]

Subfamilies of λ-terms

General terms: no restrictions on variable use

$\lambda x.\lambda y.x$

$\lambda x.\lambda y.x\ (y\ a)$

$(\lambda x.xx)(\lambda y.yy)$

Subfamilies of λ-terms

General terms: no restrictions on variable use

$\lambda x.\lambda y.x$

$\lambda x.\lambda y.x\ (y\ a)$

$(\lambda x.xx)(\lambda y.yy)$

unused abstraction

free variable

var. used twice

Subfamilies of λ-terms

General terms: no restrictions on variable use

$\lambda x.\lambda y.x$

$\lambda x.\lambda y.x\ (y\ a)$

$(\lambda x.xx)(\lambda y.yy)$

unused abstraction

free variable

var. used twice

Affine Terms: bound variables occur **at most** once

$(\lambda x.\lambda y.a)(\lambda x.x)$

$\lambda x.\lambda y.\lambda z.(x\ a)\ y$

$\lambda x.\lambda y.y$

Subfamilies of λ-terms

General terms: no restrictions on variable use

$\lambda x.\lambda y.x$

$\lambda x.\lambda y.x\ (y\ a)$

$(\lambda x.xx)(\lambda y.yy)$

unused abstraction

free variable

var. used twice

Affine Terms: bound variables occur **at most** once

$(\lambda x.\lambda y.a)(\lambda x.x)$

$\lambda x.\lambda y.\lambda z.(x\ a)\ y$

$\lambda x.\lambda y.y$

Linear Terms: bound variables occur **exactly** once

$\lambda x.\lambda y.(y\ x)a$

$\lambda x.\lambda y.(y\ a)(b\ x)$

$\lambda x.a(\lambda z.(\lambda y.y\ (x\ z)))$

β-reducing closed linear terms

β-reducing closed linear terms

- Closed linear lambda calculus is *strongly normalising*!

β-reducing closed linear terms

- Closed linear lambda calculus is *strongly normalising*!
- Repeated β-reduction is guaranteed to terminate, there exists a unique normal form, and reduction order doesn't matter!

β-reducing closed linear terms

- Closed linear lambda calculus is *strongly normalising*!
- Repeated β-reduction is guaranteed to terminate, there exists a unique normal form, and reduction order doesn't matter!
- No longer Turing-complete, many interesting connections with complexity theory (e.g PTIME-completeness [M04])

β-reducing closed linear terms

- Closed linear lambda calculus is *strongly normalising*!
- Repeated β-reduction is guaranteed to terminate, there exists a unique normal form, and reduction order doesn't matter!
- No longer Turing-complete, many interesting connections with complexity theory (e.g PTIME-completeness [M04])
- How many β-reduction steps, on average, does one need to reach a normal form starting from a random λ-term?

β-reducing closed linear terms

- Closed linear lambda calculus is *strongly normalising*!
- Repeated β-reduction is guaranteed to terminate, there exists a unique normal form, and reduction order doesn't matter!
- No longer Turing-complete, many interesting connections with complexity theory (e.g PTIME-completeness [M04])
- How many β-reduction steps, on average, does one need to reach a normal form starting from a random λ-term?

β-reducing closed linear terms

- Closed linear lambda calculus is *strongly normalising*!
- Repeated β-reduction is guaranteed to terminate, there exists a unique normal form, and reduction order doesn't matter!
- No longer Turing-complete, many interesting connections with complexity theory (e.g PTIME-completeness [M04])
- How many β-reduction steps, on average, does one need to reach a normal form starting from a random λ-term?

  A *lower bound* is given by the number of β-redices!

  This motivates the central question of this work:

What is the number of β-redices in a random linear λ-term?

β-reducing closed linear terms

- Closed linear lambda calculus is *strongly normalising*!
- Repeated β-reduction is guaranteed to terminate, there exists a unique normal form, and reduction order doesn't matter!
- No longer Turing-complete, many interesting connections with complexity theory (e.g PTIME-completeness [M04])
- How many β-reduction steps, on average, does one need to reach a normal form starting from a random λ-term?

  A *lower bound* is given by the number of β-redices!

  This motivates the central question of this work:

What is the number of β-redices in a random linear λ-term?

random variable!

uniform distribution

β-reducing closed linear terms

- Closed linear lambda calculus is *strongly normalising*!
- Repeated β-reduction is guaranteed to terminate, there exists a unique normal form, and reduction order doesn't matter!
- No longer Turing-complete, many interesting connections with complexity theory (e.g PTIME-completeness [M04])
- How many β-reduction steps, on average, does one need to reach a normal form starting from a random λ-term?

  A *lower bound* is given by the number of β-redices!

  This motivates the central question of this work:

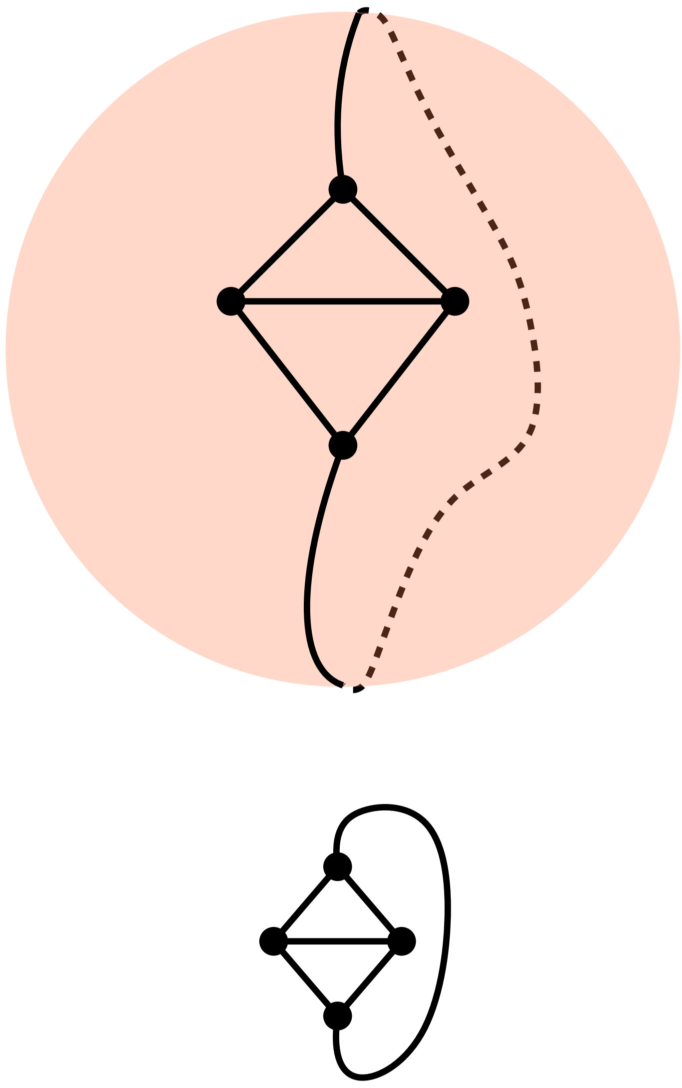
What is the number of β-redices in a random linear λ-term?

asymptotically!

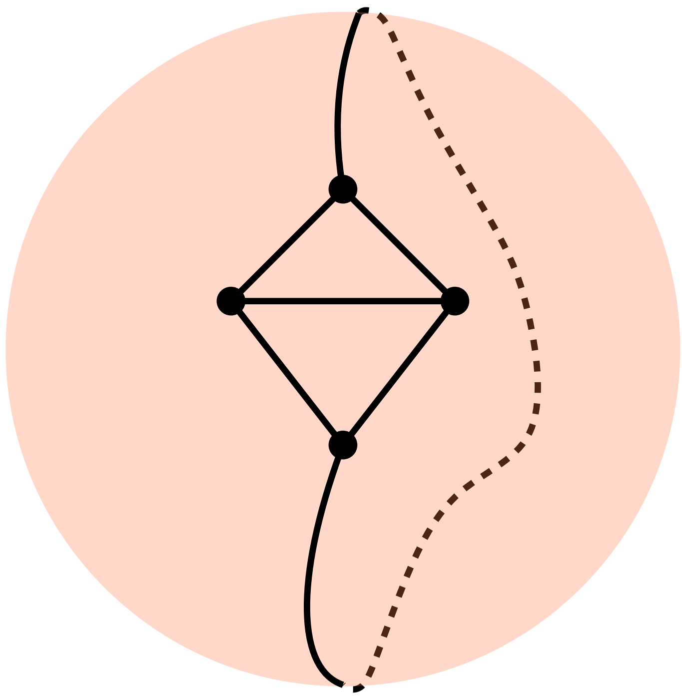
seq. of random variables!

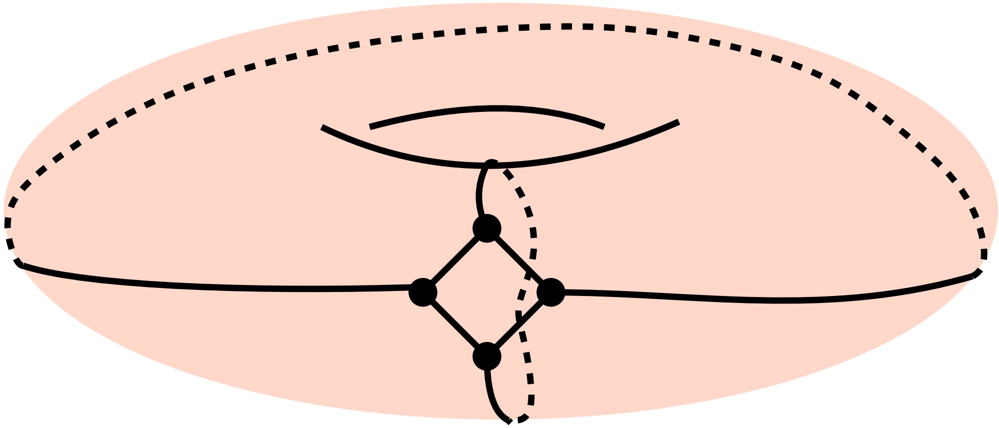
uniform distribution
on the set of terms of size n

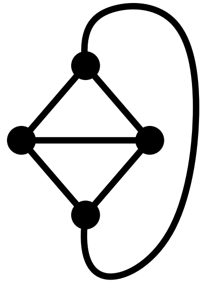
What are maps?

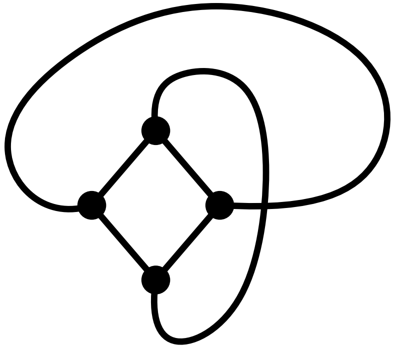


What are maps?

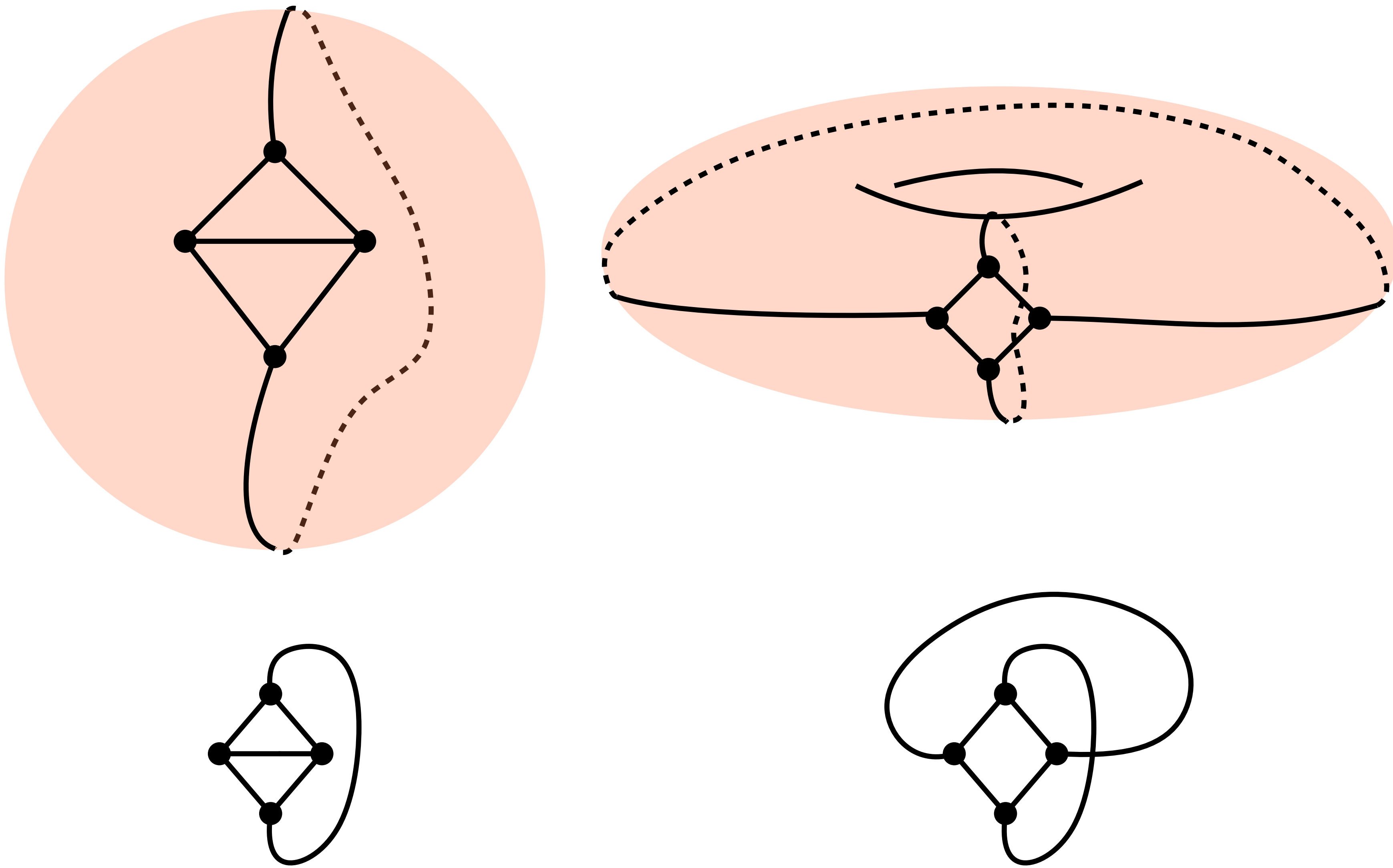








What are maps?

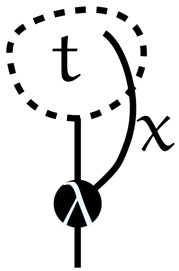


We're interested in unrestricted genus, restricted vertex degrees

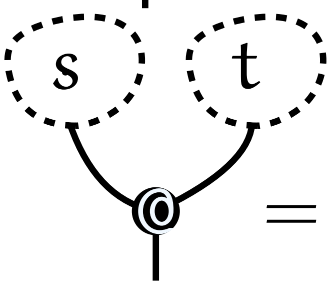
Why should you, a logician, be interested in maps?

