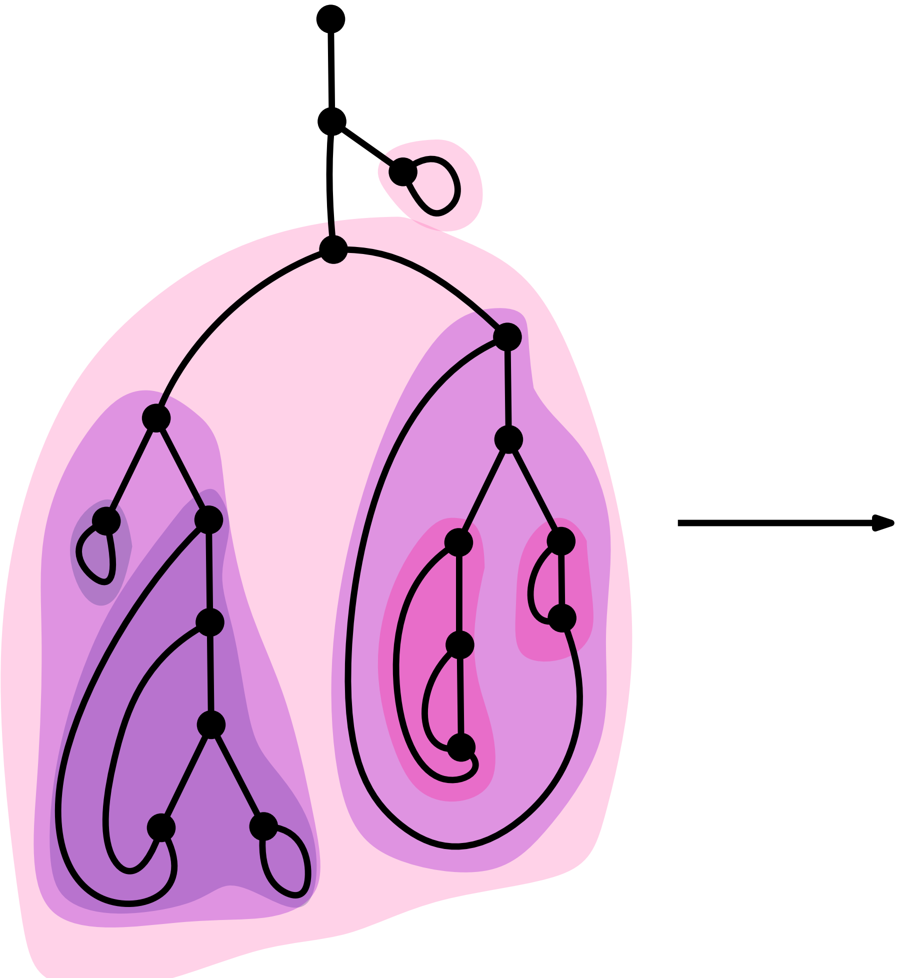
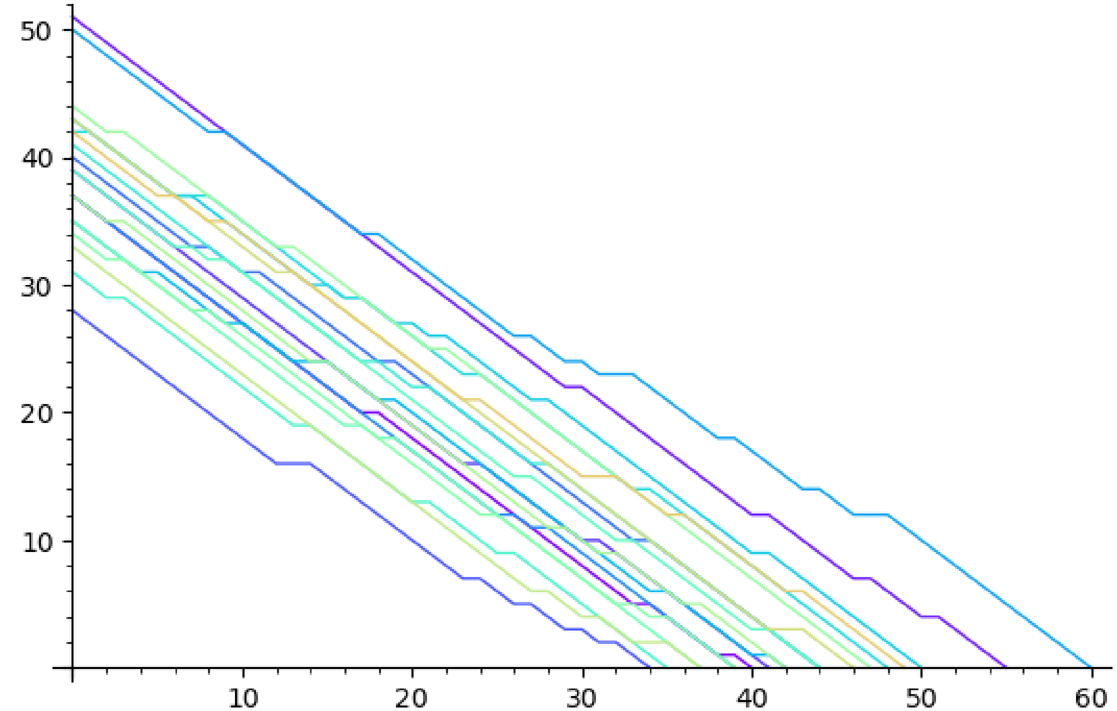
A lower bound on reduction length for random closed linear λ-terms





Algorithmic and Enumerative Combinatorics, 7 July 2022

Olivier Bodini (LIPN, Paris 13)
Michael Wallner (TU Wien)
Bernhard Gittenberger (TU WIen)
Noam Zeilberger (LIX, Polytechnique)
Alexandros Singh (LIPN, Paris 13)

The linear λ-calculus

- A **PTIME-complete** system of computation [M04]

The linear λ-calculus

- A **PTIME-complete** system of computation [M04]
- Its terms are formed inductively

$$f, t := x \mid \lambda x.t \mid (f\ t)$$

variables

The linear λ-calculus

- A **PTIME-complete** system of computation [M04]
- Its terms are formed inductively

$$f, t := x \mid \lambda x.t \mid (f\ t)$$

variables

abstractions
represent functions "$x \mapsto t$"
x appears **exactly once** inside t

The linear λ-calculus

- A **PTIME-complete** system of computation [M04]
- Its terms are formed inductively

$$f, t := x \mid \lambda x.t \mid (f\ t)$$

variables

abstractions
represent functions "$x \mapsto t$"
x appears **exactly once** inside t

applications
represent "$f(t)$"

- Terms considered up to (careful) renaming of variables:

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

Examples of linear λ-terms

$(\lambda x.(x\ y))$ open term

$(\lambda x.x)$ closed term

$(y\ (\lambda z.z))$ open term with closed subterm

Examples of linear λ-terms

$(\lambda x.(x\ y))$ open term

$(\lambda x.x)$ closed term

$(y\ (\lambda z.z))$ open term with closed subterm

Dynamics of the λ-calculus: β-reductions

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

represents:

$$f = x \mapsto t_1$$

$f(t_2)$: replace x with t_2 inside t_1

Examples of linear λ-terms

$(\lambda x.(x\ y))$ open term

$(\lambda x.x)$ closed term

$(y\ (\lambda z.z))$ open term with closed subterm

Dynamics of the λ-calculus: β-reductions

redex

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

represents:

$$f = x \mapsto t_1$$

$f(t_2)$: replace x with t_2 inside t_1

More on β-reductions

Examples of reductions

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

More on β-reductions

Examples of reductions

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

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

More on β-reductions

Examples of reductions

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

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

A term with no redices is called a normal form

More on β-reductions

Examples of reductions

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

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

A term with no redices is called a normal form

- Repeated β-reduction terminates with a unique normal form
- Starting from a random term, how many steps to reach the normal form?

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

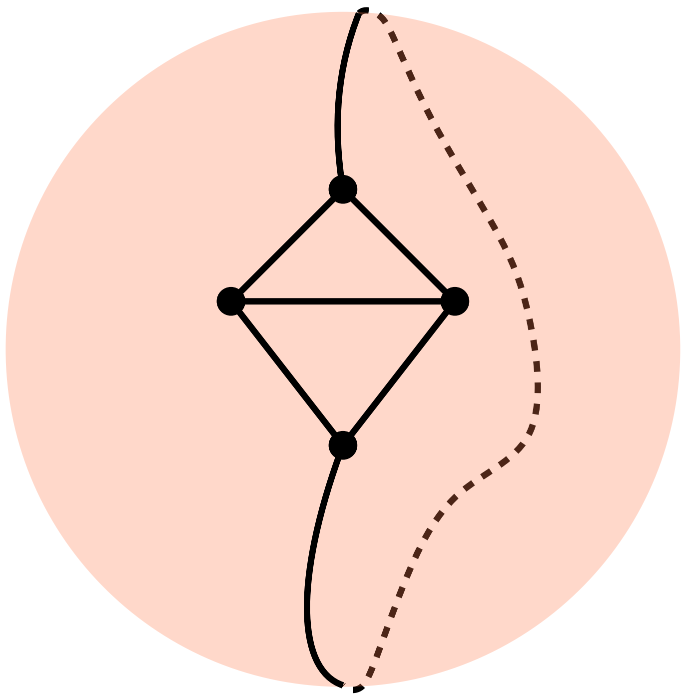
This motivates our first problem-to-solve:

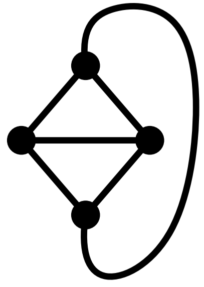
What is the number of β-redices in a random linear λ-term?

random variable (number of β-redices)

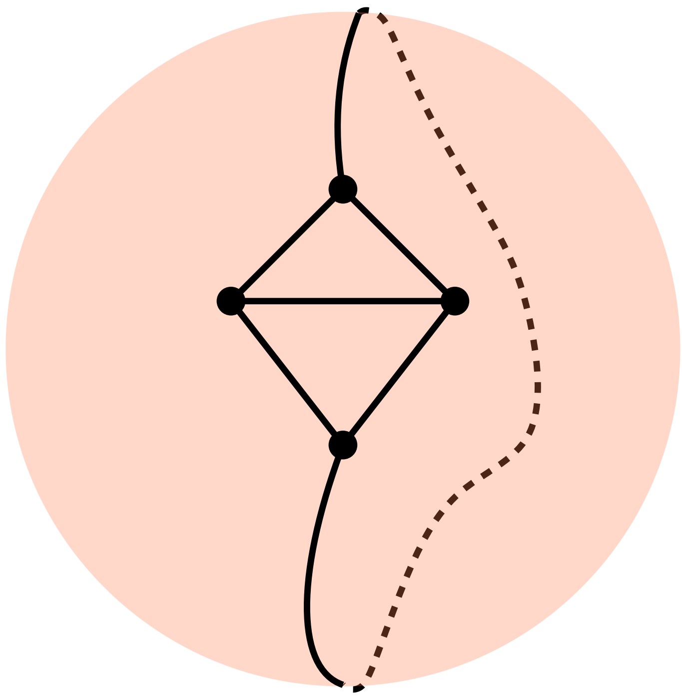
uniform distribution (random linear λ-term)

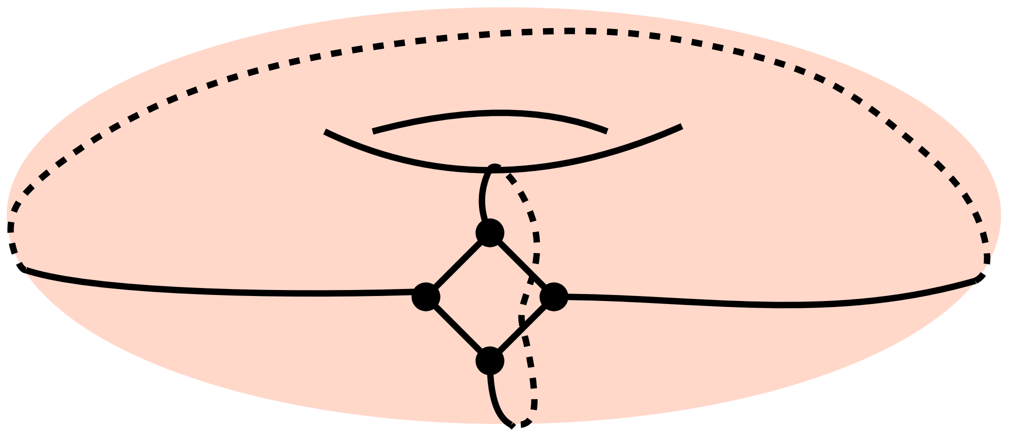
What are maps?

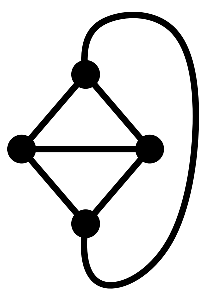


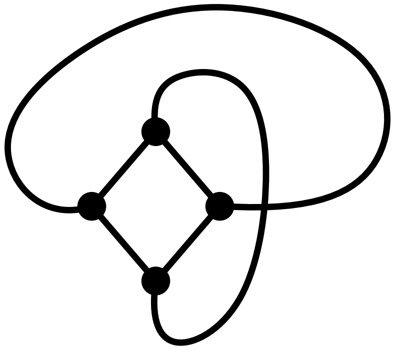


What are maps?

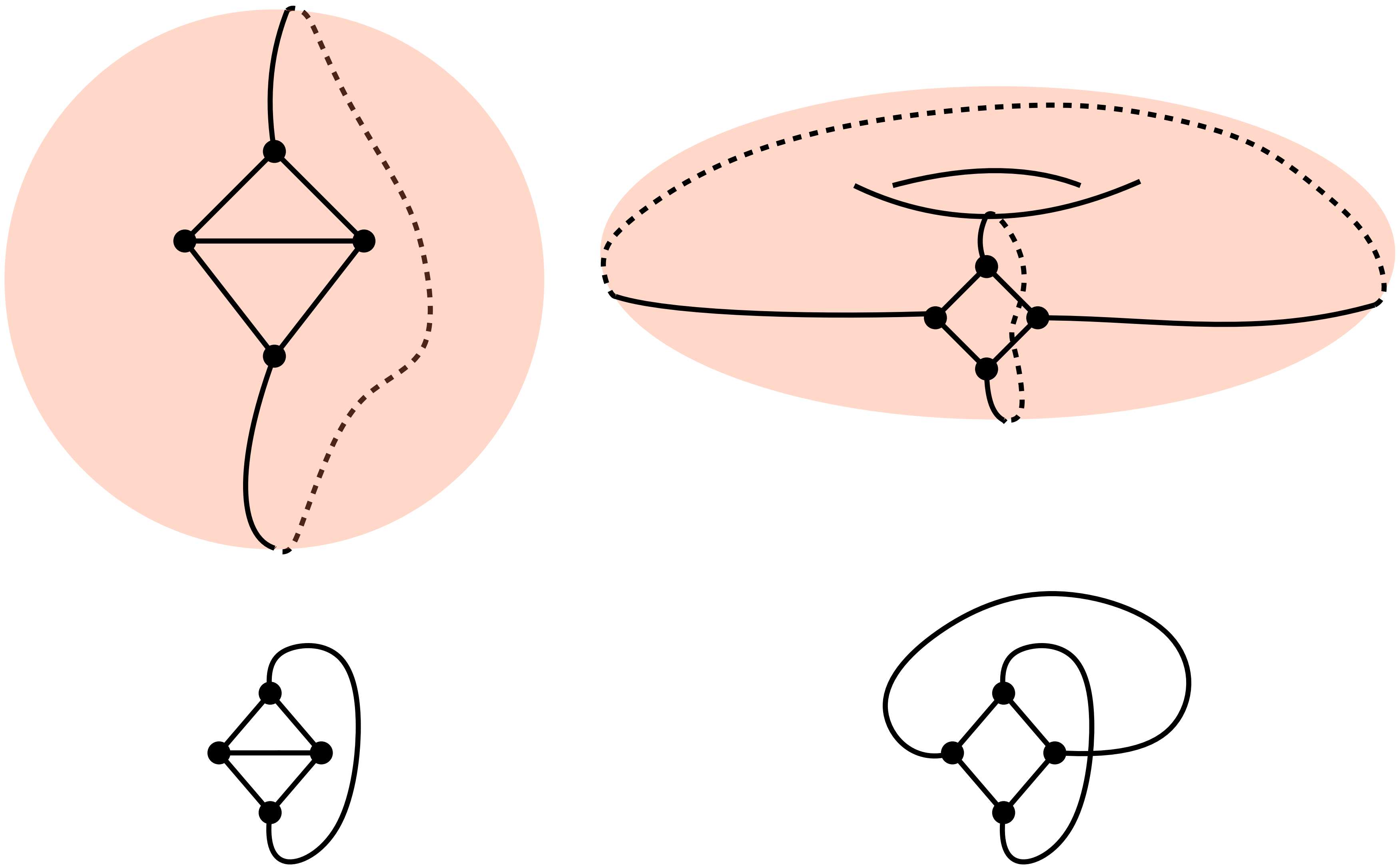








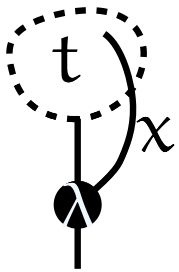
What are maps?



