Object Complexity - Examples and Tools

Edward Szczypka
(TCS UJ)

6 June 2008

Algorithm Complexity

Definition (Algorithm Complexity)

Computer resources needed to run the algorithm.

1. time complexity - how many steps does it take,
2. memory complexity - how much memory does it take.

Algorithm Complexity

Definition (Algorithm Complexity)

Computer resources needed to run the algorithm.

1. time complexity - how many steps does it take,
2. memory complexity - how much memory does it take.

Certainly, the required resources strictly depend on the complexity of input data.

Complexity of Input Data

How can we define an input complexity function?

1. usually defined as a number of isolated parts,
2. would be nice to get the same input complexity for the data with similar time and size algorithm complexity.

Complexity of Input Data

How can we define an input complexity function?

1. usually defined as a number of isolated parts,
2. would be nice to get the same input complexity for the data with similar time and size algorithm complexity.

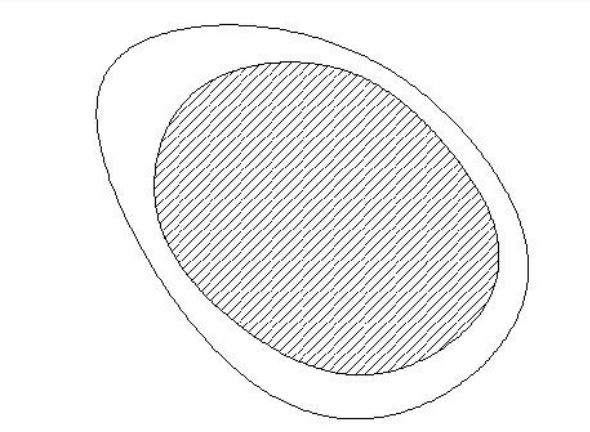
Las Vegas and Monte Carlo Algorithms

Most problems are difficult (a lot of resources).

Solution - algorithm that

1. works fast on *almost all* inputs (Las Vegas algorithm),
2. gives correct answers for *almost all* cases (Monte Carlo algorithm).

Problem (what we mean by *almost all* and/or *sufficiently many*.)



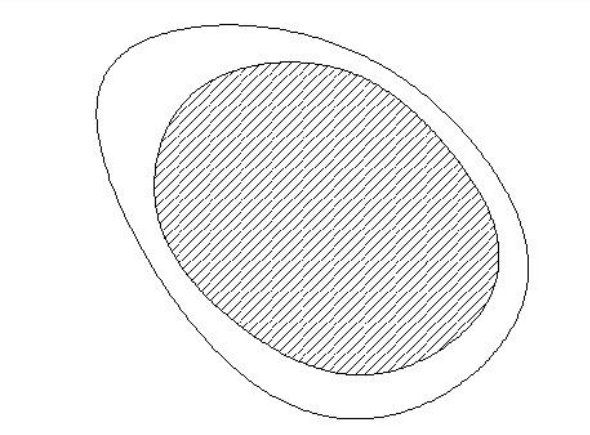
Las Vegas and Monte Carlo Algorithms

Most problems are difficult (a lot of resources).

Solution - algorithm that

1. works fast on *almost all* inputs (Las Vegas algorithm),
2. gives correct answers for *almost all* cases (Monte Carlo algorithm).

Problem (what we mean by *almost all* and/or *sufficiently many*.)



Randomization Algorithms

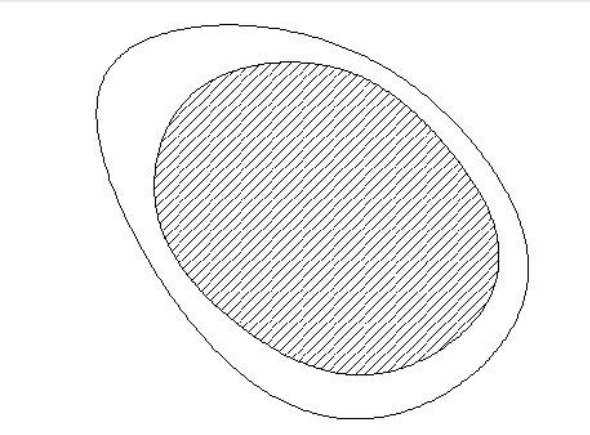
Las Vegas and Monte Carlo Algorithms

Most problems are difficult (a lot of resources).

Solution - algorithm that

1. works fast on *almost all* inputs (Las Vegas algorithm),
2. gives correct answers for *almost all* cases (Monte Carlo algorithm).

Problem (what we mean by *almost all* and/or *sufficiently many*.)



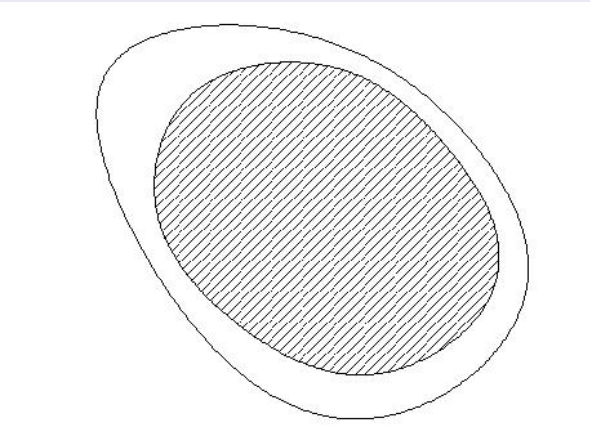
Las Vegas and Monte Carlo Algorithms

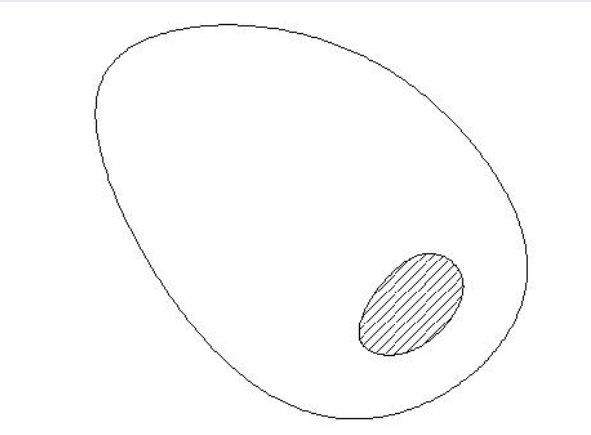
Most problems are difficult (a lot of resources).

Solution - algorithm that

1. works fast on *almost all* inputs (Las Vegas algorithm),
2. gives correct answers for *almost all* cases (Monte Carlo algorithm).

Problem (what we mean by *almost all* and/or *sufficiently many*.)





Object Complexity Function and Complexity Measure

Definition (Object Complexity)

$f : T \to \mathbb{N}$ is an input(object) complexity function for the set T of objects with the order $\prec$ of such objects if

1. $f(\mu) < f(\tau)$, for $\mu \prec \tau$,
2. $f^{-1}(i)$ finite,
3. $\liminf_{i \to \infty, i \in f(T)} \left\| f^{-1}(i) \right\| = \infty$.