String diagrams! [BGJ13, Z16]

$\bullet = x$



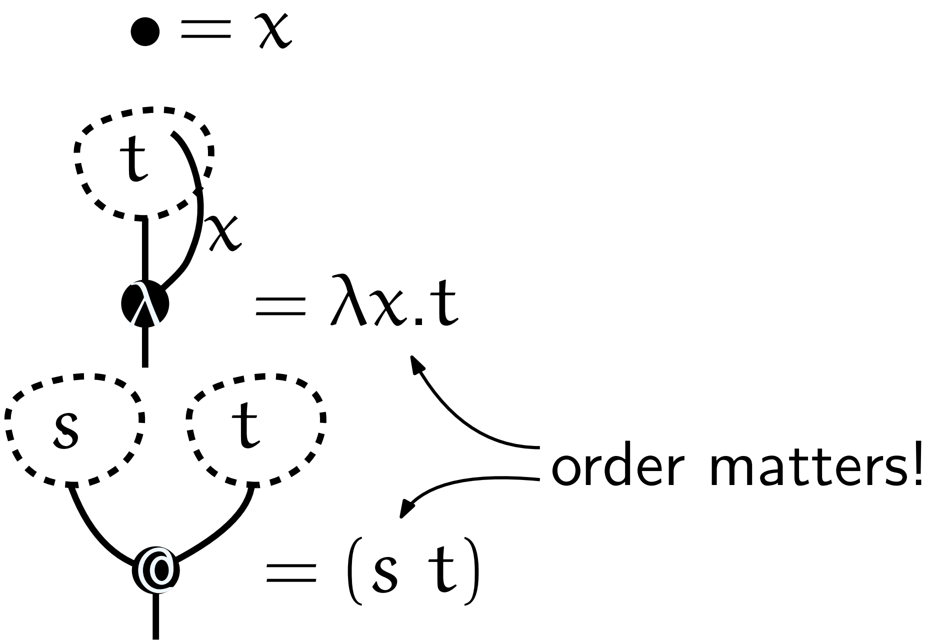
$= \lambda x.t$

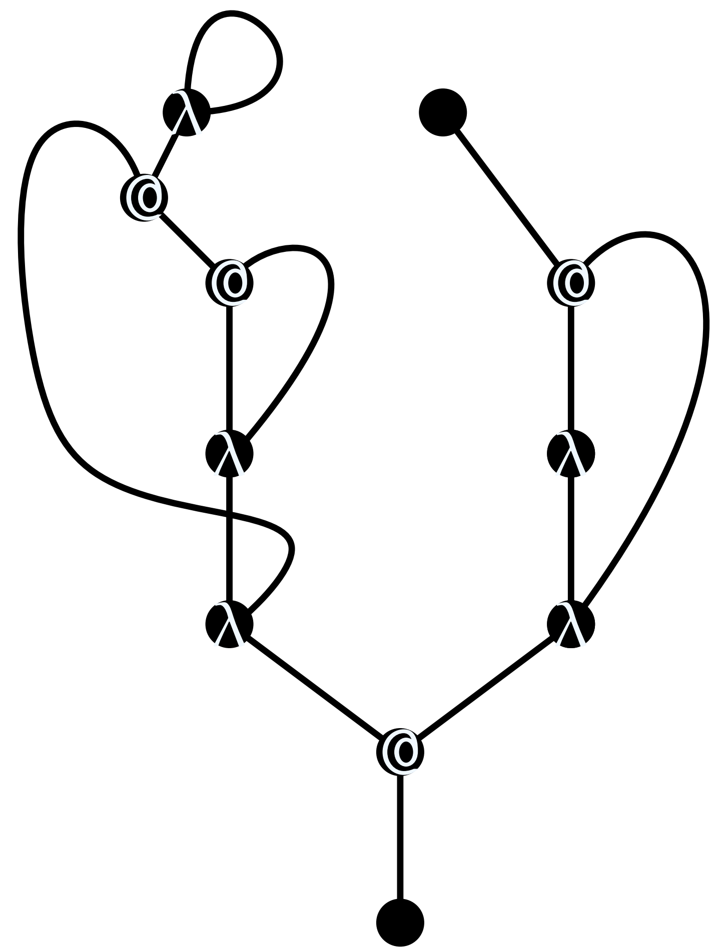


$= (s\ t)$

Why should you, a logician, be interested in maps?

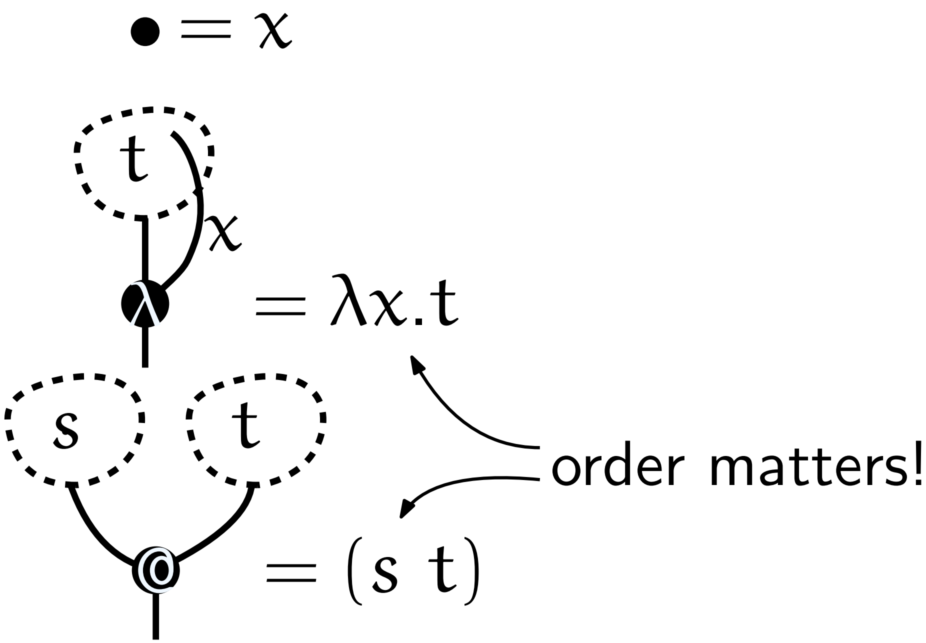
String diagrams! [BGJ13, Z16] $(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.\lambda v.a\ u)$





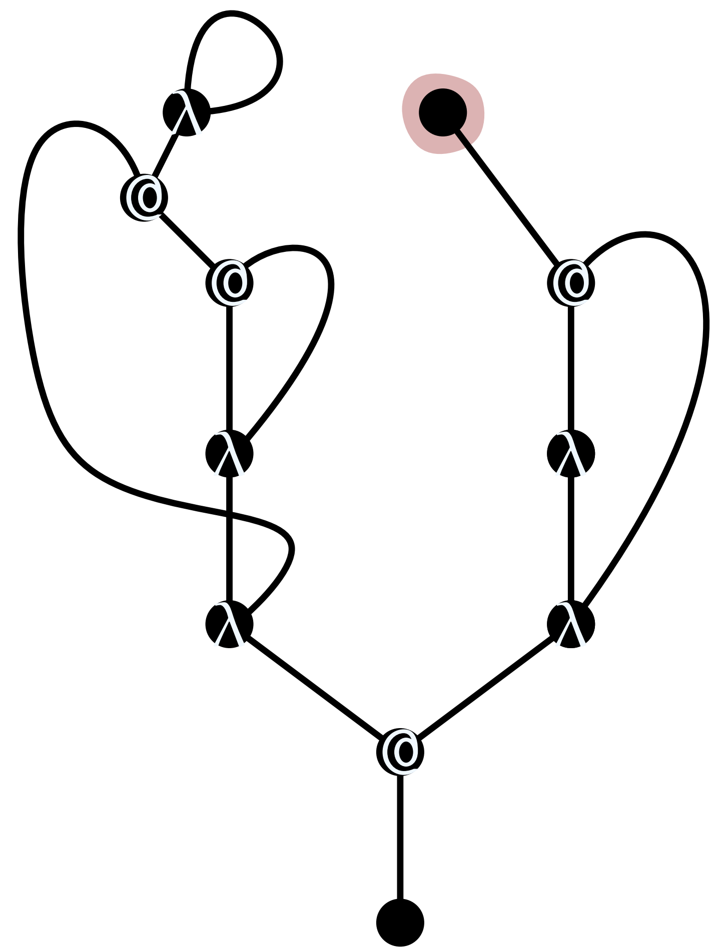
Why should you, a logician, be interested in maps?

String diagrams! [BGJ13, Z16] $(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.\lambda v.a\ u)$



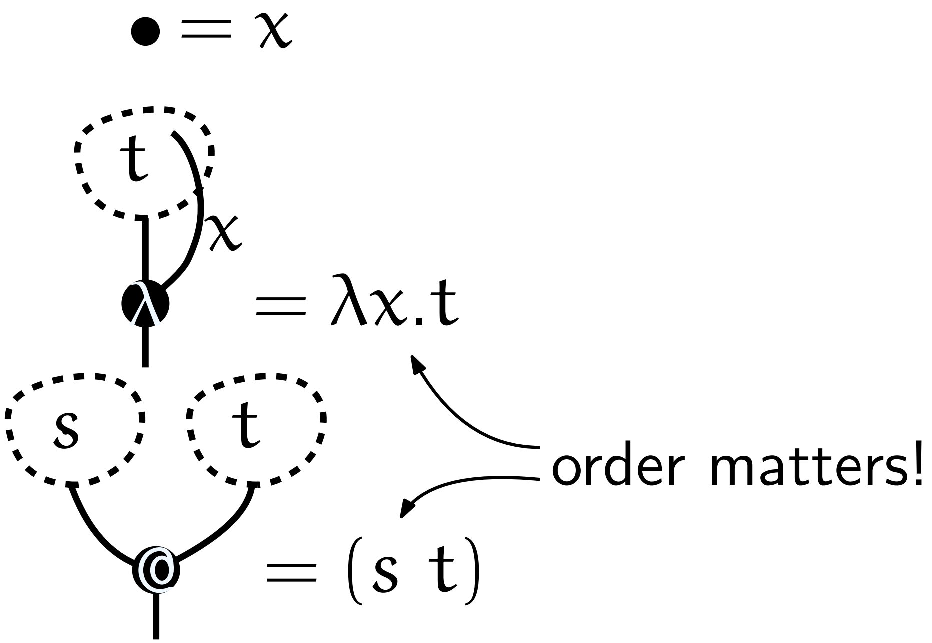
Dictionary

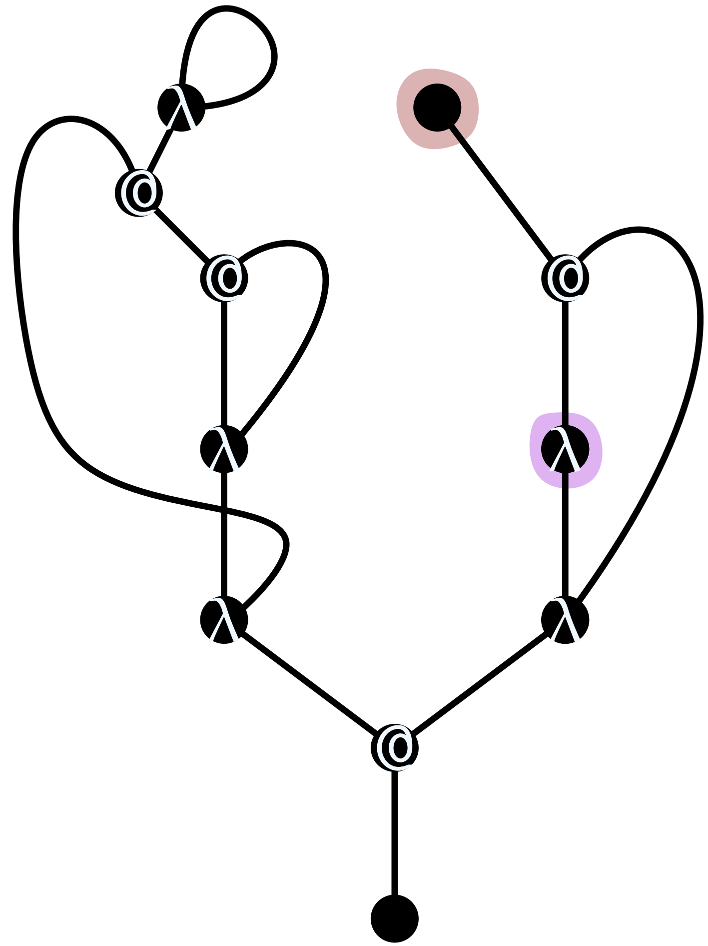
- Free var $\leftrightarrow$ unary vertex



Why should you, a logician, be interested in maps?

String diagrams! [BGJ13, Z16] $(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.\lambda v.a\ u)$





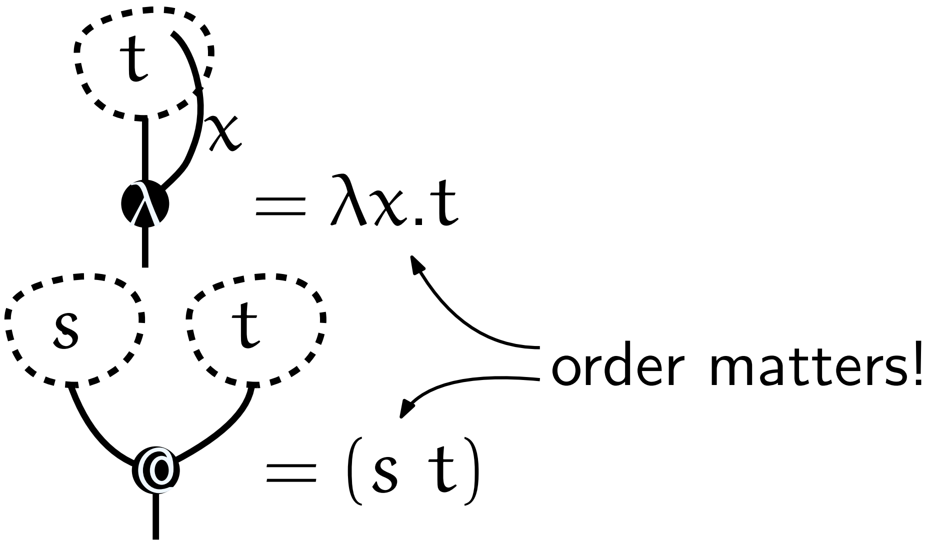
Dictionary

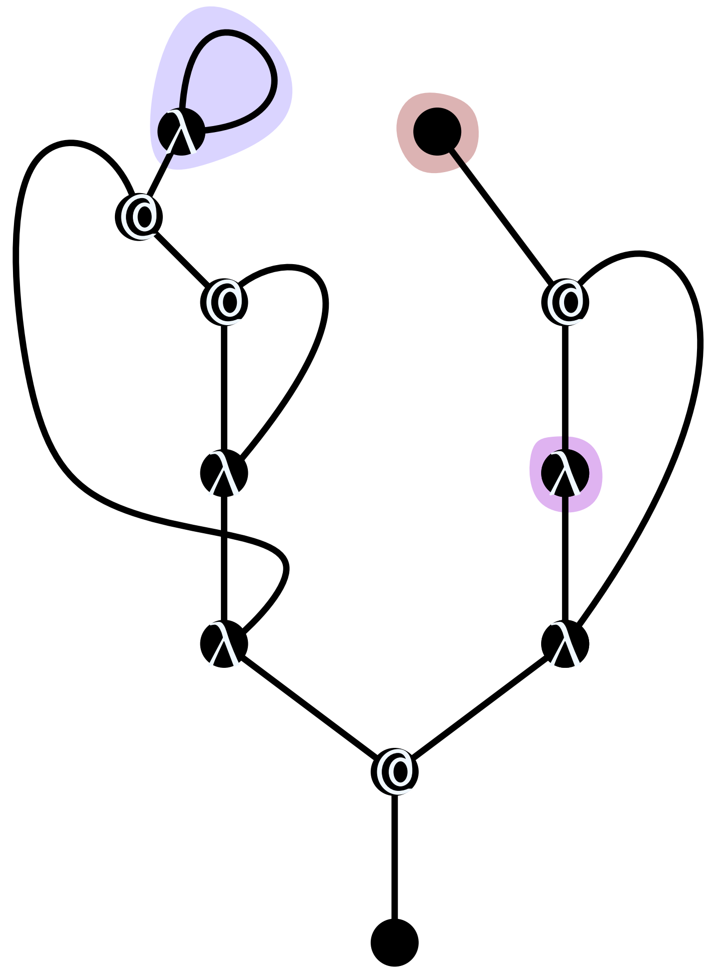
- Free var $\leftrightarrow$ unary vertex
- Unused λ $\leftrightarrow$ binary vertex

Why should you, a logician, be interested in maps?