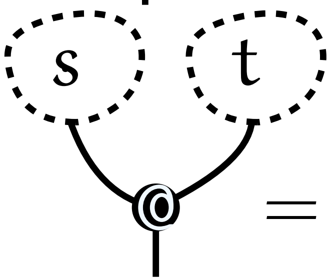
We're interested in unrestricted genus cubic maps

Why should logicians be interested in maps?

$= x$

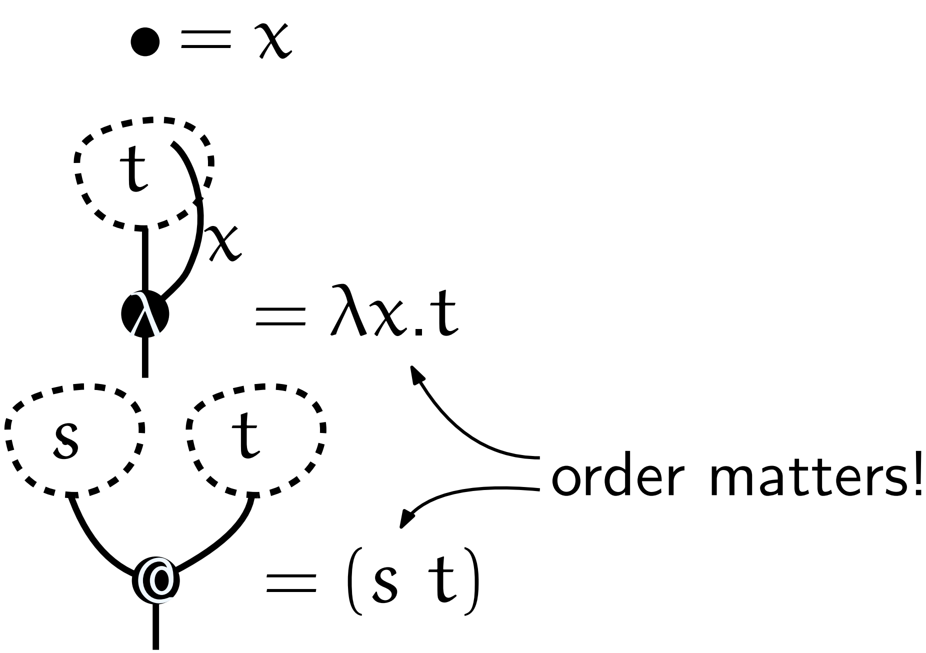


$= \lambda x.t$

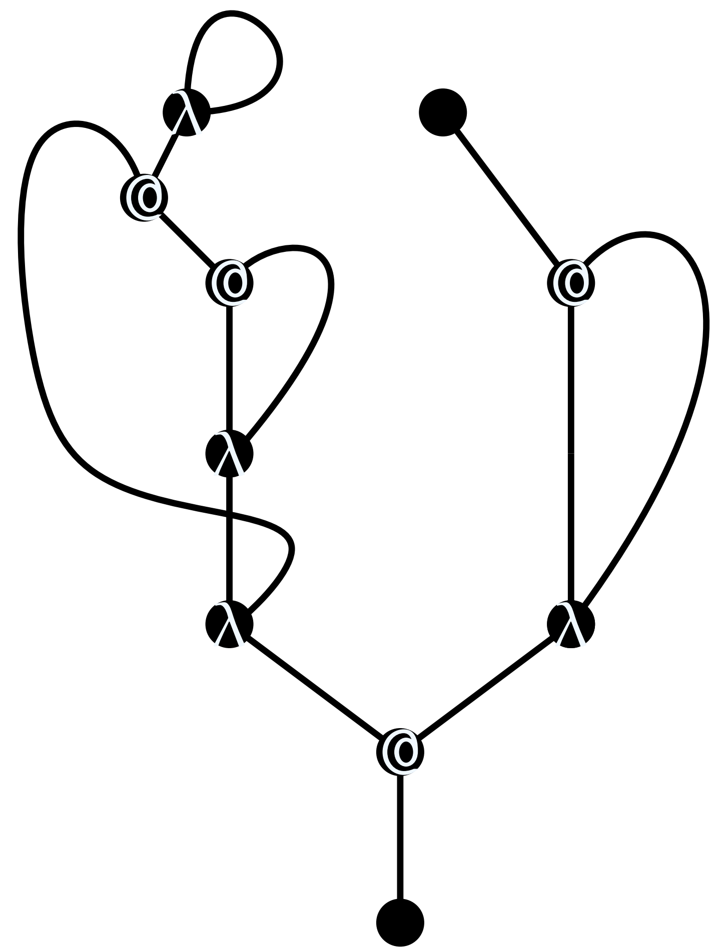


$= (s\ t)$

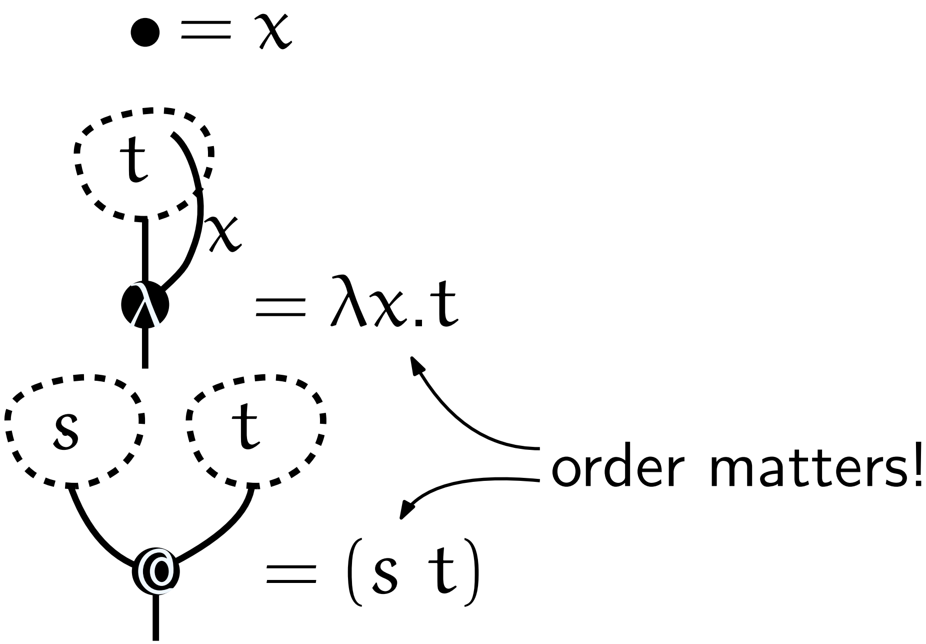
Why should logicians be interested in maps?



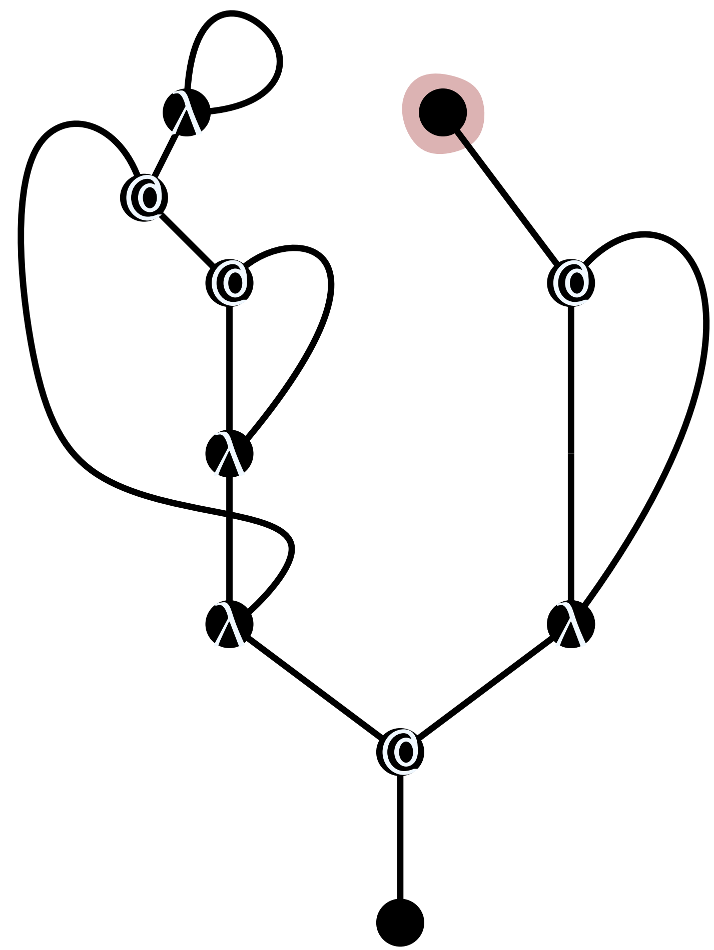
$$(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.a\ u)$$



Why should logicians be interested in maps?



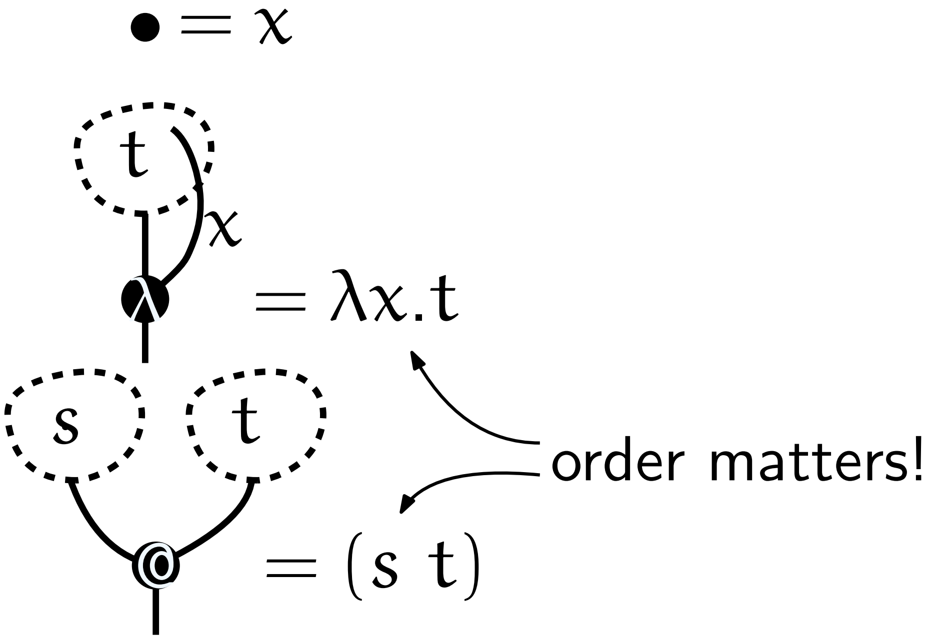
$(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.a\ u)$



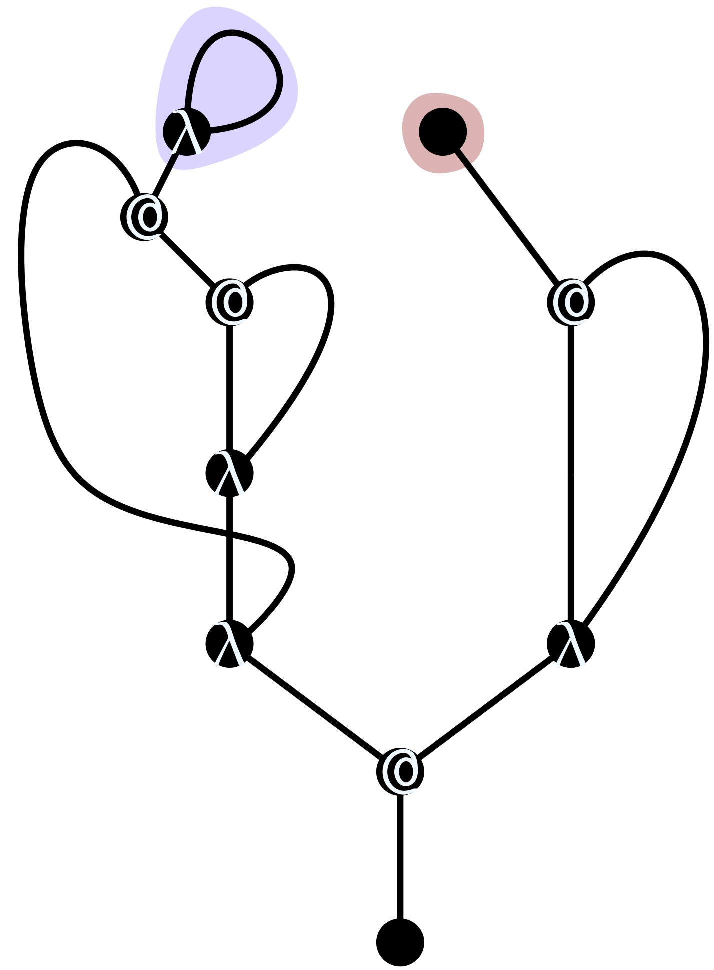
Dictionary

- Free var ↔ unary vertex

Why should logicians be interested in maps?