Definition (Complexity Measure)

for $A \subseteq T$:

$$\mu_f(A) = \lim_{i \to \infty, i \in f(T)} \frac{\left\| A \cap f^{-1}(i) \right\|}{\left\| f^{-1}(i) \right\|}.$$

Object Complexity Function and Complexity Measure

Definition (Object Complexity)

$f : T \to \mathbb{N}$ is an input(object) complexity function for the set T of objects with the order $\prec$ of such objects if

1. $f(\mu) < f(\tau)$, for $\mu \prec \tau$,
2. $f^{-1}(i)$ finite,
3. $\liminf_{i \to \infty, i \in f(T)} \left\| f^{-1}(i) \right\| = \infty$.

Definition (Complexity Measure)

for $A \subseteq T$:

$$\mu_f(A) = \lim_{i \to \infty, i \in f(T)} \frac{\left\| A \cap f^{-1}(i) \right\|}{\left\| f^{-1}(i) \right\|}.$$

Almost Every

Definition (Almost Every)

For a given property P, we say that almost every element of T has this property if

$$\mu_f\left(\{t \in T : \ t \text{ satisfies } P\}\right) = 1.$$

Examples

Example (Object Complexity Functions on Trees)

1. the number of leaves,
2. the number of nodes,
3. the number of branches,

and

4. the hight of tree.

Examples

Example (Object Complexity Functions on Trees)

1. the number of leaves,
2. the number of nodes,
3. the number of branches,

and

1. the hight of tree.

Similarity and Equivalence

Definition (Similarity of Object Complexity Functions)

$f, g : T \to \mathbb{N}$ are similar $(f \sim g)$
if and only if
$\mu_f(A) = \mu_g(A)$ for any $A \subseteq T$ when $\mu_f(A)$ and $\mu_g(A)$ exist.

Definition (Equivalence of Object Complexity Functions)

$f, g : T \to \mathbb{N}$ are equivalent $(f \approx g)$
if and only if
f and g are similar
and in addition $\mu_f(A)$ i $\mu_g(A)$ simultaneously exist or do not exist

Note

Equivalence $\approx$ is an equivalence relation,
Similarity $\sim$ usually not.

Similarity and Equivalence

Definition (Similarity of Object Complexity Functions)

$f, g : T \to \mathbb{N}$ are similar $(f \sim g)$
if and only if
$\mu_f(A) = \mu_g(A)$ for any $A \subseteq T$ when $\mu_f(A)$ and $\mu_g(A)$ exist.

Definition (Equivalence of Object Complexity Functions)

$f, g : T \to \mathbb{N}$ are equivalent $(f \approx g)$
if and only if
f and g are similar
and in addition $\mu_f(A)$ i $\mu_g(A)$ simultaneously exist or do not exist

Note

Equivalence $\approx$ is an equivalence relation,
Similarity $\sim$ usually not.

Similarity and Equivalence

Definition (Similarity of Object Complexity Functions)

$f, g : T \to \mathbb{N}$ are similar $(f \sim g)$
if and only if
$\mu_f(A) = \mu_g(A)$ for any $A \subseteq T$ when $\mu_f(A)$ and $\mu_g(A)$ exist.

Definition (Equivalence of Object Complexity Functions)

$f, g : T \to \mathbb{N}$ are equivalent $(f \approx g)$
if and only if
f and g are similar
and in addition $\mu_f(A)$ i $\mu_g(A)$ simultaneously exist or do not exist

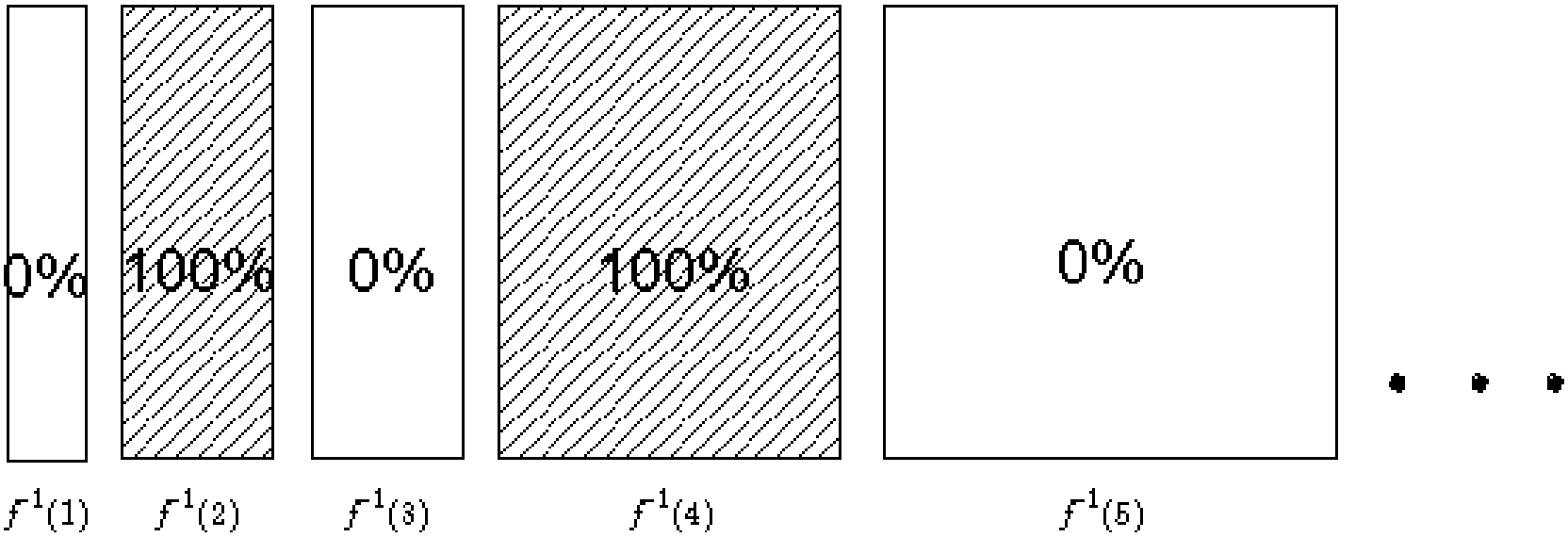
Note

Equivalence $\approx$ is an equivalence relation,
Similarity $\sim$ usually not.

