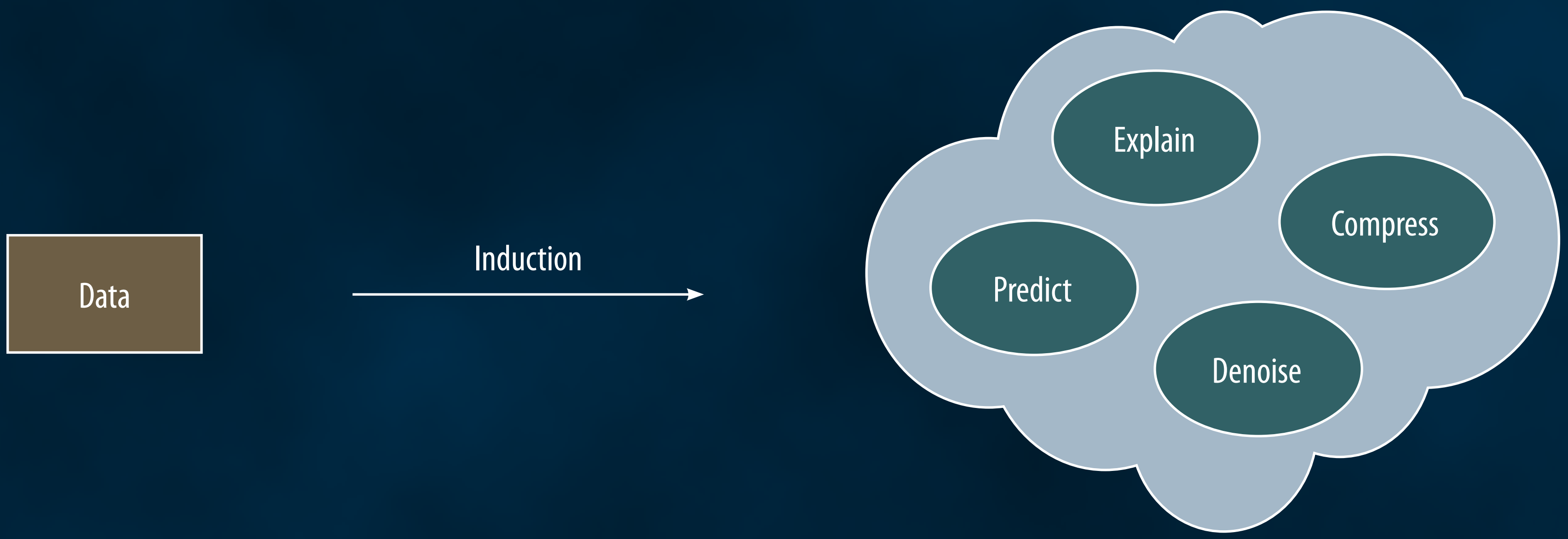
Minimum Message Length and Kolmogorov Complexity

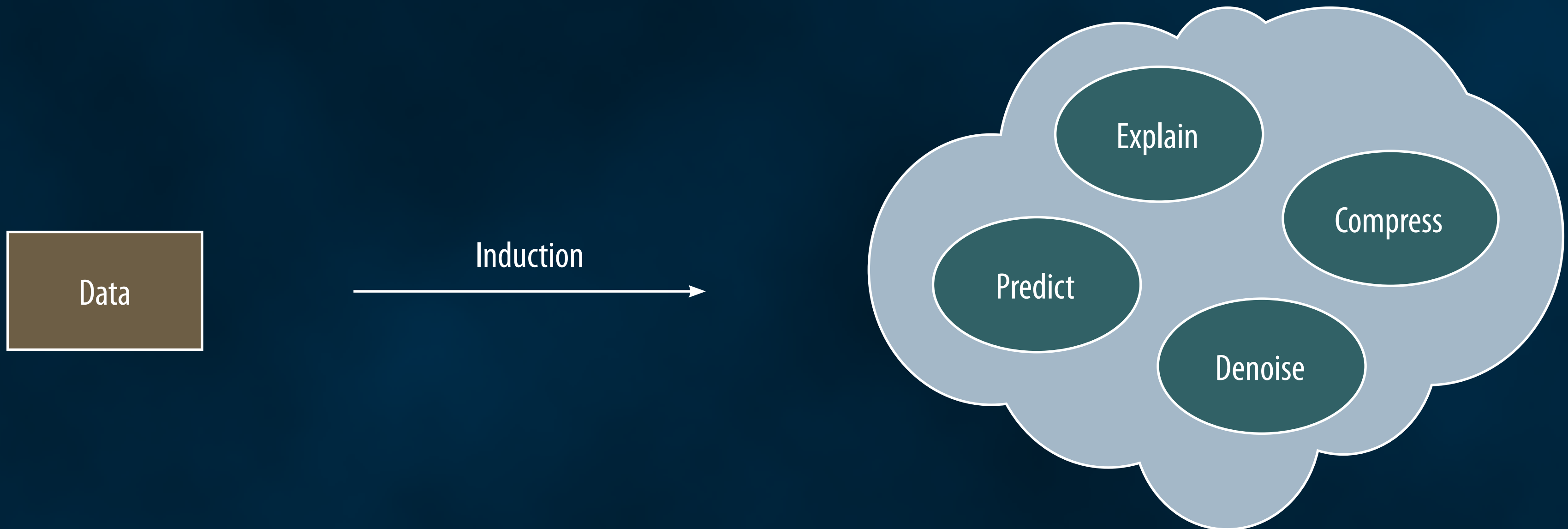
C. S. Wallace and D. L. Dowe

Presentation by Alkis Gkotovos

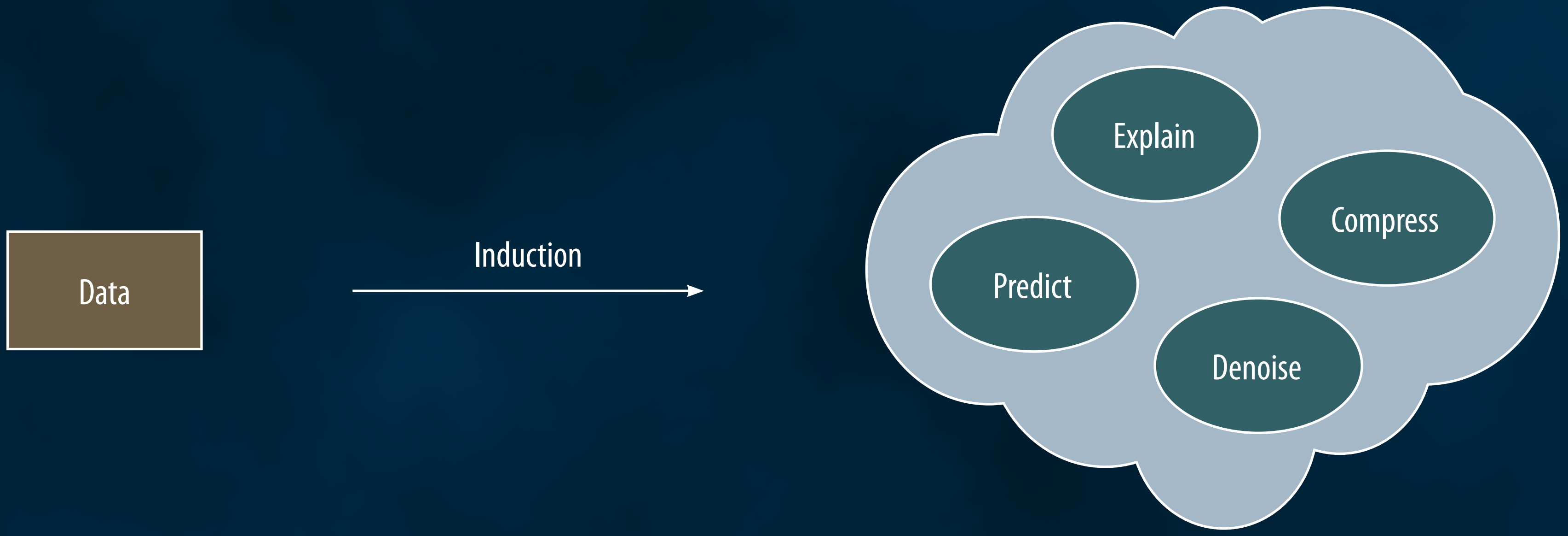
Data


Data
Explain
Compress
Predict
Denoise

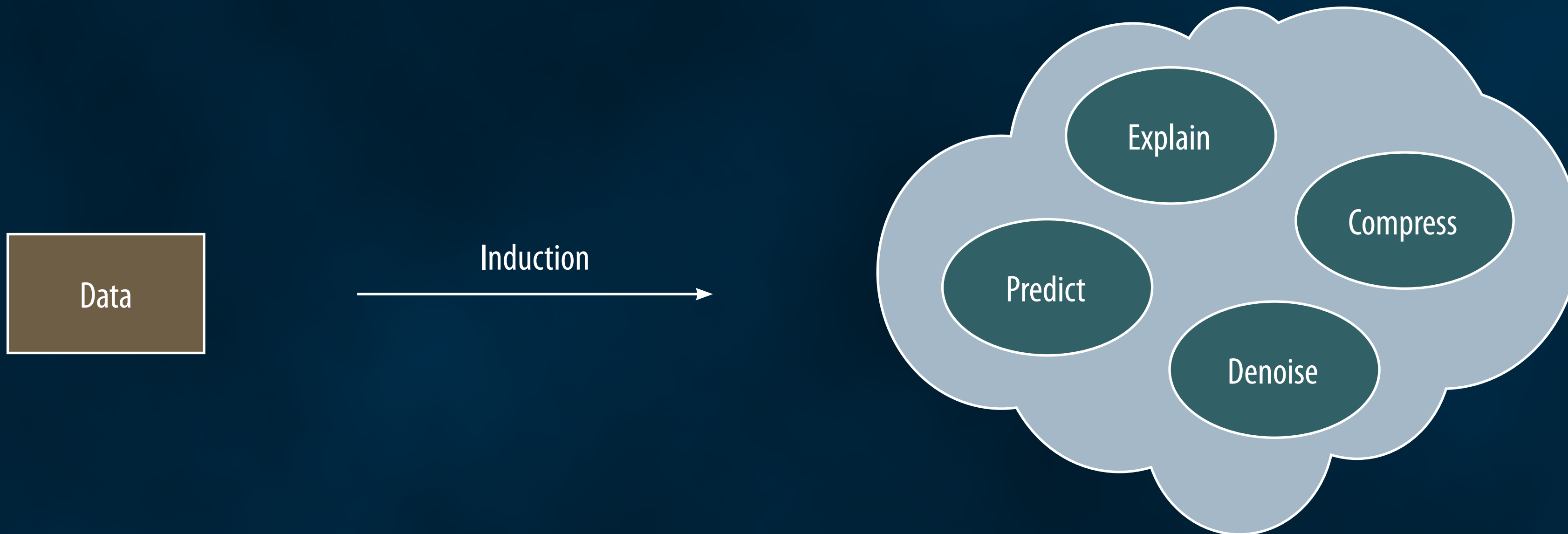

Data
Induction
Explain
Compress
Predict
Denoise



- Use some principle to guide inductive process, e.g. Occam's razor (bias towards "simplicity")