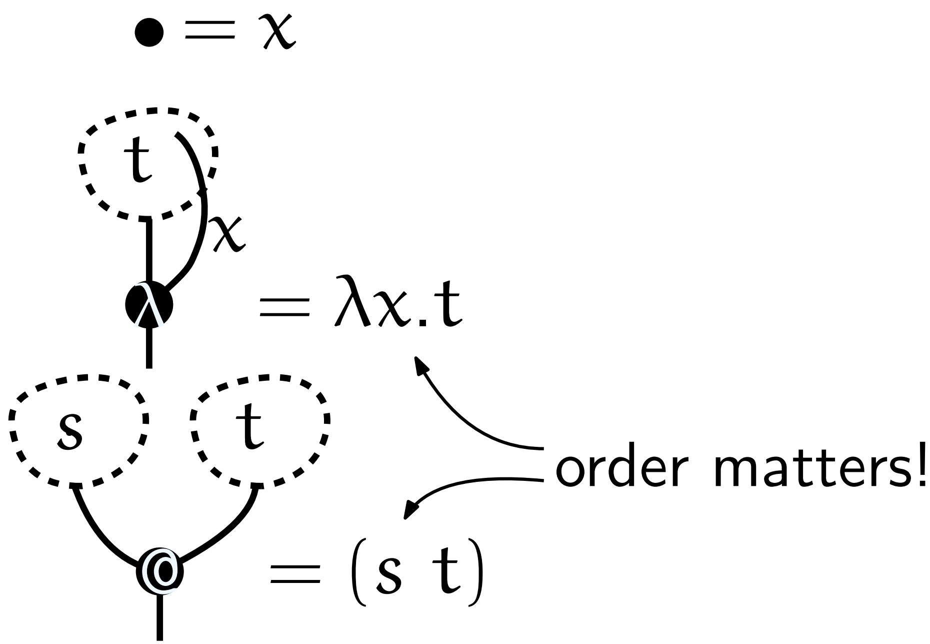
$$(\lambda y.\lambda z.(y\ \lambda w.w)z))(\lambda u.a\ u)$$

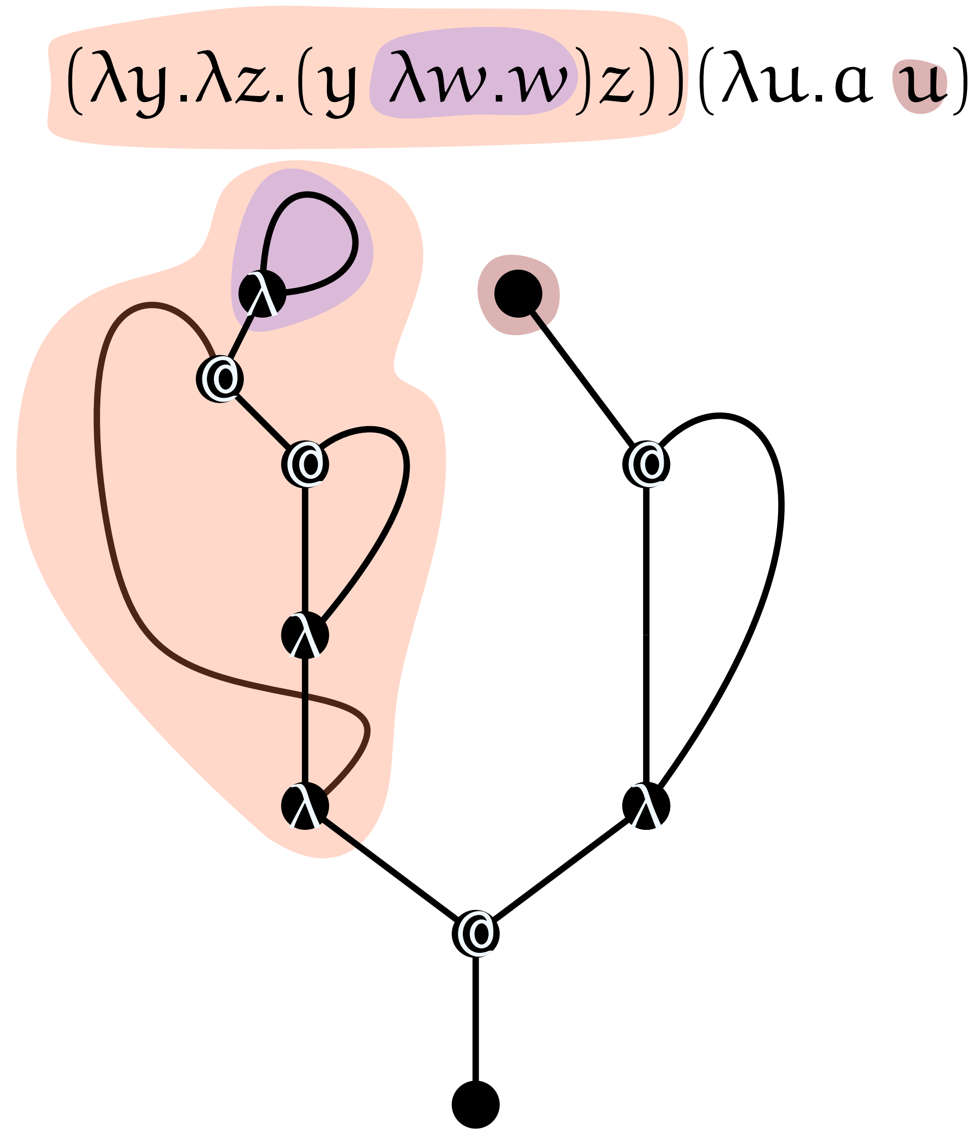


Dictionary

- Free var ↔ unary vertex
- Identity-subterm ↔ loop

Why should logicians be interested in maps?

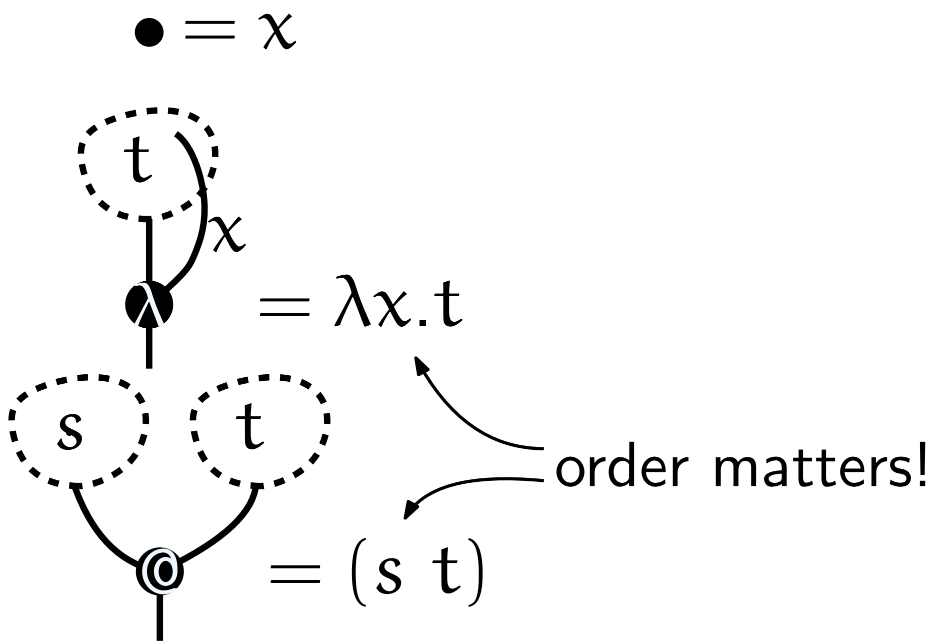


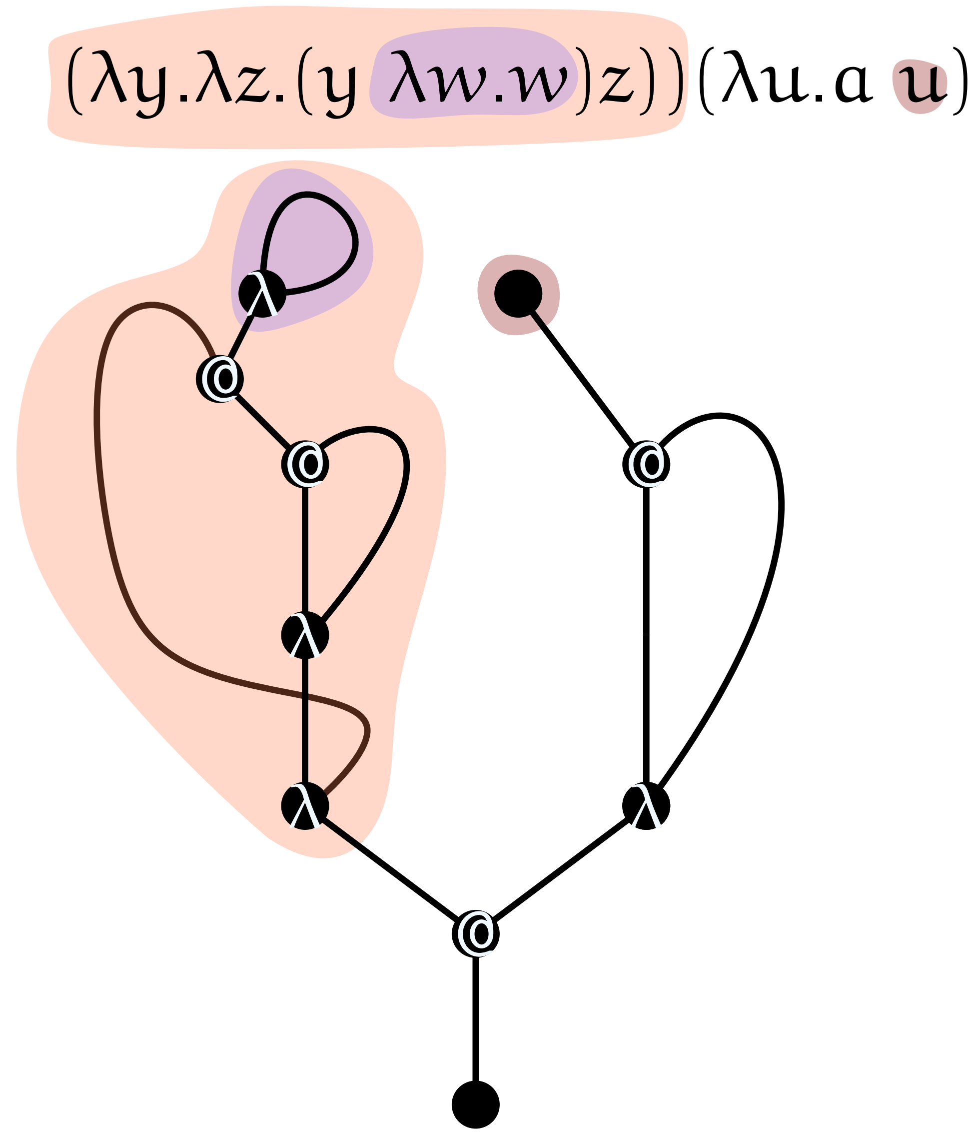


Dictionary

- Free var ↔ unary vertex
- Identity-subterm ↔ loop
- Closed subterm ↔ bridge

Why should logicians be interested in maps?

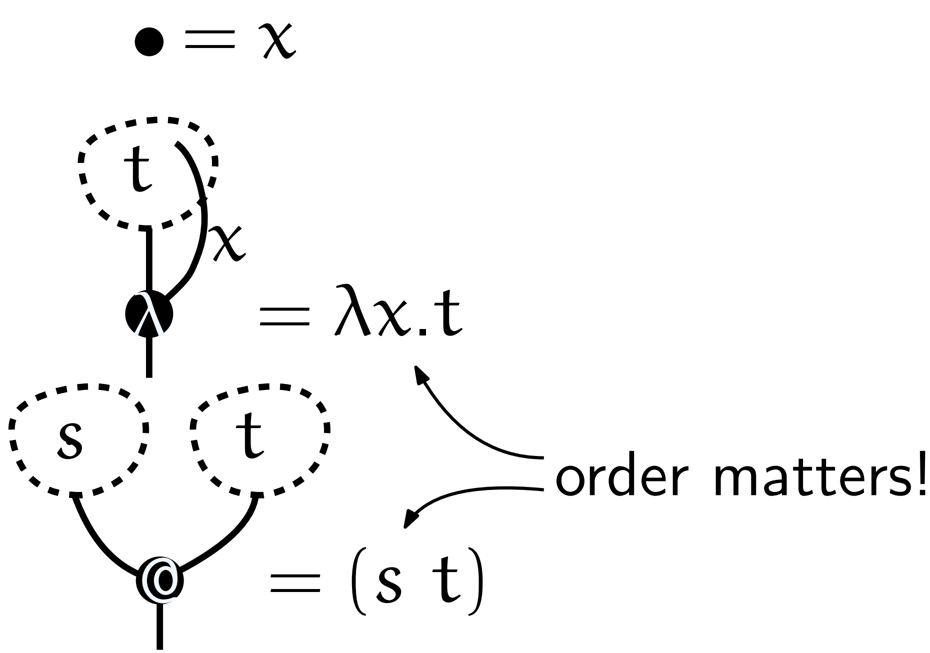


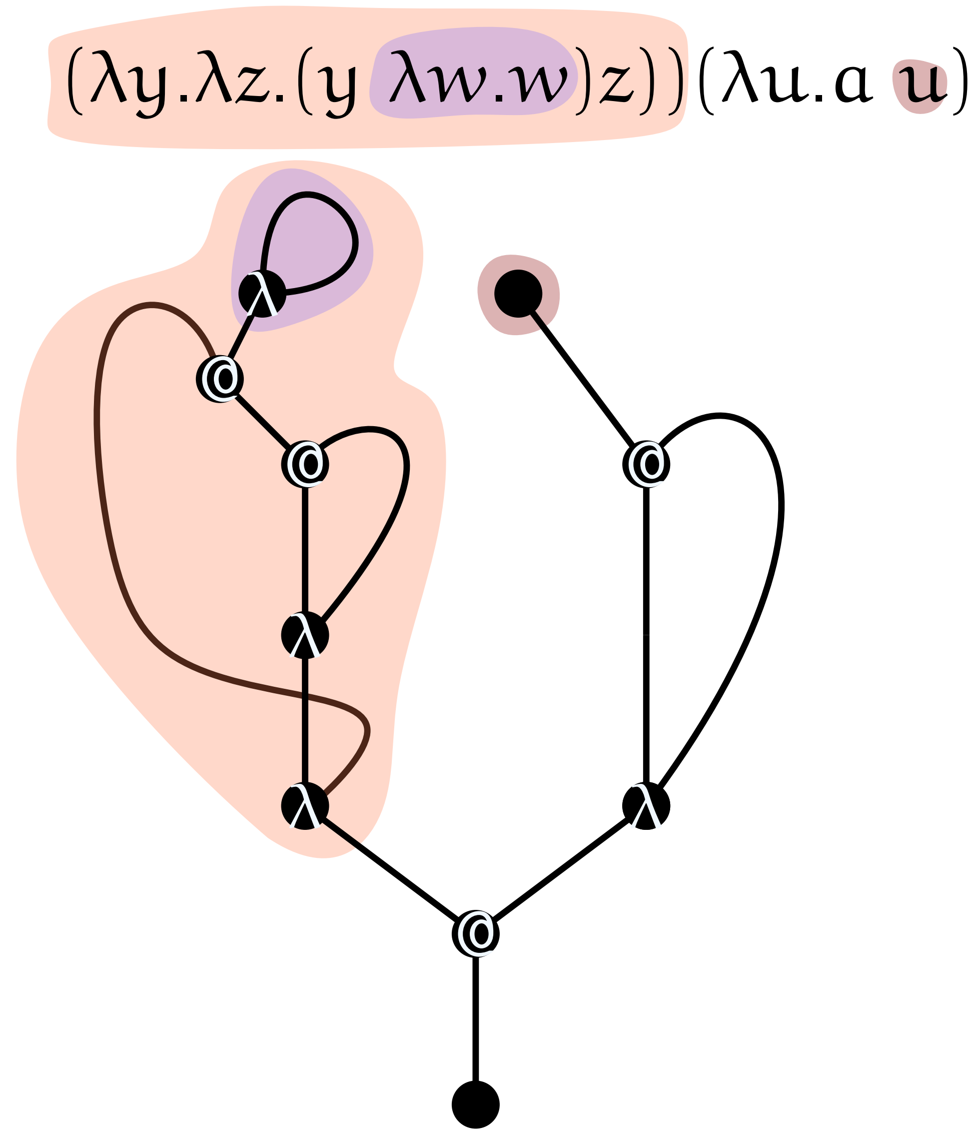


Dictionary

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

Why should logicians be interested in maps?