Basic Properties of Complexity Measure

There exists a set that is not measurable by f

The set $\bigcup_{i=1}^{\infty} f^{-1}(2i)$ is not measurable by μ_f.



$$A = \bigcup_{i=1}^{\infty} f^{-1}(2i)$$

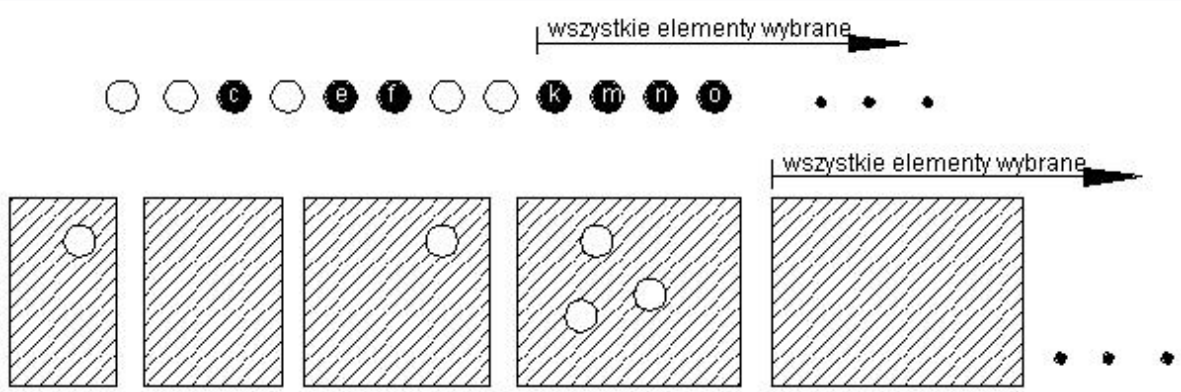
Basic Properties of Complexity Measure

Definition (Code)

Code is an object complexity function with at most one element in every slice.

Similarity and code

1. every object complexity function is similar to code,
2. but lack of transitivity.



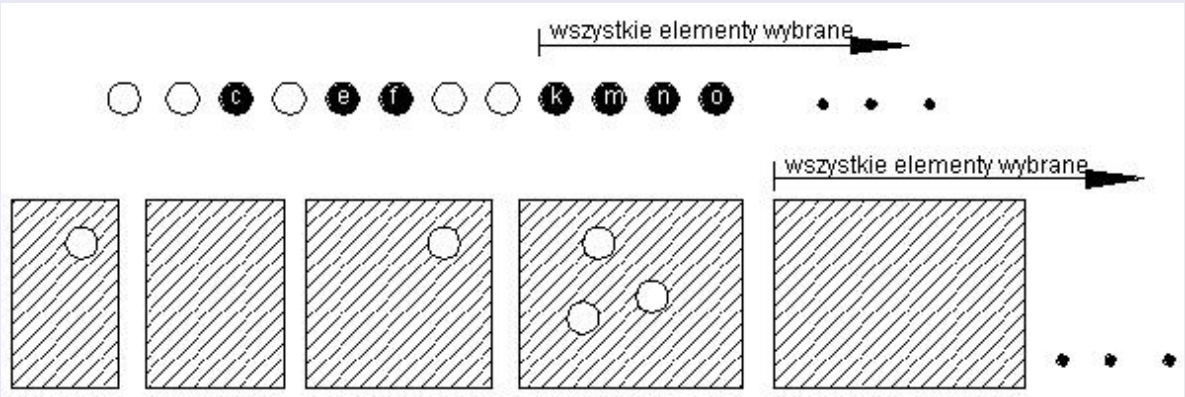
Basic Properties of Complexity Measure

Definition (Code)

Code is an object complexity function with at most one element in every slice.

Similarity and code

1. every object complexity function is similar to code,
2. but lack of transitivity.