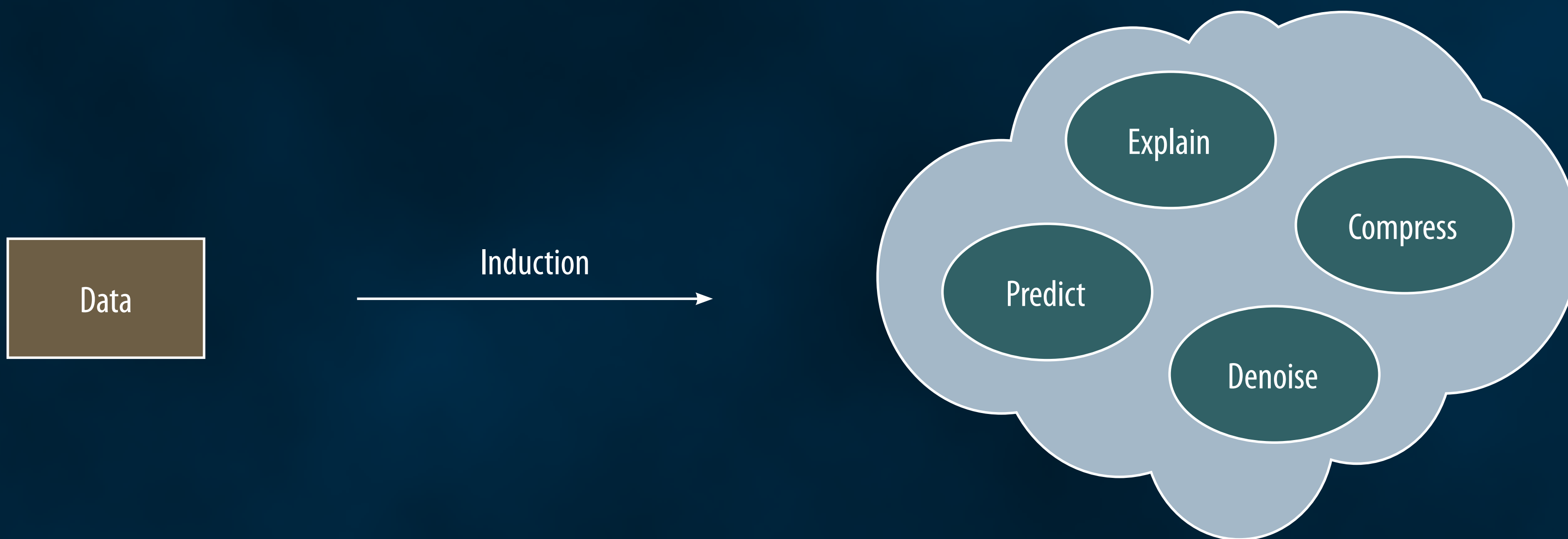
- Use some principle to guide inductive process, e.g. Occam's razor (bias towards "simplicity")
- How do we quantify complexity?
  - Probability theory (e.g. AIC, BIC)
  - Information theory (e.g. MDL, MML)



- Use some principle to guide inductive process, e.g. Occam's razor (bias towards "simplicity")

- How do we quantify complexity?
  - Probability theory (e.g. AIC, BIC)
  - Information theory (e.g. MDL, MML)

  intimately related (probability $\longleftrightarrow$ codeword length)



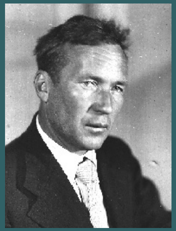
- Use some principle to guide inductive process, e.g. Occam's razor (bias towards "simplicity")

- How do we quantify complexity?
  - Probability theory (e.g. AIC, BIC)
  - Information theory (e.g. MDL, MML)
  - Algorithmic complexity

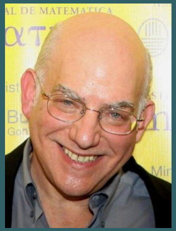
intimately related (probability $\longleftrightarrow$ codeword length)

Kolmogorov complexity

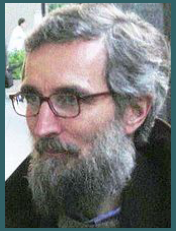
Quantify complexity of binary strings via Turing Machines (early '60s)



A. Kolmogorov



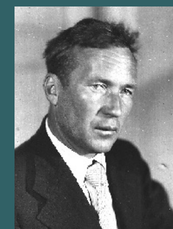
G. Chaitin



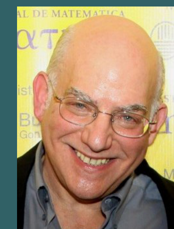
P. Martin-Löf

Kolmogorov complexity

Quantify complexity of binary strings via Turing Machines (early '60s)



A. Kolmogorov



G. Chaitin

P. Martin-Löf

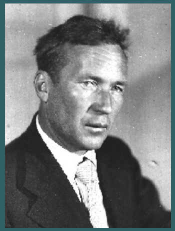
Universal induction

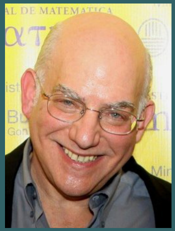
Define algorithmic probability via Turing Machines and use it for induction (early '60s)

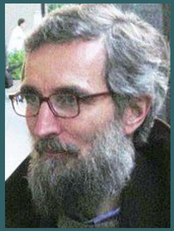
R. Solomonoff

Kolmogorov complexity

Quantify complexity of binary strings via Turing Machines (early '60s)


A. Kolmogorov


G. Chaitin


P. Martin-Löf

Universal induction

Define algorithmic probability via Turing Machines and use it for induction (early '60s)


R. Solomonoff

MML/MDL

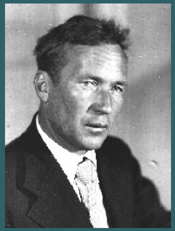
Infer a hypothesis about the data via two-part coding (late '60s and '70s)

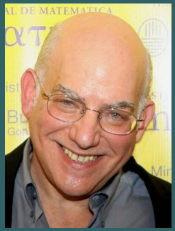

C. Wallace

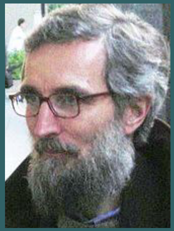

J. Rissanen

Kolmogorov complexity

Quantify complexity of binary strings via Turing Machines (early '60s)


A. Kolmogorov


G. Chaitin


P. Martin-Löf

Universal induction

Define algorithmic probability via Turing Machines and use it for induction (early '60s)


R. Solomonoff

MML/MDL

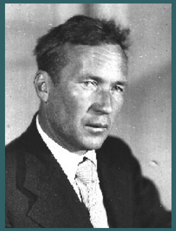
Infer a hypothesis about the data via two-part coding (late '60s and '70s)

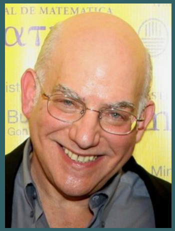

C. Wallace

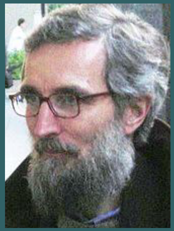

J. Rissanen

<u>Kolmogorov complexity</u>

Quantify complexity of binary strings via Turing Machines (early '60s)


A. Kolmogorov


G. Chaitin


P. Martin-Löf

?

<u>Universal induction</u>

Define algorithmic probability via Turing Machines and use it for induction (early '60s)


R. Solomonoff

<u>MML/MDL</u>

Infer a hypothesis about the data via two-part coding (late '60s and '70s)