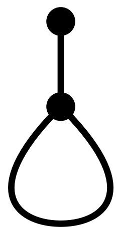


Dictionary

- Free var ↔ unary 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, Z16]

Why should combinatorialists be interested in λ-terms?
Decomposing rooted cubic maps

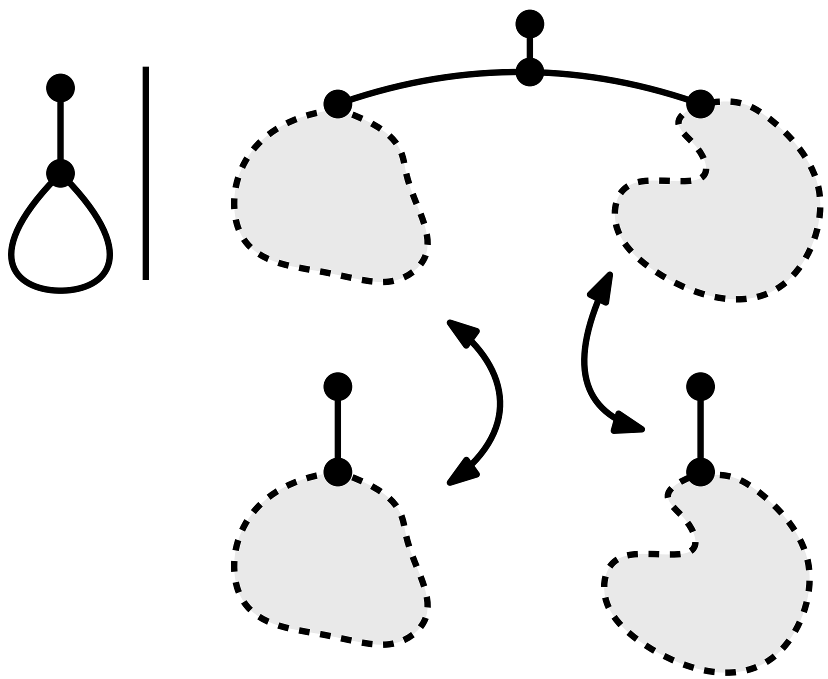


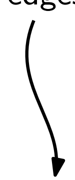
edges

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

Why should combinatorialists be interested in λ-terms?

Decomposing rooted cubic maps

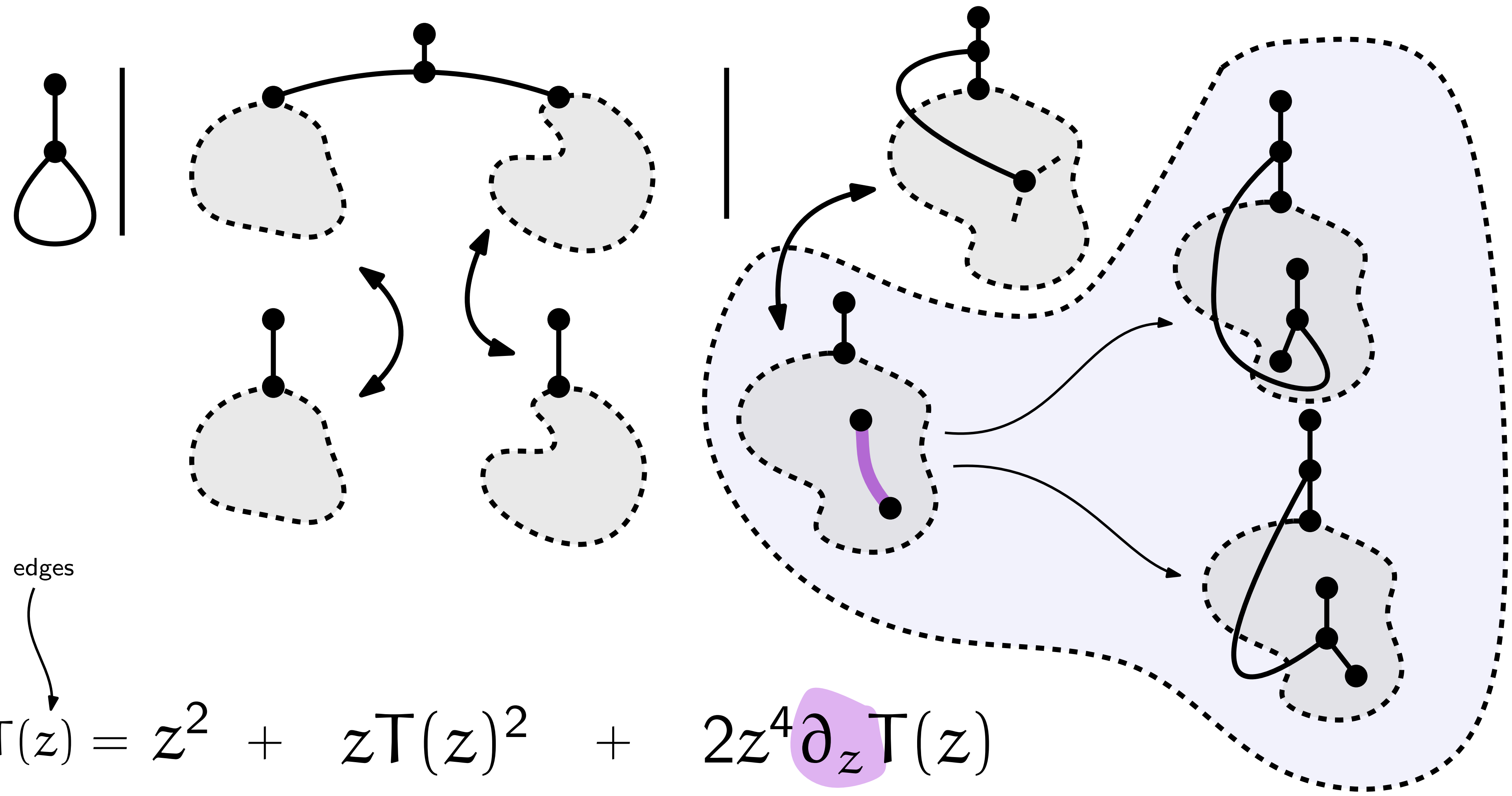




$$\mathsf{T}(z) = z^2 \ + \ z\mathsf{T}(z)^2$$

Why should combinatorialists be interested in λ-terms?

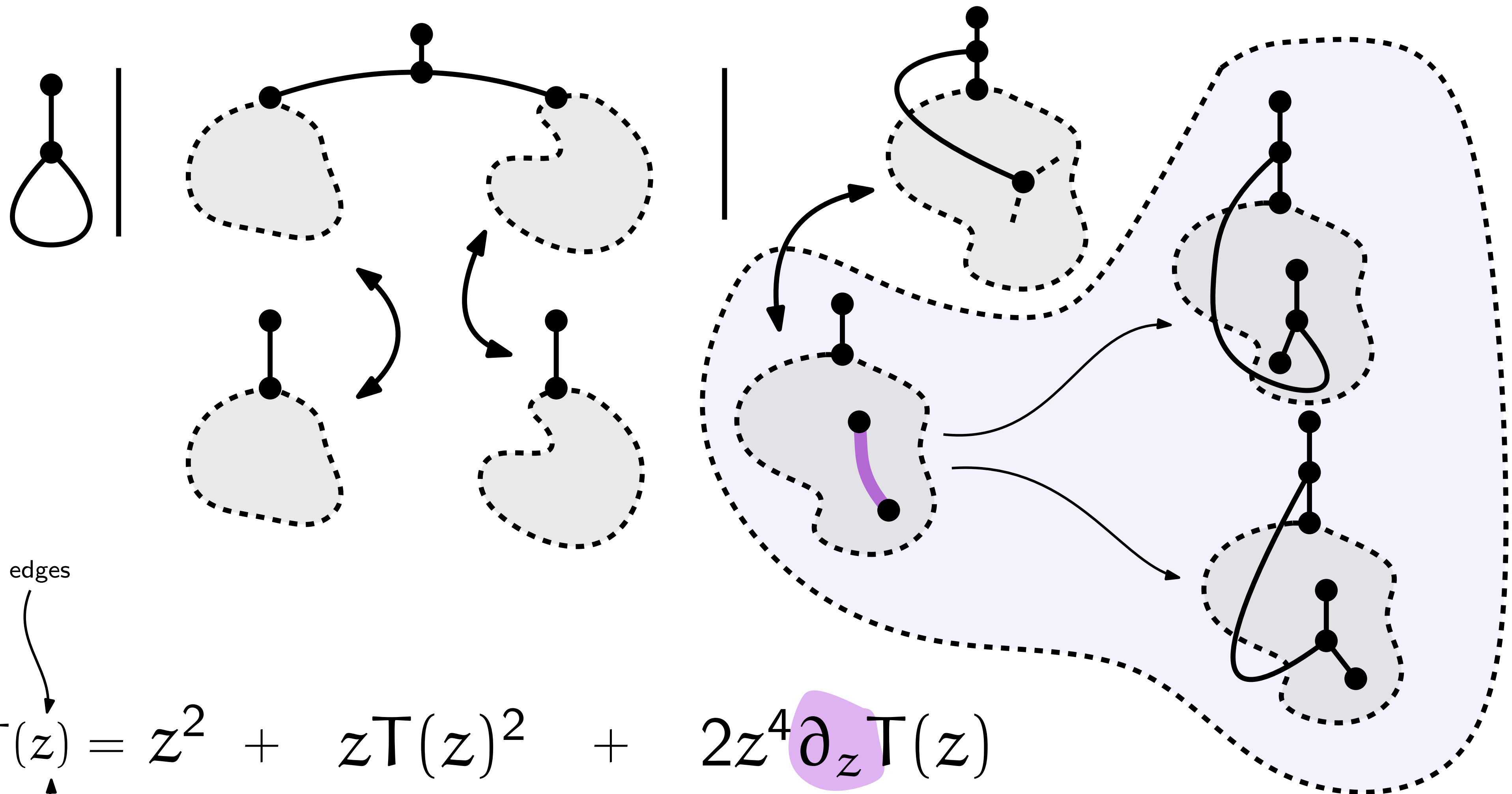
Decomposing rooted cubic maps



$$T(z) = z^2 + zT(z)^2 + 2z^4 \partial_z T(z)$$

Why should combinatorialists be interested in λ-terms?

Decomposing rooted cubic maps and closed linear terms!

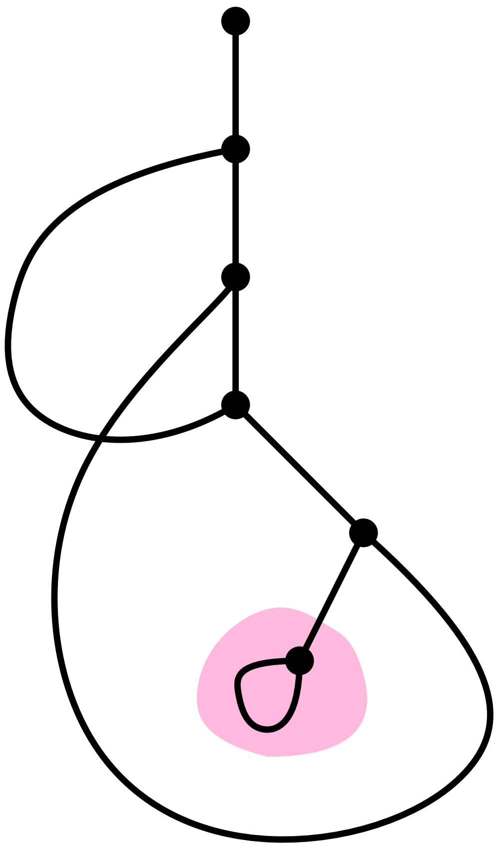


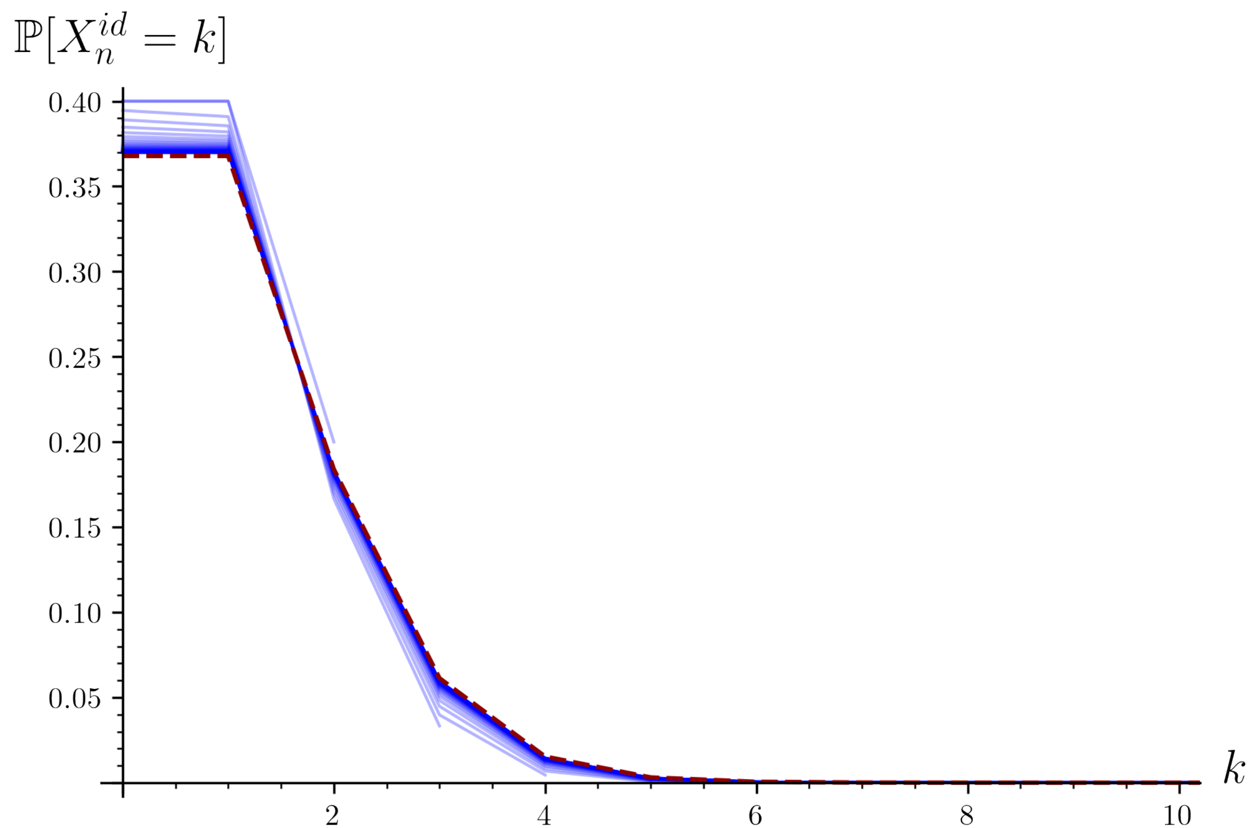
$$T(z) = z^2 + zT(z)^2 + 2z^4 \partial_z T(z)$$

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

Some of our previous results: limit distributions

loops = # "$\lambda x.x$"



