Mapping the space of Programs — Searching the space of Languages

Fritz Obermeyer

Department of Mathematics
Carnegie-Mellon University

2007:07:09

Outline

Pick a programming langauge.

Pick a programming langauge.

What are the simplest few programs?

Pick a simple programming langauge.

What are the simplest few programs?

Pick a simple programming langauge.

What are the simplest few program behaviors?

Pick a simple programming langauge family.

What are the simplest few program behaviors?

Pick a simple programming langauge family.

What are the simplest few program behaviors?

What are the simplest programming languages?

Pick a simple programming langauge family.

Combinators / λ-Calculus

What are the simplest few program behaviors?

What are the simplest programming languages?

Pick a simple programming langauge family.

Combinators / λ-Calculus

What are the simplest few program behaviors?

Map space of programs

What are the simplest programming languages?

Pick a simple programming langauge family.

Combinators / λ-Calculus

What are the simplest few program behaviors?

Map space of programs

What are the simplest programming languages?

Learn from examples

Programming as Algebra

Programming as Algebra

What can we do with programs?

- apply one program to another

$$f(x)$$

or just

$$f\ x$$

Programming as Algebra

What can we do with programs?

- apply one program to another
- compose programs

$$\lambda x.f(g\ x)$$

Programming as Algebra

What can we do with programs?

- apply one program to another
- compose programs
- copy programs

$$\lambda x.f\ x\ x$$

Programming as Algebra

$$\lambda x, y.f\ \ y\ \ x$$

What can we do with programs?

- apply one program to another
- compose programs
- copy programs
- permute arguments

Programming as Algebra

$\lambda x.f$

What can we do with programs?

- apply one program to another
- compose programs
- copy programs
- permute arguments
- ignore arguments

Programming as Algebra

$x \mid y$

concurrency
or
non-determinism

What can we do with programs?

- apply one program to another
- compose programs
- copy programs
- permute arguments
- ignore arguments
- run two programs at once

Programming as Algebra

Programming as Algebra

What can we do with programs?

- apply one program to another
- compose programs
- copy programs
- permute arguments
- ignore arguments
- run two programs at once

This is combinatory algebra.

Big Basis?

Big Basis?

not really:

Big Basis?

not really:

consider program

Behavior

Program Behavior: an abstract view

Behavior space is denser than program space.

Program Behavior: an abstract view

Behavior space is denser than program space.
$\longrightarrow$ we can build a bigger map.

Program Behavior: an abstract view

Behavior space is denser than program space.
$\longrightarrow$ we can build a bigger map.

When do two programs behave the same?

Program Behavior: an abstract view

Behavior space is denser than program space.
$\longrightarrow$ we can build a bigger map.

When do two programs behave the same?

- all programs that do nothing are equivalent.

Program Behavior: an abstract view

Behavior space is denser than program space.
$\longrightarrow$ we can build a bigger map.

When do two programs behave the same?

- all programs that do nothing are equivalent.
- equivalent in every `context` $\implies$ equivalent.

Program Behavior: an abstract view

Behavior space is denser than program space.
$\longrightarrow$ we can build a bigger map.

When do two programs behave the same?

- all programs that do nothing are equivalent.
- equivalent in every `context` $\implies$ equivalent.

$\longrightarrow$ defines a maximally dense space.

Program Behavior: an abstract view

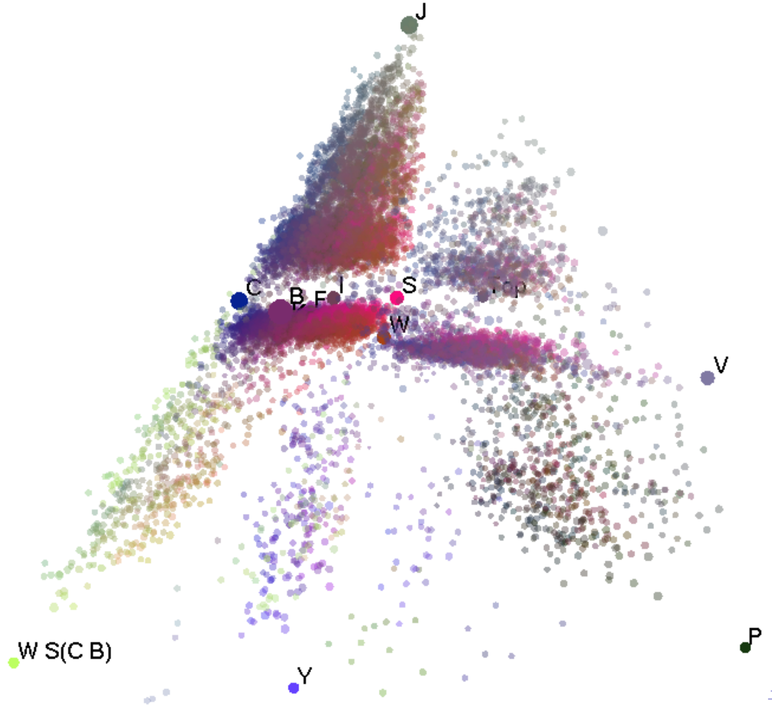
Behavior space is denser than program space.
$\longrightarrow$ we can build a bigger map.