String diagrams! [BGJ13, Z16] $(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.\lambda v.a\ u)$

$\bullet = x$



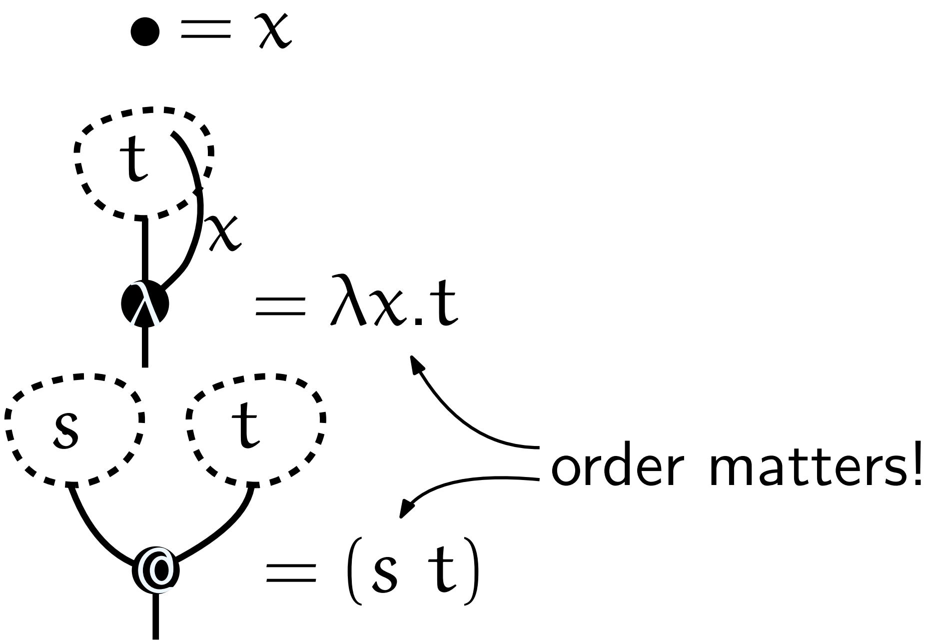


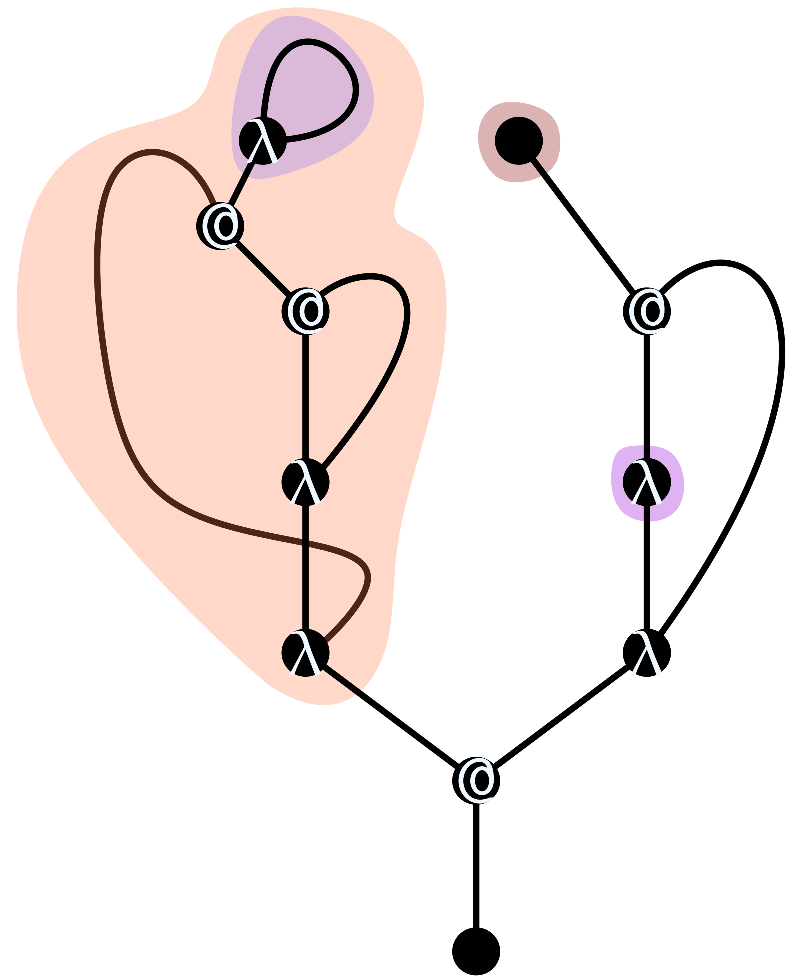
Dictionary

- Free var $\leftrightarrow$ unary vertex
- Unused λ $\leftrightarrow$ binary vertex
- Identity-subterm $\leftrightarrow$ loop

Why should you, a logician, be interested in maps?

String diagrams! [BGJ13, Z16] $(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.\lambda v.a\ u)$



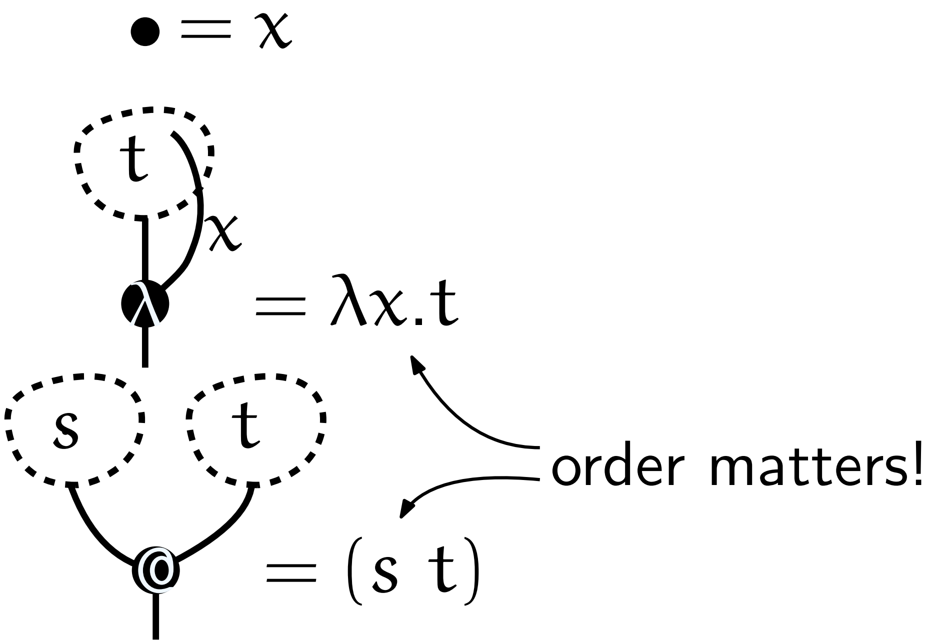


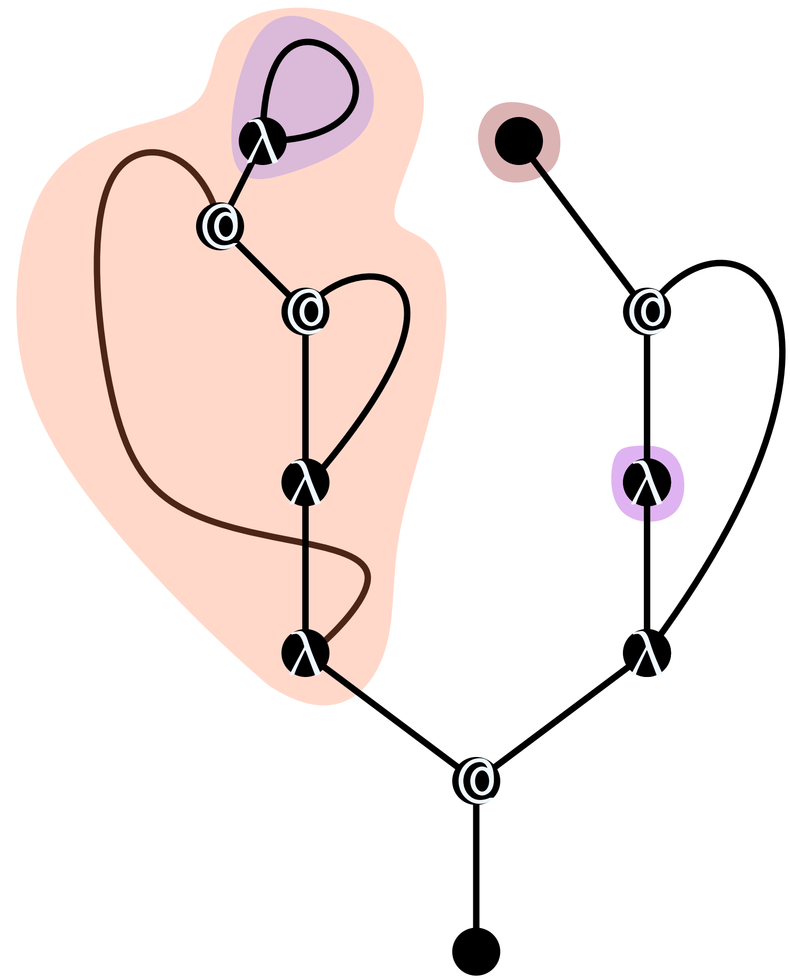
Dictionary

- Free var ↔ unary vertex
- Unused λ ↔ binary vertex
- Identity-subterm ↔ loop
- Closed subterm ↔ bridge

Why should you, a logician, be interested in maps?

String diagrams! [BGJ13, Z16] $(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.\lambda v.a\ u)$





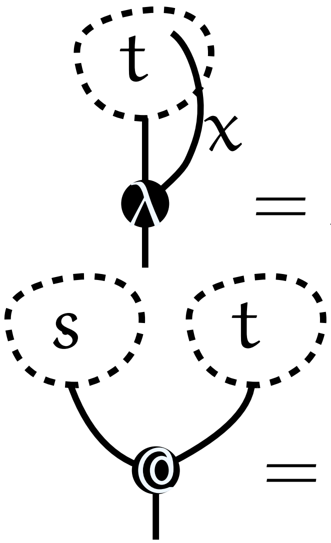
Dictionary

- Free var $\leftrightarrow$ unary vertex
- Unused λ $\leftrightarrow$ binary vertex
- Identity-subterm $\leftrightarrow$ loop
- Closed subterm $\leftrightarrow$ bridge
- # subterms $\leftrightarrow$ # edges

Why should you, a logician, be interested in maps?

String diagrams! [BGJ13, Z16] $(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.\lambda v.a\ u)$

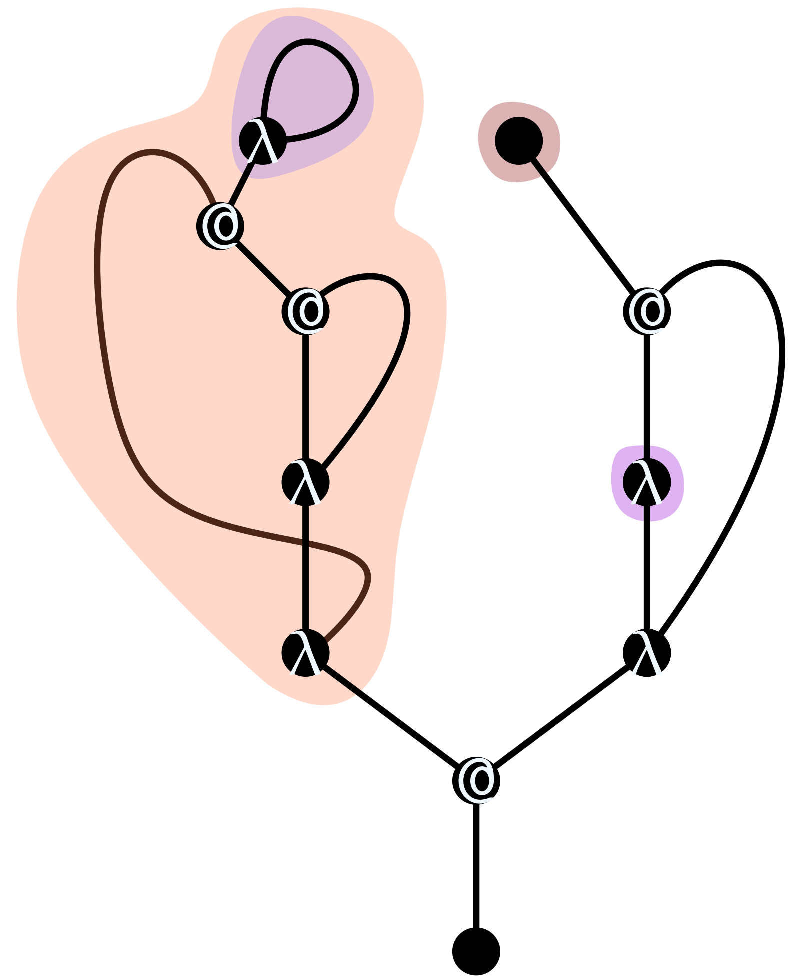
$\bullet = x$



$= \lambda x.t$

order matters!