C. Wallace

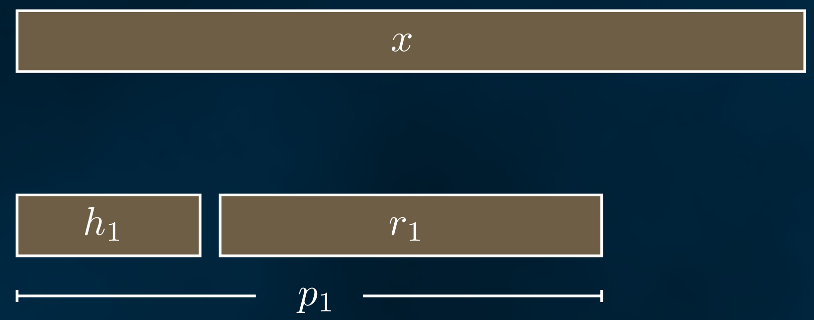

J. Rissanen

Data string

$\boxed{x}$

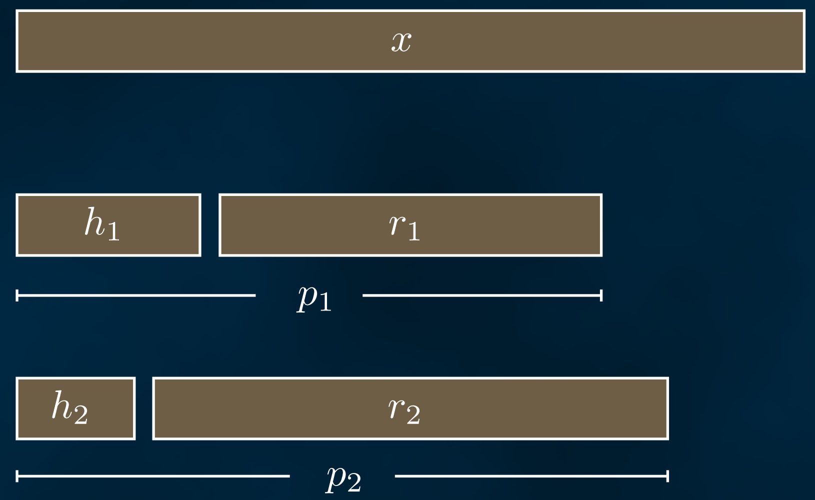
Data string

Encode x using a *two-part* scheme



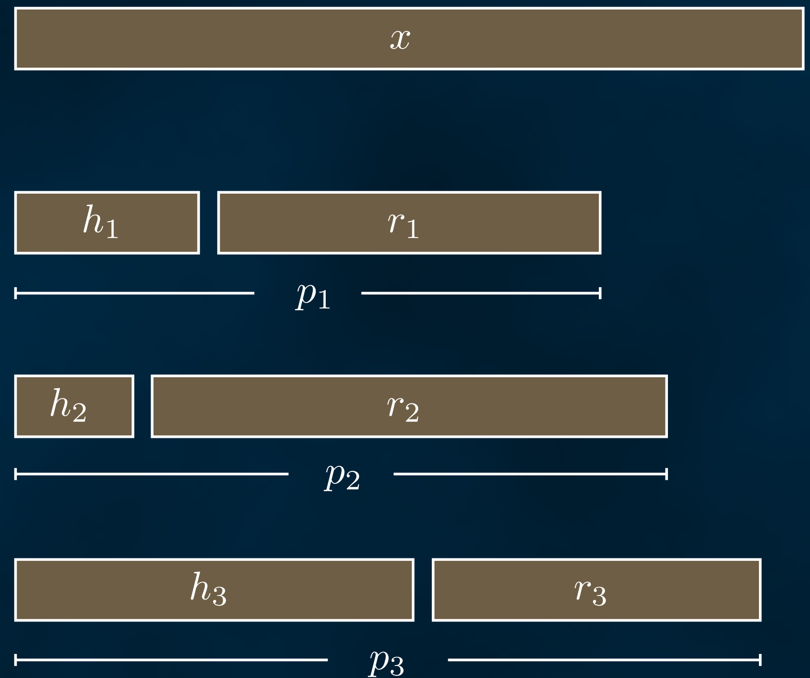
Data string

Encode x using a *two-part* scheme



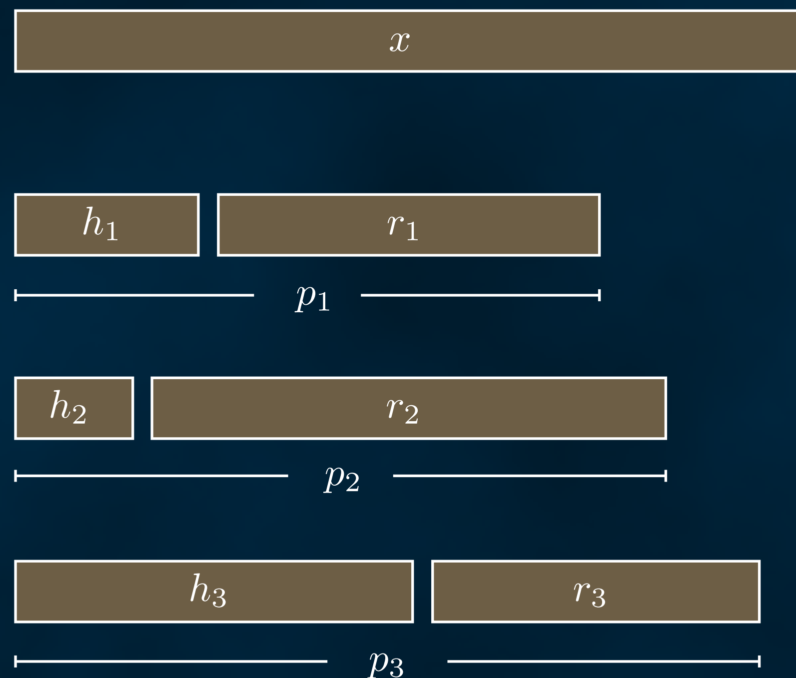
Data string

Encode x using a *two-part* scheme



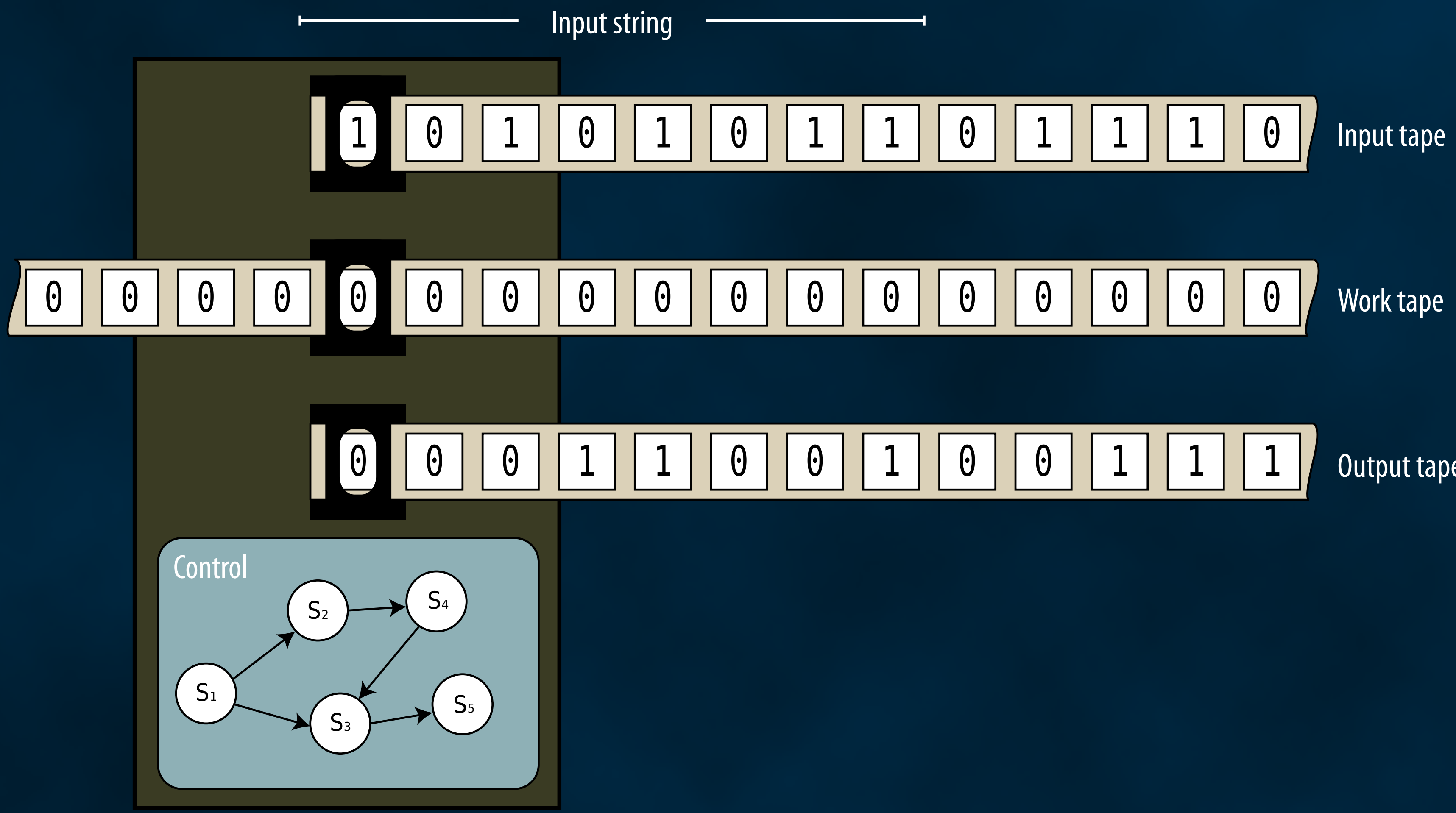
Data string

Encode x using a *two-part* scheme

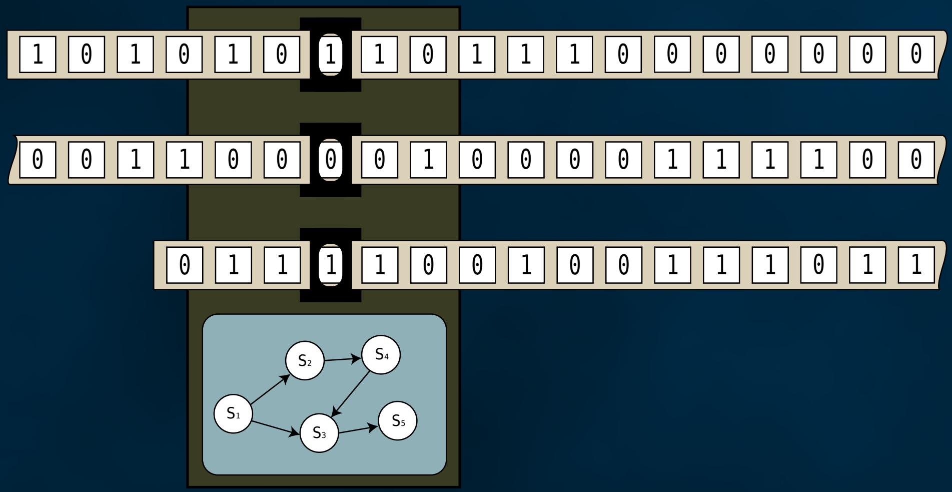


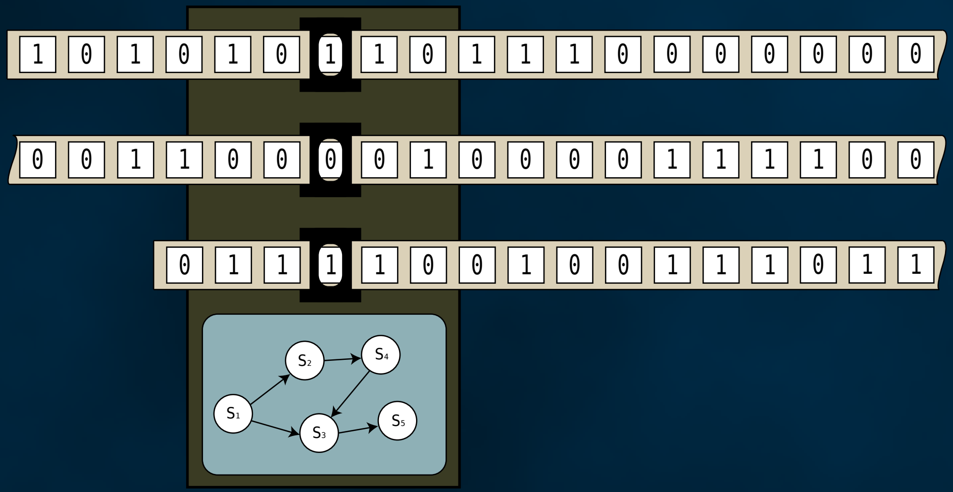
Pick the hypothesis that results in the minimum encoding length

$$l(p_i) = l(h_i) + l(r_i) = -\log_2(p_H(h_i)) - \log_2(p_X(x \mid h_i))$$


Input string
1 0 1 0 1 0 1 1 0 1 1 1 0
Input tape
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Work tape
0 0 0 1 1 0 0 1 0 0 1 1 1
Output tape
Control
S1
S2
S3
S4
S5

$T(p) = x$ if


1 0 1 0 1 0 1 1 0 1 1 1 0 0 0 0 0 0 0
0 0 1 1 0 0 0 0 1 0 0 0 0 1 1 1 1 0 0
0 1 1 1 1 0 0 1 0 0 1 1 1 0 1 1
S1
S2
S3
S4
S5