When do two programs behave the same?

- all programs that do nothing are equivalent.
- equivalent in every `context` $\implies$ equivalent.

$\longrightarrow$ defines a maximally dense space.

this is the space to map


J
C
B
F
I
S
W
V
W S(C B)
Y
P

Mapping the space of programs

where to start?

Mapping the space of programs

where to start?

Programs are just elements of an algebra

where to start?

Programs are just elements of an algebra

- a few constants

Mapping the space of programs

where to start?

Programs are just elements of an algebra

- a few constants

(which constants = which language in family)

where to start?

Programs are just elements of an algebra

- a few constants

  (which constants = which language in family)

- a binary operation (function application)

Mapping the space of programs

where to start?

Programs are just elements of an algebra

- a few constants

  (which constants = which language in family)

- a binary operation (function application)

- a few equations

Mapping the space of programs

where to start?

Programs are just elements of an algebra

- a few constants
  (which constants = which language in family)
- a binary operation (function application)
- a few equations (or inequalities)

Mapping the space of programs

where to start?

Programs are just elements of an algebra

- a few constants
(which constants = which language in family)
- a binary operation (function application)
- a few equations (or inequalities)

Try computational algebra methods:

Mapping the space of programs

where to start?

Programs are just elements of an algebra

- a few constants
  (which constants = which language in family)
- a binary operation (function application)
- a few equations (or inequalities)

Try computational algebra methods:
Todd-Coxeter algorithm builds a group

Mapping the space of programs

where to start?

Programs are just elements of an algebra

- a few constants

  (which constants = which language in family)
- a binary operation (function application)
- a few equations (or inequalities)

Try computational algebra methods:

Todd-Coxeter algorithm builds a group

Generalize to non-associative algebra

How to make a map?

How to make a map?

Todd-Coxeter-like algorithm:

How to make a map?

Todd-Coxeter-like algorithm:

Start with basic programs,

Todd-Coxeter-like algorithm:

Start with basic programs, say $\mathbf{S}, \mathbf{K}, \mathbf{J}$

Todd-Coxeter-like algorithm:

Start with basic programs, say $\mathbf{S}, \mathbf{K}, \mathbf{J}$

Then enlarge the map:

Todd-Coxeter-like algorithm:

Start with basic programs, say $\mathbf{S}, \mathbf{K}, \mathbf{J}$

Then enlarge the map:

- choose two random programs

How to make a map?

Todd-Coxeter-like algorithm:

Start with basic programs, say $\mathbf{S}, \mathbf{K}, \mathbf{J}$

Then enlarge the map:

- choose two random programs
- apply one to the other (add row+column)

How to make a map?

Todd-Coxeter-like algorithm:

Start with basic programs, say $\mathbf{S}, \mathbf{K}, \mathbf{J}$

Then enlarge the map:

- choose two random programs
- apply one to the other (add row+column)
- enforce simple algebraic rules

How to make a map?

Todd-Coxeter-like algorithm:

Start with basic programs, say $\mathbf{S}, \mathbf{K}, \mathbf{J}$

Then enlarge the map:

- choose two random programs
- apply one to the other (add row+column)
- enforce simple algebraic rules

sometimes merging programs

Todd-Coxeter-like algorithm:

Start with basic programs, say $\mathbf{S}, \mathbf{K}, \mathbf{J}$

Then enlarge the map:

- choose two random programs
- apply one to the other (add row+column)
- enforce simple algebraic rules

sometimes merging programs (slow)

How to make a map?

