



Computational Foundations of Informatics

Vasant G. Honavar

Edward Frymoyer Professor of Information Sciences and Technology
Artificial Intelligence Research Laboratory
Informatics Graduate Program
Computer Science and Engineering Graduate Program
Bioinformatics and Genomics Graduate Program
Neuroscience Graduate Program
Data Sciences Undergraduate Program
Center for Big Data Analytics and Discovery Informatics
Huck Institutes of the Life Sciences
Institute for Computational and Data Sciences
Clinical and Translational Sciences Institute
Northeast Big Data Hub
Pennsylvania State University

Sudha Murty Distinguished Visiting Chair of Neurocomputing and Data Science
Indian Institute of Science

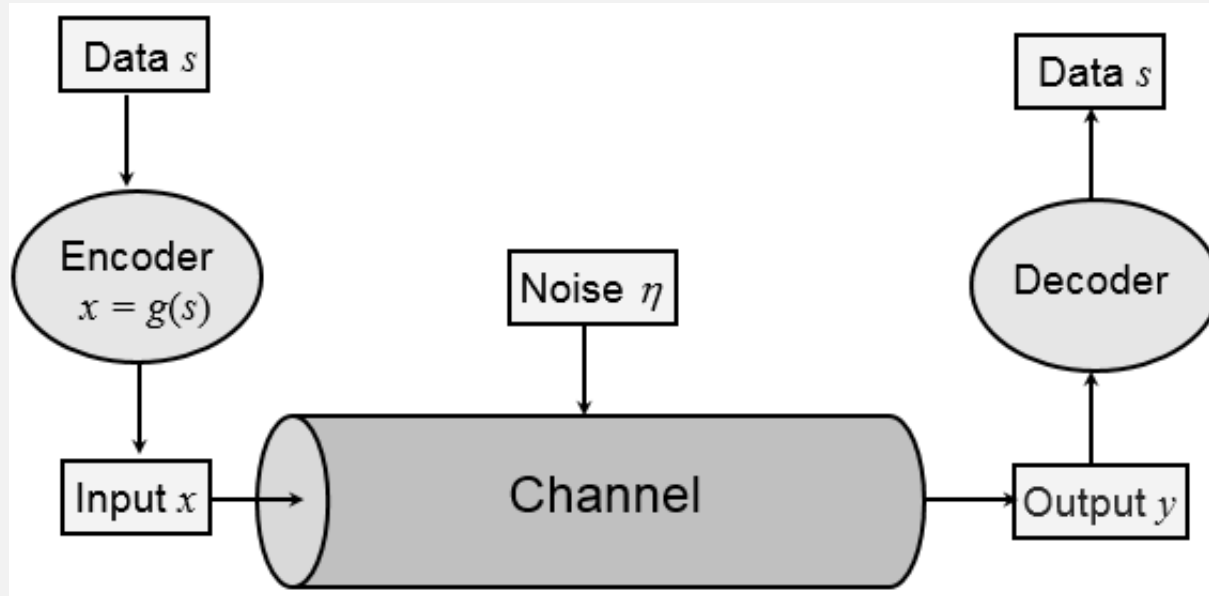


Elements of Communication Theory



Shannon's Source Coding Theorem

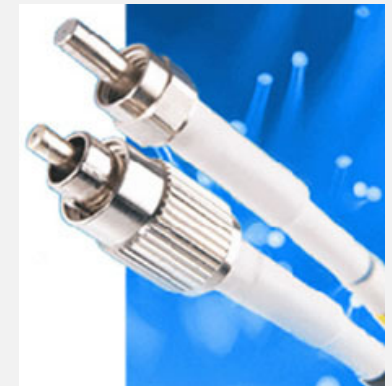
The Communication Channel



- Source generates message s – sequence of symbols
- s is encoded into channel input code words $x = g(s)$
- Channel input x leads to output y decoded to s