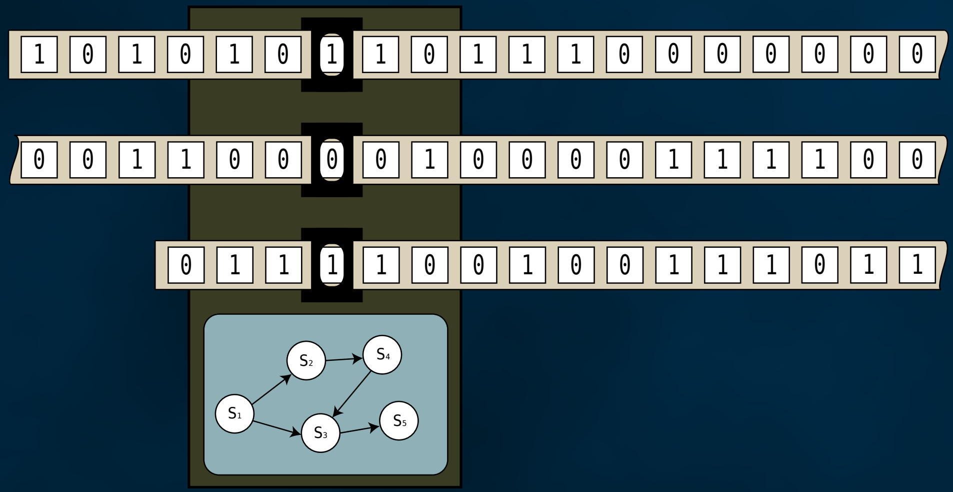


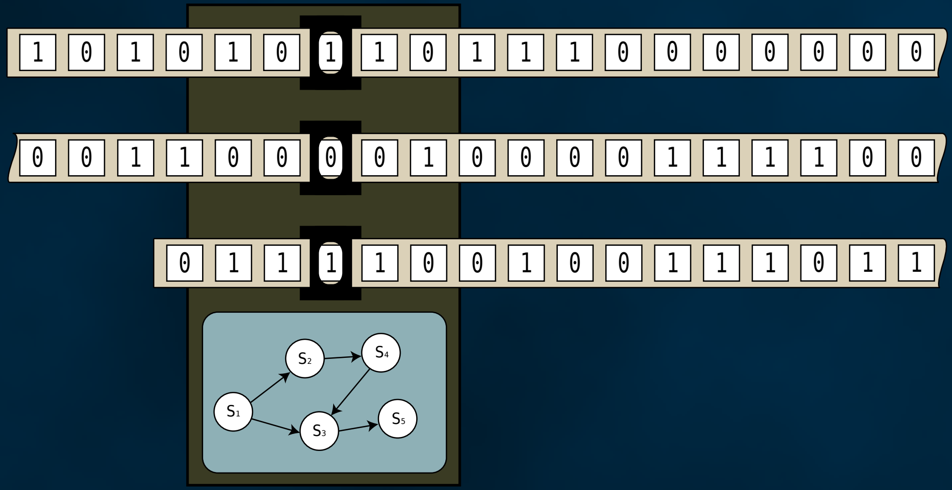
$T(p) = x$ if

- T reads all of the input p

$T(p) = x$ if

- T reads all of the input p
- T writes x to the output



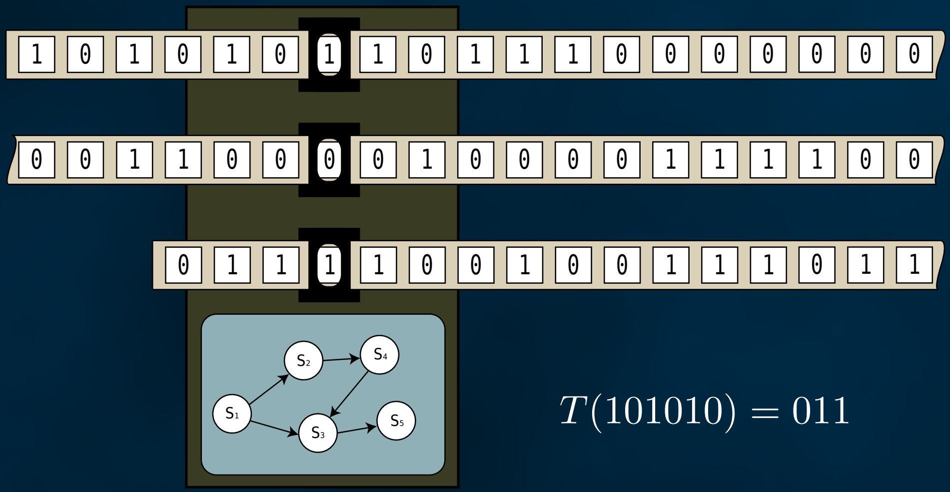


$T(p) = x$ if

- T reads all of the input p
- T writes x to the output
- T halts without reading/writing anything else

$T(p) = x$ if

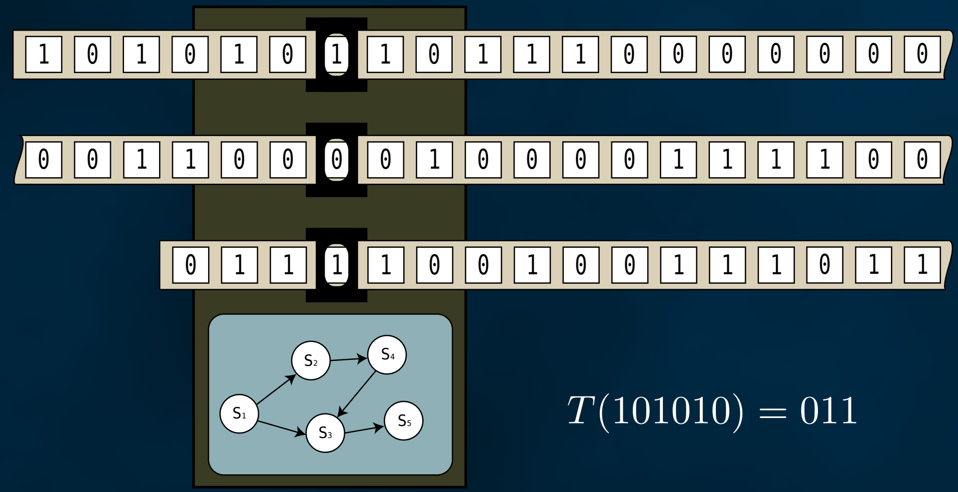
- T reads all of the input p
- T writes x to the output
- T halts without reading/writing anything else



$T(101010) = 011$

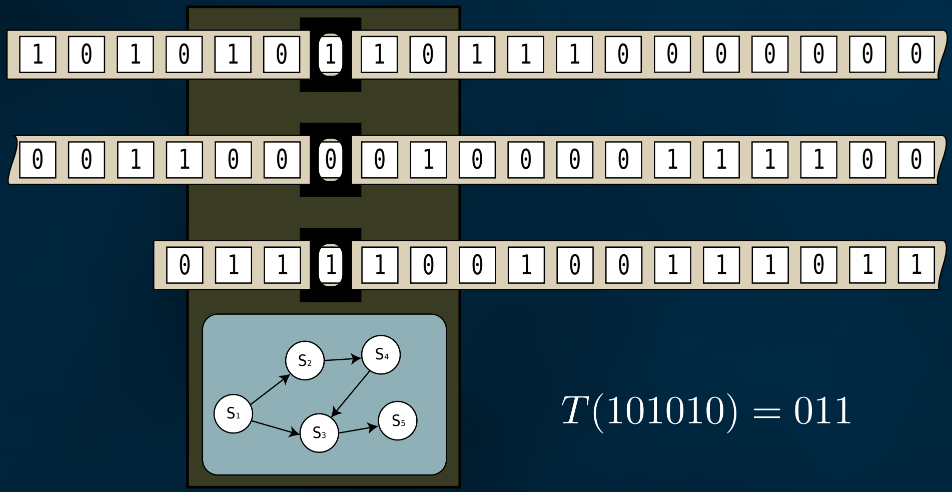
$T(p) = x$ if

- T reads all of the input p
- T writes x to the output
- T halts without reading/writing anything else



$T(101010) = 011$

T is a decoder of a prefix code (why prefix?)



$T(p) = x$ if

- T reads all of the input p
- T writes x to the output
- T halts without reading/writing anything else

$T(101010) = 011$

T is a decoder of a prefix code (why prefix?)

Universal (prefix) TM = TM that can emulate any other (prefix) TM, e.g. $T(\langle i, p \rangle) = T_i(p)$

(Prefix) Kolmogorov complexity of x $=$ Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

(Prefix) Kolmogorov complexity of x $=$ Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

(Prefix) Kolmogorov complexity of x = Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

- Invariance theorem: Choose a *universal* prefix TM (among a special class) $\longrightarrow$ as good as any TM (up to a constant)

(Prefix) Kolmogorov complexity of x $=$ Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

- Invariance theorem: Choose a *universal* prefix TM (among a special class) $\longrightarrow$ as good as any TM (up to a constant)
- Intuition: You can do (almost) as good as any TM by writing a "compiler" for it

(Prefix) Kolmogorov complexity of x $=$ Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

- Invariance theorem: Choose a *universal* prefix TM (among a special class) $\longrightarrow$ as good as any TM (up to a constant)
- Intuition: You can do (almost) as good as any TM by writing a "compiler" for it
- Caveat 1: Kolmogorov complexity is not computable (approximate, e.g. via compressor)

(Prefix) Kolmogorov complexity of x $=$ Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

- Invariance theorem: Choose a *universal* prefix TM (among a special class) $\longrightarrow$ as good as any TM (up to a constant)
- Intuition: You can do (almost) as good as any TM by writing a "compiler" for it
- Caveat 1: Kolmogorov complexity is not computable (approximate, e.g. via compressor)
- Caveat 2: Constants can be large (more on this later)

(Prefix) Kolmogorov complexity of x = Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

- Invariance theorem: Choose a *universal* prefix TM (among a special class) $\longrightarrow$ as good as any TM (up to a constant)
- Intuition: You can do (almost) as good as any TM by writing a "compiler" for it
- Caveat 1: Kolmogorov complexity is not computable (approximate, e.g. via compressor)
- Caveat 2: Constants can be large (more on this later)

Can define a "probability" measure from Kolmogorov complexity

$$P_T(x) = 2^{-K_T(x)}$$

(Prefix) Kolmogorov complexity of x $=$ Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

- Invariance theorem: Choose a *universal* prefix TM (among a special class) $\longrightarrow$ as good as any TM (up to a constant)
- Intuition: You can do (almost) as good as any TM by writing a "compiler" for it
- Caveat 1: Kolmogorov complexity is not computable (approximate, e.g. via compressor)
- Caveat 2: Constants can be large (more on this later)

Can define a "probability" measure from Kolmogorov complexity

$$P_T(x) = 2^{-K_T(x)} \qquad \text{with} \quad \sum_x P_T(x) \leq 1 \qquad \text{(why?)}$$

(Prefix) Kolmogorov complexity of x = Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

- Invariance theorem: Choose a *universal* prefix TM (among a special class) $\longrightarrow$ as good as any TM (up to a constant)
- Intuition: You can do (almost) as good as any TM by writing a "compiler" for it
- Caveat 1: Kolmogorov complexity is not computable (approximate, e.g. via compressor)
- Caveat 2: Constants can be large (more on this later)

Can define a "probability" measure from Kolmogorov complexity

$$P_T(x) = 2^{-K_T(x)} \qquad \text{with} \quad \sum_x P_T(x) \leq 1 \qquad \text{(why?)}$$

Why don't we normalize?

(Prefix) Kolmogorov complexity of x = Length of the shortest input string required to output x

$$K_T(x) = \min\{l(p) \mid T(p) = x\}$$

Definition dependent on T! Does this make any sense?

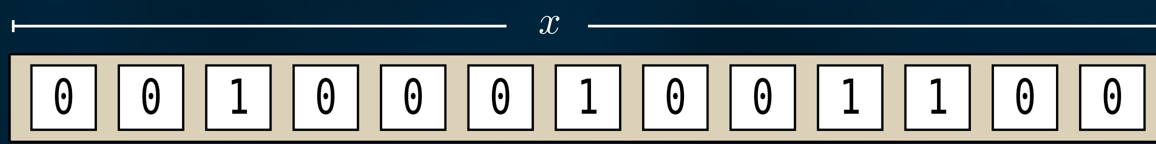
- Invariance theorem: Choose a *universal* prefix TM (among a special class) $\longrightarrow$ as good as any TM (up to a constant)
- Intuition: You can do (almost) as good as any TM by writing a "compiler" for it
- Caveat 1: Kolmogorov complexity is not computable (approximate, e.g. via compressor)
- Caveat 2: Constants can be large (more on this later)

Can define a "probability" measure from Kolmogorov complexity

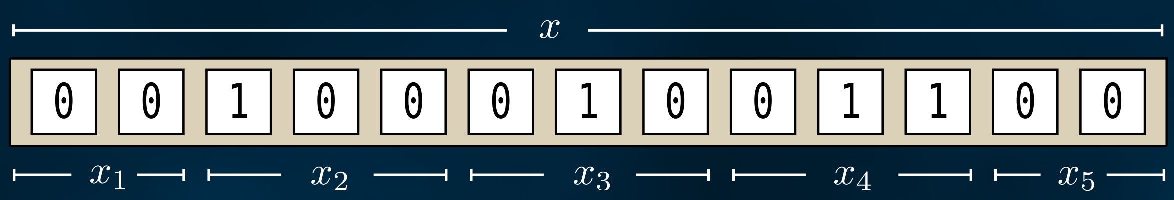
$$P_T(x) = 2^{-K_T(x)} \qquad \text{with} \qquad \sum_x P_T(x) \leq 1 \qquad \text{(why?)}$$