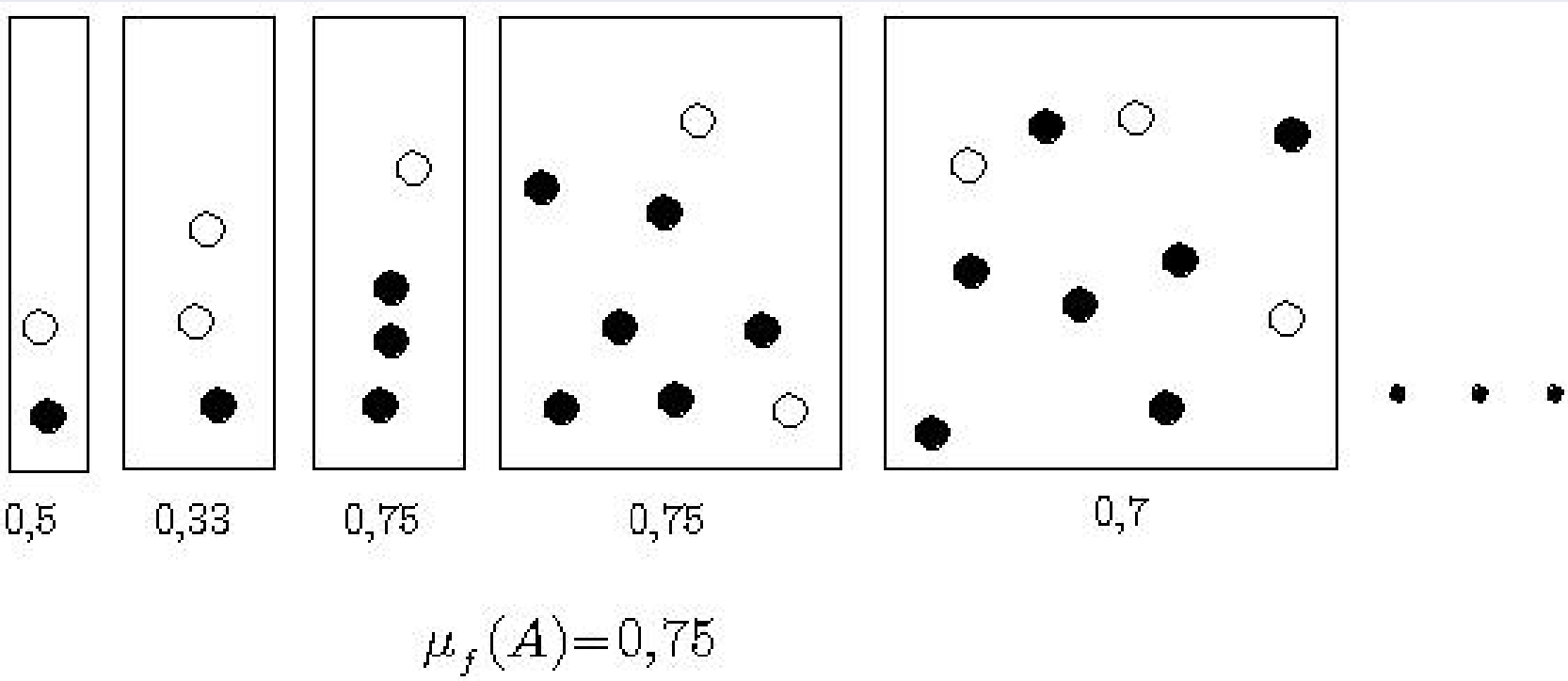


Basic Properties of Complexity Measure

There is a set for any fixed measure

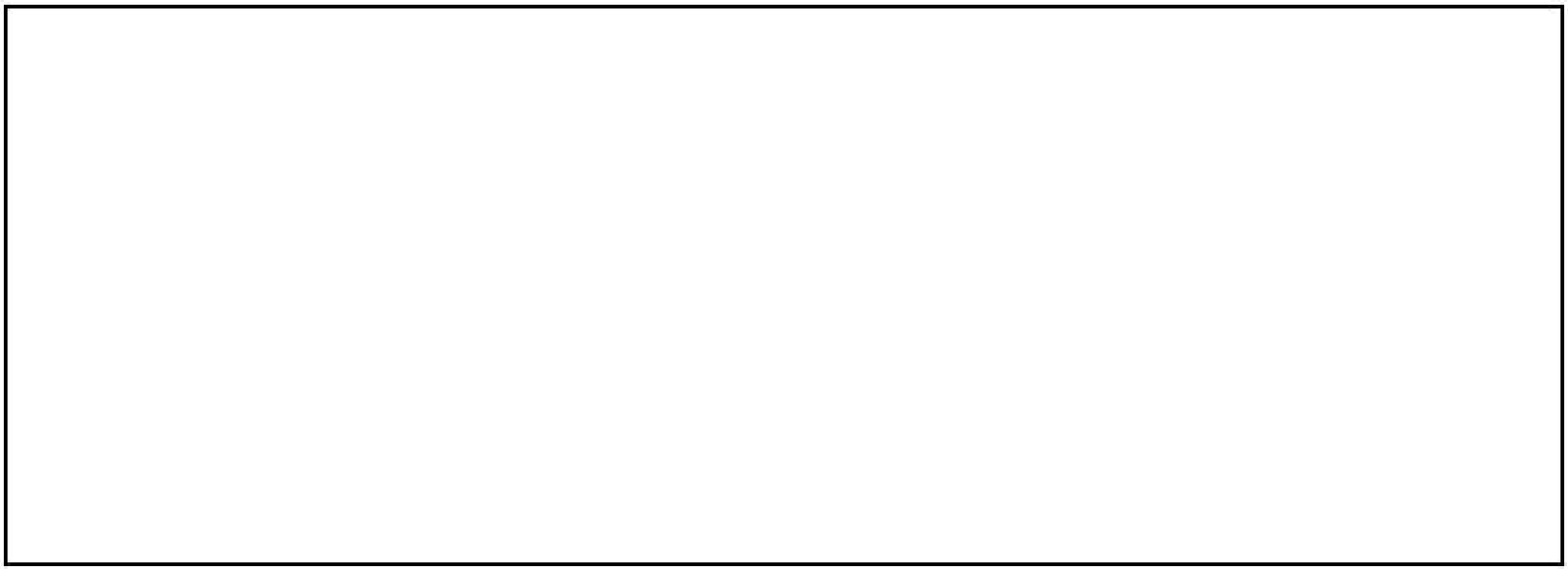
Fix $m \in [0, 1]$

There is a set A, such that $\mu_f(A) = m$.



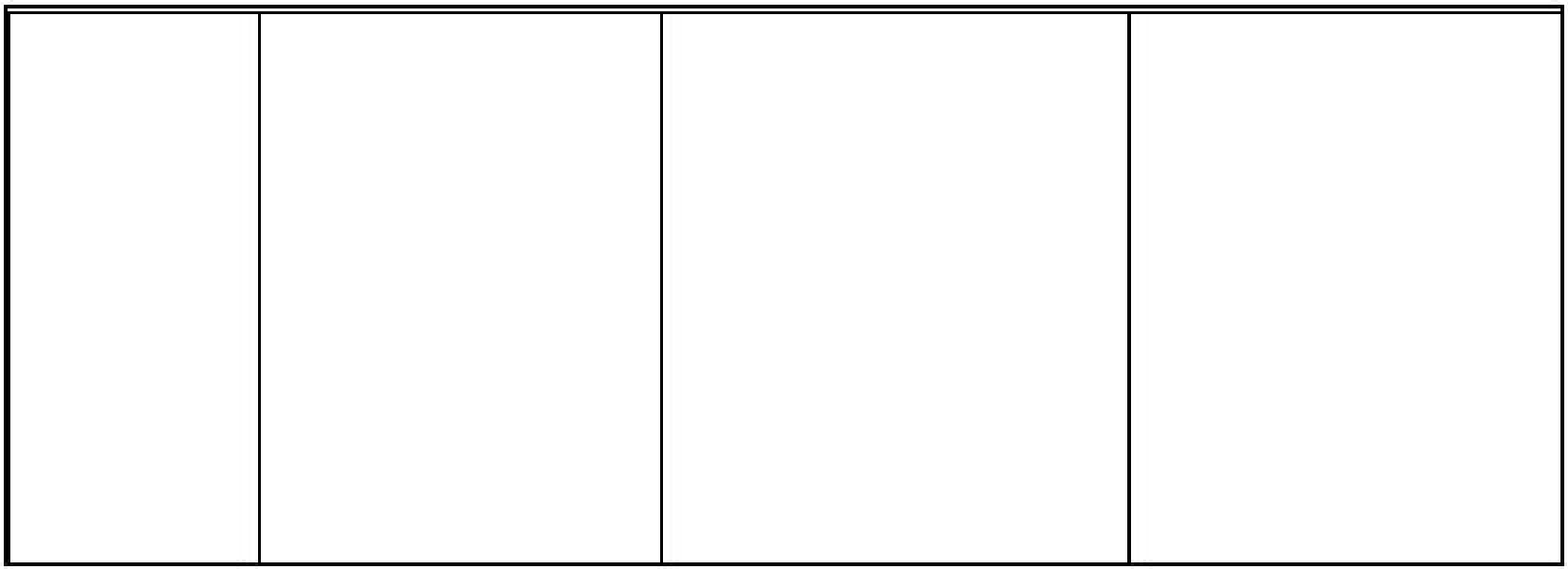
Inclusion vs Similarity

If each slice of f is included in exactly one slice g (with possibly finitely many exceptions) then f and g are similar ($f \sim g$).