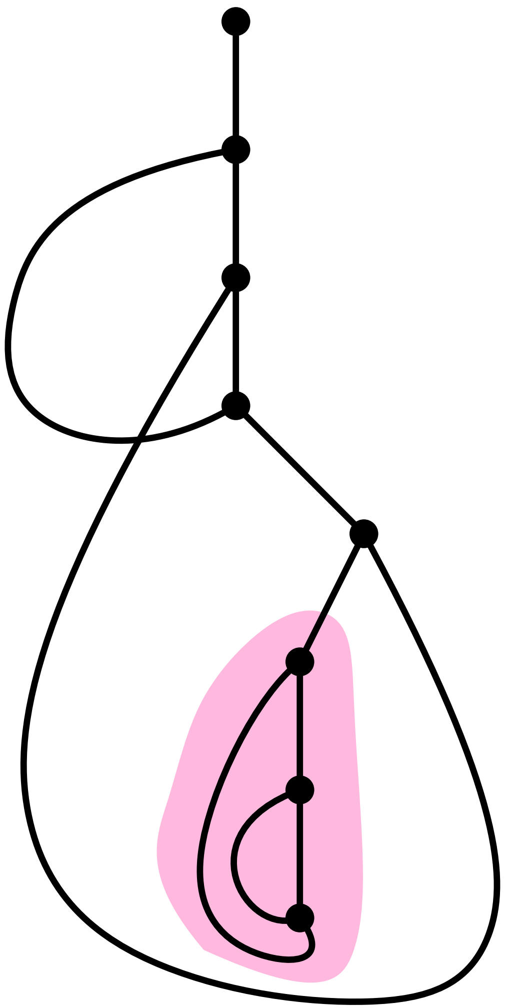


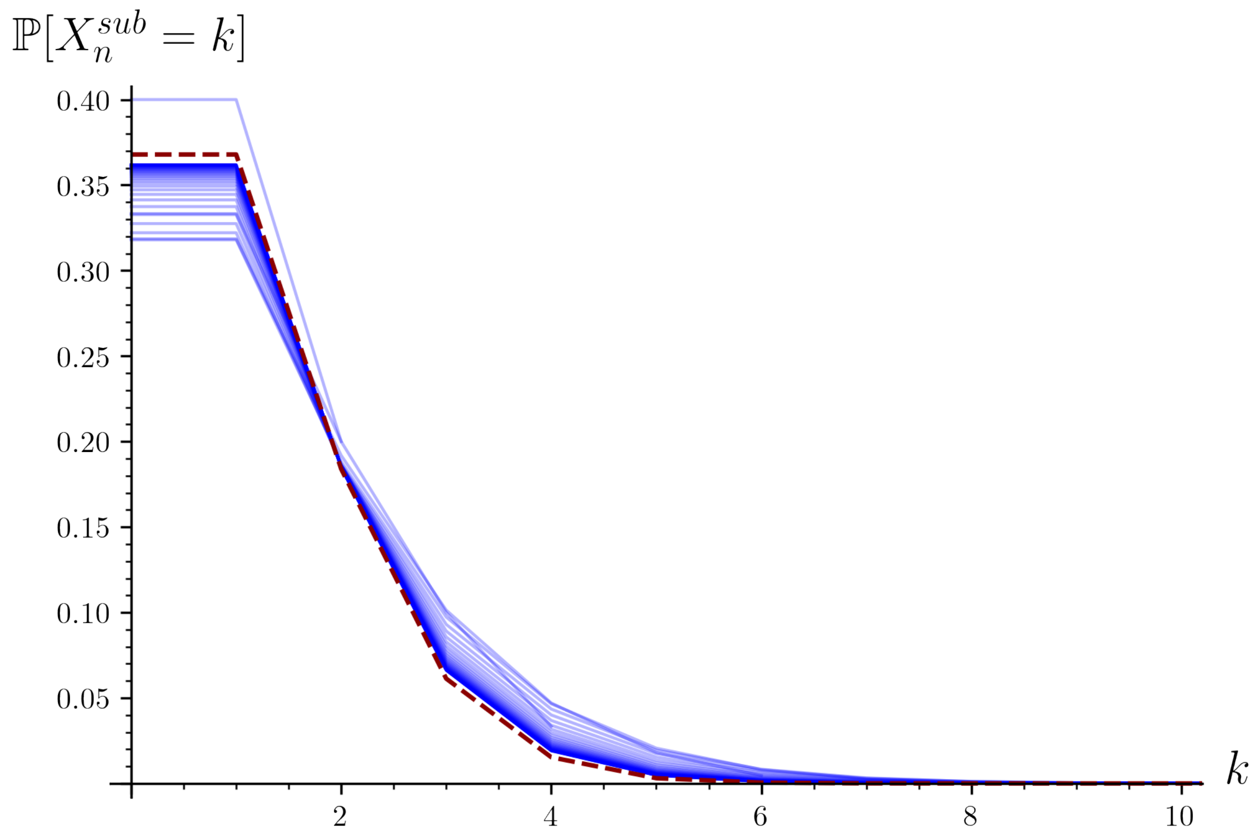
$$\lambda x.\lambda y.(y\ \lambda w.w)x$$

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

Some of our previous results: limit distributions

bridges = # closed subterms



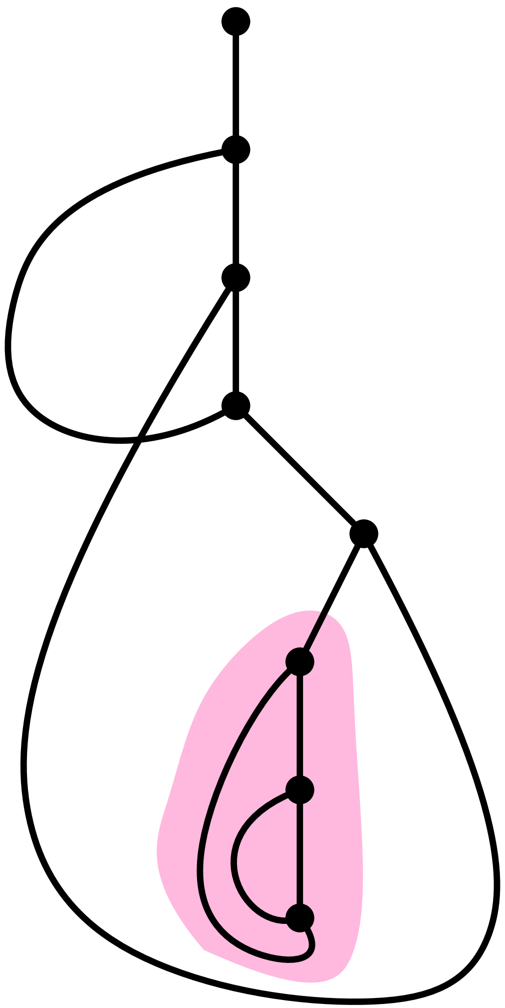


$\lambda x.\lambda y.(y\ \lambda z.\lambda w.zw)x$

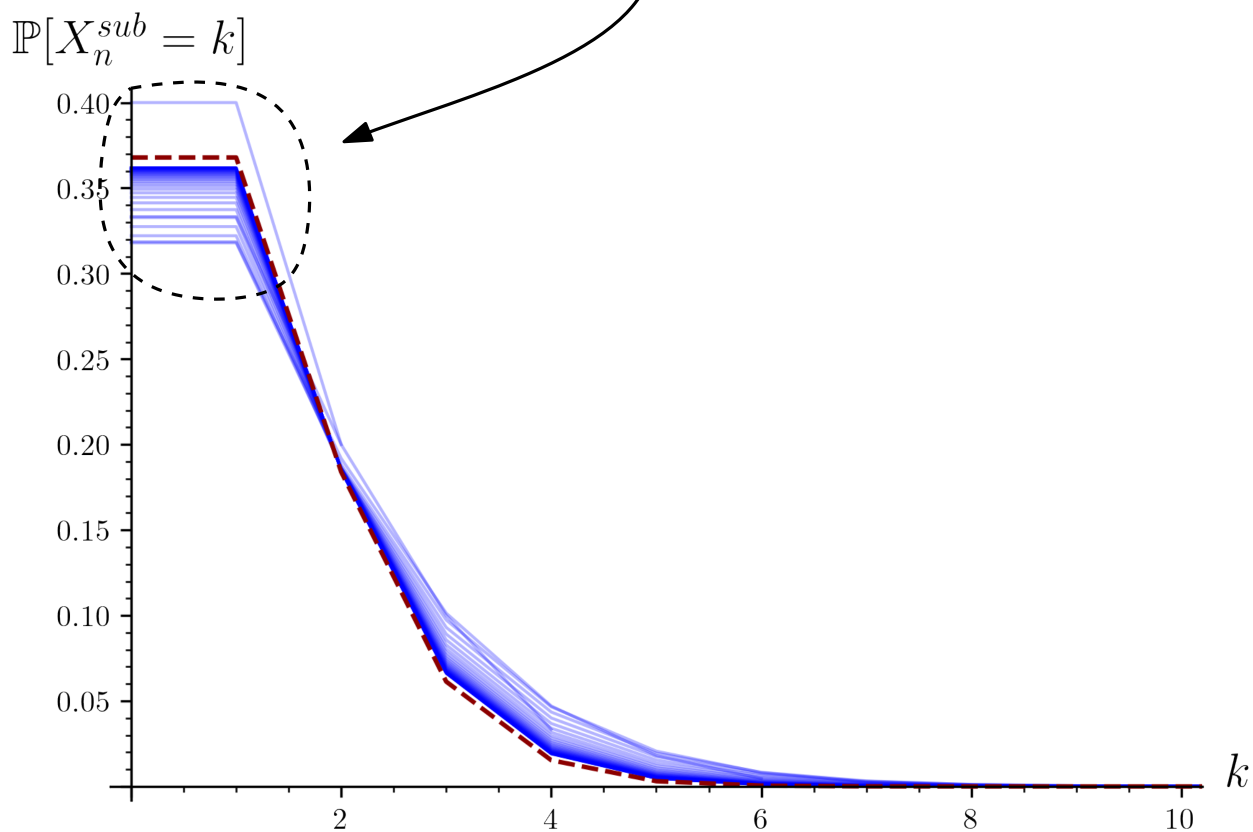
$X_n^{sub} \xrightarrow{D} \mathrm{Poisson}(1)$

Some of our previous results: limit distributions

\# bridges = # closed subterms



maps with one bridge ↔ bridgeless maps



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

Our strategy:

1) Track evolution of parameter in decompositions of cubic maps/λ-terms

Our strategy:

1) Track evolution of parameter in decompositions of cubic maps/λ-terms

different decompositions $\rightsquigarrow$ differential equations, Hadamard products, ...

Our strategy:

1) Track evolution of parameter in decompositions of cubic maps/λ-terms

different decompositions ⇝ differential equations, Hadamard products, . . .

divergent generating functions

Our strategy:

1) Track evolution of parameter in decompositions of cubic maps/λ-terms

different decompositions $\rightsquigarrow$ differential equations, Hadamard products, . . .

divergent generating functions

2) Extend tools for rapidly growing coefficients:

Our strategy:

1) Track evolution of parameter in decompositions of cubic maps/λ-terms

different decompositions $\rightsquigarrow$ differential equations, Hadamard products, ...

divergent generating functions

2) Extend tools for rapidly growing coefficients:

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

$$[z^n](A(z) \cdot B(z)) \sim a_n b_0 + a_0 b_n + O(a_{n-1} + b_{n-1})$$

for A, B, G divergent

Our strategy:

1) Track evolution of parameter in decompositions of cubic maps/λ-terms

different decompositions ⇝ differential equations, Hadamard products, ...

divergent generating functions

2) Extend tools for rapidly growing coefficients:

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

$$[z^n](A(z) \cdot B(z)) \sim a_n b_0 + a_0 b_n + O(a_{n-1} + b_{n-1})$$

for A, B, G divergent

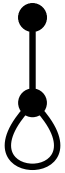
Mean number of β-redices in closed terms

- Tracking redices during the decomposition

Mean number of β-redices in closed terms

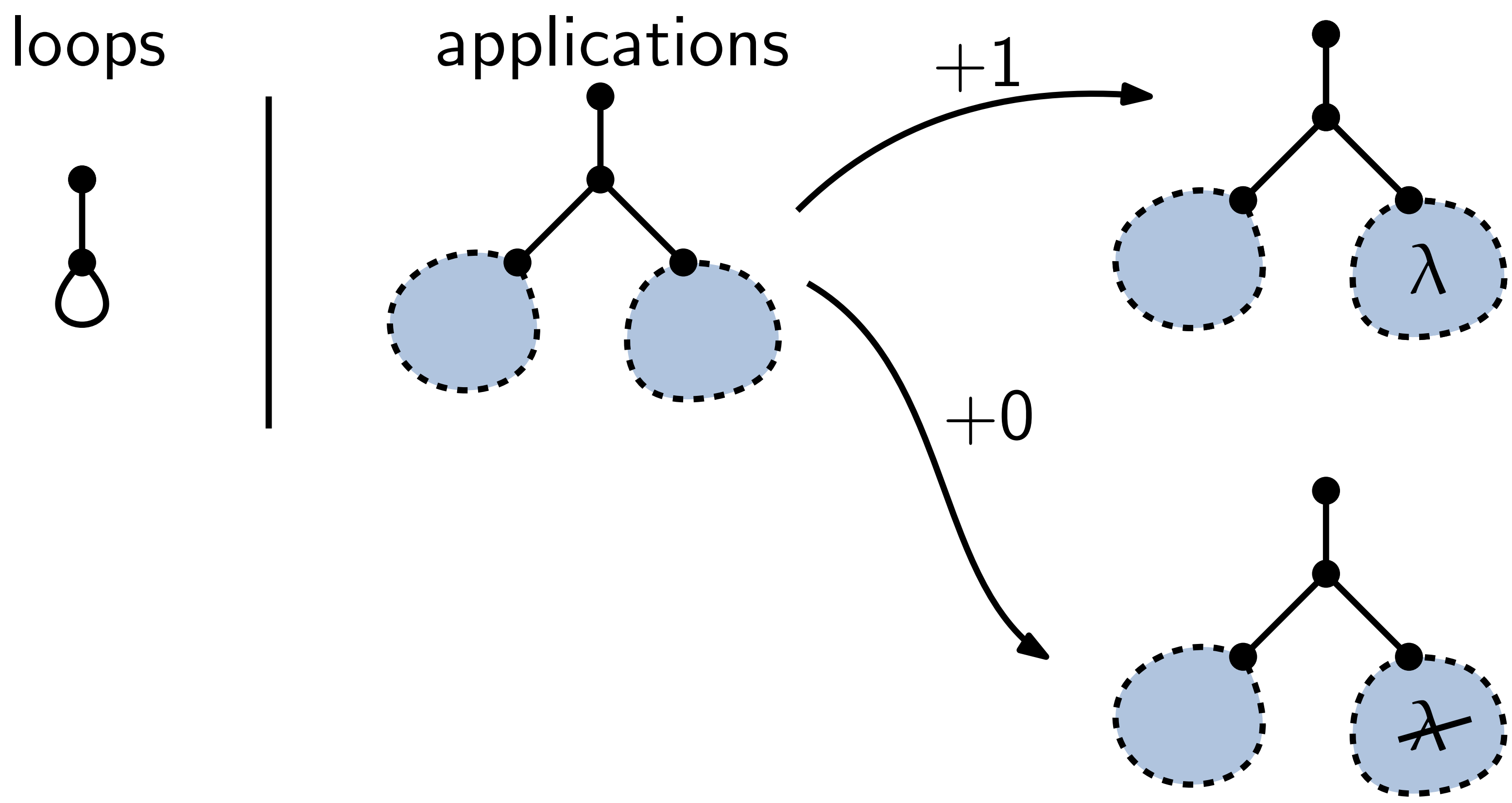
- Tracking redices during the decomposition

loops



Mean number of β-redices in closed terms

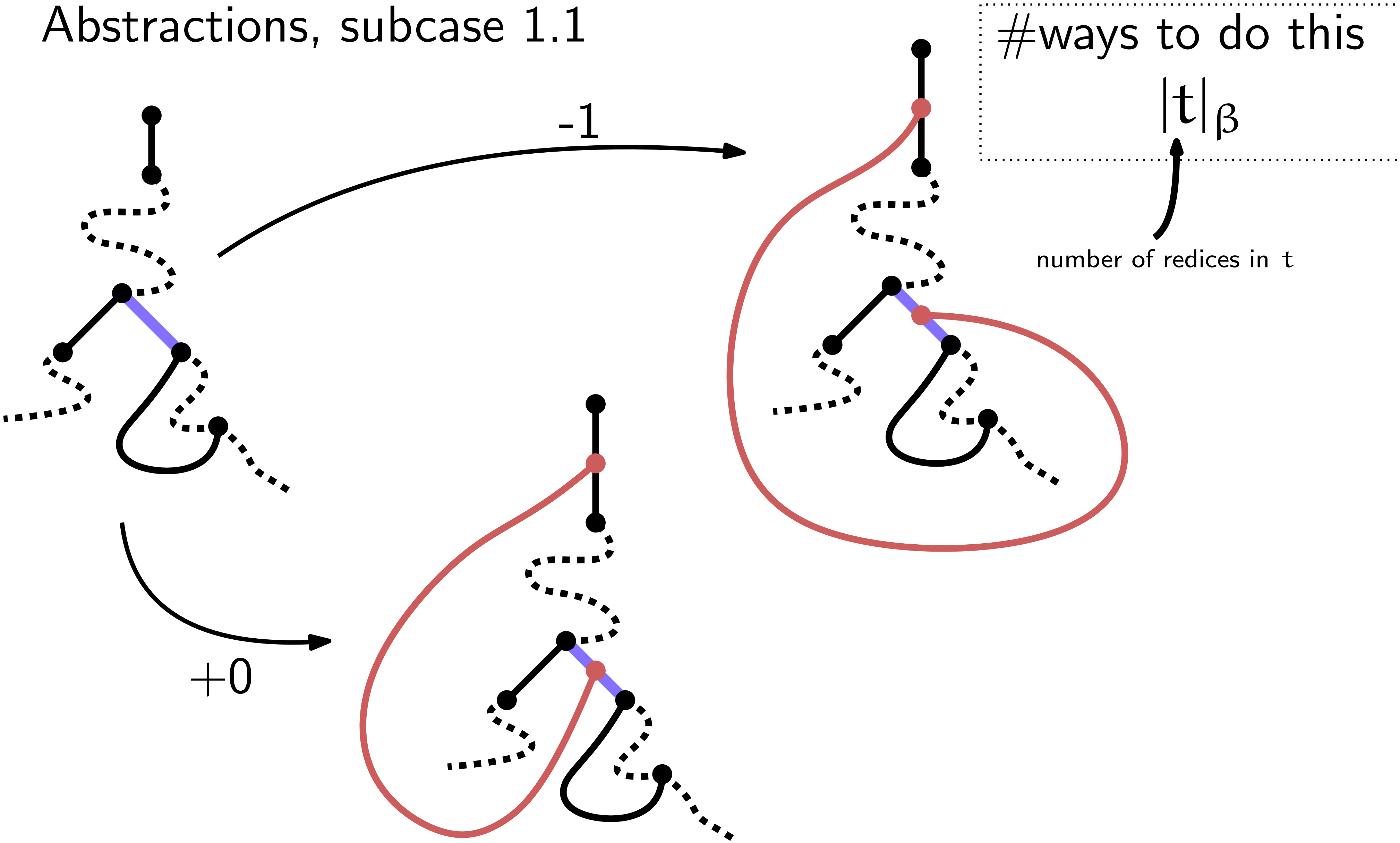
- Tracking redices during the decomposition