Todd-Coxeter-like algorithm:

Start with basic programs, say $\mathbf{S}, \mathbf{K}, \mathbf{J}$

Then enlarge the map:

- choose two random programs
- apply one to the other (add row+column)
- enforce simple algebraic rules

  sometimes merging programs (slow)

When map gets too big, randomly prune programs

Making a map: a simple example

	S	K
S	?	?
K	?	?

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K
S	?	?
K	**?**	?

$$S\ x\ y\ z = x\ z(y\ z)$$

$$K\ x\ y = x$$

Making a map: a simple example

	S	K
S	?	?
K	**KS**	?

$$S\ x\ y\ z = x\ z(y\ z)$$

$$K\ x\ y = x$$

Making a map: a simple example

	S	K	**KS**
S	?	?	**?**
K	**KS**	?	**?**
KS	**?**	**?**	**?**

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS
S	?	?	?
K	KS	?	?
KS	?	?	?

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	**K**	KS
S	?	?	?
K	KS	?	?
KS	?	**S**	?

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS
S	?	?	?
K	KS	?	?
KS	**?**	S	?

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS
S	?	?	?
K	KS	?	?
KS	**S**	S	?

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	**KS**
S	?	?	?
K	KS	?	?
KS	S	S	**?**

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	**KS**
S	?	?	?
K	KS	?	?
KS	S	S	**S**

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS
S	?	?	?
K	KS	?	?
KS	S	S	S

$$S\ x\ y\ z = x\ z(y\ z)$$

$$K\ x\ y = x$$

Making a map: a simple example

	S	K	KS
S	?	**?**	?
K	KS	?	?
KS	S	S	S

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS
S	?	**SK**	?
K	KS	?	?
KS	S	S	S

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS	**SK**
S	?	**SK**	?	**?**
K	KS	?	?	**?**
KS	S	S	S	**?**
SK	**?**	**?**	**?**	**?**

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS	SK
S	?	SK	?	?
K	KS	?	?	?
KS	S	S	S	?
SK	?	?	?	?

no rules apply

$$S\ x\ y\ z = x\ z(y\ z)$$

$$K\ x\ y = x$$

Making a map: a simple example

	S	K	KS	SK
S	?	SK	?	?
K	KS	?	?	?
KS	S	S	S	?
SK	?	**?**	?	?

$$S\ x\ y\ z = x\ z(y\ z)$$

$$K\ x\ y = x$$

Making a map: a simple example

	S	K	KS	SK
S	?	SK	?	?
K	KS	?	?	?
KS	S	S	S	?
SK	?	**SKK**	?	?

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS	SK	**SKK**
S	?	SK	?	?	**?**
K	KS	?	?	?	**?**
KS	S	S	S	?	**?**
SK	?	**SKK**	?	?	**?**
SKK	**?**	**?**	**?**	**?**	**?**

$$S\ x\ y\ z = x\ z (y\ z)$$

$$K\ x\ y = x$$

Making a map: a simple example

	S	K	KS	SK	SKK
S	?	SK	?	?	?
K	KS	?	?	?	?
KS	S	S	S	?	?
SK	?	SKK	?	?	?
SKK	?	?	?	?	?

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS	SK	SKK
S	?	SK	?	?	?
K	KS	?	?	?	?
KS	S	S	S	?	?
SK	?	SKK	?	?	?
SKK	?	?	?	?	?

(SKK)(S) = (KS)(KS)

S x y z = x z(y z)

K x y = x

Making a map: a simple example

	S	K	KS	SK	SKK
S	?	SK	?	?	?
K	KS	?	?	?	?
KS	S	S	S	?	?
SK	?	SKK	?	?	?
SKK	S	?	?	?	?

(SKK)(S) = (KS)(KS

S x y z = x z(y z

K x y = x

Making a map: a simple example

	S	K	KS	SK	SKK
S	?	SK	?	?	?
K	KS	?	?	?	?
KS	S	S	S	?	?
SK	?	SKK	?	?	?
SKK	S	?	?	?	?