Why don't we normalize?
(Hereafter: "probability" $\longleftrightarrow$ semimeasure)

Data string x is a representation of observational data from a real world phenomenon



Data string x is a representation of observational data from a real world phenomenon



$$L = \{00, 100, 010, 011\}$$

- "Sentences" $x_i \in L$, where L is a prefix-free set (data "language")

Data string x is a representation of observational data from a real world phenomenon

$$\overbrace{\underbrace{\texttt{0}\;\texttt{0}}_{x_1}\;\underbrace{\texttt{1}\;\texttt{0}\;\texttt{0}}_{x_2}\;\underbrace{\texttt{0}\;\texttt{1}\;\texttt{0}}_{x_3}\;\underbrace{\texttt{0}\;\texttt{1}\;\texttt{1}}_{x_4}\;\underbrace{\texttt{0}\;\texttt{0}}_{x_5}}^{x}$$

$$L = \{00, 100, 010, 011\}$$

- "Sentences" $x_i \in L$, where L is a prefix-free set (data "language")
- Distinct sentences represent distinct real-world facts

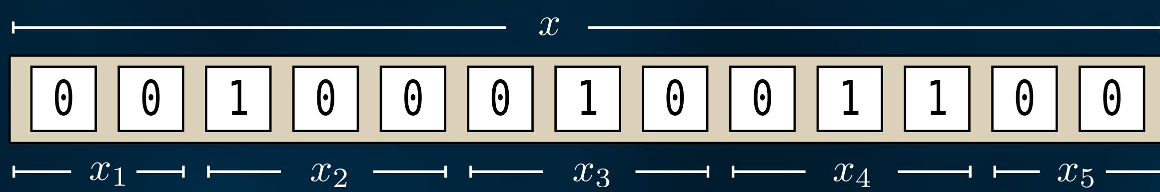
Data string x is a representation of observational data from a real world phenomenon

$$\overbrace{\underbrace{0\;0}_{x_1}\;\underbrace{1\;0\;0}_{x_2}\;\underbrace{0\;1\;0}_{x_3}\;\underbrace{0\;1\;1}_{x_4}\;\underbrace{0\;0}_{x_5}}^{x}$$

$$L = \{00, 100, 010, 011\}$$

- "Sentences" $x_i \in L$, where L is a prefix-free set (data "language")
- Distinct sentences represent distinct real-world facts
- Sentences are conditionally independent given full knowledge of the phenomenon

Data string x is a representation of observational data from a real world phenomenon



$$L = \{00, 100, 010, 011\}$$

- “Sentences” $x_i \in L$, where L is a prefix-free set (data “language”)
- Distinct sentences represent distinct real-world facts
- Sentences are conditionally independent given full knowledge of the phenomenon
- Strings are invariant to sentence permutation

Hypothesis Q is a (computable) probability distribution over L

Hypothesis Q is a (computable) probability distribution over L

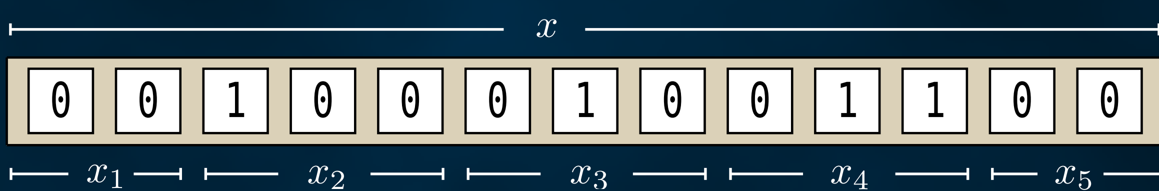
Conditional independence of sentences implies

$$x = x_1 \ldots x_n \Rightarrow Q(x) = Q(x_1) \times \ldots \times Q(x_n)$$

Hypothesis Q is a (computable) probability distribution over L

Conditional independence of sentences implies

$$x = x_1 \dots x_n \Rightarrow Q(x) = Q(x_1) \times \dots \times Q(x_n)$$

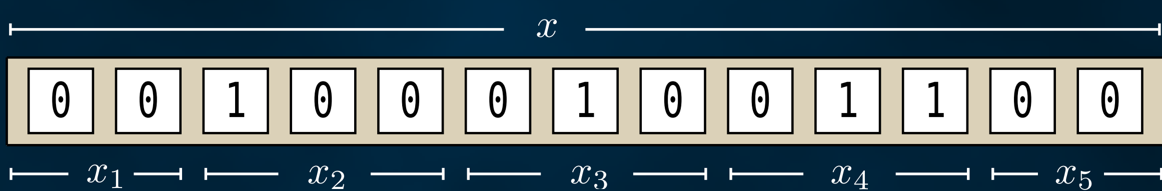


$$L = \{00, 100, 010, 011\}$$

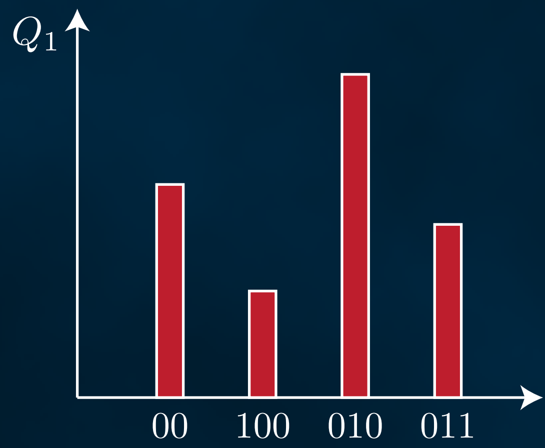
Hypothesis Q is a (computable) probability distribution over L

Conditional independence of sentences implies

$$x = x_1 \ldots x_n \Rightarrow Q(x) = Q(x_1) \times \ldots \times Q(x_n)$$



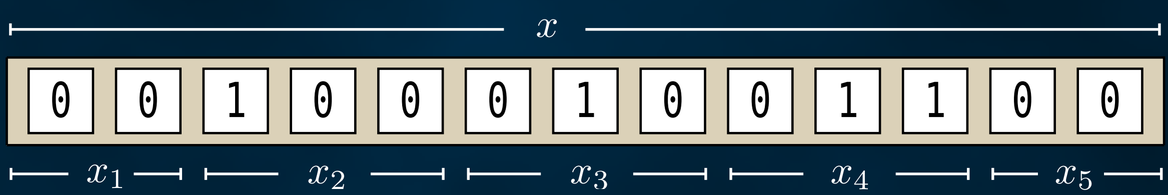
$$L = \{00, 100, 010, 011\}$$



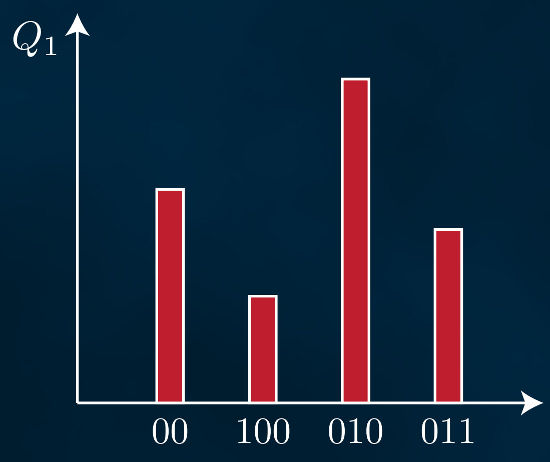
Hypothesis Q is a (computable) probability distribution over L

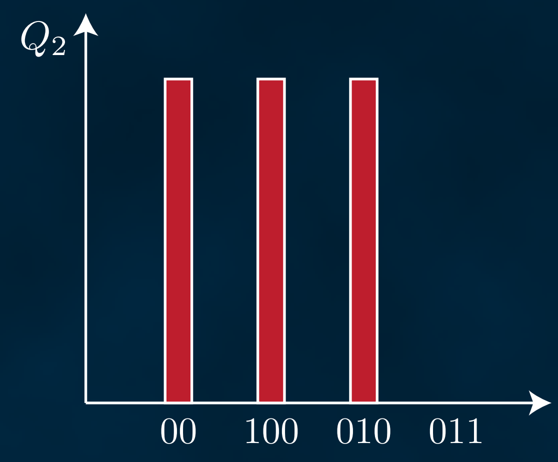
Conditional independence of sentences implies

$$x = x_1 \dots x_n \Rightarrow Q(x) = Q(x_1) \times \dots \times Q(x_n)$$



$$L = \{00, 100, 010, 011\}$$





How do we acquire a hypothesis-based encoding of data in the Algorithmic Complexity framework?

How do we acquire a hypothesis-based encoding of data in the Algorithmic Complexity framework?

Idea

- Use conditional Kolmogorov complexity

$$K_T(x \mid y) = \min\{l(p) \mid T(\langle y, p \rangle) = x\}$$

and interpret y as hypothesis and x as data

How do we acquire a hypothesis-based encoding of data in the Algorithmic Complexity framework?

Idea

- Use conditional Kolmogorov complexity

$$K_T(x \mid y) = \min\{l(p) \mid T(\langle y, p \rangle) = x\}$$

  and interpret y as hypothesis and x as data

- Corresponding conditional algorithmic probability

$$P_T(x \mid y) = 2^{-K_T(x|y)}$$

How do we acquire a hypothesis-based encoding of data in the Algorithmic Complexity framework?

Idea

- Use conditional Kolmogorov complexity

$$K_T(x \mid y) = \min\{l(p) \mid T(\langle y, p \rangle) = x\}$$

  and interpret y as hypothesis and x as data

- Corresponding conditional algorithmic probability

$$P_T(x \mid y) = 2^{-K_T(x|y)}$$

Problem

Probability can never be 0, i.e. Popper-falsification not possible, because

$$K_T(x \mid y) < K_T(x) + O(1) \Rightarrow P_T(x \mid y) > P_T(x) + O(1)$$

How do we acquire a hypothesis-based encoding of data in the Algorithmic Complexity framework?

Idea

- Use conditional Kolmogorov complexity

$$K_T(x \mid y) = \min\{l(p) \mid T(\langle y, p \rangle) = x\}$$

and interpret y as hypothesis and x as data

- Corresponding conditional algorithmic probability

$$P_T(x \mid y) = 2^{-K_T(x|y)}$$

Problem

Probability can never be 0, i.e. Popper-falsification not possible, because

$$K_T(x \mid y) < K_T(x) + O(1) \Rightarrow P_T(x \mid y) > P_T(x) + O(1)$$

Why?

How do we acquire a hypothesis-based encoding of data in the Algorithmic Complexity framework?

Idea

- Use conditional Kolmogorov complexity

$$K_T(x \mid y) = \min\{l(p) \mid T(\langle y, p\rangle) = x\}$$

and interpret y as hypothesis and x as data

- Corresponding conditional algorithmic probability

$$P_T(x \mid y) = 2^{-K_T(x|y)}$$

Problem

Probability can never be 0, i.e. Popper-falsification not possible, because

$$K_T(x \mid y) < K_T(x) + O(1) \Rightarrow P_T(x \mid y) > P_T(x) + O(1)$$

Why? Hypothesis y acts as “extra info”, instead of assertively

How do we acquire a hypothesis-based encoding of data in the Algorithmic Complexity framework?