Mean number of β-redices in closed terms

- Tracking redices during the decomposition

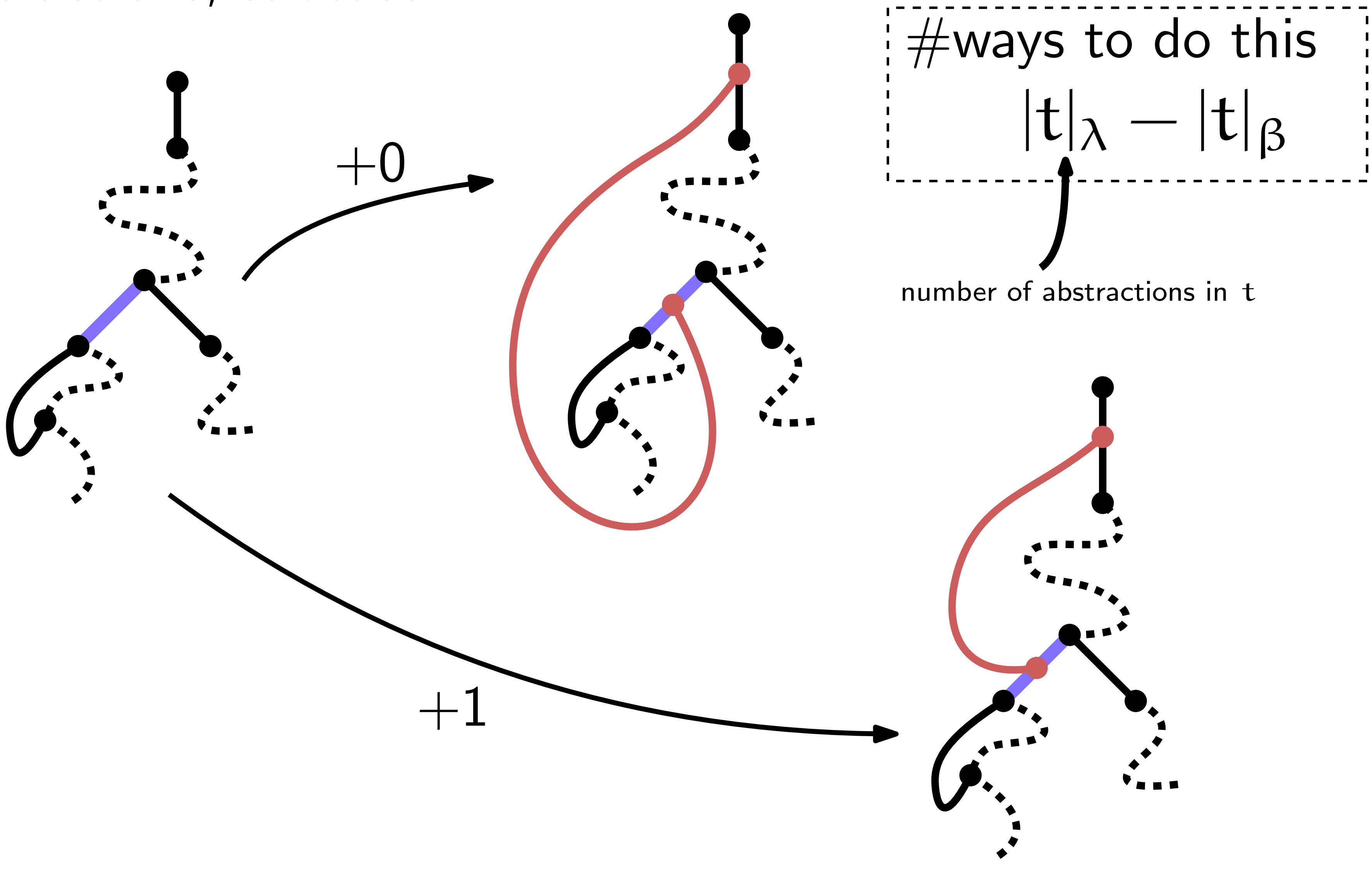
Abstractions, subcase 1.1



Mean number of β-redices in closed terms

- Tracking redices during the decomposition

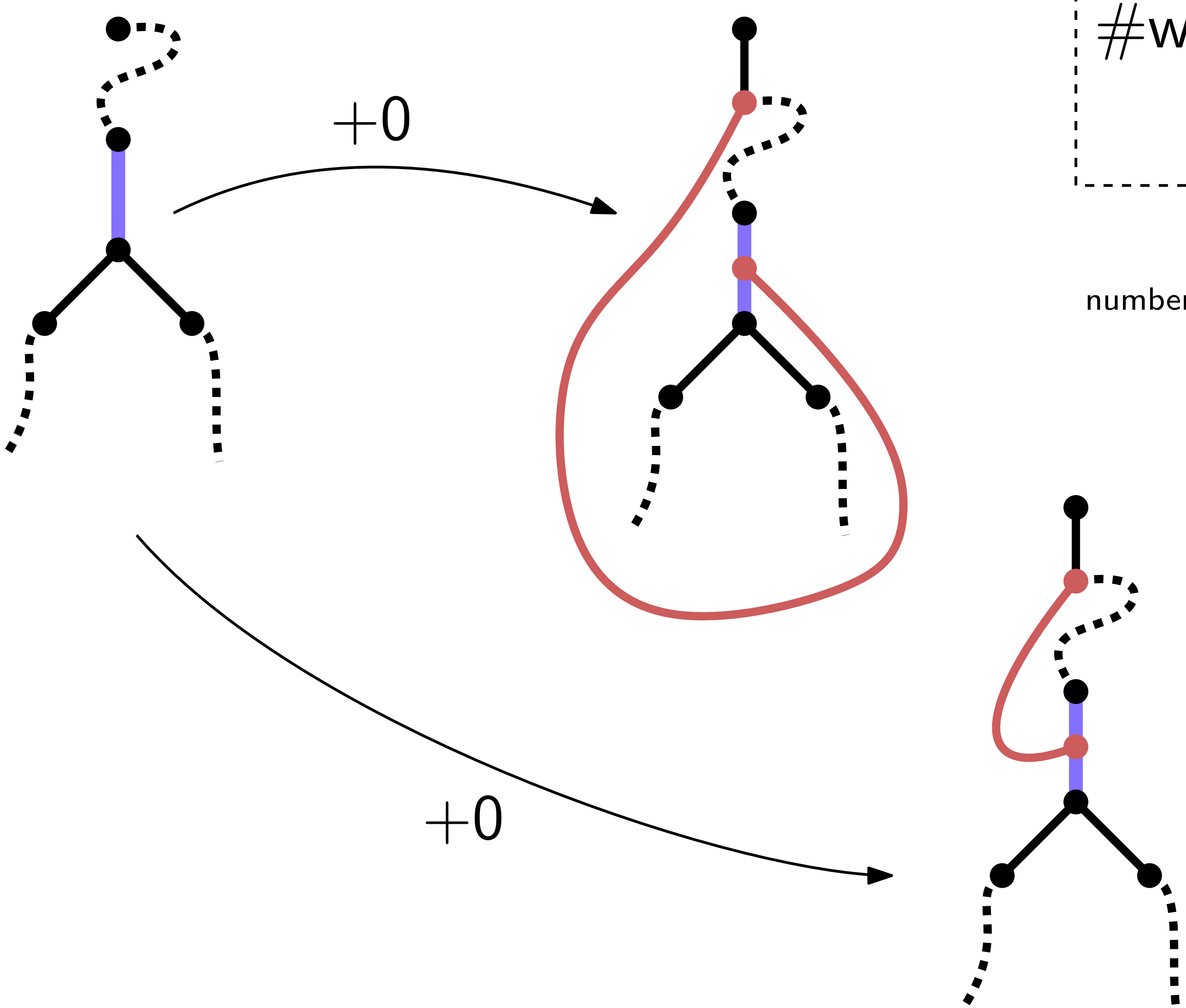
Abstractions, subcase 1.2



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

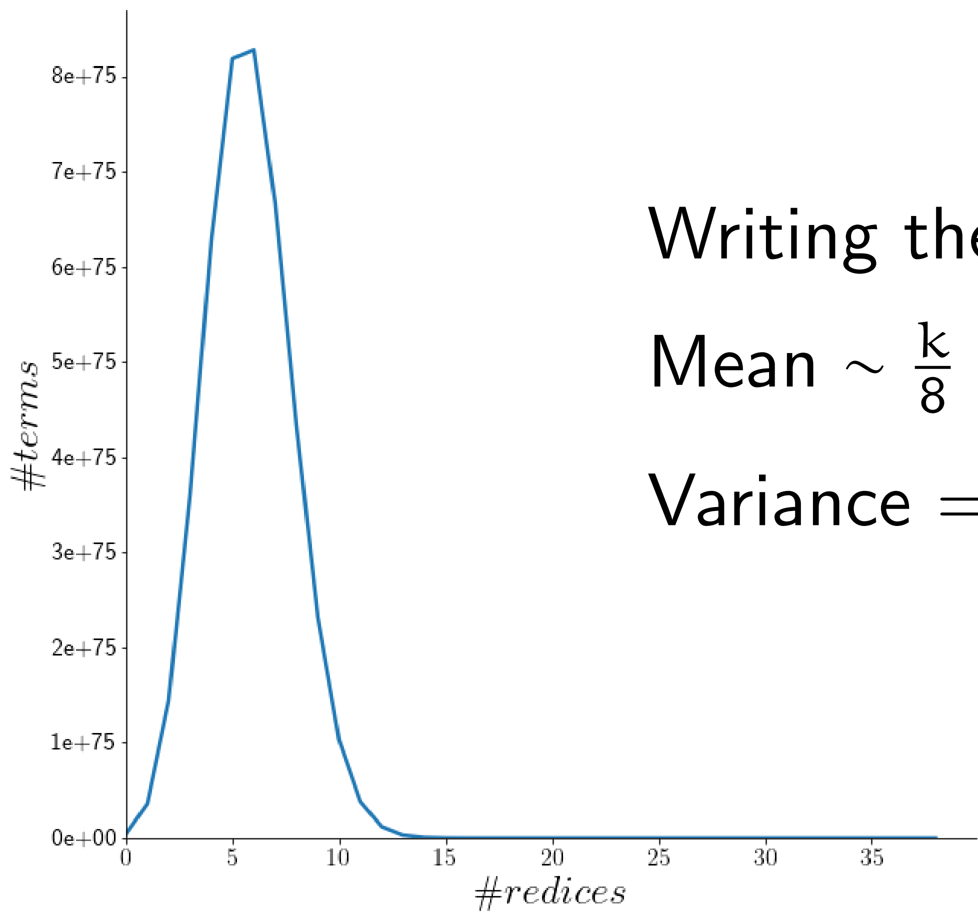
$$\mathsf{T} = -z\left(z^2(\mathrm{r}+1)(1+(\mathrm{r}-1)z\mathsf{T})(\mathrm{r}-1)\partial_{\mathrm{r}}\mathsf{T} - \frac{(1+z(\mathrm{r}-1)\mathsf{T})z^3(\mathrm{r}+5)\partial_z\mathsf{T}}{3} - \frac{z^3(\mathrm{r}-1)^2\mathsf{T}^2}{3} - \frac{4z^2(\mathrm{r}-1)\mathsf{T}}{3} - z - \mathsf{T}^2\right)$$

Mean number of β-redices in closed terms

- Translating to a differential equation and pumping

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

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 $= O(k)$

A better lower bound

A better lower bound

- Consider the following three patterns of redices

$$(\lambda x.C[(x\ u)])(\lambda y.t_2) \quad (p_1) \qquad ((\lambda x.\lambda y.t_1)t_2)t_3 \quad (p_2)$$

$$(\lambda x.x)(\lambda y.t_1)t_2 \quad (p_3)$$

A better lower bound

- Consider the following three patterns of redices

$$(\lambda x.C[(x\ u)])(\lambda y.t_2) \quad (p_1) \qquad ((\lambda x.\lambda y.t_1)t_2)t_3 \quad (p_2)$$

$$(\lambda x.x)(\lambda y.t_1)t_2 \quad (p_3)$$

- These are the only patterns whose reduction leaves the number of redices invariant.

A better lower bound

- Consider the following three patterns of redices

$$(\lambda x.C[(x\ u)])(\lambda y.t_2) \quad (p_1) \qquad ((\lambda x.\lambda y.t_1)t_2)t_3 \quad (p_2)$$

$$(\lambda x.x)(\lambda y.t_1)t_2 \quad (p_3)$$

- These are the only patterns whose reduction leaves the number of redices invariant.