S x y z = x z(y z)

K x y = x

Making a map: a large example

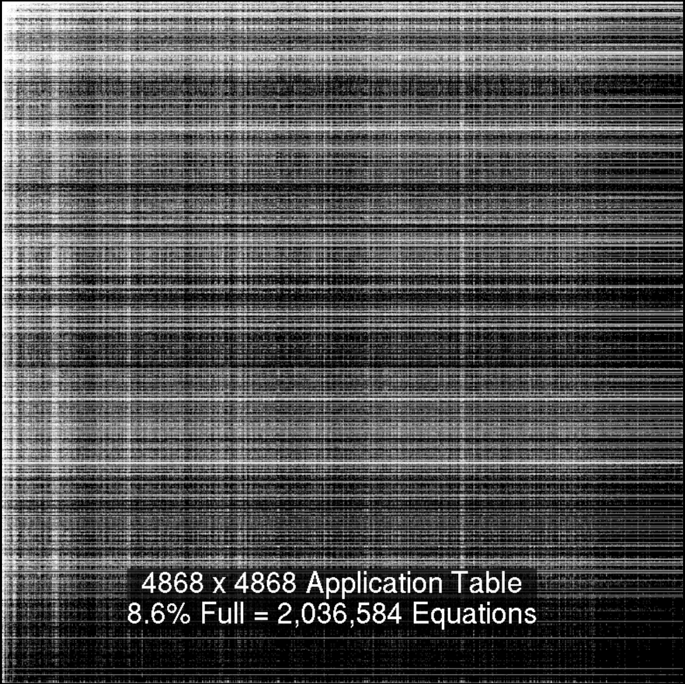
Making a map: a large example

Two data structures

take most of the space

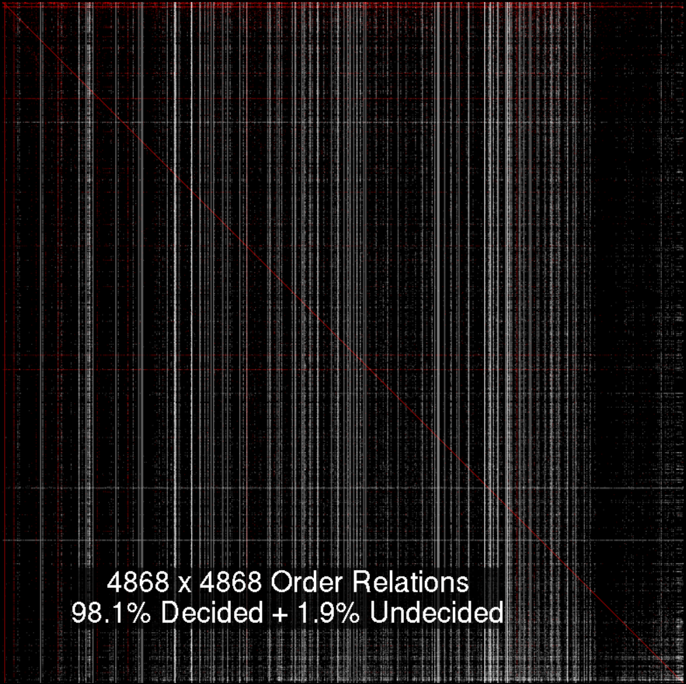
Making a map: a large example

1. a multiplication table


4868 x 4868 Application Table
8.6% Full = 2,036,584 Equations

Making a map: a large example

2. an order relation table


4868 x 4868 Order Relations
98.1% Decided + 1.9% Undecided

How much does it cost?

In Theory:

N programs,

How much does it cost?

In Theory:

N programs, N^2 space,

How much does it cost?

In Theory:

N programs, N^2 space, N^3 time,

How much does it cost?

In Theory:

N programs, N^2 space, N^3 time,

equivalence is undecidable

How much does it cost?

In Theory:

N programs, N^2 space, N^3 time,

equivalence is undecidable

In Practice:

12K programs,

How much does it cost?

In Theory:

N programs, N^2 space, N^3 time,

equivalence is undecidable

In Practice:

12K programs, 1G Bytes,

How much does it cost?

In Theory:

N programs, N^2 space, N^3 time,

equivalence is undecidable

In Practice:

12K programs, 1G Bytes, 1 month

How much does it cost?

In Theory:

N programs, N^2 space, N^3 time,

equivalence is undecidable

In Practice:

12K programs, 1G Bytes, 1 month

equivalence is over 96% decided

A space of programs

What shape is the algebra of programs?

A space of programs

What shape is the algebra of programs?

- Program size gives a norm $|x|$

A space of programs

What shape is the algebra of programs?

- Program size gives a norm $|\, x \,|$ (Kolmogorov complexity)

A space of programs

What shape is the algebra of programs?