Idea

- Use conditional Kolmogorov complexity

$$K_T(x \mid y) = \min\{l(p) \mid T(\langle y, p\rangle) = x\}$$

and interpret y as hypothesis and x as data

- Corresponding conditional algorithmic probability

$$P_T(x \mid y) = 2^{-K_T(x|y)}$$

Problem

Probability can never be 0, i.e. Popper-falsification not possible, because

$$K_T(x \mid y) < K_T(x) + O(1) \Rightarrow P_T(x \mid y) > P_T(x) + O(1)$$

Why? Hypothesis y acts as "extra info", instead of assertively

Proposal

- Have hypothesis be a prefix of input string p

How do we acquire a hypothesis-based encoding of data in the Algorithmic Complexity framework?

Idea

- Use conditional Kolmogorov complexity

$$K_T(x \mid y) = \min\{l(p) \mid T(\langle y, p\rangle) = x\}$$

  and interpret y as hypothesis and x as data

- Corresponding conditional algorithmic probability

$$P_T(x \mid y) = 2^{-K_T(x|y)}$$

Problem

Probability can never be 0, i.e. Popper-falsification not possible, because

$$K_T(x \mid y) < K_T(x) + O(1) \Rightarrow P_T(x \mid y) > P_T(x) + O(1)$$

Why? Hypothesis y acts as “extra info”, instead of assertively

Proposal

- Have hypothesis be a prefix of input string p
- Force intended two-part encoding by imposing conditions on p

Input p is an acceptable MML message encoding data string x, if

Input p is an acceptable MML message encoding data string x, if

1) $T(p) = x$ p encodes x

Input p is an acceptable MML message encoding data string x, if

1) $T(p) = x$ — p encodes x

2) $l(p) < l(x)$ — some compression is achieved

Input p is an acceptable MML message encoding data string x, if

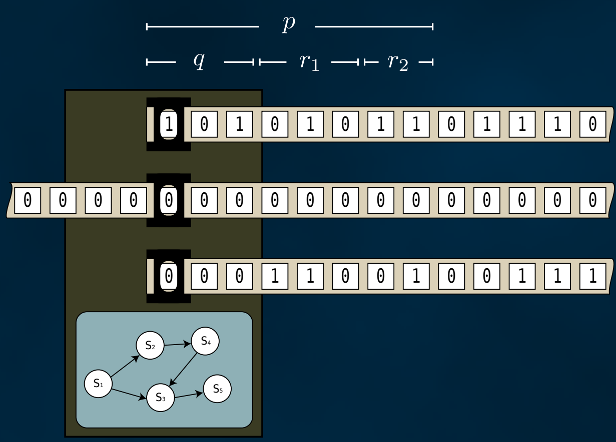
1) $T(p) = x$ — p encodes x

2) $l(p) < l(x)$ — some compression is achieved

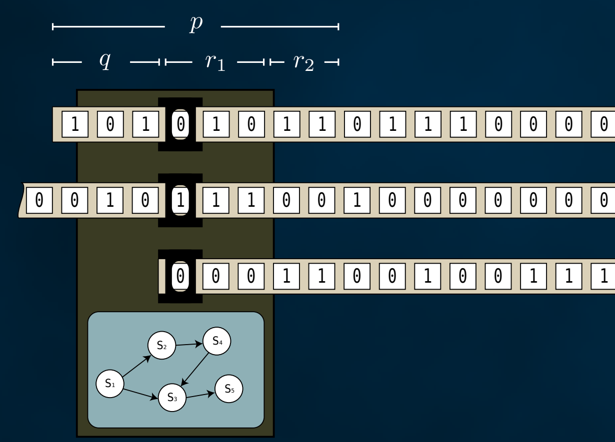
3) $p = qr$ — two-part encoding

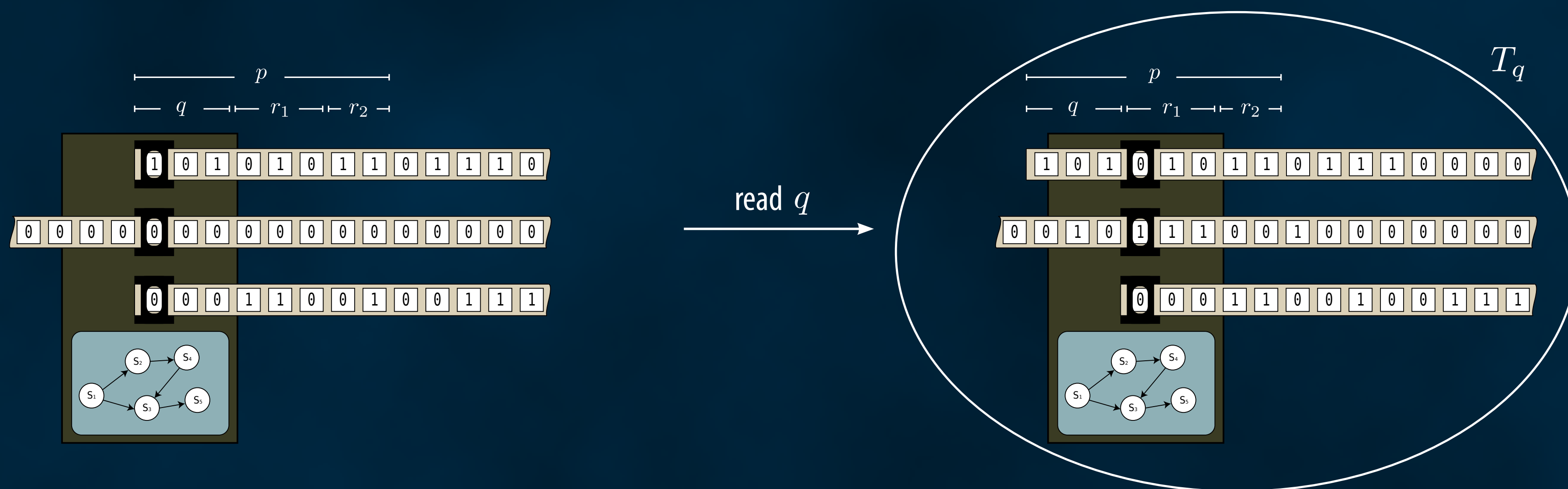
Input p is an acceptable MML message encoding data string x, if

1)	$T(p) = x$	p encodes x
2)	$l(p) < l(x)$	some compression is achieved
3)	$p = qr$	two-part encoding
4)	$T(q) = \epsilon$	hypothesis q does not determine data



read q $\longrightarrow$




p
q
r1
r2
S1
S2
S3
S4
S5
read q
Tq

Input p is an acceptable MML message encoding data string x, if

1) $T(p) = x$ — p encodes x

2) $l(p) < l(x)$ — some compression is achieved

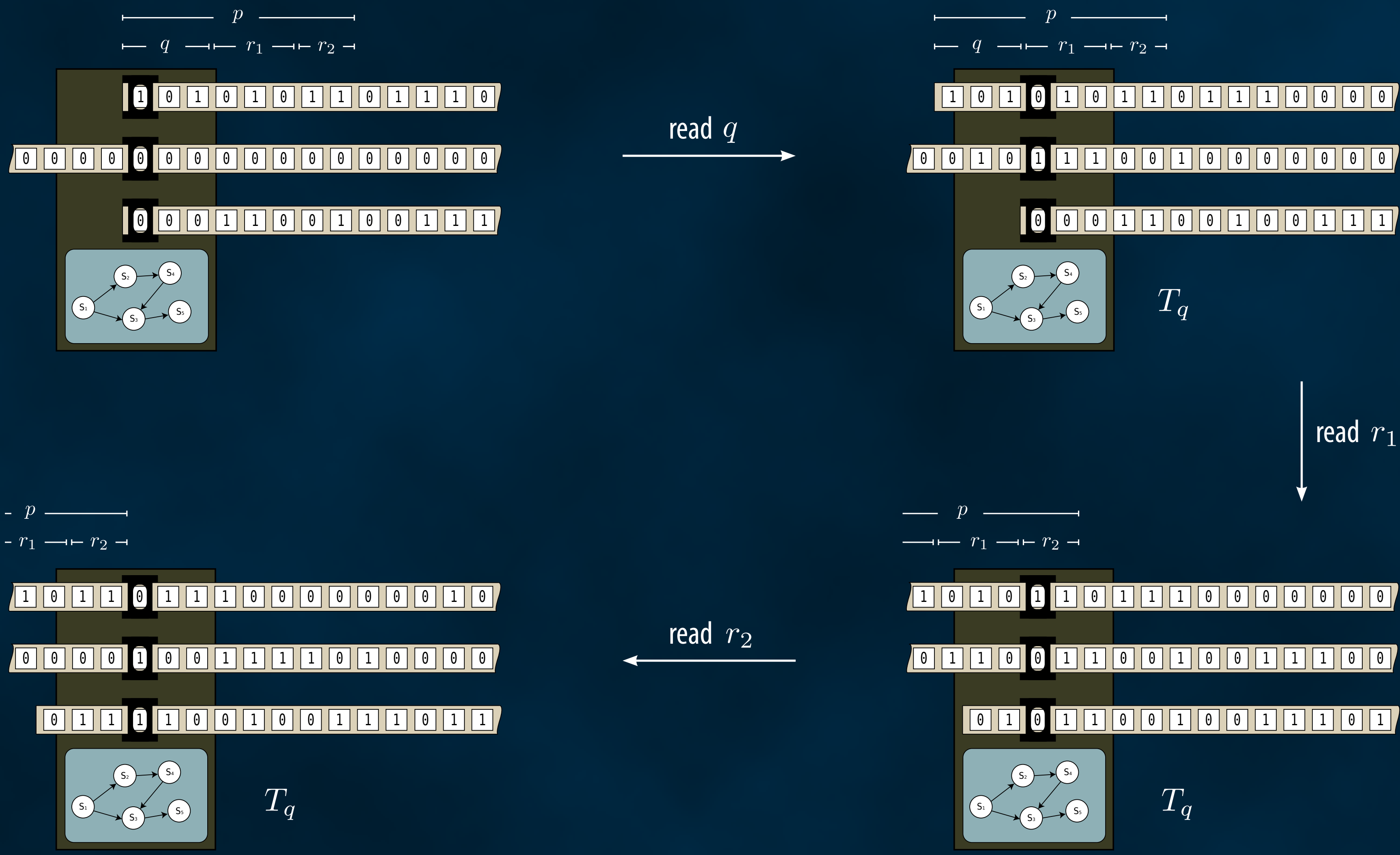
3) $p = qr$ — two-part encoding

4) $T(q) = \epsilon$ — hypothesis q does not determine data

5) $T_q(rs) = xT_q(s)$ — reading r does not alter the state of T

Input p is an acceptable MML message encoding data string x, if

1)	$T(p) = x$	p encodes x
2)	$l(p) < l(x)$	some compression is achieved
3)	$p = qr$	two-part encoding
4)	$T(q) = \epsilon$	hypothesis q does not determine data
5)	$T_q(rs) = xT_q(s)$	reading r does not alter the state of T
6)	$x = x_1 \ldots x_n \Rightarrow \begin{cases} r = r_1 \ldots r_n \\ T_q(r_i) = x_i, \; i = 1 \ldots n \end{cases}$	conditionally independent sentences


p
q
r1
r2
read q
Tq
read r1
read r2
S1
S2
S3
S4
S5

Input p is an acceptable MML message encoding data string x, if

1) $T(p) = x$ — p encodes x

2) $l(p) < l(x)$ — some compression is achieved

3) $p = qr$ — two-part encoding

4) $T(q) = \epsilon$ — hypothesis q does not determine data

5) $T_q(rs) = xT_q(s)$ — reading r does not alter the state of T

6) $x = x_1 \ldots x_n \Rightarrow \begin{cases} r = r_1 \ldots r_n \\ T_q(r_i) = x_i, \; i = 1 \ldots n \end{cases}$ — conditionally independent sentences