$= (s\ t)$



Dictionary

- Free var ↔ unary vertex
- Unused λ ↔ binary vertex
- Identity-subterm ↔ loop
- Closed subterm ↔ bridge
- # subterms ↔ # edges

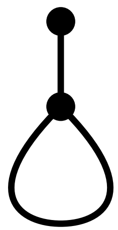
Closed linear terms ↔ trivalent maps
Closed affine terms ↔ (2,3)-valent maps
Established in [BGJ13, BGGJ13]

Why should you, a combinatorialist, be interested in λ-terms?

Decomposing (closed) rooted trivalent maps [BGJ13]

Why should you, a combinatorialist, be interested in λ-terms?

Decomposing (closed) rooted trivalent maps [BGJ13]

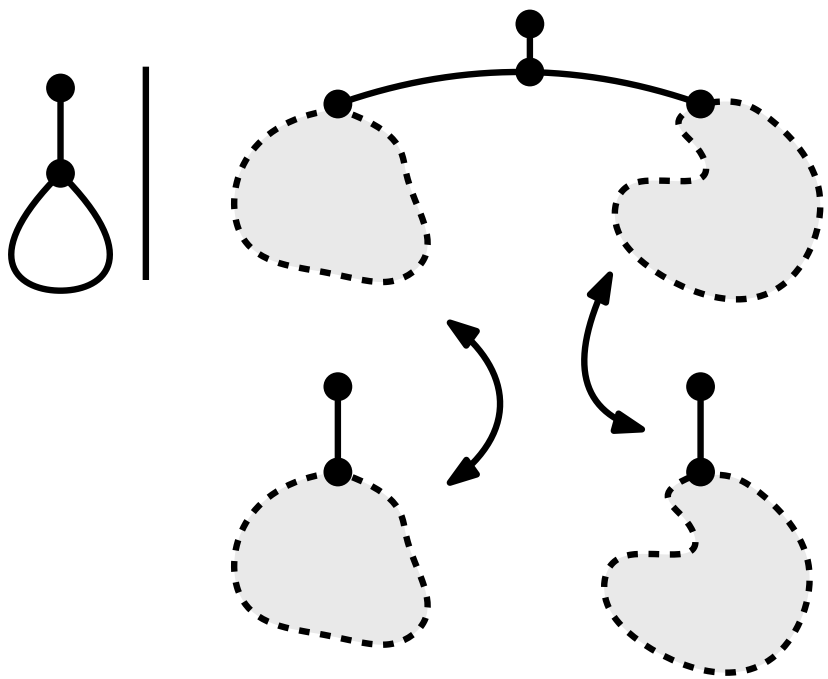


edges

$$\mathsf{T}(z) = z^2$$

Why should you, a combinatorialist, be interested in λ-terms?

Decomposing (closed) rooted trivalent maps [BGJ13]

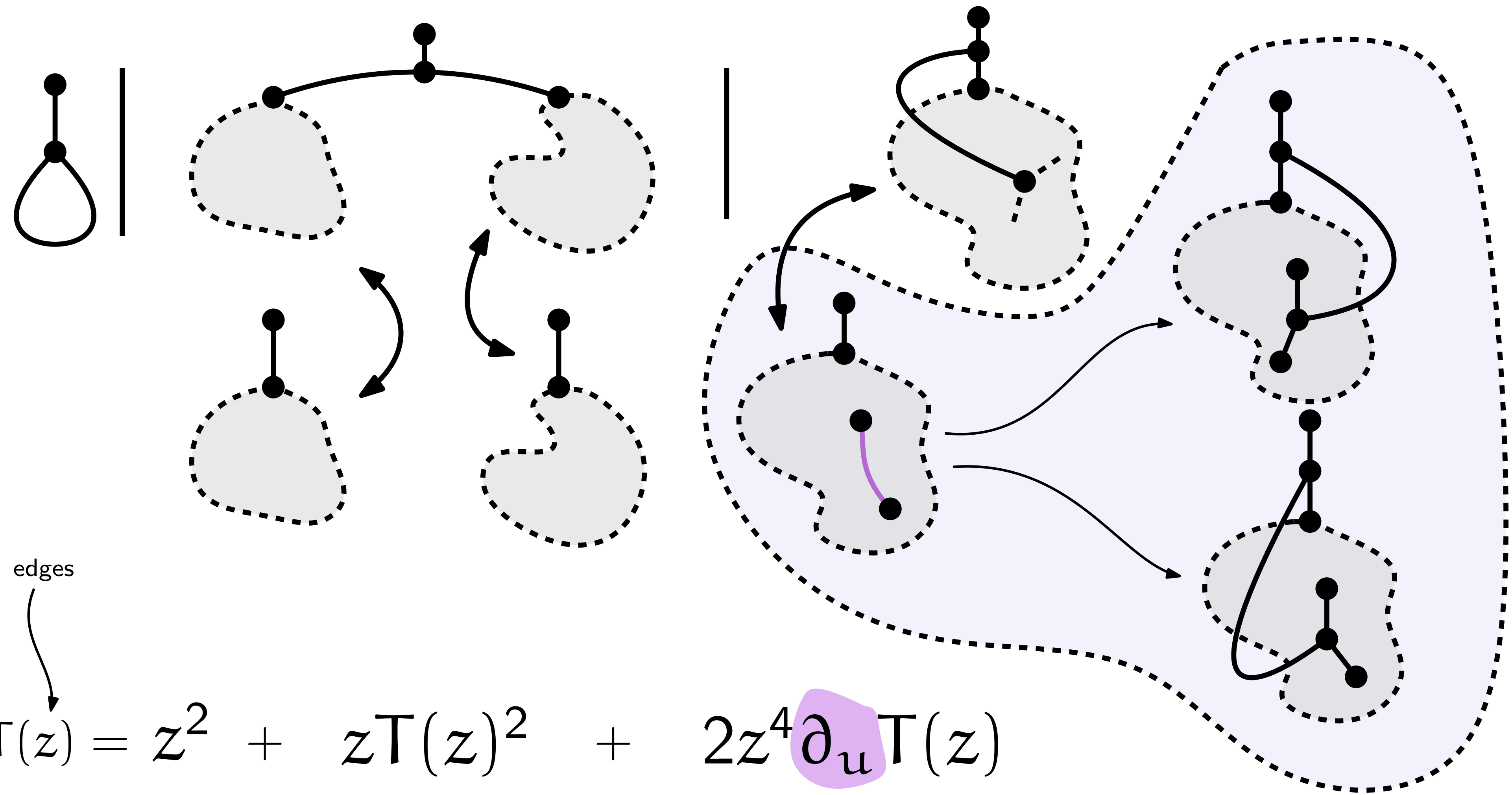


edges

$$\mathsf{T}(z) = z^2 \; + \; z\mathsf{T}(z)^2$$

Why should you, a combinatorialist, be interested in λ-terms?

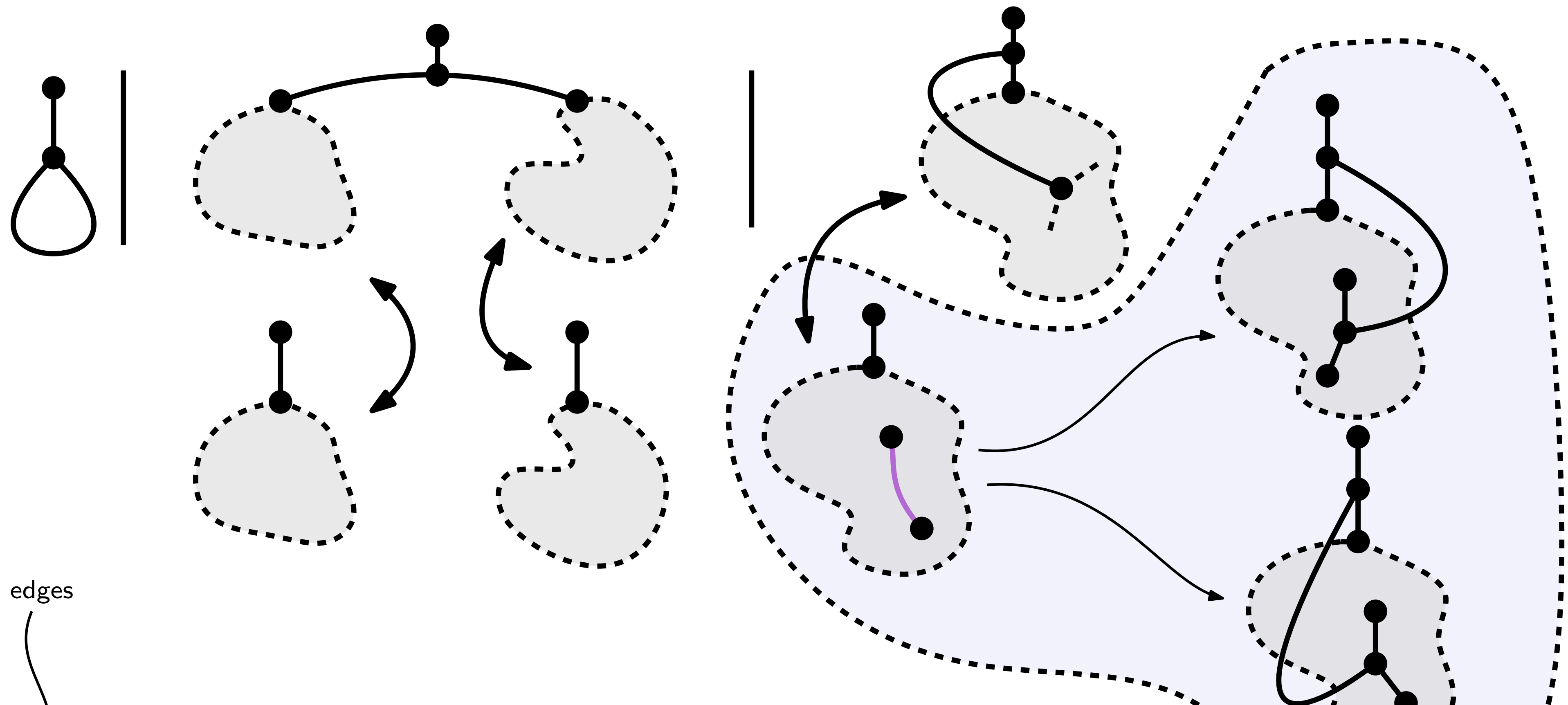
Decomposing (closed) rooted trivalent maps [BGJ13]



$$\mathsf{T}(z) = z^2 + z\mathsf{T}(z)^2 + 2z^4\partial_u \mathsf{T}(z)$$

Why should you, a combinatorialist, be interested in λ-terms?

Decomposing (closed) rooted trivalent maps [BGJ13]
and closed linear terms!



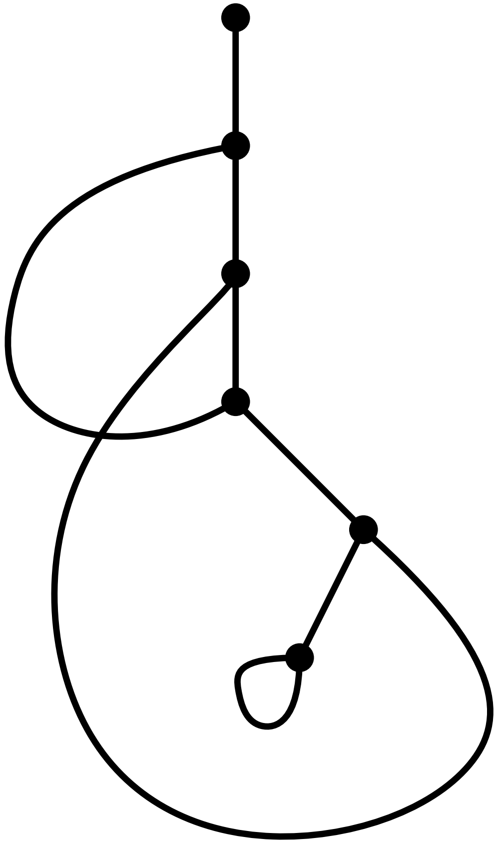
edges

$$\mathsf{T}(z) = z^2 \; + \; z\mathsf{T}(z)^2 \; + \; 2z^4 \partial_u \mathsf{T}(z)$$

subterms

$$\text{lin.term} = \lambda x.x \;\Big|\; (s\ t) \;\Big|\; \begin{array}{l}\lambda x.t[u := (x\ u)] \text{ or} \\ \lambda x.t[u := (u\ x)]\end{array}$$

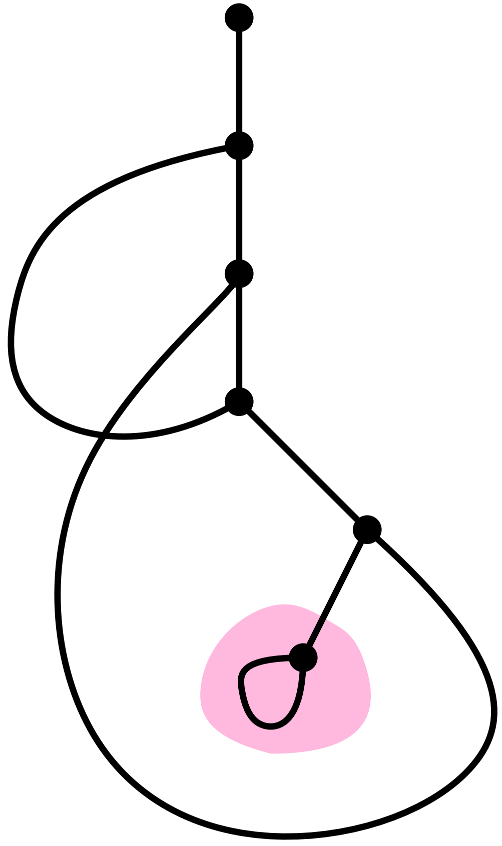
Some of our previous results: limit distributions