- Program size gives a norm $|\ x\ |$ (Kolmogorov complexity)
- also relative complexity $|\ x\ |_y$

A space of programs

What shape is the algebra of programs?

- Program size gives a norm $|x|$ (Kolmogorov complexity)
- also relative complexity $|x|_y$
- symmetrizing gives a distance $d(x,y) = |x|_y + |y|_x$

A space of programs

What shape is the algebra of programs?

- Program size gives a norm $|\ x\ |$ (Kolmogorov complexity)
- also relative complexity $|\ x\ |_y$
- symmetrizing gives a distance $d(x, y) = |\ x\ |_y \quad + \quad |\ y\ |_x$
- this space is asymptotically hyperbolic: volume of sphere is exponential in radius

A space of programs

What shape is the algebra of programs?

- Program size gives a norm $|\, x \,|$ (Kolmogorov complexity)
- also relative complexity $|\, x \,|_y$
- symmetrizing gives a distance $d(x, y) = |\, x \,|_y + |\, y \,|_x$
- this space is asymptotically hyperbolic: volume of sphere is exponential in radius

Gromov studied the geometry of groups

A space of programs

What shape is the algebra of programs?

- Program size gives a norm $|\, x \,|$ (Kolmogorov complexity)
- also relative complexity $|\, x \,|_y$
- symmetrizing gives a distance $d(x, y) = |\, x \,|_y \quad + \quad |\, y \,|_x$
- this space is asymptotically hyperbolic: volume of sphere is exponential in radius

Gromov studied the geometry of groups
this is a non-associative generalization

Visualizing the space of programs

Goal Programming styles are local

Visualizing the space of programs

Goal Programming styles are local

- atoms are far-out

Visualizing the space of programs

Goal Programming styles are local

- atoms are far-out
- related programs are close together

Visualizing the space of programs

Goal Programming styles are local

- atoms are far-out
- related programs are close together
- parse trees are small

How Pose as eigenvector problem

Visualizing the space of programs

Goal Programming styles are local

- atoms are far-out
- related programs are close together
- parse trees are small

How Pose as eigenvector problem

- linear springs between programs

Visualizing the space of programs

Goal Programming styles are local

- atoms are far-out
- related programs are close together
- parse trees are small

How Pose as eigenvector problem

- linear springs between programs
- simpler programs are heavier

Visualizing the space of programs

Goal Programming styles are local

- atoms are far-out
- related programs are close together
- parse trees are small

How Pose as eigenvector problem

- linear springs between programs
- simpler programs are heavier
- project to first few dimensions:

Visualizing the space of programs

Goal Programming styles are local

- atoms are far-out
- related programs are close together
- parse trees are small

How Pose as eigenvector problem

- linear springs between programs
- simpler programs are heavier
- project to first few dimensions: 3 space

Visualizing the space of programs

Goal Programming styles are local

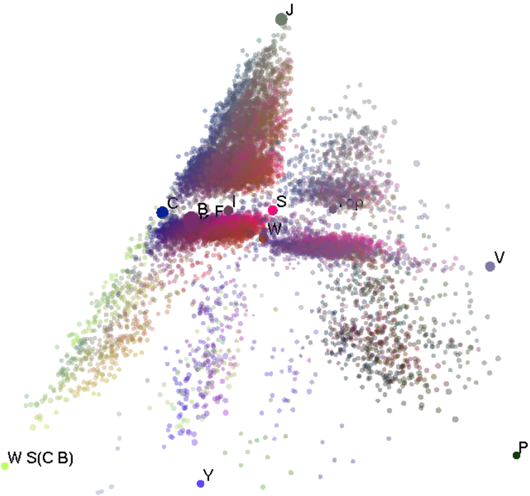
- atoms are far-out
- related programs are close together
- parse trees are small

How Pose as eigenvector problem

- linear springs between programs
- simpler programs are heavier
- project to first few dimensions:
  3 space + 3 color

see interactive maps...

www.math.cmu.edu/~fho/johann/



Where to map?

Time complexity is cubic:

Where to map?

Time complexity is cubic: mis-fitting is expensive!

Where to map?

Time complexity is cubic: mis-fitting is expensive!

A simple basis,

Where to map?

Time complexity is cubic: mis-fitting is expensive!