- Gives a lower bound on the number of steps to reach normal form:

$$\#steps \geqslant |t|_\beta + |t|_{p1} + |t|_{p_2} + |t|_{p_3}$$

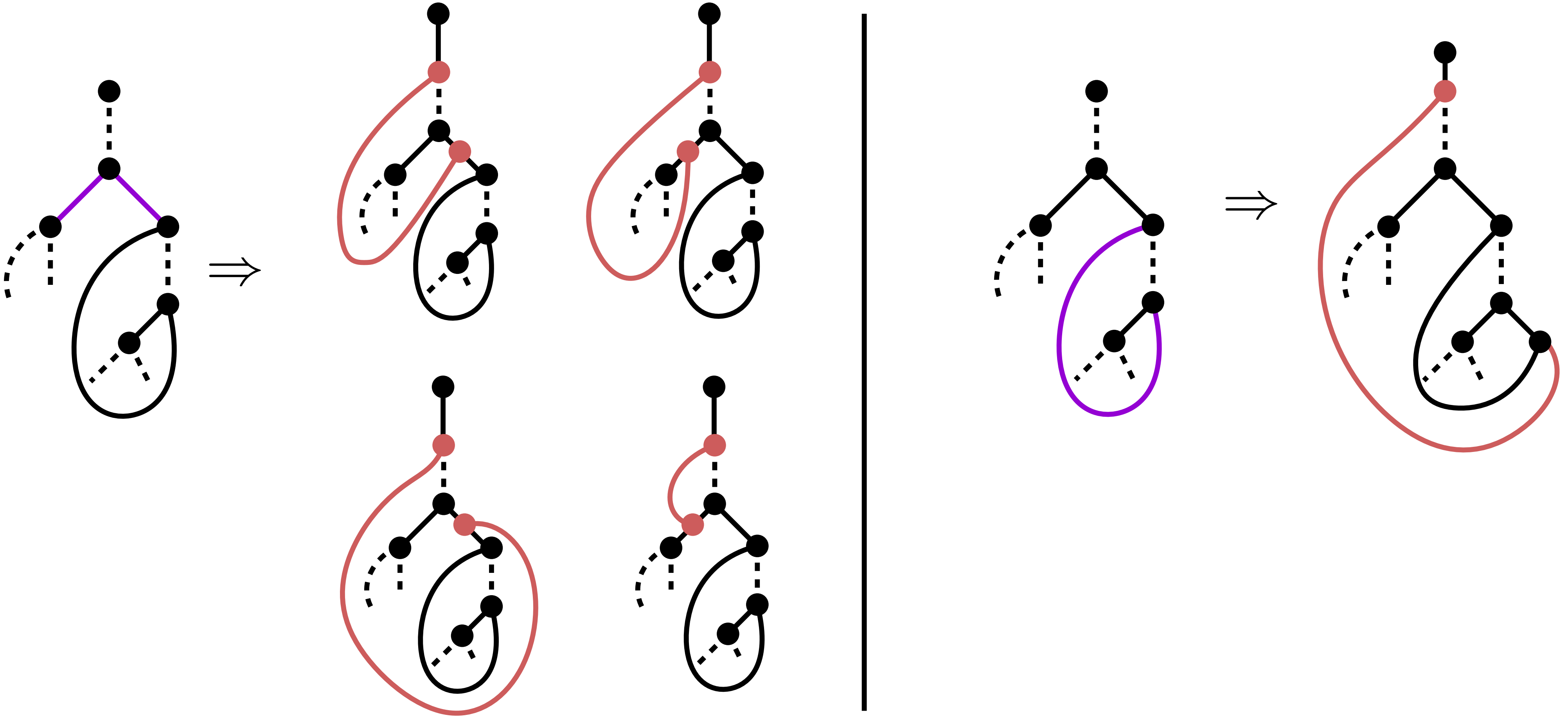
Enumerating $\mathfrak{p}_1$-patterns

- Tracking the creation/destruction of patterns during the recursive decomposition:

Enumerating p_1-patterns

- Tracking the creation/destruction of patterns during the recursive decomposition:

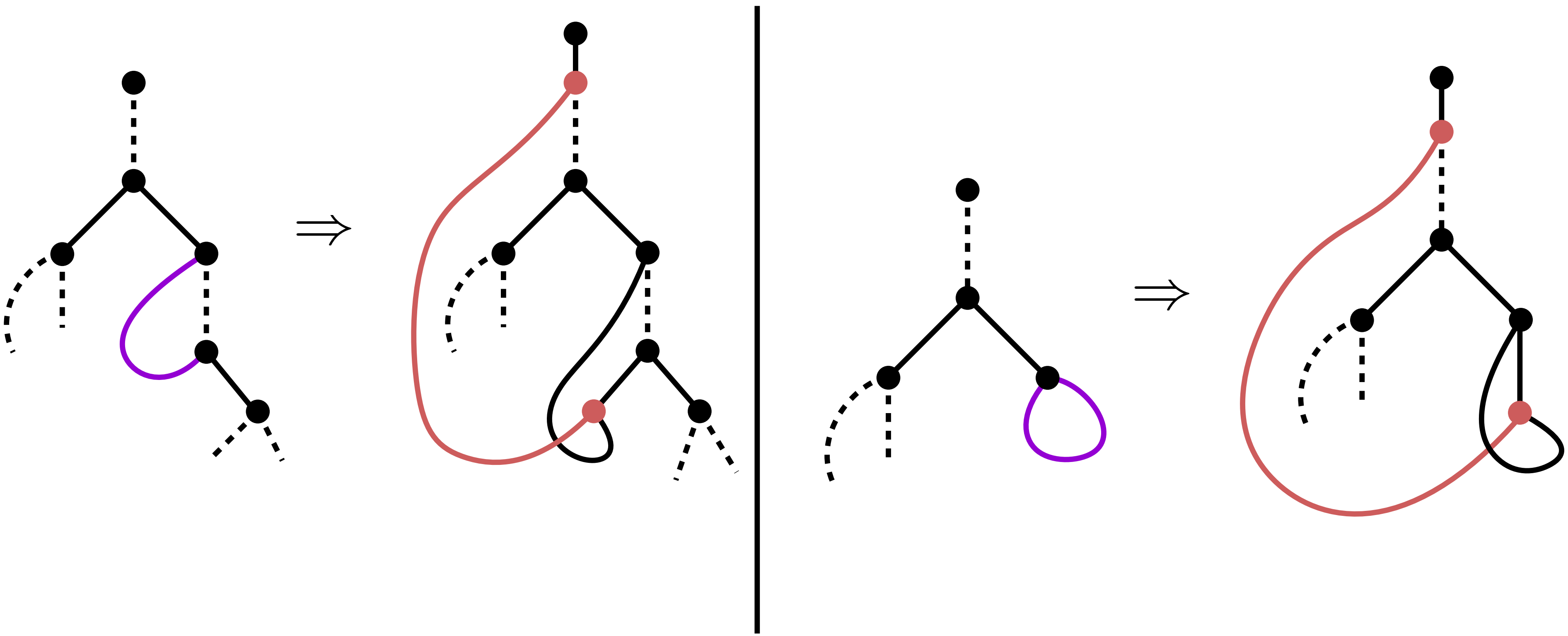
  Cuts destroying a p_1-pattern:



Enumerating p_1-patterns

- Tracking the creation/destruction of patterns during the recursive decomposition:

  Cuts creating a p_1-pattern:



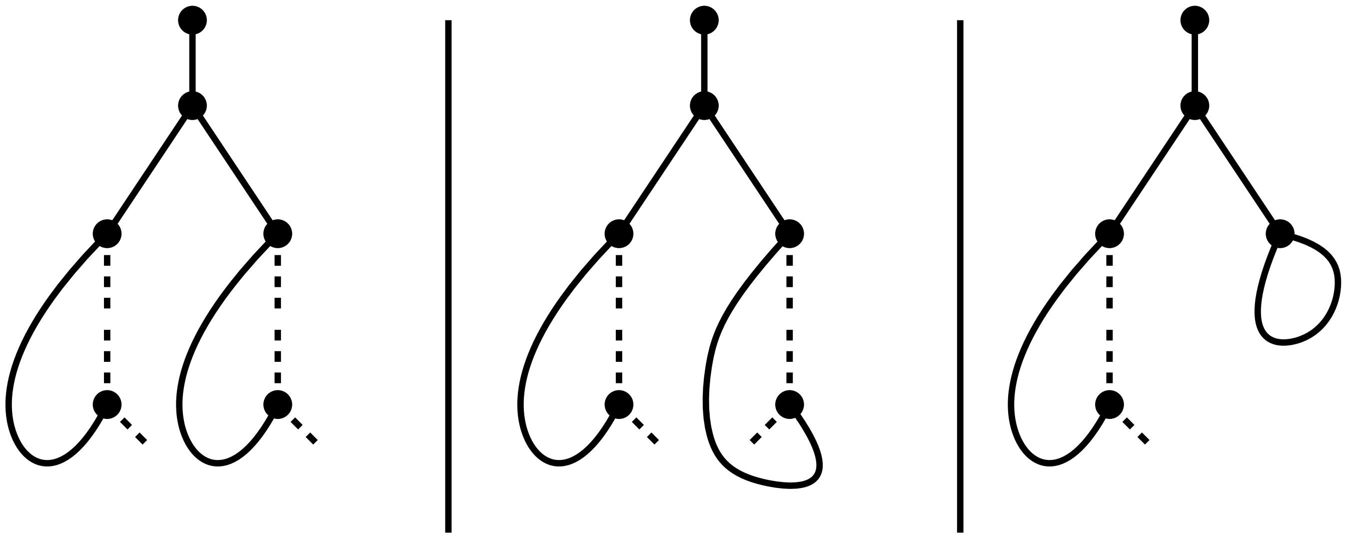
Thus we also need to keep track of:

$$C_1[\lambda x.C_2[(t_1\ x)]](\lambda y.t_2)] \quad C_1[(\lambda x.x)(\lambda y.t_2)]$$

Enumerating p_1-patterns

- Tracking the creation/destruction of patterns during the recursive decomposition:

  Applications creating p_1 and auxilliary patterns:



Thus, for an app. of the form $(l_1\ \lambda y.t_1)$ we need to consider how l_1 was formed.

Enumerating p_1-patterns

- Thus we have the following equations:

$$S = \Lambda + A$$

$$\Lambda = z^2 + 2z^4 S_z + (v - u + 4(1-u))z^3 S_u + (u - v + 4(1-v))z^3 S_v$$

$$A = zS^2 + (u-1)z(z^4 S_z + (v - u + 2(1-u))z^3 S_u + 2(1-v)z^3 S_v) \cdot \Lambda$$
$$+(v-1)z(z^2 + z^4 S_z + (u - v + 2(1-v))z^3 S_u + 2(1-u)z^3 S_u) \cdot \Lambda$$

Enumerating p_1-patterns

- Thus we have the following equations:

$$S = \Lambda + A$$

$$\Lambda = z^2 + 2z^4 S_z + (v - u + 4(1-u))z^3 S_u + (u - v + 4(1-v))z^3 S_v$$

$$A = zS^2 + (u-1)z(z^4 S_z + (v - u + 2(1-u))z^3 S_u + 2(1-v)z^3 S_v) \cdot \Lambda$$
$$+(v-1)z(z^2 + z^4 S_z + (u - v + 2(1-v))z^3 S_u + 2(1-u)z^3 S_u) \cdot \Lambda$$

- Extracting the mean:

$$\partial_u S|_{u=1,v=1}$$
$$= \left(2zS\partial_u S + 2z^4 \partial_{z,u} S + z^7 \partial_z S + 2z^9 (\partial_z S)^2 - 5z^3 \partial_u S + z^3 \partial_v S\right)|_{u=1,v=1}$$

Enumerating p_1-patterns

- Thus we have the following equations:

$$S = \Lambda + A$$

$$\Lambda = z^2 + 2z^4 S_z + (v - u + 4(1-u))z^3 S_u + (u - v + 4(1-v))z^3 S_v$$

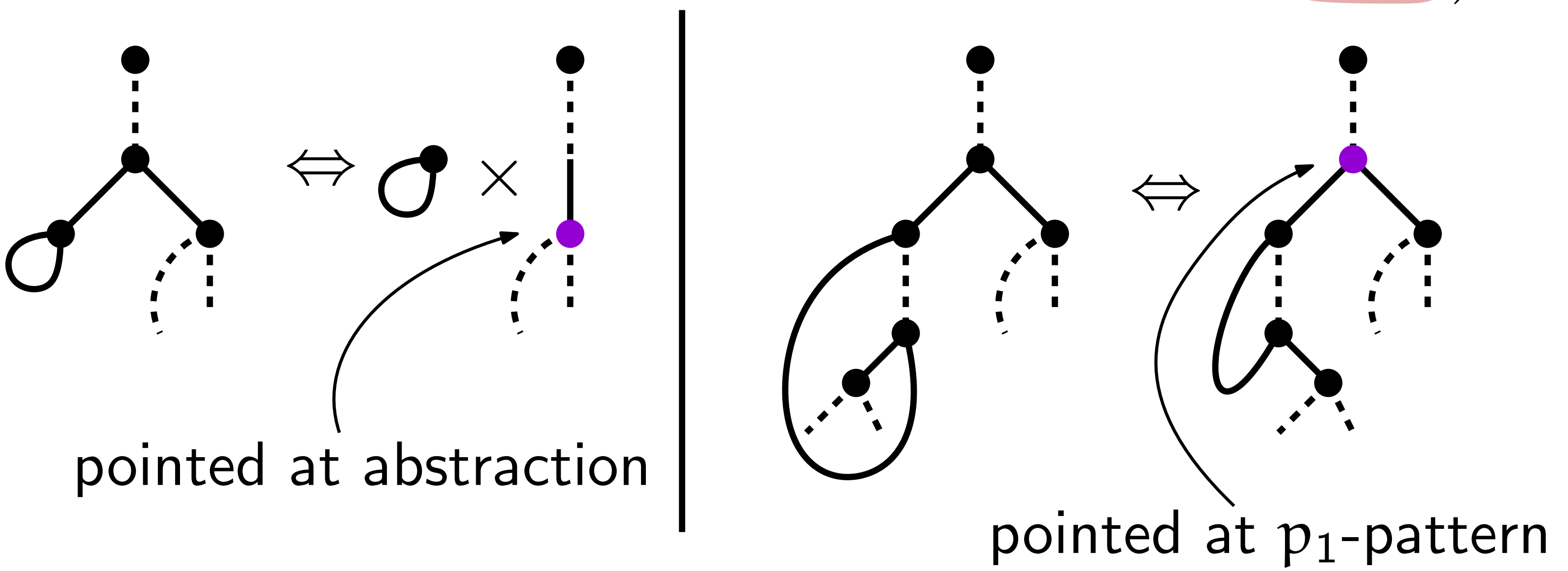
$$A = zS^2 + (u-1)z(z^4 S_z + (v - u + 2(1-u))z^3 S_u + 2(1-v)z^3 S_v) \cdot \Lambda$$
$$+(v-1)z(z^2 + z^4 S_z + (u - v + 2(1-v))z^3 S_u + 2(1-u)z^3 S_u) \cdot \Lambda$$

- Extracting the mean:

$$\partial_u S|_{u=1,v=1}$$

bijection: $\partial_v \leftrightarrow \partial_u$

$$= \left(2zS\partial_u S + 2z^4 \partial_{z,u} S + z^7 \partial_z S + 2z^9 (\partial_z S)^2 - 5z^3 \partial_u S + z^3 \partial_v S\right)|_{u=1,v=1}$$



Enumerating $\mathfrak{p}_1$-patterns

- Finally we obtain a mean number of occurences:

$$\mathbb{E}[\# \ \mathfrak{p}_1 \text{ patterns}] \sim \frac{1}{6}$$

Enumerating p_1-patterns, p_2-patterns, and p_3-patterns

- Finally we obtain a mean number of occurences:

$$\mathbb{E}[\#\ p_1 \text{ patterns}] \sim \frac{1}{6}$$

- Analogously, we have a mean number of occurences for p_2:

$$\mathbb{E}[\#\ p_2 \text{ patterns}] \sim \frac{1}{48}$$

Both are asymptotically constant in expectation!

- Via different methods, we obtain:

$$\mathbb{E}[\#\ p_3 \text{ patterns}] \geqslant \frac{n}{240}$$

Asymptotically linear in n!

Conclusion

- Expected #steps required to reduce a random term to its normal form?

Conclusion

- Expected #steps required to reduce a random term to its normal form?

- Combined techniques from combinatorics and logic to count patterns in decorated cubic maps and linear λ-terms

Conclusion

- Expected #steps required to reduce a random term to its normal form?
- Combined techniques from combinatorics and logic to count patterns in decorated cubic maps and linear λ-terms
- Lower bound obtained, for terms of size n:

$$\mathbb{E}[\#\text{steps to reach normal form}] \geqslant \frac{11n}{240}$$

which is quite close to Noam Zeilberger's conjecture of $\mathbb{E}[\#steps] = \frac{n}{21}$!

Conclusion

- Expected #steps required to reduce a random term to its normal form?

- Combined techniques from combinatorics and logic to count patterns in decorated cubic maps and linear λ-terms

- Lower bound obtained, for terms of size n:

$$\mathbb{E}[\#\text{steps to reach normal form}] \geqslant \frac{11n}{240}$$

which is quite close to Noam Zeilberger's conjecture of $\mathbb{E}[\#steps] = \frac{n}{21}$!