$$\lambda x.\lambda y.(y\ \lambda w.w)x$$

Some of our previous results: limit distributions

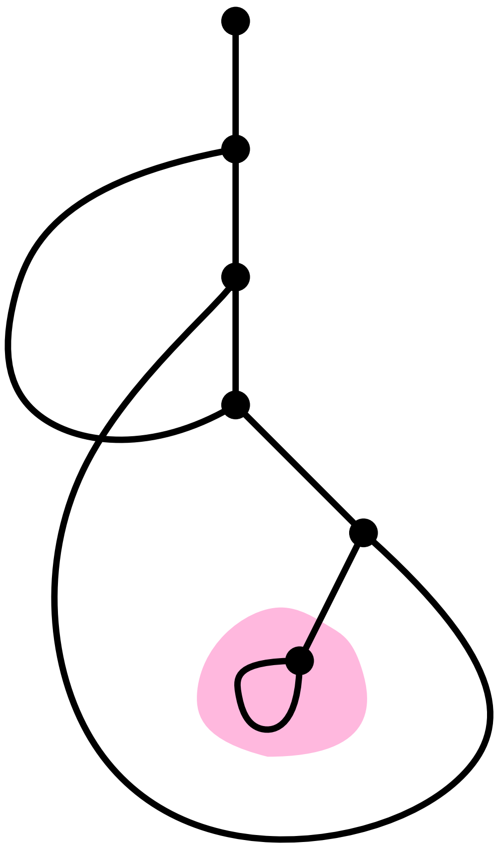
\# loops = \# id-subterms

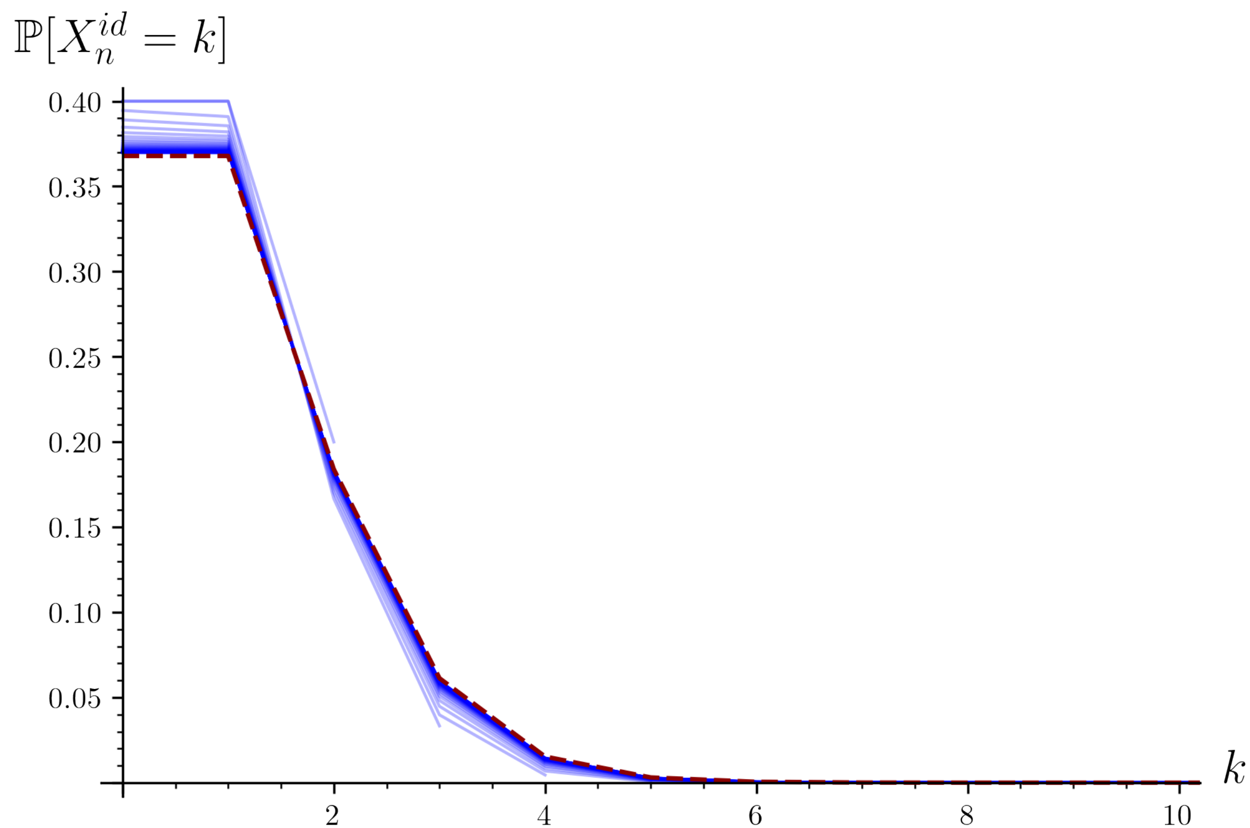


$$\lambda x.\lambda y.(y\ \lambda w.w)x$$

Some of our previous results: limit distributions

\# loops = # id-subterms

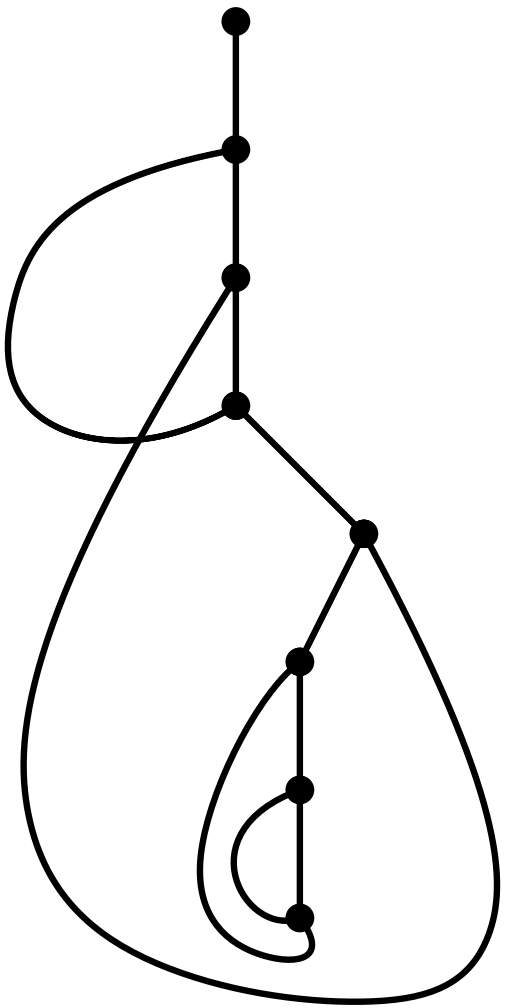




$$\lambda x.\lambda y.(y\ \lambda w.w)x$$

$$X_n^{\mathrm{id}} \xrightarrow{D} \mathrm{Poisson}(1)$$

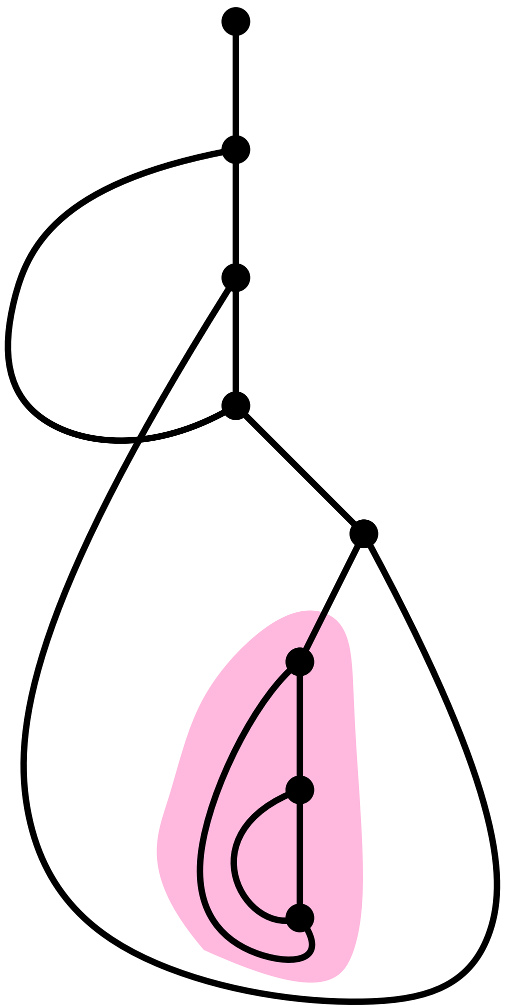
Some of our previous results: limit distributions