7) $l(r) < K_T(x)$ — hypothesis q is "significant"

Input p is an acceptable MML message encoding data string x, if

1)	$T(p) = x$	p encodes x
2)	$l(p) < l(x)$	some compression is achieved
3)	$p = qr$	two-part encoding
4)	$T(q) = \epsilon$	hypothesis q does not determine data
5)	$T_q(rs) = xT_q(s)$	reading r does not alter the state of T
6)	$x = x_1 \ldots x_n \Rightarrow \begin{cases} r = r_1 \ldots r_n \\ T_q(r_i) = x_i, \ i = 1 \ldots n \end{cases}$	conditionally independent sentences
7)	$l(r) < K_T(x)$	hypothesis q is "significant"
8)	$\begin{matrix} x' = x^{(1)}x^{(2)} \\ j' = j^{(1)}j^{(2)} \end{matrix} \Rightarrow \begin{matrix} T_q(j^{(1)}) = x^{(1)}, \ j^{(1)} < K_T(x^{(1)}) \\ T_q(j^{(2)}) = x^{(2)}, \ j^{(2)} < K_T(x^{(2)}) \end{matrix}$	hypothesis q is "general"

Input p is an acceptable MML message encoding data string x, if

1)	$T(p) = x$	p encodes x
2)	$l(p) < l(x)$	some compression is achieved
3)	$p = qr$	two-part encoding
4)	$T(q) = \epsilon$	hypothesis q does not determine data
5)	$T_q(rs) = xT_q(s)$	reading r does not alter the state of T
6)	$x = x_1 \dots x_n \Rightarrow \begin{cases} r = r_1 \dots r_n \\ T_q(r_i) = x_i,\ i = 1 \dots n \end{cases}$	conditionally independent sentences
7)	$l(r) < K_T(x)$	hypothesis q is "significant"
8)	$\begin{matrix} x' = x^{(1)}x^{(2)} \\ j' = j^{(1)}j^{(2)} \end{matrix} \Rightarrow \begin{matrix} T_q(j^{(1)}) = x^{(1)},\ j^{(1)} < K_T(x^{(1)}) \\ T_q(j^{(2)}) = x^{(2)},\ j^{(2)} < K_T(x^{(2)}) \end{matrix}$	hypothesis q is "general"
9)	No prefix of q satisfies all the above conditions	all of q is required

- The division of p into q and r is unique

- The division of p into q and r is unique
- In what way exactly does hypothesis string q affect T?

- The division of p into q and r is unique

- In what way exactly does hypothesis string q affect T?

Remember $T \xrightarrow{q} T_q$

T_q is a decoder of “second parts”

$$T_q : S \to W$$

- The division of p into q and r is unique

- In what way exactly does hypothesis string q affect T?

Remember $T \xrightarrow{q} T_q$

T_q is a decoder of “second parts”

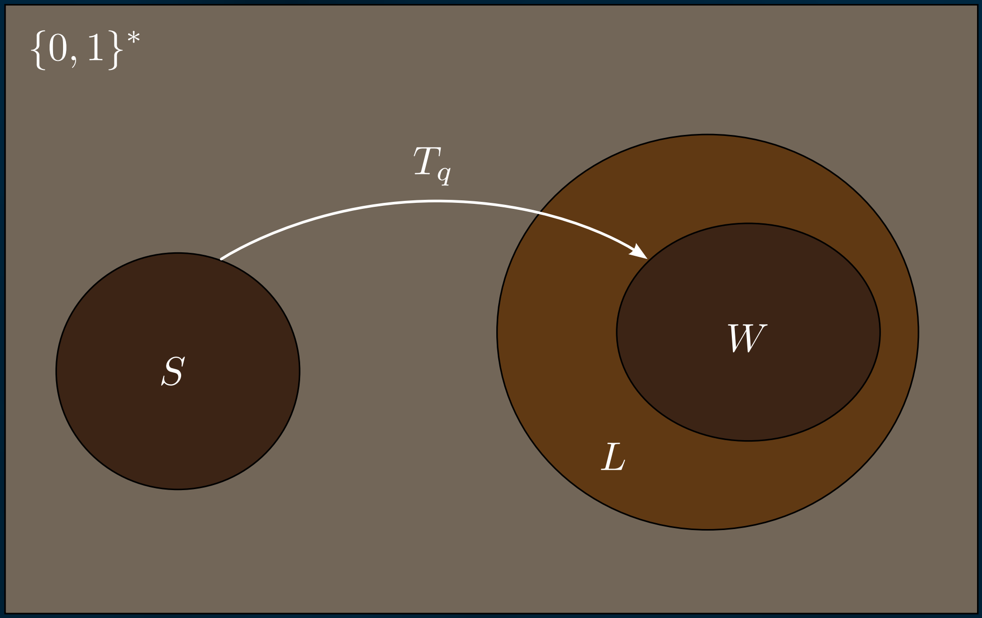
$$T_q : S \to W$$

Code words

$$S = \{r_i \in \{0,1\}^* \mid T_q(r_i) \in L\}$$

Subset of L that is coded

$$W = \{x_i \in L \mid \exists r_i \in S : T_q(r_i) = x_i\}$$



- The division of p into q and r is unique

- In what way exactly does hypothesis string q affect T?

Remember $T \xrightarrow{q} T_q$

T_q is a decoder of "second parts"

$$T_q : S \to W$$

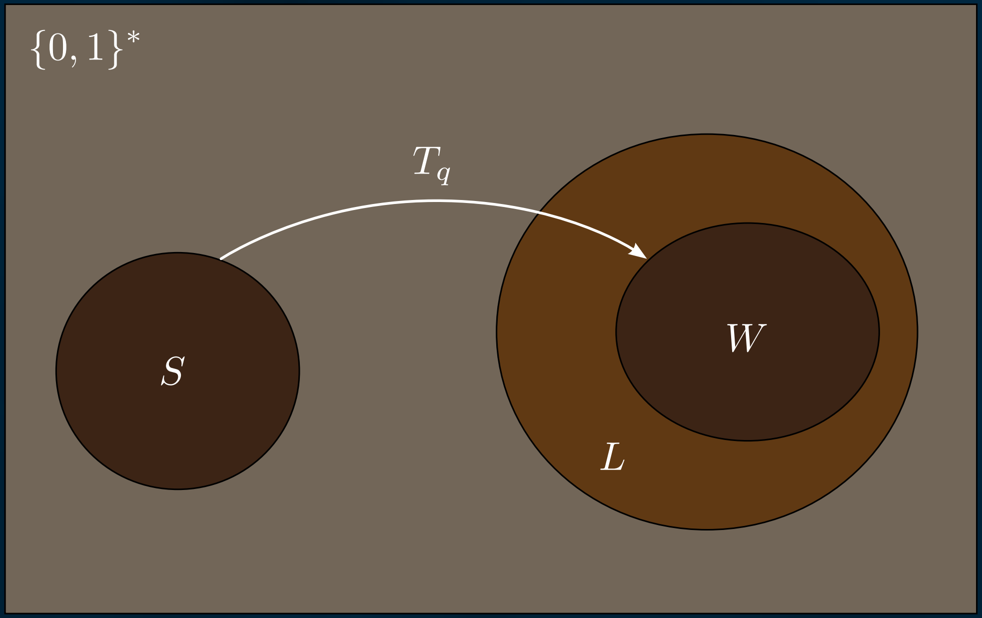
Code words

$$S = \{r_i \in \{0,1\}^* \mid T_q(r_i) \in L\}$$

Subset of L that is coded

$$W = \{x_i \in L \mid \exists r_i \in S : T_q(r_i) = x_i\}$$

In fact, T_q decodes a prefix code (why?)



- What is the hypothesis (probability distribution) Q implied by hypothesis string q?

- What is the hypothesis (probability distribution) Q implied by hypothesis string q?

$$Q(x_i) = \begin{cases} 2^{-l(p)} & \text{, if } p \text{ is a shortest codeword for sentence } x_i \in L \\ 0 & \text{, if there is no codeword for sentence } x_i \in L \end{cases}$$

- What is the hypothesis (probability distribution) Q implied by hypothesis string q?

$$Q(x_i) = \begin{cases} 2^{-l(p)} & \text{, if } p \text{ is a shortest codeword for sentence } x_i \in L \\ 0 & \text{, if there is no codeword for sentence } x_i \in L \end{cases}$$

Because of prefix code

$$\sum_{x_i \in L} Q(x_i) = \sum_{x_i \in W} 2^{-l(p)} \overset{\text{Kraft}}{\leq} 1$$

- What is the hypothesis (probability distribution) Q implied by hypothesis string q?

$$Q(x_i) = \begin{cases} 2^{-l(p)} & \text{, if } p \text{ is a shortest codeword for sentence } x_i \in L \\ 0 & \text{, if there is no codeword for sentence } x_i \in L \end{cases}$$

Because of prefix code

$$\sum_{x_i \in L} Q(x_i) = \sum_{x_i \in W} 2^{-l(p)} \overset{\text{Kraft}}{\leq} 1$$

- In this setting, hypotheses are falsifiable

- What is the hypothesis (probability distribution) Q implied by hypothesis string q?

$$Q(x_i) = \begin{cases} 2^{-l(p)} & \text{, if } p \text{ is a shortest codeword for sentence } x_i \in L \\ 0 & \text{, if there is no codeword for sentence } x_i \in L \end{cases}$$

Because of prefix code

$$\sum_{x_i \in L} Q(x_i) = \sum_{x_i \in W} 2^{-l(p)} \overset{\text{Kraft}}{\leq} 1$$

- In this setting, hypotheses are falsifiable

2) $l(p) < l(x) \Rightarrow l(r) < l(x)$

- What is the hypothesis (probability distribution) Q implied by hypothesis string q?

$$Q(x_i) = \begin{cases} 2^{-l(p)} & \text{, if } p \text{ is a shortest codeword for sentence } x_i \in L \\ 0 & \text{, if there is no codeword for sentence } x_i \in L \end{cases}$$

Because of prefix code

$$\sum_{x_i \in L} Q(x_i) = \sum_{x_i \in W} 2^{-l(p)} \overset{\text{Kraft}}{\leq} 1$$

- In this setting, hypotheses are falsifiable

2) $l(p) < l(x) \Rightarrow l(r) < l(x)$

If Q assigns low probability (eq. high codeword length) to a sentence x_i, then adding enough such sentences to the data string will violate the above condition and falsify the hypothesis

- What is the hypothesis (probability distribution) Q implied by hypothesis string q?

$$Q(x_i) = \begin{cases} 2^{-l(p)} & \text{, if } p \text{ is a shortest codeword for sentence } x_i \in L \\ 0 & \text{, if there is no codeword for sentence } x_i \in L \end{cases}$$

Because of prefix code

$$\sum_{x_i \in L} Q(x_i) = \sum_{x_i \in W} 2^{-l(p)} \overset{\text{Kraft}}{\leq} 1$$

- In this setting, hypotheses are falsifiable

2) $l(p) < l(x) \Rightarrow l(r) < l(x)$

If Q assigns low probability (eq. high codeword length) to a sentence x_i, then adding enough such sentences to the data string will violate the above condition and falsify the hypothesis

Can Q assign lower codeword length to every sentence?
(L is a complete prefix code for "data facts")

- What is the hypothesis (probability distribution) Q implied by hypothesis string q?

$$Q(x_i) = \begin{cases} 2^{-l(p)} & \text{, if } p \text{ is a shortest codeword for sentence } x_i \in L \\ 0 & \text{, if there is no codeword for sentence } x_i \in L \end{cases}$$

Because of prefix code

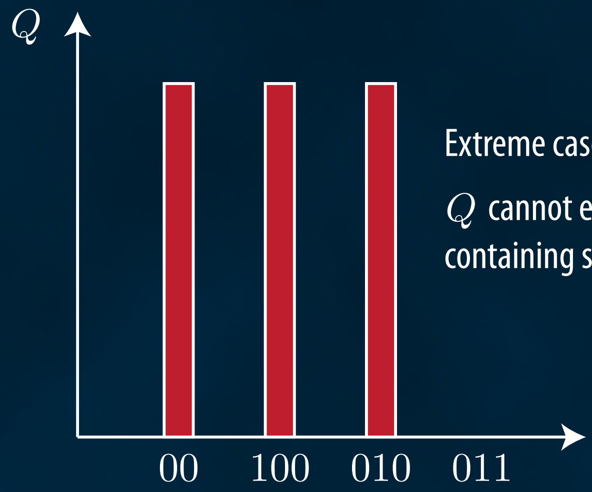
$$\sum_{x_i \in L} Q(x_i) = \sum_{x_i \in W} 2^{-l(p)} \overset{\text{Kraft}}{\leq} 1$$

- In this setting, hypotheses are falsifiable

2) $l(p) < l(x) \Rightarrow l(r) < l(x)$

If Q assigns low probability (eq. high codeword length) to a sentence x_i, then adding enough such sentences to the data string will violate the above condition and falsify the hypothesis

Can Q assign lower codeword length to every sentence? (L is a complete prefix code for "data facts")



Extreme case:

Q cannot encode *any* data string containing sentence 011.