A simple basis,

$$
\begin{aligned}
\text{program} \;::=\;& \mathbf{S} \\
\mid\;& \mathbf{K} \\
\mid\;& \mathbf{J} \\
\mid\;& (\text{program}\ \ \text{program})
\end{aligned}
$$

Where to map?

Time complexity is cubic: mis-fitting is expensive!

A simple basis, with simple weights

$$
\begin{array}{rcll}
\text{program} & ::= & \mathbf{S} & @\ 1/6 \\
 & | & \mathbf{K} & @\ 1/6 \\
 & | & \mathbf{J} & @\ 1/6 \\
 & | & (\text{program}\ \ \text{program}) & @\ 1/2
\end{array}
$$

Where to map?

Time complexity is cubic: mis-fitting is expensive!

A simple basis, with simple weights

$$\begin{array}{rcll} \text{program} & ::= & \mathbf{S} & @\ 1/6 \\ & | & \mathbf{K} & @\ 1/6 \\ & | & \mathbf{J} & @\ 1/6 \\ & | & (\text{program program}) & @\ 1/2 \end{array}$$

Choice of small basis is arbitrary

Where to map?

Time complexity is cubic: mis-fitting is expensive!

A simple basis, with simple weights

$$
\begin{aligned}
\text{program} ::= \; & \mathbf{S} & @\ 1/6 \\
\mid \; & \mathbf{K} & @\ 1/6 \\
\mid \; & \mathbf{J} & @\ 1/6 \\
\mid \; & (\text{program program}) & @\ 1/2
\end{aligned}
$$

Choice of small basis is arbitrary

$\Longrightarrow$ extra information

Where to map?

Time complexity is cubic: mis-fitting is expensive!

A simple basis, with simple weights

$$
\begin{array}{rcll}
\text{program} & ::= & \mathbf{S} & @\ 1/6 \\
& | & \mathbf{K} & @\ 1/6 \\
& | & \mathbf{J} & @\ 1/6 \\
& | & (\text{program program}) & @\ 1/2
\end{array}
$$

Choice of small basis is arbitrary

$\Longrightarrow$ extra information $\Longrightarrow$ not simple

Where to map?

Time complexity is cubic: mis-fitting is expensive!

A simple basis, with simple weights

$$
\begin{aligned}
\text{program} \; ::= \; & \mathbf{S} && \texttt{@}\ 1/6 \\
\mid \; & \mathbf{K} && \texttt{@}\ 1/6 \\
\mid \; & \mathbf{J} && \texttt{@}\ 1/6 \\
\mid \; & (\text{program}\ \ \text{program}) && \texttt{@}\ 1/2
\end{aligned}
$$

Choice of small basis is arbitrary
$\Longrightarrow$ extra information $\Longrightarrow$ not simple

which languages are simple?

Complexity

Kolmogorov's view: complexity is a norm

Complexity

Kolmogorov's view: complexity is a norm
Solomonoff's view: complexity is $-\log(\text{probability})$

Complexity → Probability

Kolmogorov's view: complexity is a norm
Solomonoff's view: complexity is $-\log(\text{probability})$

so consider...

a probability space of programs

Complexity → Probability

Kolmogorov's view: complexity is a norm
Solomonoff's view: complexity is $-\log(\text{probability})$

so consider...

a probability space of programs
parametrized by a language

Complexity $\longrightarrow$ Probability

Kolmogorov's view: complexity is a norm
Solomonoff's view: complexity is $-\log(\text{probability})$

so consider...

a probability space of programs
parametrized by a language
($=$ weighted set of basic programs)

Complexity → Probability → Geometry

Kolmogorov's view: complexity is a norm
Solomonoff's view: complexity is $-\log(\text{probability})$

so consider...

a probability space of programs
parametrized by a language
($=$ weighted set of basic programs)

now we have...

a space of languages

Complexity $\to$ Probability $\to$ Geometry

Kolmogorov's view: complexity is a norm
Solomonoff's view: complexity is $-\log(\text{probability})$

so consider...

a probability space of programs
parametrized by a language
($=$ weighted set of basic programs)

now we have...

a space of languages: information manifold

Complexity $\longrightarrow$ Probability $\longrightarrow$ Geometry

Kolmogorov's view: complexity is a norm
Solomonoff's view: complexity is $-\log$(probability)

so consider...

a probability space of programs
parametrized by a language
($=$ weighted set of basic programs)

now we have...