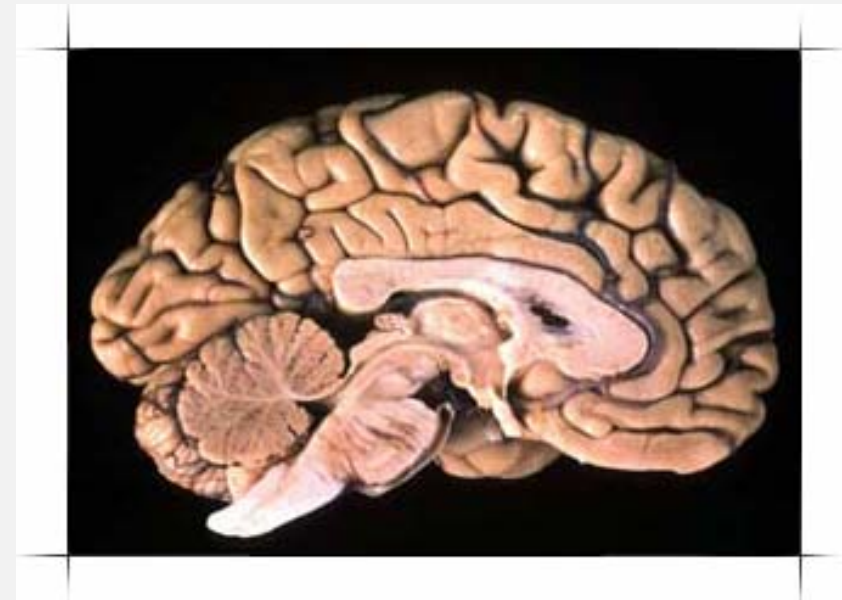
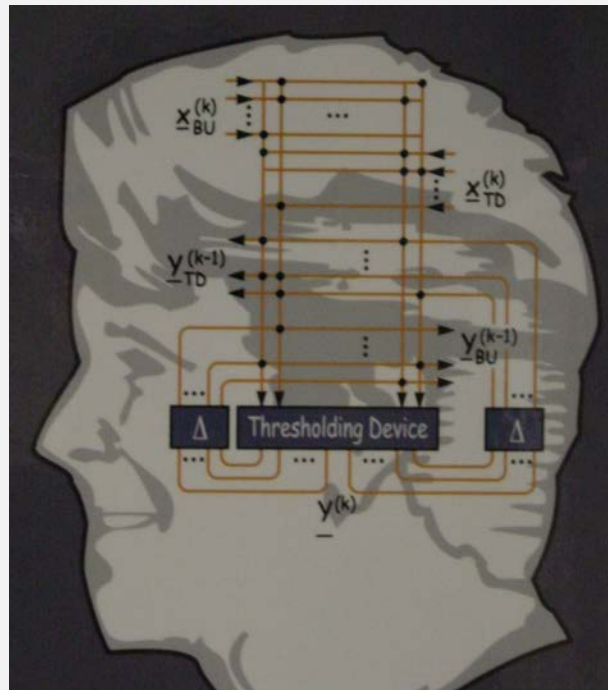


Communication channels





Communication channels





Shannon's Definition of Communication

“The fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.”

“Frequently the messages have *meaning*”