- What do we “pay” for enforcing a two-part encoding scheme?

- What do we “pay” for enforcing a two-part encoding scheme?

Shortest acceptable MML input string: $M_T(x)$
Shortest unconstrained string: $K_T(x)$

with $M_T(x) \geq K_T(x)$

- What do we "pay" for enforcing a two-part encoding scheme?

Shortest acceptable MML input string: $M_T(x)$
Shortest unconstrained string: $K_T(x)$

with $M_T(x) \geq K_T(x)$

$$\begin{aligned} M_T(x) - K_T(x) &= l(q) + l(r) - K_T(x) \\ &= K_T(Q) - \log_2(Q(x)) - K_T(x) \\ &= -\log_2\left(\frac{P_T(Q)Q(x)}{P_T(x)}\right) \\ &\approx -\log_2(\Pr(Q \mid x)) \end{aligned}$$

$$\boxed{P_T(x) = 2^{-K_T(x)}}$$

- What do we "pay" for enforcing a two-part encoding scheme?

Shortest acceptable MML input string: $M_T(x)$
Shortest unconstrained string: $K_T(x)$

with $M_T(x) \geq K_T(x)$

$$\begin{aligned}
M_T(x) - K_T(x) &= l(q) + l(r) - K_T(x) \\
&= K_T(Q) - \log_2(Q(x)) - K_T(x) \\
&= -\log_2\left(\frac{P_T(Q)Q(x)}{P_T(x)}\right) \\
&\approx -\log_2(\Pr(Q \mid x))
\end{aligned}$$

$$\boxed{P_T(x) = 2^{-K_T(x)}}$$

Finding the shortest MML string is like MAP, where $P_T(Q)$ plays the role of the prior

- What do we "pay" for enforcing a two-part encoding scheme?

Shortest acceptable MML input string: $M_T(x)$
Shortest unconstrained string: $K_T(x)$
with $M_T(x) \geq K_T(x)$

$$\begin{aligned} M_T(x) - K_T(x) &= l(q) + l(r) - K_T(x) \\ &= K_T(Q) - \log_2(Q(x)) - K_T(x) \\ &= -\log_2\left(\frac{P_T(Q)Q(x)}{P_T(x)}\right) \\ &\approx -\log_2(\Pr(Q \mid x)) \end{aligned}$$

$$\boxed{P_T(x) = 2^{-K_T(x)}}$$

Finding the shortest MML string is like MAP, where $P_T(Q)$ plays the role of the prior

The log posterior odds ratio of two hypotheses is

$$\log_2\left(\frac{\Pr(Q_1 \mid x)}{\Pr(Q_2 \mid x)}\right) = l(p_2) - l(p_1)$$

where p_1 and p_2 are shortest input strings for their respective hypotheses

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible “theories” (semicomputable semimeasures)

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible “theories” (semicomputable semimeasures)

- Hutter: Many (all?) interesting problems $\longrightarrow$ Sequence prediction $\longrightarrow$ (Universal) induction + Decision theory

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible "theories" (semicomputable semimeasures)

- Hutter: Many (all?) interesting problems $\longrightarrow$ Sequence prediction $\longrightarrow$ (Universal) induction + Decision theory

- Why would we ever need to pick <u>one</u> "theory"?!

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible "theories" (semicomputable semimeasures)

- Hutter: Many (all?) interesting problems ⟶ Sequence prediction ⟶ (Universal) induction + Decision theory

- Why would we ever need to pick <u>one</u> "theory"?!
  - Universal induction is uncomputable

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible "theories" (semicomputable semimeasures)

- Hutter: Many (all?) interesting problems ⟶ Sequence prediction ⟶ (Universal) induction + Decision theory

- Why would we ever need to pick <u>one</u> "theory"?!
  - Universal induction is uncomputable
  - Even constrained version is *highly* infeasible (right now, for real problems)
    1) Encode all human knowledge into a huge string to use as prior
    2) Compute Bayesian posterior for each event we want to predict

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible "theories" (semicomputable semimeasures)

- Hutter: Many (all?) interesting problems $\longrightarrow$ Sequence prediction $\longrightarrow$ (Universal) induction + Decision theory

- Why would we ever need to pick <u>one</u> "theory"?!
  - Universal induction is uncomputable
  - Even constrained version is *highly* infeasible (right now, for real problems)
    1) Encode all human knowledge into a huge string to use as prior
    2) Compute Bayesian posterior for each event we want to predict
  - General theories are compact, efficient (but imperfect) summaries of prior knowledge

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible "theories" (semicomputable semimeasures)

- Hutter: Many (all?) interesting problems $\longrightarrow$ Sequence prediction $\longrightarrow$ (Universal) induction + Decision theory

- Why would we ever need to pick <u>one</u> "theory"?!
  - Universal induction is uncomputable
  - Even constrained version is *highly* infeasible (right now, for real problems)
    1) Encode all human knowledge into a huge string to use as prior
    2) Compute Bayesian posterior for each event we want to predict
  - General theories are compact, efficient (but imperfect) summaries of prior knowledge
  - Humans prefer to understand the world as general theories, instead of mixtures

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible "theories" (semicomputable semimeasures)

- Hutter: Many (all?) interesting problems $\longrightarrow$ Sequence prediction $\longrightarrow$ (Universal) induction + Decision theory

- Why would we ever need to pick <u>one</u> "theory"?!
  - Universal induction is uncomputable
  - Even constrained version is *highly* infeasible (right now, for real problems)
    1) Encode all human knowledge into a huge string to use as prior
    2) Compute Bayesian posterior for each event we want to predict
  - General theories are compact, efficient (but imperfect) summaries of prior knowledge
  - Humans prefer to understand the world as general theories, instead of mixtures

- Compromise: Instead of a single theory, retain the few best of them

- Solomonoff: *Truly* Bayesian universal induction

  Universal prior = Bayesian mixture over all possible "theories" (semicomputable semimeasures)

- Hutter: Many (all?) interesting problems ⟶ Sequence prediction ⟶ (Universal) induction + Decision theory

- Why would we ever need to pick <u>one</u> "theory"?!
  - Universal induction is uncomputable
  - Even constrained version is *highly* infeasible (right now, for real problems)
    1) Encode all human knowledge into a huge string to use as prior
    2) Compute Bayesian posterior for each event we want to predict
  - General theories are compact, efficient (but imperfect) summaries of prior knowledge
  - Humans prefer to understand the world as general theories, instead of mixtures

> *[...] Bayes/Solomonoff is the Gold standard for prediction, but MML/MDL is (often) a good "approximation/simplification" for explanation and understanding.*
>
> M. Hutter

- Compromise: Instead of a single theory, retain the few best of them

- Results in Kolmogorov complexity are almost always “up to a constant”

- Results in Kolmogorov complexity are almost always "up to a constant"

- Constants correspond to length of "compiler" and can be large

- Results in Kolmogorov complexity are almost always "up to a constant"
- Constants correspond to length of "compiler" and can be large
- Remember $\log_2\left(\dfrac{\Pr(Q_1 \mid x)}{\Pr(Q_2 \mid x)}\right) = l(p_2) - l(p_1)$

Example $l(p_2) - l(p_1) = 10 \Rightarrow \Pr(Q_1 \mid x) = 1024 \times \Pr(Q_2 \mid x)$

- Results in Kolmogorov complexity are almost always "up to a constant"

- Constants correspond to length of "compiler" and can be large

- Remember $$\log_2\left(\frac{\Pr(Q_1 \mid x)}{\Pr(Q_2 \mid x)}\right) = l(p_2) - l(p_1)$$

  Example $$l(p_2) - l(p_1) = 10 \Rightarrow \Pr(Q_1 \mid x) = 1024 \times \Pr(Q_2 \mid x)$$

- In practice, one has to choose priors very carefully in order to avoid unwanted biases
  (authors claim that MML school is more considerate in this regard than Kolmogorov and MDL ones)

- Results in Kolmogorov complexity are almost always "up to a constant"

- Constants correspond to length of "compiler" and can be large

- Remember $$\log_2\left(\frac{\Pr(Q_1 \mid x)}{\Pr(Q_2 \mid x)}\right) = l(p_2) - l(p_1)$$

  Example $$l(p_2) - l(p_1) = 10 \Rightarrow \Pr(Q_1 \mid x) = 1024 \times \Pr(Q_2 \mid x)$$

- In practice, one has to choose priors very carefully in order to avoid unwanted biases
  (authors claim that MML school is more considerate in this regard than Kolmogorov and MDL ones)

- Maybe time to change the paradigm?

- Results in Kolmogorov complexity are almost always "up to a constant"
- Constants correspond to length of "compiler" and can be large
- Remember $$\log_2\left(\frac{\Pr(Q_1 \mid x)}{\Pr(Q_2 \mid x)}\right) = l(p_2) - l(p_1)$$

  Example $$l(p_2) - l(p_1) = 10 \Rightarrow \Pr(Q_1 \mid x) = 1024 \times \Pr(Q_2 \mid x)$$
- In practice, one has to choose priors very carefully in order to avoid unwanted biases
  (authors claim that MML school is more considerate in this regard than Kolmogorov and MDL ones)
- Maybe time to change the paradigm?

  ART

  Handcraft priors,
  parameter encodings,
  features, etc.

- Results in Kolmogorov complexity are almost always "up to a constant"

- Constants correspond to length of "compiler" and can be large

- Remember $\log_2\left(\frac{\Pr(Q_1 \mid x)}{\Pr(Q_2 \mid x)}\right) = l(p_2) - l(p_1)$

  Example $l(p_2) - l(p_1) = 10 \Rightarrow \Pr(Q_1 \mid x) = 1024 \times \Pr(Q_2 \mid x)$

- In practice, one has to choose priors very carefully in order to avoid unwanted biases (authors claim that MML school is more considerate in this regard than Kolmogorov and MDL ones)

- Maybe time to change the paradigm?

ART		SCIENCE
Handcraft priors, parameter encodings, features, etc.	$\longrightarrow$	Automatically select suitable priors, encodings, features, etc.

T H A N K Y O U !