a space of languages: information manifold
$\implies$ Riemannian manifold

Complexity $\to$ Probability $\to$ Geometry

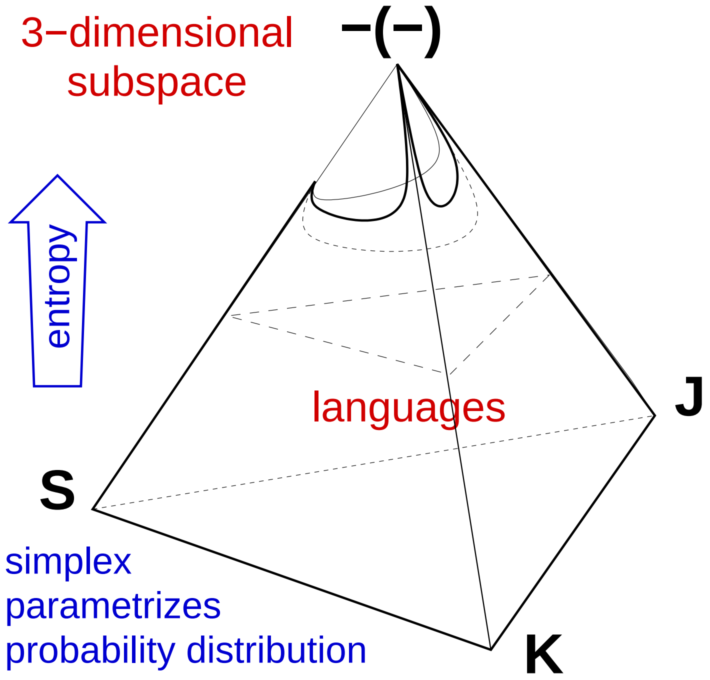
Kolmogorov's view: complexity is a norm
Solomonoff's view: complexity is $-\log(\text{probability})$

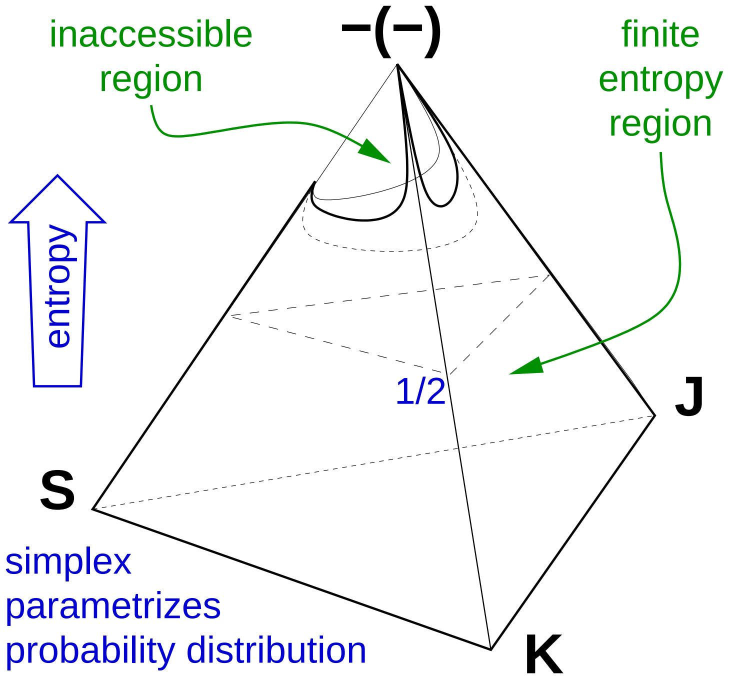
so consider...

a probability space of programs
parametrized by a language
($=$ weighted set of basic programs)

now we have...

a space of languages: information manifold
$\implies$ Riemannian manifold
$\implies$ differential manifold


3−dimensional
subspace
−(−)
entropy
languages
J
S
K
simplex
parametrizes
probability distribution



inaccessible region
−(−)
finite entropy region
entropy
1/2
J
S
K
simplex parametrizes probability distribution

So What?

How to find a simple language

Goal: map interesting programs

How to find a simple language

Goal: map interesting programs

Constraint: limited space and time

How to find a simple language

Goal: map interesting programs

Constraint: limited space and time

so find a language that makes
interesting programs simple

How to find a simple language

Goal: map interesting programs

Constraint: limited space and time

so find a language that makes
interesting programs simple

We know many interesting programs.