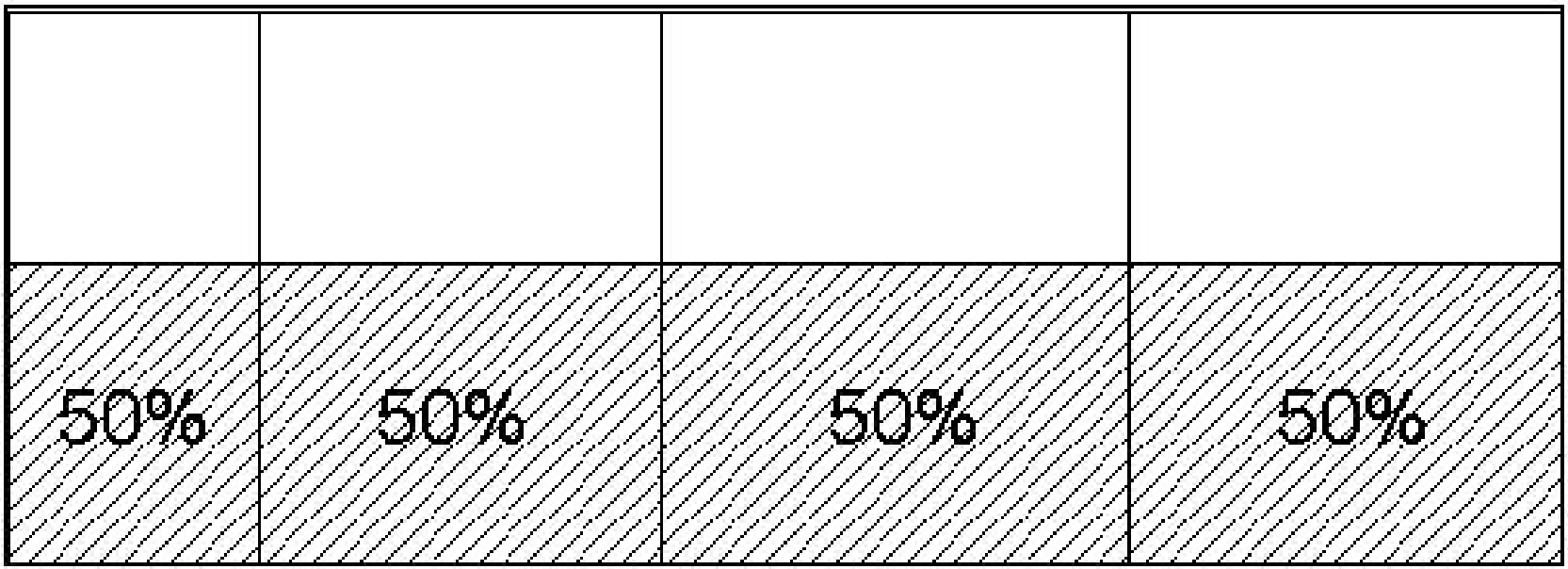
Inclusion vs Similarity

If each slice of f is included in exactly one slice g (with possibly finitely many exceptions) then f and g are similar ($f \sim g$).



Inclusion vs Similarity

If each slice of f is included in exactly one slice g (with possibly finitely many exceptions) then f and g are similar ($f \sim g$).



Examples of Object Complexity Functions

Natural Numbers

There is only one surjective object complexity function for $(\mathbb{N}, <)$, i.e. $f(n) = n$.

Examples of Object Complexity Functions

Definition (Words)

Words

$$\Sigma^* = \bigcup_{n \in \omega} \Sigma^n$$

Order

$u \leq v$ wtw

$xuy = v$ for some $x, y \in \Sigma^*$.

Complexity of $\{a, b\}^*$

Any function $f_k(u)$ with

$$f_k(u) = (\text{the number of } a \text{ in } u) + k \cdot (\text{the number of } b \text{ in } u)$$

is an object complexity function.

Examples of Object Complexity Functions

Definition (Words)

Words

$$\Sigma^* = \bigcup_{n \in \omega} \Sigma^n$$

Order

$u \leq v$ wtw

$xuy = v$ for some $x, y \in \Sigma^*$.

Complexity of $\{a, b\}^*$

Any function $f_k(u)$ with

$$f_k(u) = (\text{the number of } a \text{ in } u) + k \cdot (\text{the number of } b \text{ in } u)$$

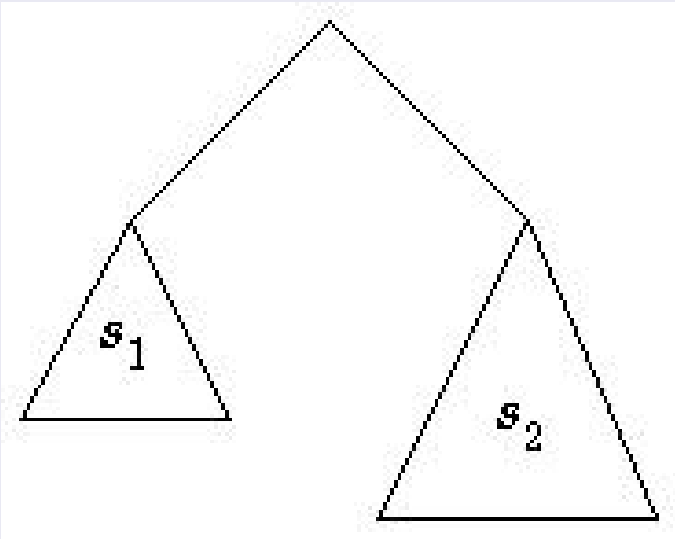
is an object complexity function.

Examples of Object Complexity Functions

Definition (Binary Trees)

Point $\cdot$ is a tree.
If s_1 and s_2 are trees
then $t = s_1 \hat{} s_2$ is a tree.

$s \leq t$ if and only if
s is a subtree of t



Object Complexity on Trees

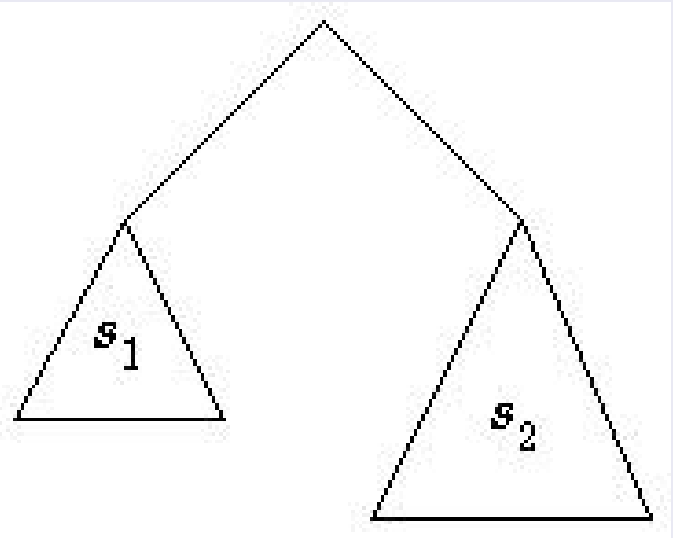
1. The width of tree f_c (the number of leaves)
   $f_c(\cdot) = 1, f_c(s_1 \hat{} s_2) = f_c(s_1) + f_c(s_2),$
2. The height of tree f_h
   $f_h(\cdot) = 1, f_h(s_1 \hat{} s_2) = \max\{f_c(s_1), f_c(s_2)\} + 1,$

Examples of Object Complexity Functions

Definition (Binary Trees)

Point $\cdot$ is a tree.
If s_1 and s_2 are trees
then $t = s_1 \hat{} s_2$ is a tree.