$\lambda x.\lambda y.(y\ \lambda z.\lambda w.zw)x$

Some of our previous results: limit distributions

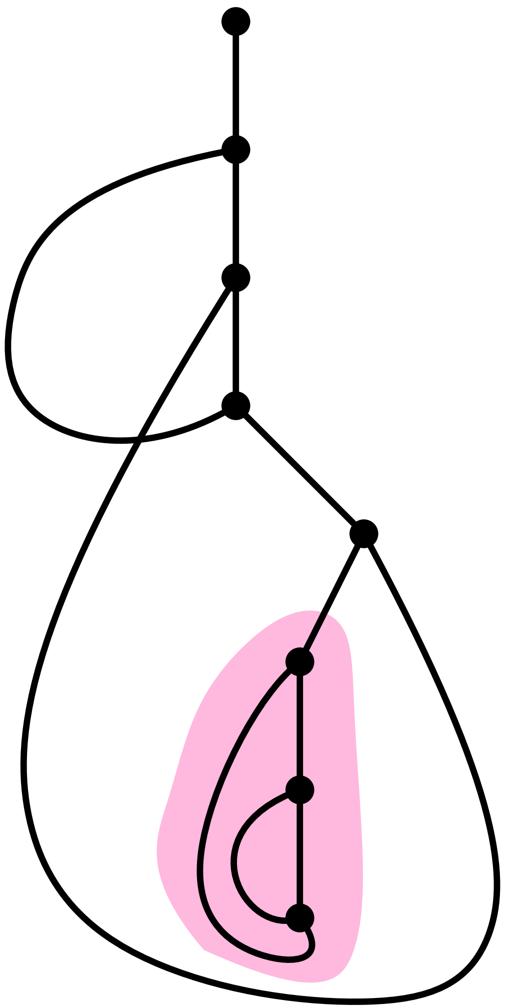
$\#$ bridges $= \#$ closed subterms

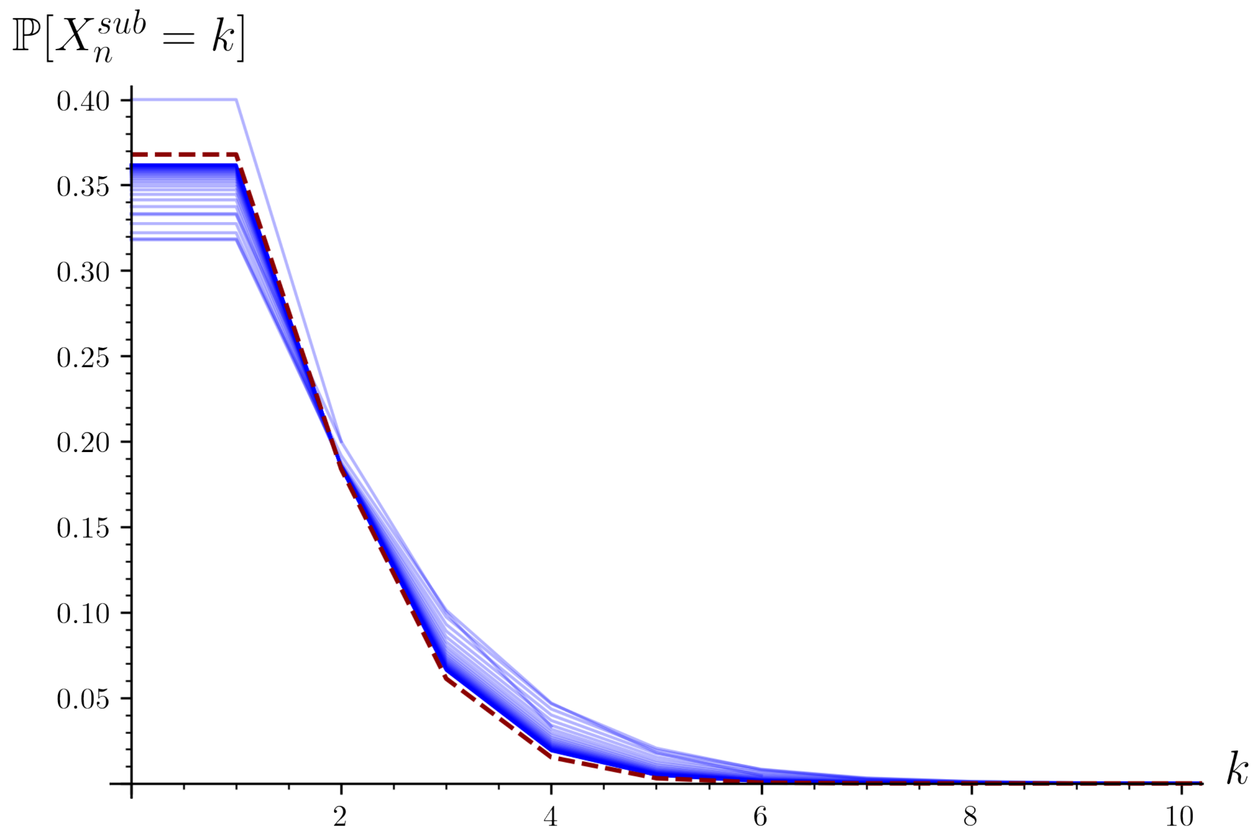


$\lambda x.\lambda y.(y\ \lambda z.\lambda w.zw)x$

Some of our previous results: limit distributions

\# bridges = # closed subterms

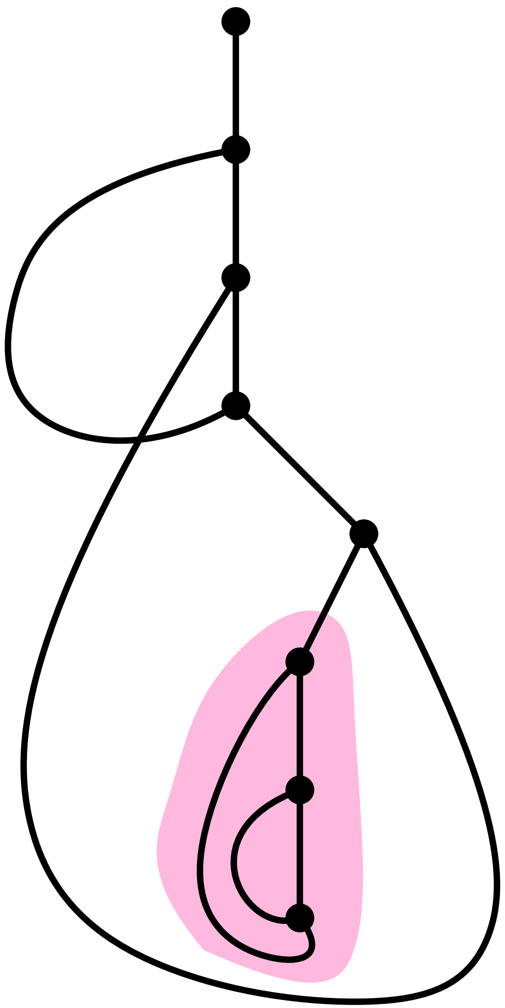


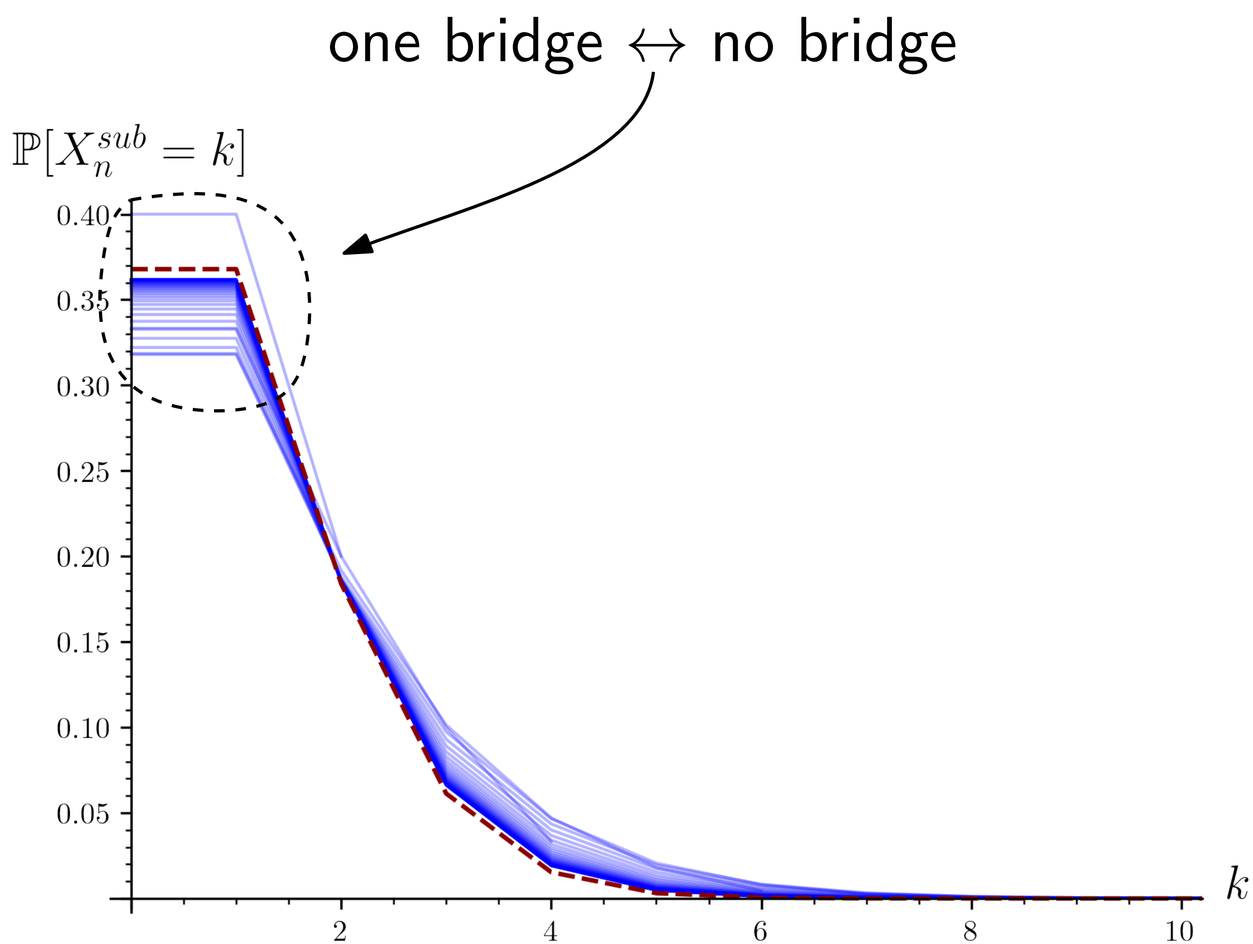


$\lambda x.\lambda y.(y\ \lambda z.\lambda w.zw)x$ $\qquad X_n^{sub} \xrightarrow{D} \mathrm{Poisson}(1)$

Some of our previous results: limit distributions

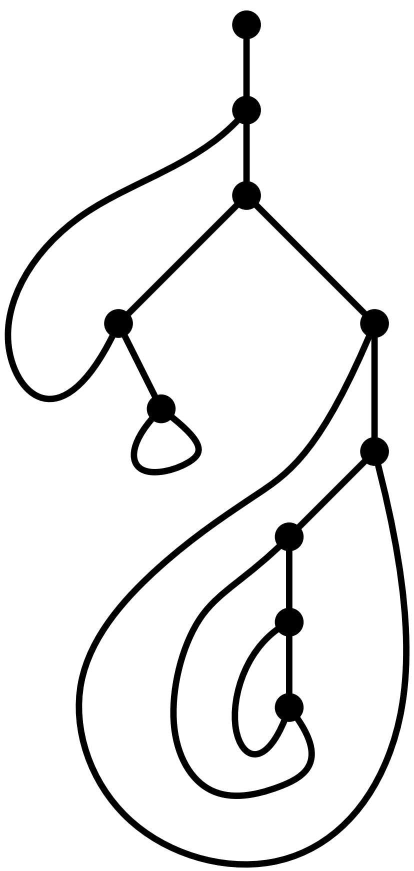
\# bridges = # closed subterms





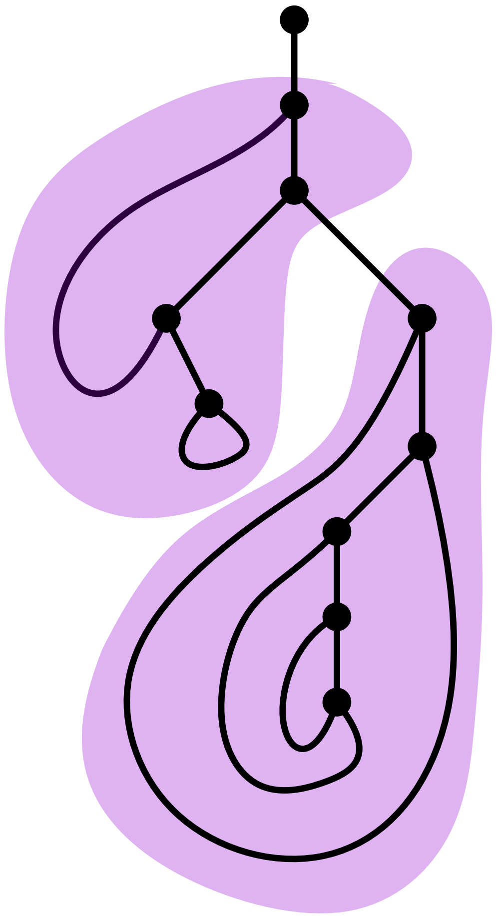
$\lambda x.\lambda y.(y\ \lambda z.\lambda w.zw)x$ $\qquad X_n^{sub} \xrightarrow{D} \mathrm{Poisson}(1)$

β-reduction as map rewriting



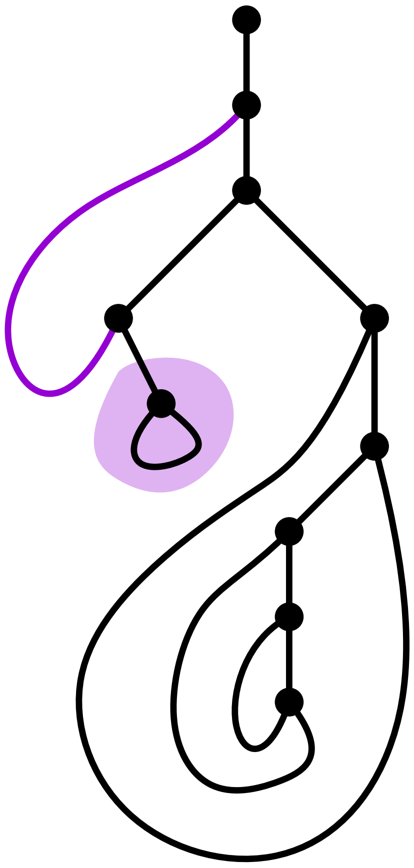
$\lambda x.(\lambda y.y(\lambda z.\lambda w.zw))((\lambda u.u)x)$

β-reduction as map rewriting



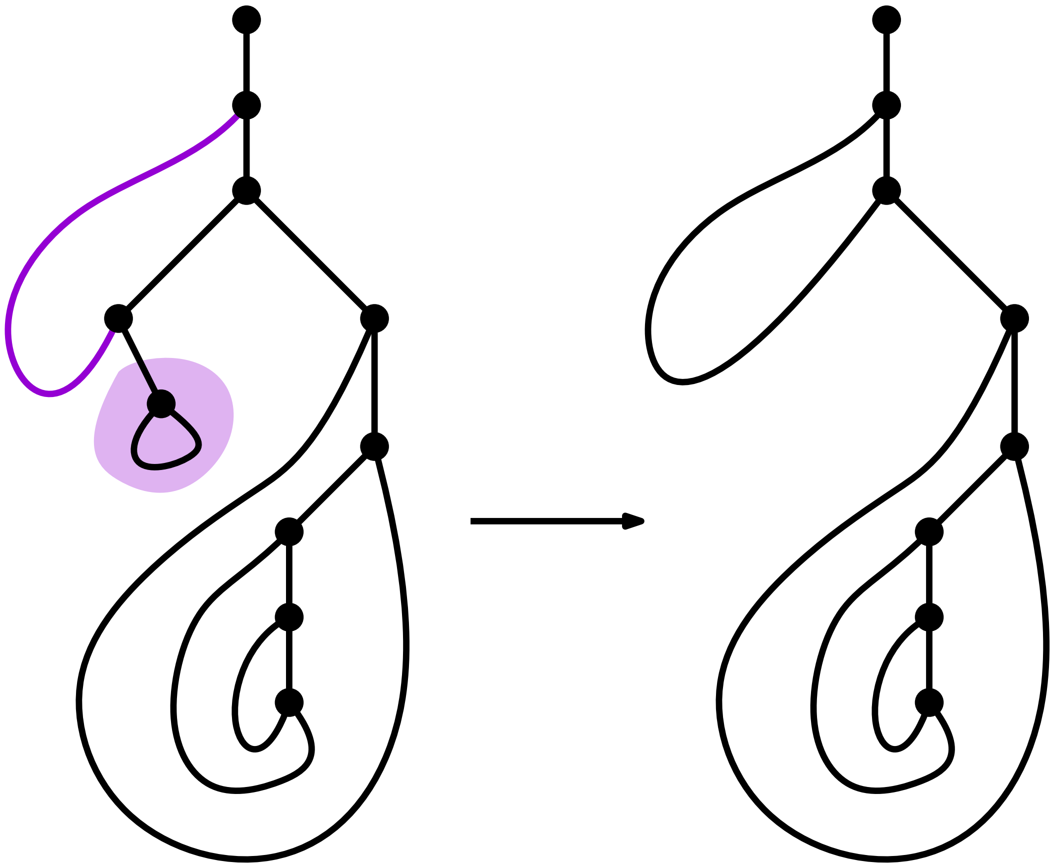
$\lambda x.(\lambda y.y(\lambda z.\lambda w.zw))((\lambda u.u)x)$

β-reduction as map rewriting



$$\lambda x.(\lambda y.y(\lambda z.\lambda w.zw))((\lambda u.u)x)$$

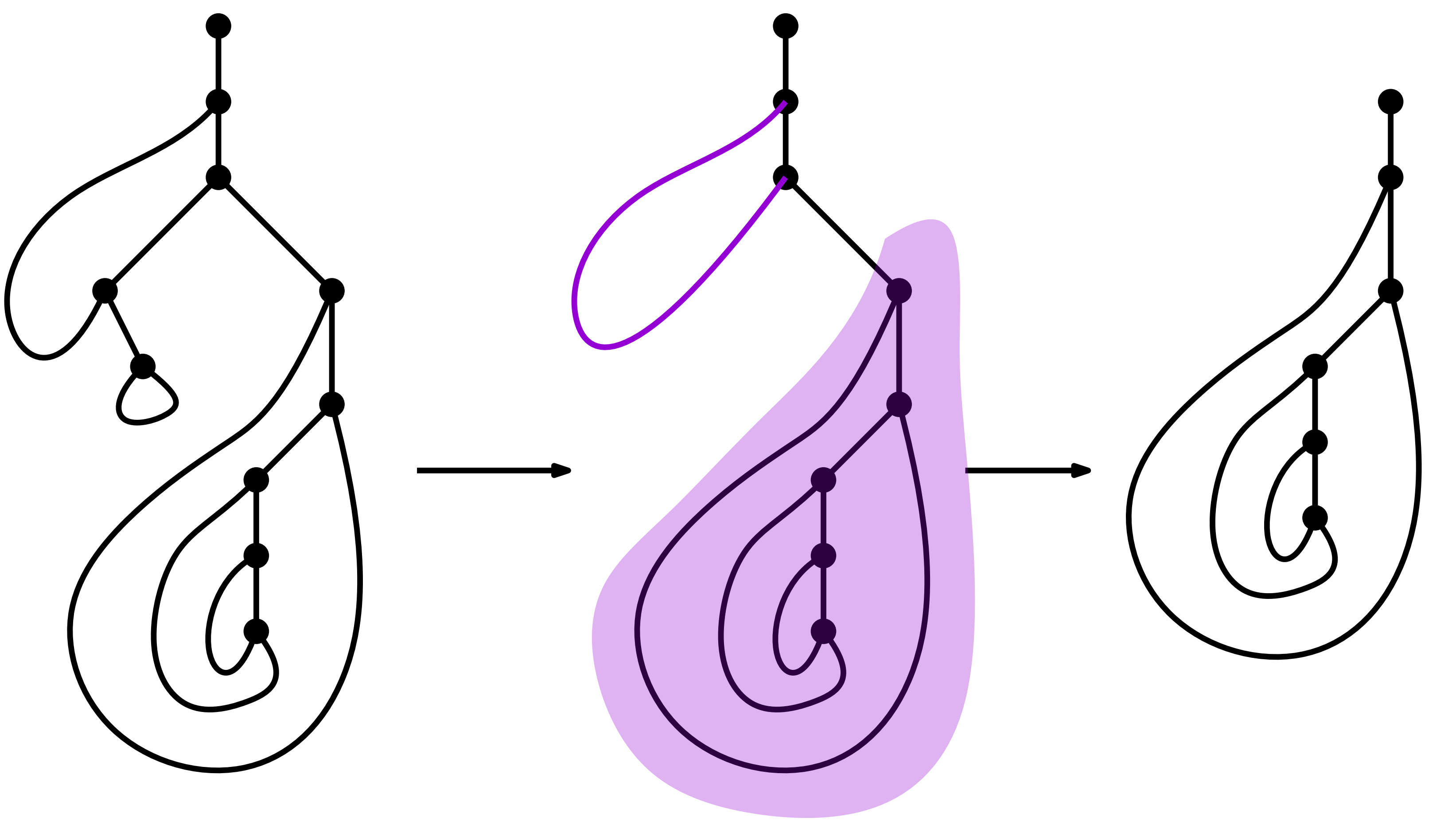
β-reduction as map rewriting



$$\lambda x.(\lambda y.y(\lambda z.\lambda w.zw))((\lambda u.u)x)$$

$$\longrightarrow \lambda x.((\lambda y.y(\lambda z.\lambda w.zw))x)$$

β-reduction as map rewriting



$\lambda x.(\lambda y.y(\lambda z.\lambda w.zw))((\lambda u.u)x)$

$\lambda x.((\lambda y.y(\lambda z.\lambda w.zw))x)$

$\lambda x.x\ (\lambda z.\lambda w.zw)$

Our workflow:

Our workflow:

1) Track evolution of parameter during an appropriately chosen decomposition of closed linear terms/trivalent maps.

Our workflow:

1) Track evolution of parameter during an appropriately chosen decomposition of closed linear terms/trivalent maps.

There's a lot, based on: differential equations, exponential Hadamard products, etc

Our workflow:

1) Track evolution of parameter during an appropriately chosen decomposition of closed linear terms/trivalent maps.

There's a lot, based on: differential equations, exponential Hadamard products, etc

resulting OGFs are purely formal, which makes them difficult to analyse!