How to find a simple language

Goal: map interesting programs
Constraint: limited space and time

so find a language that makes
interesting programs simple

We know many interesting programs.
Language space is a Riemannian manifold.

How to find a simple language

Goal: map interesting programs
Constraint: limited space and time

so find a language that makes
interesting programs simple

We know many interesting programs.
Language space is a Riemannian manifold.

so collect a training set of programs

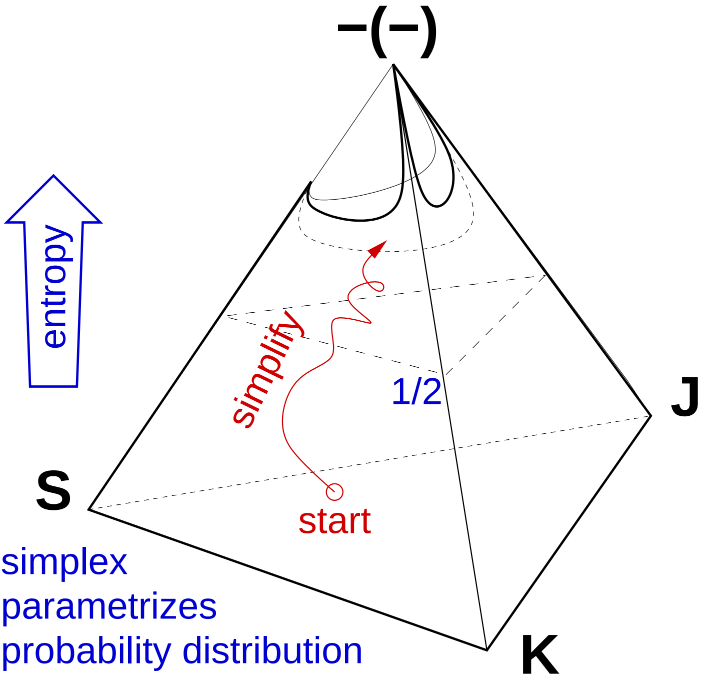
How to find a simple language

Goal: map interesting programs
Constraint: limited space and time

so find a language that makes
interesting programs simple

We know many interesting programs.
Language space is a Riemannian manifold.

so collect a training set of programs, and
do gradient descent to minimize its complexity



simplex
parametrizes
probability distribution

Potential Applications

Program simplification

Potential Applications

Program simplification

using database of simplest rewrites

Potential Applications

Program simplification

using database of simplest rewrites

Software analysis

Potential Applications

Program simplification

using database of simplest rewrites

Software analysis

refactor based on spatial proximity

Potential Applications

Program simplification

using database of simplest rewrites

Software analysis

refactor based on spatial proximity

Universal Bayesian filtering

Potential Applications

Program simplification

using database of simplest rewrites

Software analysis

refactor based on spatial proximity

Universal Bayesian filtering

practical Solomonoff induction?

Potential Applications

Program simplification

using database of simplest rewrites

Software analysis

refactor based on spatial proximity

Universal Bayesian filtering

practical Solomonoff induction?

Programming by searching

Potential Applications

Program simplification

using database of simplest rewrites

Software analysis

refactor based on spatial proximity

Universal Bayesian filtering

practical Solomonoff induction?

Programming by searching

calibrate search with examples

Potential Applications

Potential Applications

Program simplification
using database of simplest rewrites

Software analysis
refactor based on spatial proximity

Universal Bayesian filtering
practical Solomonoff induction?

Programming by searching
calibrate search with examples

Bayesian foundation for genetic programming

Summary

- the average over all languages
  is simpler than any one

Summary

- the average over all languages
  is simpler than any one

- complexity $\longrightarrow$ probability

Summary

- the average over all languages
  is simpler than any one
- complexity $\longrightarrow$ probability $\longrightarrow$ geometry

Summary

- the average over all languages
  is simpler than any one
- complexity $\longrightarrow$ probability $\longrightarrow$ geometry
- learn simplicity from examples

Summary

- the average over all languages
  is simpler than any one
- complexity $\longrightarrow$ probability $\longrightarrow$ geometry
- learn simplicity from examples

Questions

- what should those examples be?

Summary

- the average over all languages
  is simpler than any one
- complexity $\longrightarrow$ probability $\longrightarrow$ geometry
- learn simplicity from examples

Questions

- what should those examples be?
- how is real software shaped?