$s \leq t$ if and only if
s is a subtree of t



Object Complexity on Trees

1. The width of tree f_c (the number of leaves)
   $f_c(\cdot) = 1, f_c(s_1 \hat{} s_2) = f_c(s_1) + f_c(s_2),$
2. The height of tree f_h
   $f_h(\cdot) = 1, f_h(s_1 \hat{} s_2) = \max\{f_c(s_1), f_c(s_2)\} + 1,$

Object Complexity on Trees

Theorem (The smallest object complexity function on trees)

The function f_h is the smallest function on binary trees,
i.e. $f_h(t) \leq f(t)$,

The function f_h has the biggest slices

$\left\| f_h^{-1}(1,\ldots,n) \right\| \geq \left\| f^{-1}(1,\ldots,n) \right\|$, for any f
$\left\| f_h^{-1}(1,\ldots,n) \right\| = 2^{\theta(2^n)}$

Object Complexity on Trees

Theorem (The smallest object complexity function on trees)

The function f_h is the smallest function on binary trees,
i.e. $f_h(t) \leq f(t)$,

The function f_h has the biggest slices

$\left\| f_h^{-1}(1,\ldots,n) \right\| \geq \left\| f^{-1}(1,\ldots,n) \right\|$, for any f
$\left\| f_h^{-1}(1,\ldots,n) \right\| = 2^{\theta(2^n)}$

Object Complexity on Trees

Functions with small slices

1. Codes have small slices but they are not symmetric,
2. Function f_h is symmetric,
   i.e. $f_h(s_1{}^\wedge s_2) = f_h(s_2{}^\wedge s_1)$,
3. For $\gamma > 0$ exists symmetric object complexity function f, such that $\left\| f^{-1}(i) \right\| = O(i^\gamma)$,
4. For any symmetric complexity function f exist $\alpha, \beta > 0$ such that $\left\| f^{-1}(i) \right\| \geq \alpha \cdot i^\beta$.

Object Complexity on Trees

Functions with small slices

1. Codes have small slices but they are not symmetric,
2. Function f_h is symmetric,
   i.e. $f_h(s_1{}^\wedge s_2) = f_h(s_2{}^\wedge s_1)$,
3. For $\gamma > 0$ exists symmetric object complexity function f, such that $\left\| f^{-1}(i) \right\| = O(i^\gamma)$,
4. For any symmetric complexity function f exist $\alpha, \beta > 0$ such that $\left\| f^{-1}(i) \right\| \geq \alpha \cdot i^\beta$.

Object Complexity on Trees

Functions with small slices

1. Codes have small slices but they are not symmetric,
2. Function f_h is symmetric,
   i.e. $f_h(s_1{}^\wedge s_2) = f_h(s_2{}^\wedge s_1)$,
3. For $\gamma > 0$ exists symmetric object complexity function f, such that $\left\|f^{-1}(i)\right\| = O(i^\gamma)$,
4. For any symmetric complexity function f exist $\alpha, \beta > 0$ such that $\left\|f^{-1}(i)\right\| \geq \alpha \cdot i^\beta$.

Object Complexity on Trees

Functions with small slices

1. Codes have small slices but they are not symmetric,
2. Function f_h is symmetric,
   i.e. $f_h(s_1{}^\wedge s_2) = f_h(s_2{}^\wedge s_1)$,
3. For $\gamma > 0$ exists symmetric object complexity function f, such that $\left\|f^{-1}(i)\right\| = O(i^\gamma)$,
4. For any symmetric complexity function f exist $\alpha, \beta > 0$ such that $\left\|f^{-1}(i)\right\| \geq \alpha \cdot i^\beta$.

Counting Object Complexity Functions

Theorem (Counting Numbers)

There is exactly one complexity function on Numbers.

Theorem (Counting Words)

There is countable many symmetric $(f(uv) = f(vu))$ *non-similar object complexity functions on words.*

Theorem (Counting Trees)

There are continuum many non-similar complexity functions on trees.

Counting Object Complexity Functions

Theorem (Counting Numbers)

There is exactly one complexity function on Numbers.

Theorem (Counting Words)

There is countable many symmetric $(f(uv) = f(vu))$ *non-similar object complexity functions on words.*

Theorem (Counting Trees)

There are continuum many non-similar complexity functions on trees.

Counting Object Complexity Functions

Theorem (Counting Numbers)

There is exactly one complexity function on Numbers.

Theorem (Counting Words)

There is countable many symmetric $(f(uv) = f(vu))$ *non-similar object complexity functions on words.*

Theorem (Counting Trees)

There are continuum many non-similar complexity functions on trees.

Equivalence and Similarity

- Similarity and Equivalence put severe restrictions on the interaction of object complexity functions slices,

Two numbers:

- $Eqv(f_1, f_2)$-degree of equivalence,
- $Sim(f_1, f_2)$-degree of similarity.

Our Goal

- $f \approx g$ if and only if $Eqv(f, g) = 1$,
- $f \sim g$ if and if $Sim(f, g) = 1$.

Equivalence and Similarity

- Similarity and Equivalence put severe restrictions on the interaction of object complexity functions slices,

Two numbers:

- $Eqv(f_1, f_2)$-degree of equivalence,
- $Sim(f_1, f_2)$-degree of similarity.

Our Goal

- $f \approx g$ if and only if $Eqv(f, g) = 1$,
- $f \sim g$ if and if $Sim(f, g) = 1$.

Equivalence and Similarity

- Similarity and Equivalence put severe restrictions on the interaction of object complexity functions slices,

Two numbers:

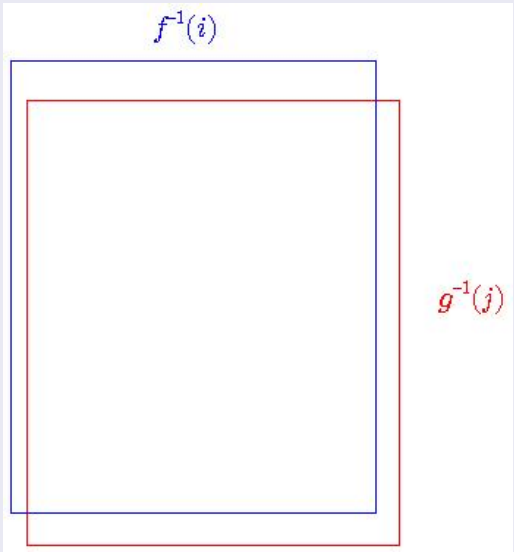
- $Eqv(f_1, f_2)$-degree of equivalence,
- $Sim(f_1, f_2)$-degree of similarity.

Our Goal

- $f \approx g$ if and only if $Eqv(f, g) = 1$,
- $f \sim g$ if and if $Sim(f, g) = 1$.

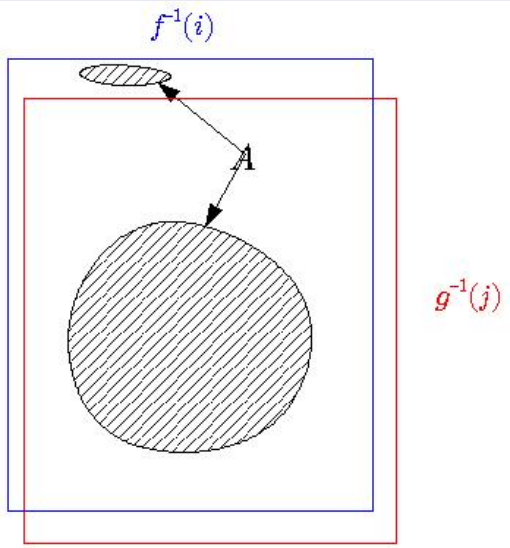
Degree of equivalence

$$Eqv(f,g) := \frac{1}{2}\left(\liminf_{i\to\infty} \max_{j} \frac{\left\| f^{-1}(i) \cap g^{-1}(j) \right\|}{\left\| f^{-1}(i) \cup g^{-1}(j) \right\|} + \liminf_{j\to\infty} \max_{i} \frac{\left\| f^{-1}(i) \cap g^{-1}(j) \right\|}{\left\| f^{-1}(i) \cup g^{-1}(j) \right\|} \right),$$