Our workflow:

1) Track evolution of parameter during an appropriately chosen decomposition of closed linear terms/trivalent maps.

There's a lot, based on: differential equations, exponential Hadamard products, etc

resulting OGFs are purely formal, which makes them difficult to analyse!

2) Find appropriate tools to deal with their analysis.

Our workflow:

1) Track evolution of parameter during an appropriately chosen decomposition of closed linear terms/trivalent maps.

There's a lot, based on: differential equations, exponential Hadamard products, etc

resulting OGFs are purely formal, which makes them difficult to analyse!

2) Find appropriate tools to deal with their analysis.

Our workflow:

1) Track evolution of parameter during an appropriately chosen decomposition of closed linear terms/trivalent maps.

There's a lot, based on: differential equations, exponential Hadamard products, etc

resulting OGFs are purely formal, which makes them difficult to analyse!

2) Find appropriate tools to deal with their analysis.

- Bender's theorem for compositions $F(z, G(z))$

Our workflow:

1) Track evolution of parameter during an appropriately chosen decomposition of closed linear terms/trivalent maps.

There's a lot, based on: differential equations, exponential Hadamard products, etc

resulting OGFs are purely formal, which makes them difficult to analyse!

2) Find appropriate tools to deal with their analysis.

- Bender's theorem for compositions $F(z, G(z))$
- Coefficient asymptotics of Cauchy products

$$[z^n](A \cdot B) = \sum_{k=n_0}^{n} a_k b_{n-k}$$

first order asymptotics given by $k = n_0$ and $k = n$

Our workflow:

1) Track evolution of parameter during an appropriately chosen decomposition of closed linear terms/trivalent maps.

There's a lot, based on: differential equations, exponential Hadamard products, etc

resulting OGFs are purely formal, which makes them difficult to analyse!

crucial ingredient: coefficients are growing *rapidly*

2) Find appropriate tools to deal with their analysis.

- Bender's theorem for compositions $F(z, G(z))$
- Coefficient asymptotics of Cauchy products

$$[z^n](A \cdot B) = \sum_{k=n_0}^{n} a_k b_{n-k}$$

first order asymptotics given by $k = n_0$ and $k = n$

Mean number of β-redices in closed terms

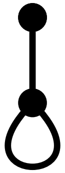
Mean number of β-redices in closed terms

- Tracking redices during the decomposition

Mean number of β-redices in closed terms

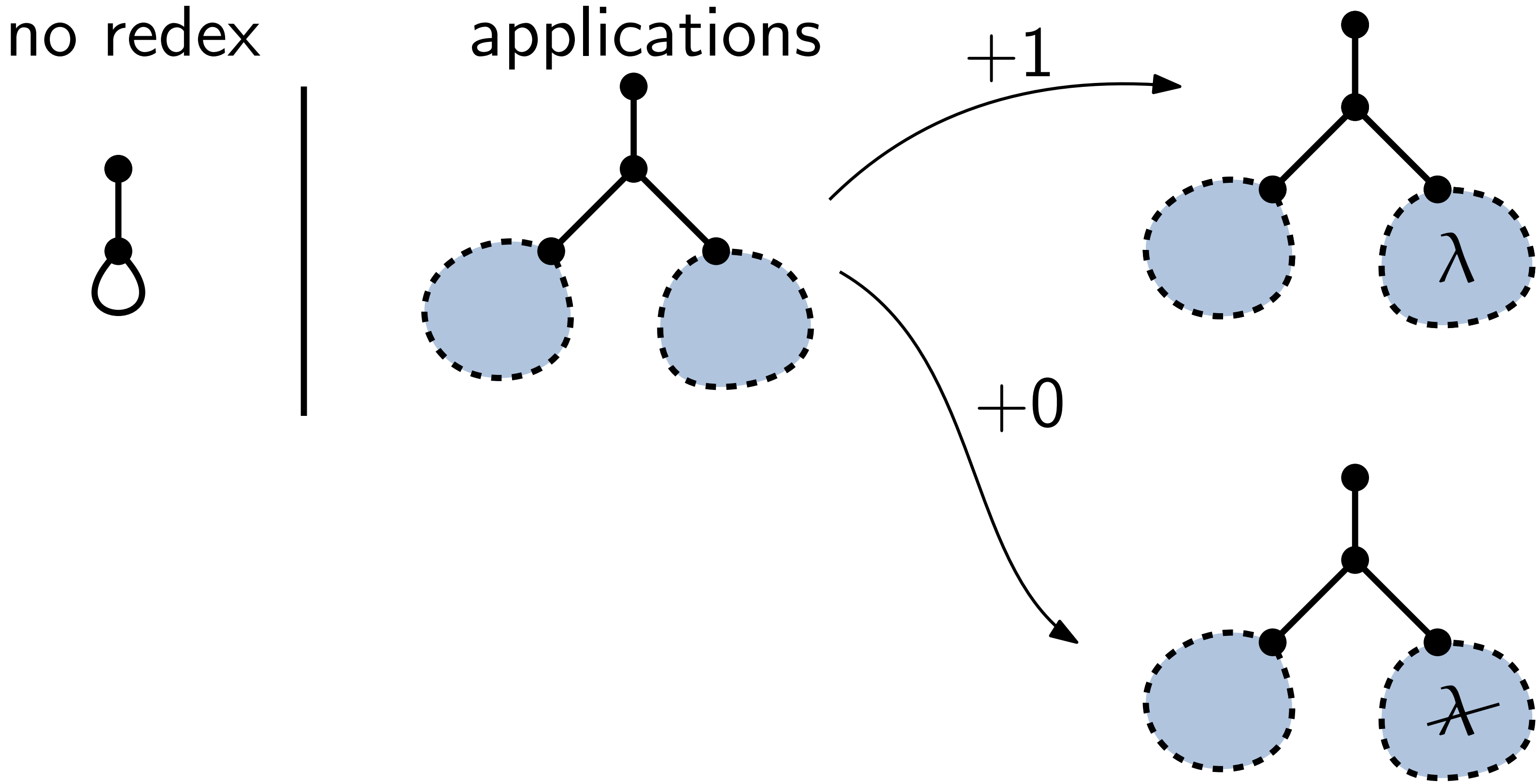
- Tracking redices during the decomposition

no redex



Mean number of β-redices in closed terms

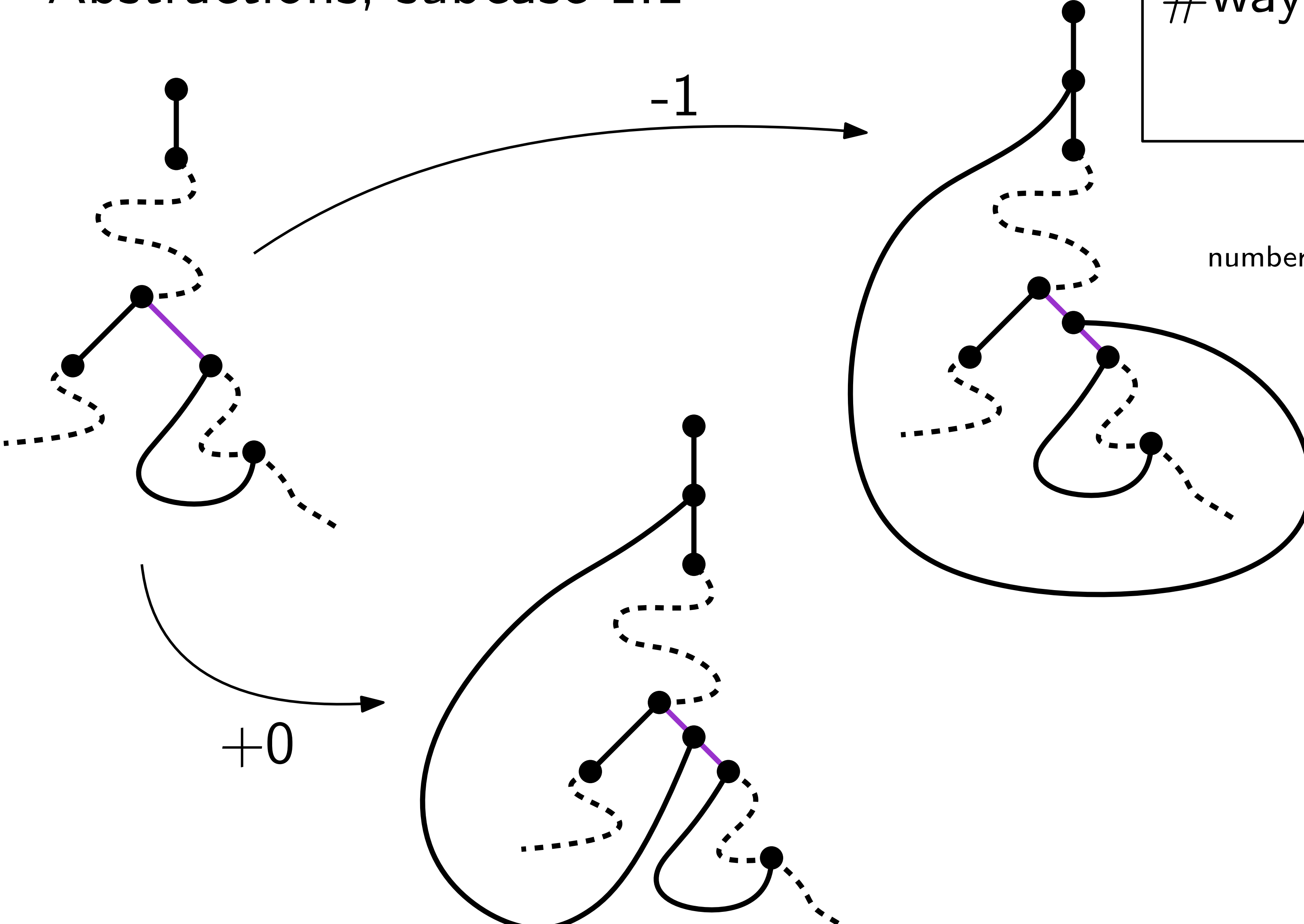
- Tracking redices during the decomposition