Thank you!

Bonus slides!

The λ-calculus

- A **universal** system of computation

The λ-calculus

- A **universal** system of computation
- Its terms are formed inductively

$$\frac{}{x \vdash x}\ \text{var}$$

The λ-calculus

- A **universal** system of computation
- Its terms are formed inductively

$$\frac{}{x \vdash x}\ \text{var}$$

free

The λ-calculus

- A **universal** system of computation
- Its terms are formed inductively

$$\dfrac{}{x \vdash x}\,\text{var} \qquad \dfrac{\Gamma, x \vdash t}{\Gamma \vdash \lambda x.t}\,\text{abs}$$

free

abstractions
represent functions "$x \mapsto t$"

The λ-calculus

- A **universal** system of computation
- Its terms are formed inductively

$$\frac{}{x \vdash x}\ \text{var} \qquad \frac{\Gamma, x \vdash t}{\Gamma \vdash \lambda x.t}\ \text{abs}$$

free

bound

abstractions
represent functions "$x \mapsto t$"

The λ-calculus

- A **universal** system of computation
- Its terms are formed inductively

$$\frac{}{x \vdash x}\,\text{var} \qquad \frac{\Gamma, x \vdash t}{\Gamma \vdash \lambda x.t}\,\text{abs} \qquad \frac{\Gamma \vdash f \quad \Delta \vdash t}{\Gamma, \Delta \vdash (f\ t)}\,\text{app}$$

free

bound

abstractions
represent functions "$x \mapsto t$"

applications
represent "$f(t)$"

The λ-calculus

- A **universal** system of computation
- Its terms are formed inductively

$$\frac{}{x \vdash x}\ \text{var} \qquad \frac{\Gamma, x \vdash t}{\Gamma \vdash \lambda x.t}\ \text{abs} \qquad \frac{\Gamma \vdash f \quad \Delta \vdash t}{\Gamma, \Delta \vdash (f\ t)}\ \text{app}$$

disjoint

free

bound

abstractions
represent functions "$x \mapsto t$"

applications
represent "$f(t)$"

The λ-calculus

- A **universal** system of computation
- Its terms are formed inductively

disjoint

$$\frac{}{x \vdash x}\,\text{var} \qquad \frac{\Gamma, x \vdash t}{\Gamma \vdash \lambda x.t}\,\text{abs} \qquad \frac{\Gamma \vdash f \quad \Delta \vdash t}{\Gamma, \Delta \vdash (f\ t)}\,\text{app}$$

free

bound

abstractions
represent functions "$x \mapsto t$"

applications
represent "$f(t)$"

$$\frac{\Gamma, x, y, \Delta \vdash t}{\Gamma, y, x, \Delta \vdash t}\,\text{exc} \qquad \frac{\Gamma \vdash t}{\Gamma, x \vdash t}\,\text{wea} \qquad \frac{\Gamma, x, y \vdash t}{\Gamma, x \vdash t[y := x]}\,\text{con}$$

Computing with the λ-calculus

- Substitution rule:

$$t_1[v := t_2]$$

"replace free occurences of v in t_1 with t_2"

(renaming variables in t_1 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"
(renaming variables in t_1 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:

$$t_1[v := t_2]$$

"replace free occurences of v in t_1 with t_2"
(renaming variables in t_1 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!)

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

Represents:

$$f = x \mapsto t_1$$

$$f(t_2) : \text{replace } x \text{ with } t_2 \text{ inside } t_1$$

Computing with the λ-calculus

- Substitution rule:

$$t_1[v := t_2]$$

"replace free occurences of v in t_1 with t_2"
(renaming variables in t_1 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!)

redex

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

Represents:

$$f = x \mapsto t_1$$

$$f(t_2) : \text{replace } x \text{ with } t_2 \text{ inside } t_1$$

β-reducing general terms

- 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!

$$(\lambda x.z)((\lambda x.(x\ x))(\lambda x.(x\ x))) \rightarrow (\lambda x.z)((x\ x)[x := (\lambda x.(x\ x))]) = \ldots$$

$$(\lambda x.z)((\lambda x.(x\ x))(\lambda x.(x\ x))) \rightarrow z[x := (\lambda x.x\ x)(\lambda x.x\ x)] = z$$

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

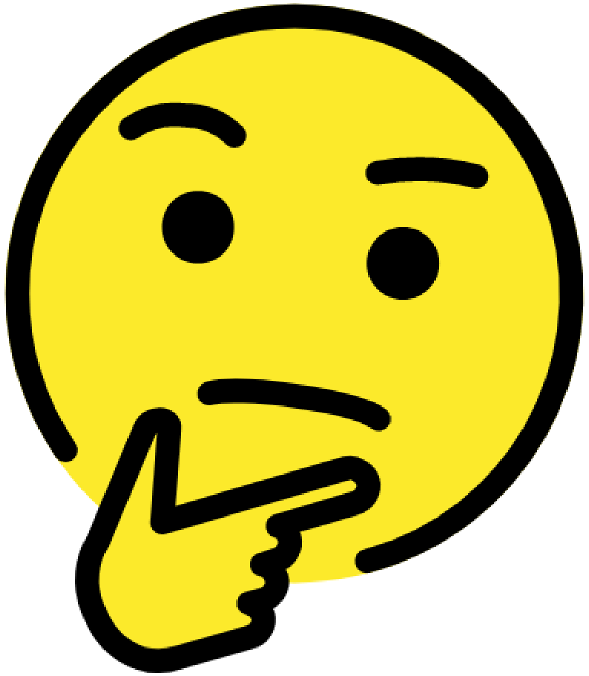
- Asymptotically almost all λ-terms are strongly normalizing.
  [DGKRTZ13]

- 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]
  Model based on 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]
  Model based on de Bruijn indices or combinators (together with appropriate size functions)

Parameter sensitive to 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

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

free variable

$\lambda x.\lambda y.x$

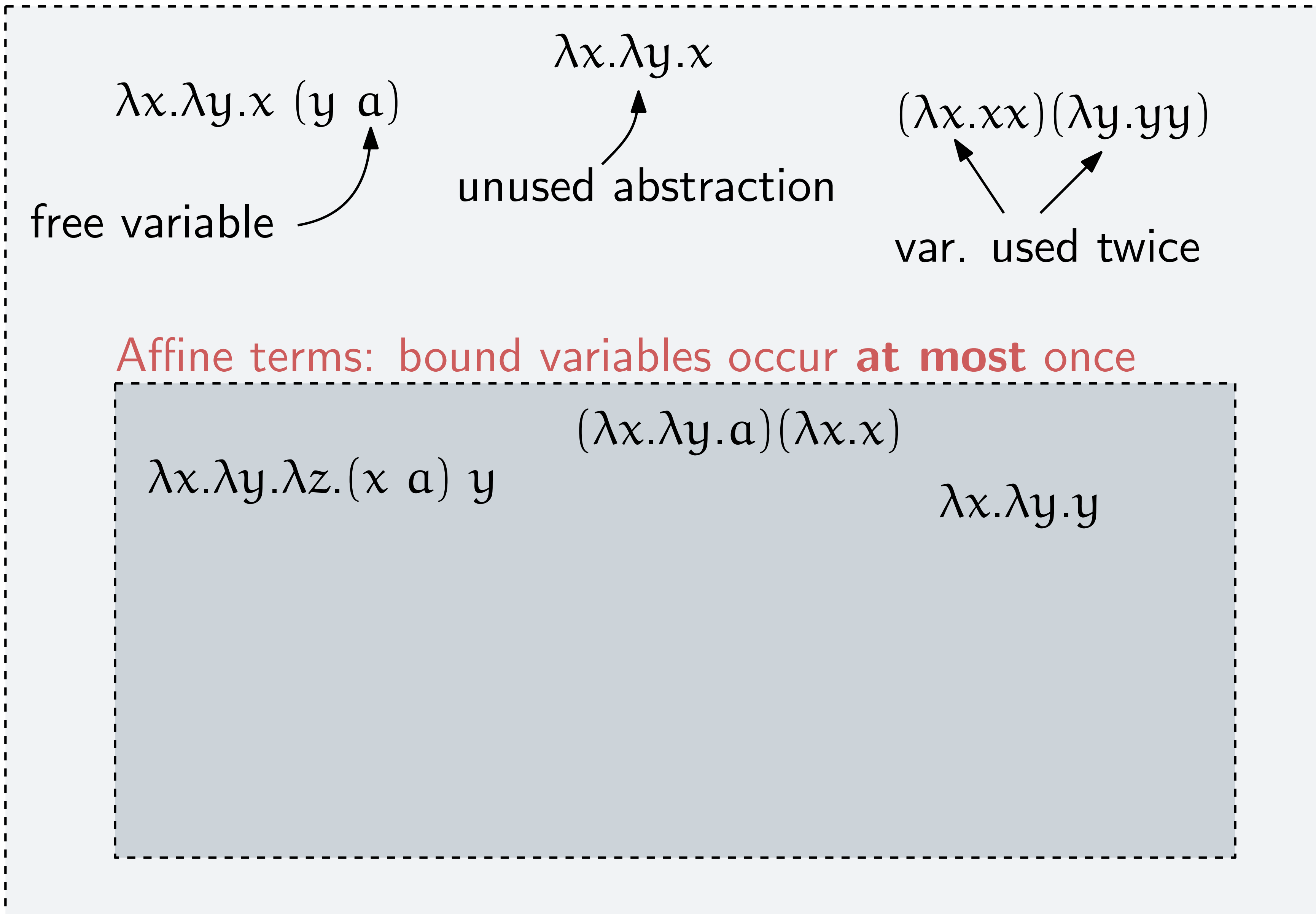
unused abstraction

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

var. used twice

Subfamilies of λ-terms

General terms: no restrictions on variable use



affine = no contraction

Subfamilies of λ-terms

General terms: no restrictions on variable use

$\lambda x.\lambda y.x\ (y\ a)$ — free variable

$\lambda x.\lambda y.x$ — unused abstraction

$(\lambda x.xx)(\lambda y.yy)$ — var. used twice

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

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

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

$\lambda x.\lambda y.y$

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

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

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

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

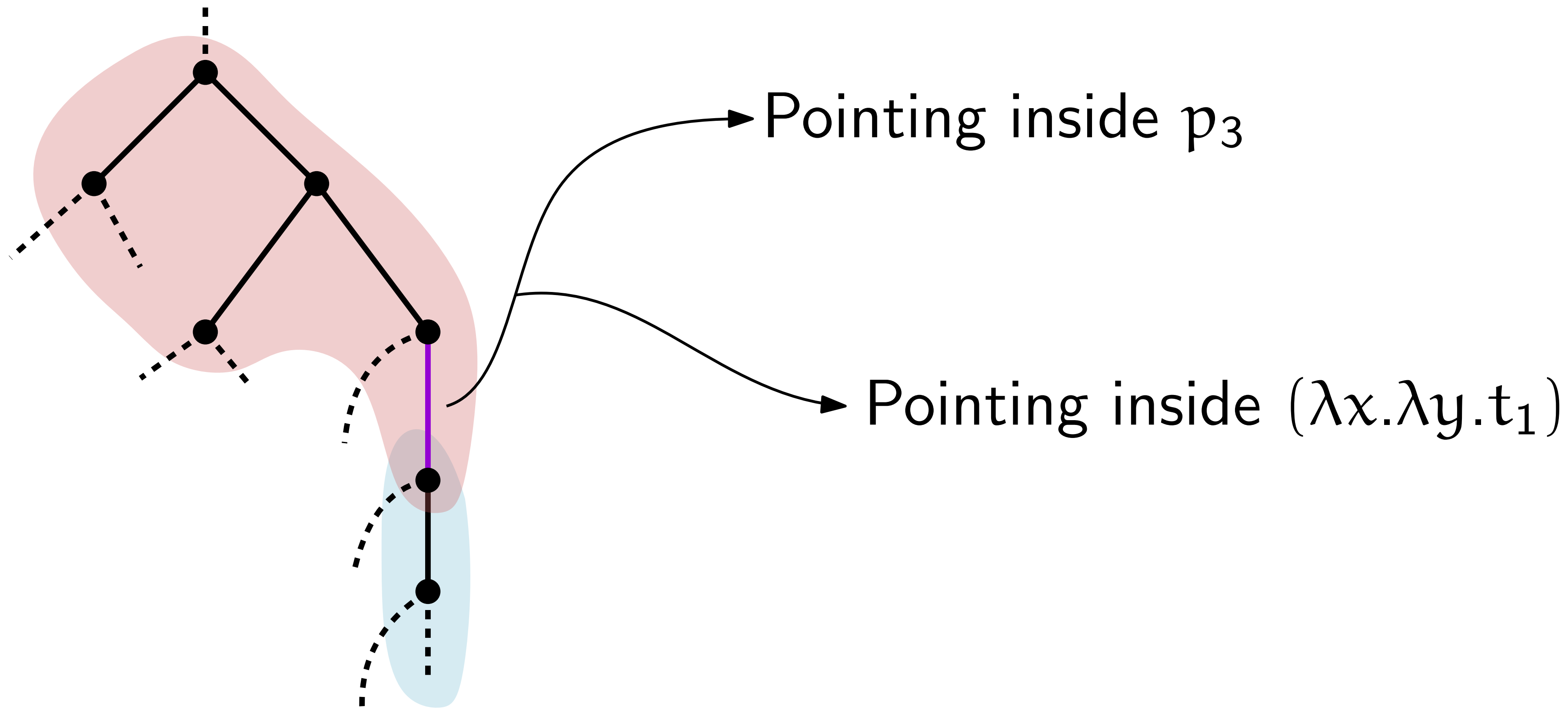
affine = no contraction linear = no contraction, no weakening

Enumerating p_3-patterns

- As before, we'll also need to enumerate auxilliary patterns:

$(\lambda x.\lambda y.t_1)$ $\quad$ $(\lambda x.\lambda y.t_1)\ t_2$ $\quad$ $(\lambda x.\lambda y.t_1)\ t_2\ t_3 \quad (p_3)$

- However we run into a problem:



Enumerating p_3-patterns

- Generatingfunctionology fails, we revert to more elementary methods:

$$\mathbb{E}(V_n) = \mathbb{E}(V_n|\Lambda_n) \cdot \frac{|\Lambda_n|}{|L_n|} + \mathbb{E}(V_n|A_n) \cdot \frac{|A_n|}{|L_n|}$$

Enumerating $\mathfrak{p}_3$-patterns

- Generatingfunctionology fails, we revert to more elementary methods:

asymptotic contribution $\approx \frac{\mathbb{E}(V_{n-3})}{n}$

$$\mathbb{E}(V_n) = \mathbb{E}(V_n|\Lambda_n) \cdot \frac{|\Lambda_n|}{|L_n|} + \mathbb{E}(V_n|A_n) \cdot \frac{|A_n|}{|L_n|}$$

Enumerating p_3-patterns

- Generatingfunctionology fails, we revert to more elementary methods:

asymptotic contribution $\approx \frac{\mathbb{E}(V_{n-3})}{n}$

$$\mathbb{E}(V_n) = \mathbb{E}(V_n | \Lambda_n) \cdot \frac{|\Lambda_n|}{|L_n|} + \mathbb{E}(V_n | A_n) \cdot \frac{|A_n|}{|L_n|}$$

Magic: linear over *families* of all possible abstractions created via cuts from a fixed term!

$$\overline{X}_n = (2n - 12)\overline{X}_{n-3} 2\overline{Y}_{n-3}$$
$$\overline{Y}_n = (2n - 6)Y_{n-3} - 6Y_{n-3}$$
$$\overline{Z}_n = 2(n - 4)(Z + \mathbf{1}_{\Lambda_n})$$

where: X_n counts # of p_1 patt. over terms of size n

Y_n is the same for the pattern $(\lambda x.\lambda y.t_1)\ t_2$, and

Z is the same for the pattern $(\lambda x.\lambda y.t_1)$

The $\overline{V}$ for $V \in \{X_n, Y_n, Z_n\}$ are cummulatives over families of abstractions

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.