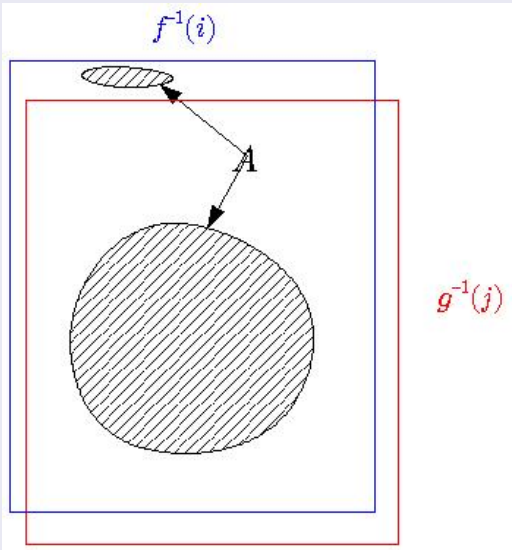
Degree of equivalence

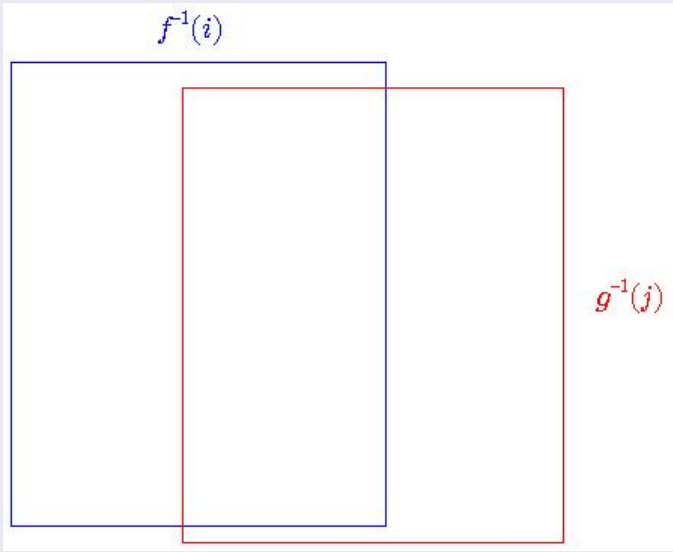
$$Eqv(f,g) := \frac{1}{2}\left(\liminf_{i\to\infty} \max_{j} \frac{\left\| f^{-1}(i) \cap g^{-1}(j) \right\|}{\left\| f^{-1}(i) \cup g^{-1}(j) \right\|} + \liminf_{j\to\infty} \max_{i} \frac{\left\| f^{-1}(i) \cap g^{-1}(j) \right\|}{\left\| f^{-1}(i) \cup g^{-1}(j) \right\|} \right),$$



Degree of equivalence

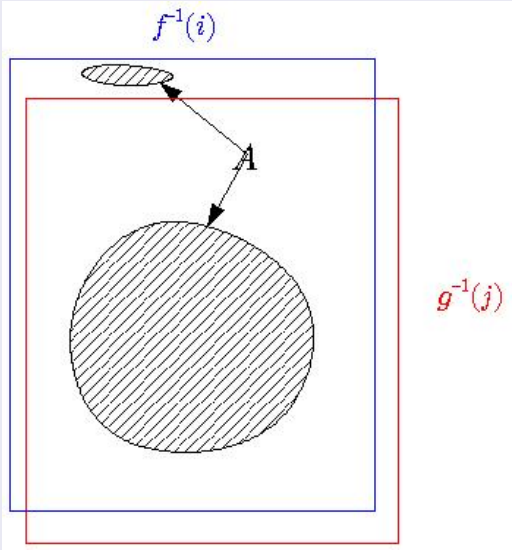
$$Eqv(f,g) := \frac{1}{2}\left(\liminf_{i\to\infty} \max_{j} \frac{\left\| f^{-1}(i) \cap g^{-1}(j) \right\|}{\left\| f^{-1}(i) \cup g^{-1}(j) \right\|} + \liminf_{j\to\infty} \max_{i} \frac{\left\| f^{-1}(i) \cap g^{-1}(j) \right\|}{\left\| f^{-1}(i) \cup g^{-1}(j) \right\|} \right),$$

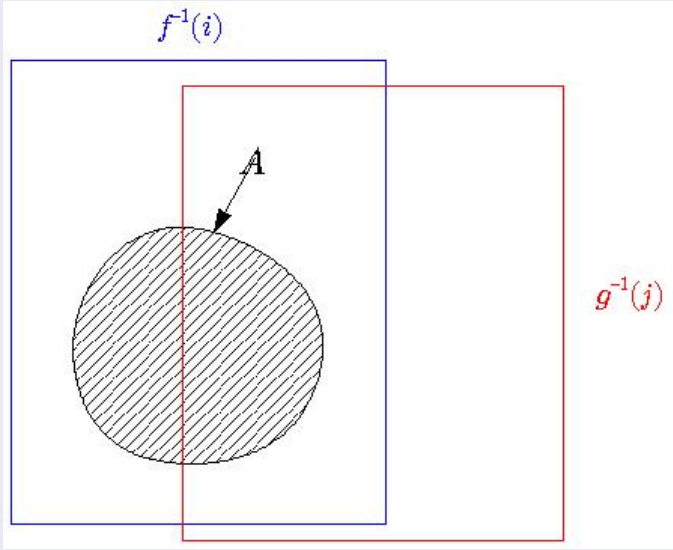




Degree of equivalence

$$Eqv(f,g) := \frac{1}{2}\left(\liminf_{i\to\infty} \max_{j} \frac{\left\| f^{-1}(i) \cap g^{-1}(j) \right\|}{\left\| f^{-1}(i) \cup g^{-1}(j) \right\|} + \liminf_{j\to\infty} \max_{i} \frac{\left\| f^{-1}(i) \cap g^{-1}(j) \right\|}{\left\| f^{-1}(i) \cup g^{-1}(j) \right\|} \right),$$





Degree of Equivalence

Theorem

The following conditions are equivalent:

1. $Eqv(f, g) = 1$,
2. $f \approx g$,
3. *f i g measure the same object sets,*
   ($\mu_f(A)$ exists if and only if $\mu_g(A)$ exists),

Degree of Equivalence

Theorem (Quick Test for Equivalence)

For two equivalent object complexity functions f and g
if $\lim_{i\to\infty} \frac{\left\|f^{-1}(i+1)\right\|}{\|f^{-1}(i)\|}$ *exists,*
then $\lim_{i\to\infty} \frac{\left\|g^{-1}(i+1)\right\|}{\|g^{-1}(i)\|}$ *also exists*
and those both limits are equal.