Mean number of β-redices in closed terms

- Tracking redices during the decomposition

Abstractions, subcase 1.1



$\#$ways to do this

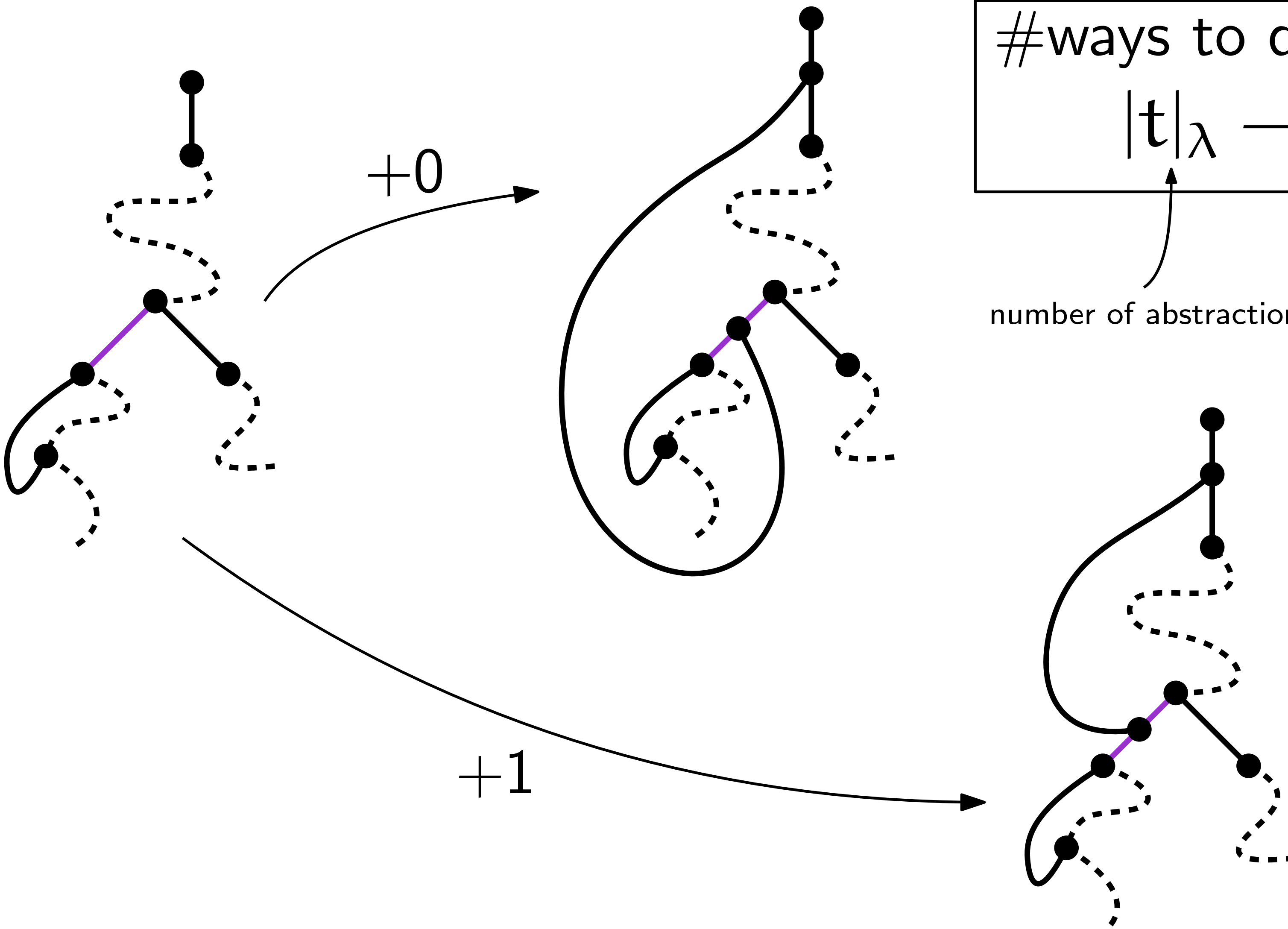
$$|t|_\beta$$

number of redices in t

Mean number of β-redices in closed terms

- Tracking redices during the decomposition

Abstractions, subcase 1.2



#ways to do this

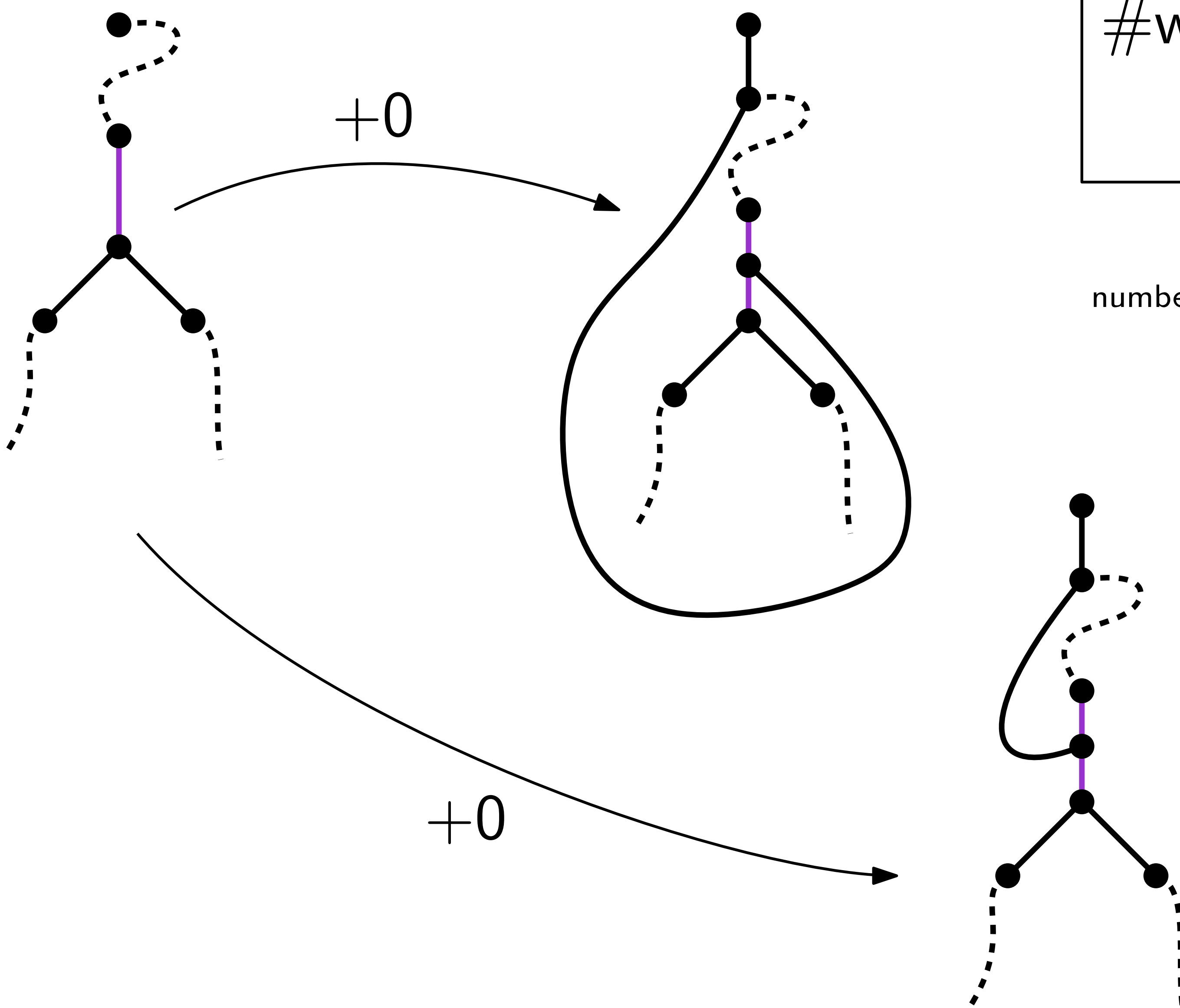
$$|t|_\lambda - |t|_\beta$$

number of abstractions in t

Mean number of β-redices in closed terms

- Tracking redices during the decomposition

Abstractions, subcase 1.3



#ways to do this

$$|t| - |t|_\lambda$$

number of subterms in t = size of t

Mean number of β-redices in closed terms

- Building the specification of the OGF

- $|t|_\lambda = \frac{|t|+1}{3}, |t| - |t|_\lambda = \frac{2|t|-1}{3}$
- $r\partial_r T_0 = \sum_{t\in T_0} |t|_\beta z^{|t|} r^{|t|_\beta}$
- $\frac{z\partial_z T_0 + T_0}{3} = \sum_{t\in T_0} \frac{|t|+1}{3} z^{|t|} v^{|t|_\beta}$
- $\frac{2z\partial_z T_0 - T_0}{3} = \sum_{t\in T_0} \frac{2|t|-1}{3} z^{|t|} v^{|t|_\beta}$

Mean number of β-redices in closed terms

- Translating to a differential equation and pumping

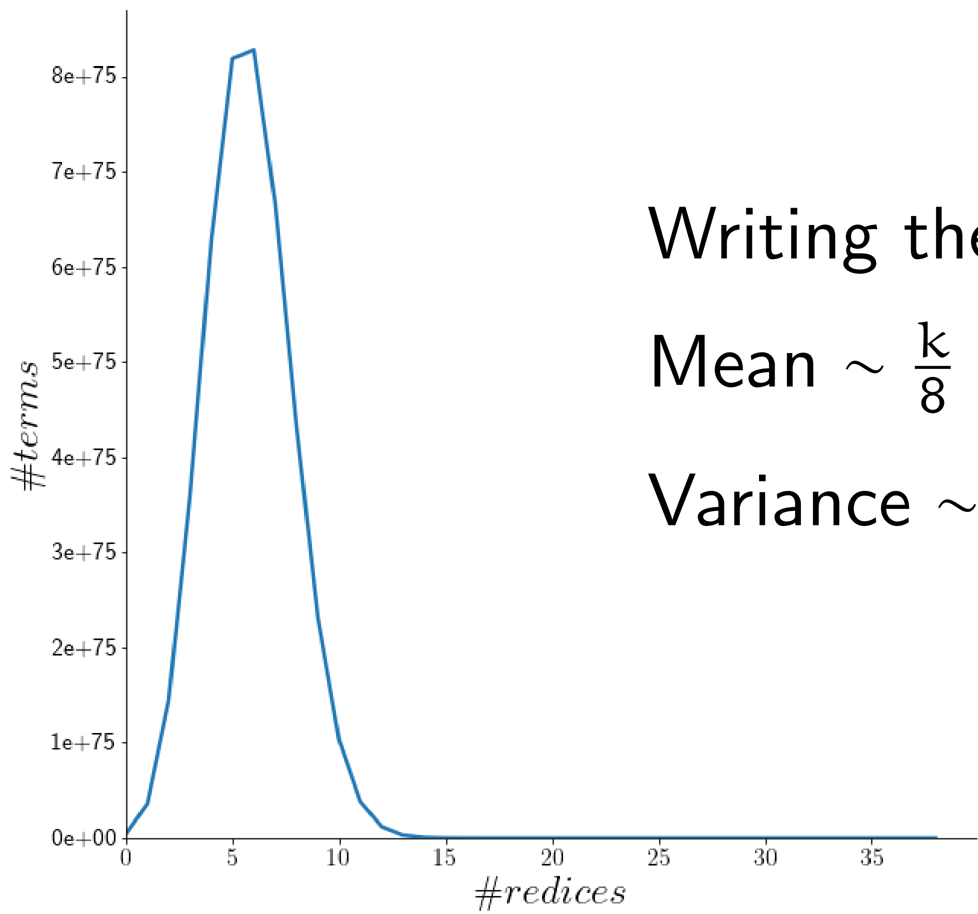
$$T_0 = -z\left(z^2(r+1)(1+(r-1)zT)(r-1)\partial_r T_0 - \frac{(1+z(r-1)T)z^3(r+5)\partial_z T_0}{3} - \frac{z^3(r-1)^2 T_0^2}{3} - \frac{4z^2(r-1)T_0}{3} - z - T_0^2\right)$$

Mean number of β-redices in closed terms

- Translating to a differential equation and pumping

$$T_0 = -z\left(z^2(r+1)(1+(r-1)zT)(r-1)\partial_r T_0 - \frac{(1+z(r-1)T)z^3(r+5)\partial_z T_0}{3} - \frac{z^3(r-1)^2 T_0^2}{3} - \frac{4z^2(r-1)T_0}{3} - z - T_0^2\right)$$

A plot of the dist. of redices for terms/maps of size $n = 119$



Writing the size as $n = 3k + 2$, we have:

Mean $\sim \frac{k}{8}$

Variance $\sim \frac{29k}{320}$

Whats next?

Whats next?

- Tracking the three patterns whose reduction alters the number of redices (WIP with Bodini, Zeilberger, Wallner, Gittenberger)

$$(\lambda x.C[(x\ u)])(\lambda y.t_2) \qquad ((\lambda x.\lambda y.t_1)t_2)t_3$$

$$(\lambda x.x)(\lambda y.t_1)t_2$$

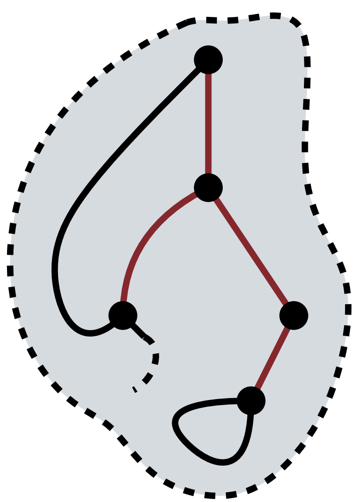
Whats next?

- Tracking the three patterns whose reduction alters the number of redices (WIP with Bodini, Zeilberger, Wallner, Gittenberger)

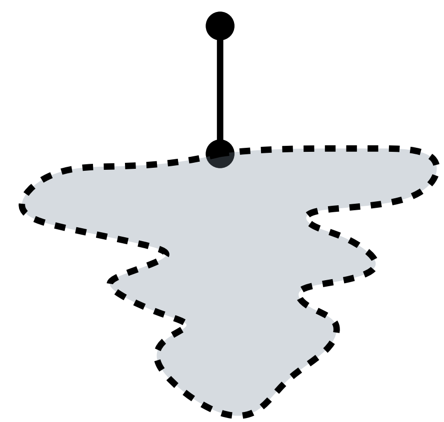
$$(\lambda x.C[(x\ u)])(\lambda y.t_2) \qquad ((\lambda x.\lambda y.t_1)t_2)t_3$$

$$(\lambda x.x)(\lambda y.t_1)t_2$$

- More parameters:



Mean path length



Profile

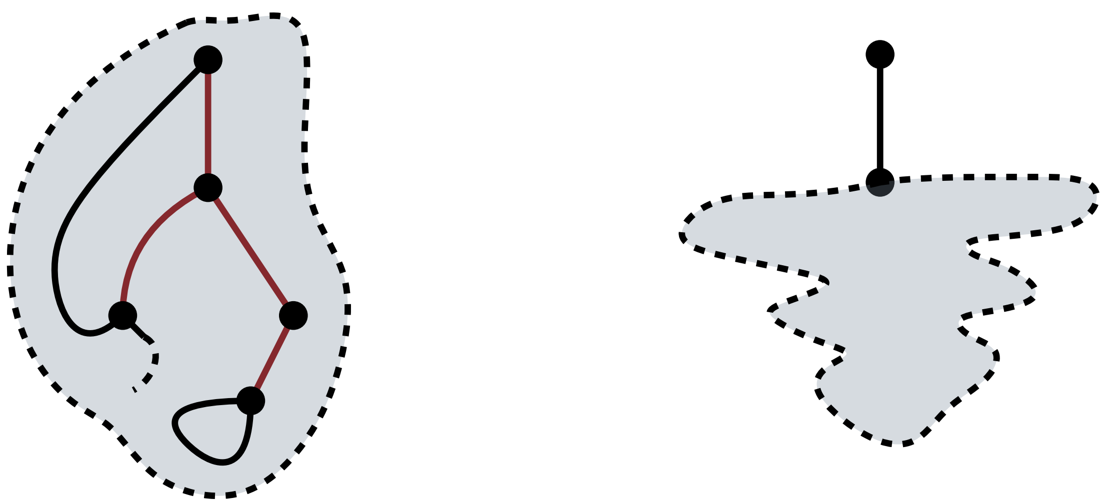
Whats next?

- Tracking the three patterns whose reduction alters the number of redices (WIP with Bodini, Zeilberger, Wallner, Gittenberger)

$$(\lambda x.C[(x\ u)])(\lambda y.t_2) \qquad ((\lambda x.\lambda y.t_1)t_2)t_3$$

$$(\lambda x.x)(\lambda y.t_1)t_2$$

- More parameters:



Mean path length   Profile

Thank you!

Bibliography

[BGGJ13] Bodini, O., Gardy, D., Gittenberger, B., & Jacquot, A. (2013).
Enumeration of Generalized BCI Lambda-terms.
The Electronic Journal of Combinatorics, P30-P30.

[Z16] Zeilberger, N. (2016).
Linear lambda terms as invariants of rooted trivalent maps.
Journal of functional programming, 26.

[AB00] Arques, D., & Béraud, J. F. (2000).
Rooted maps on orientable surfaces, Riccati's equation and continued fractions
Discrete mathematics, 215(1-3), 1-12.

[BFSS01] Banderier, C., Flajolet, P., Schaeffer, G., & Soria, M. (2001).
Random maps, coalescing saddles, singularity analysis, and Airy phenomena.
Random Structures & Algorithms, 19(3-4), 194-246.

Bibliography

[BR86] Bender, E. A., & Richmond, L. B. (1986).
A survey of the asymptotic behaviour of maps.
Journal of Combinatorial Theory, Series B, 40(3), 297-329.

[BGLZ16] Bendkowski, M., Grygiel, K., Lescanne, P., & Zaionc, M. (2016).
A natural counting of lambda terms.
In International Conference on Current Trends in Theory and Practice of Informatics (pp. 183-194). Springer, Berlin, Heidelberg.

[BBD19] Bendkowski, M., Bodini, O., & Dovgal, S. (2019).
Statistical Properties of Lambda Terms.
The Electronic Journal of Combinatorics, P4-1.

[BCDH18] Bodini, O., Courtiel, J., Dovgal, S., & Hwang, H. K. (2018, June).
Asymptotic distribution of parameters in random maps.
In 29th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms (Vol. 110, pp. 13-1)

Bibliography

[B75] Bender, E. A. (1975).
An asymptotic expansion for the coefficients of some formal power series.
Journal of the London Mathematical Society, 2(3), 451-458.

[FS93] Flajolet, P., & Soria, M. (1993).
General combinatorial schemas: Gaussian limit distributions and exponential tails.
Discrete Mathematics, 114(1-3), 159-180.

[B18] Borinsky, M. (2018).
Generating Asymptotics for Factorially Divergent Sequences.
The Electronic Journal of Combinatorics, P4-1.

[BKW21] Banderier, C., Kuba, M., & Wallner, M. (2021).
Analytic Combinatorics of Composition schemes and phase transitions
mixed Poisson distributions.
arXiv preprint arXiv:2103.03751.

Bibliography

[BGJ13] Bodini, O., Gardy, D., & Jacquot, A. (2013).
Asymptotics and random sampling for BCI and BCK lambda terms
Theoretical Computer Science, 502, 227-238.

[M04] Mairson, H. G. (2004).
Linear lambda calculus and PTIME-completeness
Journal of Functional Programming, 14(6), 623-633.

[DGKRTZ13] Zaionc, M., Theyssier, G., Raffalli, C., Kozic, J.,
J., Grygiel, K., & David, R. (2013)
Asymptotically almost all λ-terms are strongly normalizing
Logical Methods in Computer Science, 9

[SAKT17] Sin'Ya, R., Asada, K., Kobayashi, N., & Tsukada, T. (2017)
Almost Every Simply Typed λ-Term Has a Long β-Reduction Sequence
In International Conference on Foundations of Software Science and
and Computation Structures (pp. 53-68). Springer, Berlin, Heidelberg.