Example (f_c, f_h, f_e are not equivalent)

1. $\lim_{i\to\infty} \frac{\left\|f_c^{-1}(i+1)\right\|}{\|f_c^{-1}(i)\|} = 4,$
2. $\lim_{i\to\infty} \frac{\left\|f_h^{-1}(i+1)\right\|}{\|f_h^{-1}(i)\|} = \infty,$
3. $\lim_{i\to\infty} \frac{\left\|f_e^{-1}(i+1)\right\|}{\|f_e^{-1}(i)\|} \leq 2$

Degree of Equivalence

Theorem (Quick Test for Equivalence)

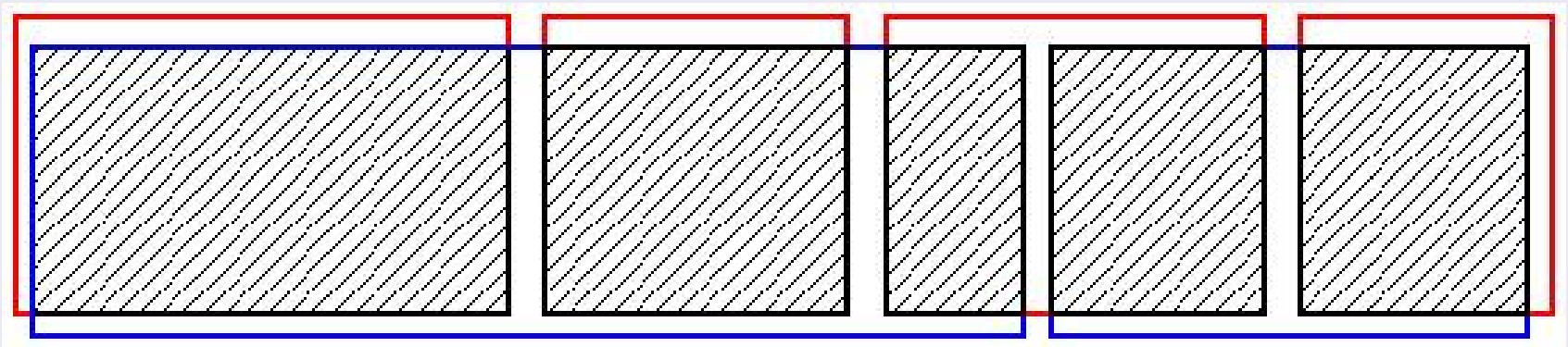
For two equivalent object complexity functions f *and* g
if $\lim_{i\to\infty} \frac{\left\|f^{-1}(i+1)\right\|}{\|f^{-1}(i)\|}$ *exists,*
then $\lim_{i\to\infty} \frac{\left\|g^{-1}(i+1)\right\|}{\|g^{-1}(i)\|}$ *also exists*
and those both limits are equal.

Example (f_c, f_h, f_e are not equivalent)

1. $\lim_{i\to\infty} \frac{\left\|f_c^{-1}(i+1)\right\|}{\left\|f_c^{-1}(i)\right\|} = 4,$
2. $\lim_{i\to\infty} \frac{\left\|f_h^{-1}(i+1)\right\|}{\left\|f_h^{-1}(i)\right\|} = \infty,$
3. $\lim_{i\to\infty} \frac{\left\|f_e^{-1}(i+1)\right\|}{\left\|f_e^{-1}(i)\right\|} \leq 2$

Degree of Similarity

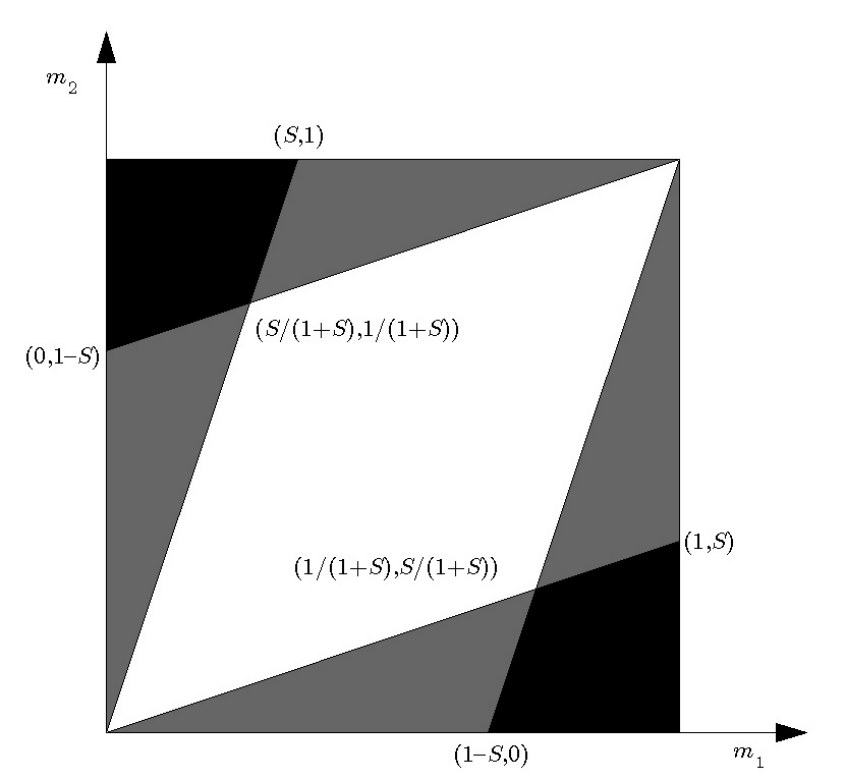
$$Sim(f,g) = \lim_{i\to\infty} \sup \left\{ \frac{\|f^{-1}(I) \cap g^{-1}(J)\|}{\|f^{-1}(I) \cup g^{-1}(J)\|} : I, J \subseteq_{fin} [i,\infty) \right\}.$$



Degree of Similarity

What describes degree of similarity?

- How much $\mu_f(A)$ and $\mu_g(A)$ can differ ,
- For $S = Sim(f, g)$ a pair of numbers $(x, y) \in [0, 1] \times [0, 1]$



1. can be always realized if it falls into the white area,
2. can not be realized if it falls into black area.

Degree of Similarity

Theorem

The following conditions are equivalent:

1. $Sim(f, g) = 1$,
2. $f \sim g$.

Degree of Similarity

Example

For two complexity functions f, g such that:

1. $\left\|f^{-1}(i)\right\| = O(2^i) = \left\|g^{-1}(i)\right\|$,
2. $f(t\hat{}\cdot) = f(\cdot\hat{}t)$ i $g(t\hat{}\cdot) = g(\cdot\hat{}t)$.

There exists $M < 1$ such that
if both $\mu_{f_1}(A)$ and $\mu_{f_2}(A)$ exist, then $|\mu_{f_1}(A) - \mu_{f_2}(A)| < M$.

Example

Functions

f_h - measures the hight of tree,
$f_e(s_1 \hat{} s_2) = f_e(s_1)^{f_e(s_2)} + f_e(s_2)^{f_e(s_1)}$

are "completely non-similar" i.e. $Sim(f, g) = 0$. In particular $\mu_{f_h}(A) = 1$ i $\mu_{f_e}(A) = 0$ for some A.