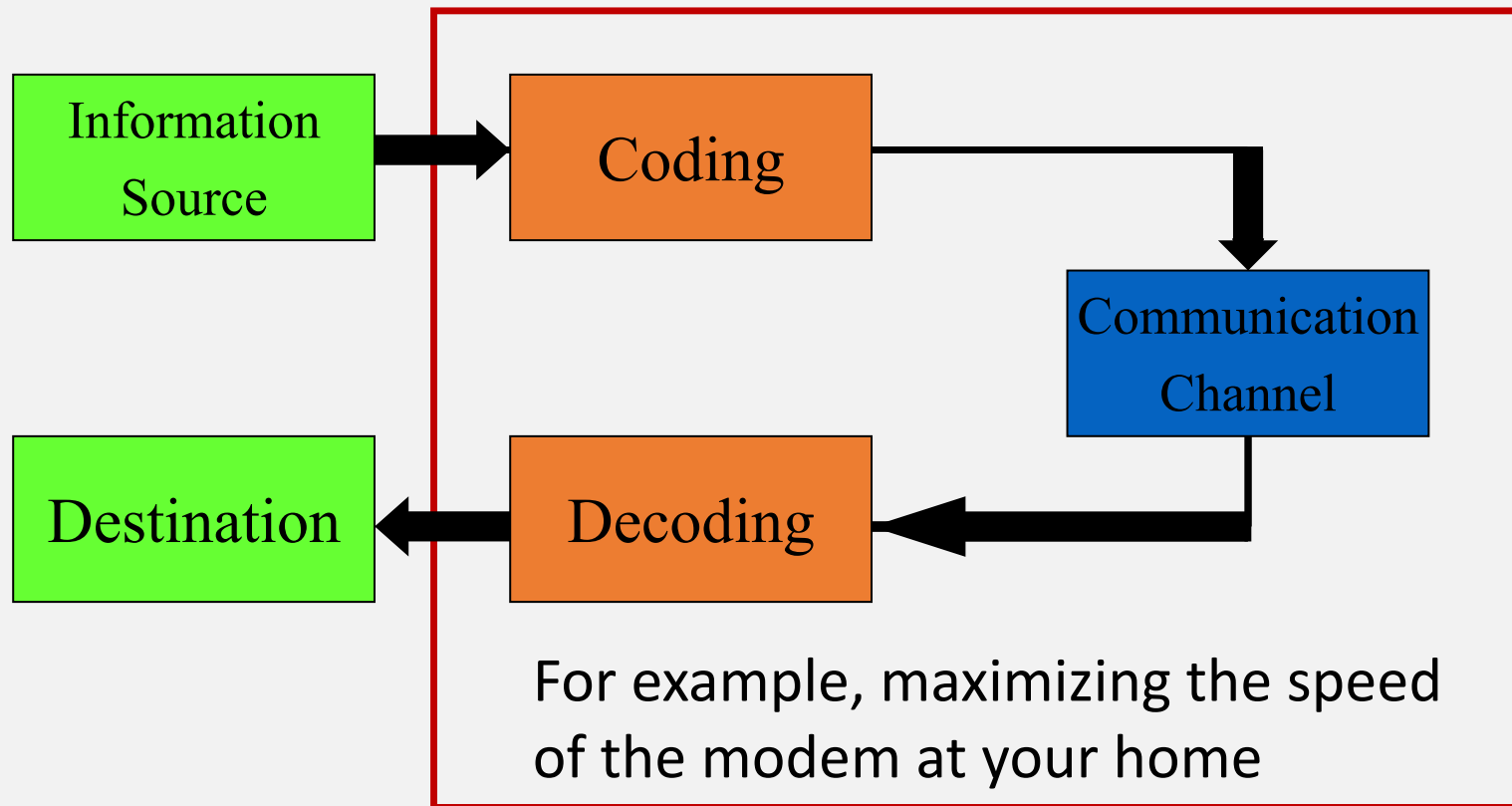
“... [which is] irrelevant to the engineering problem.”





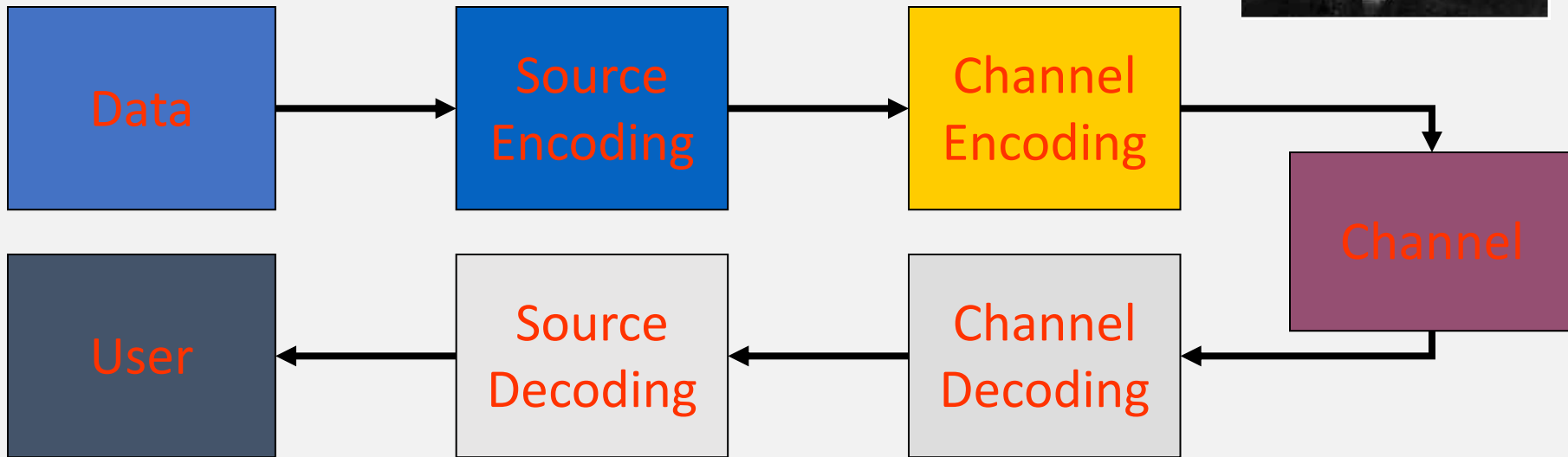
Shannon wants to...

- Find a way for “reliably” transmitting data through the channel at “maximal” possible rate.

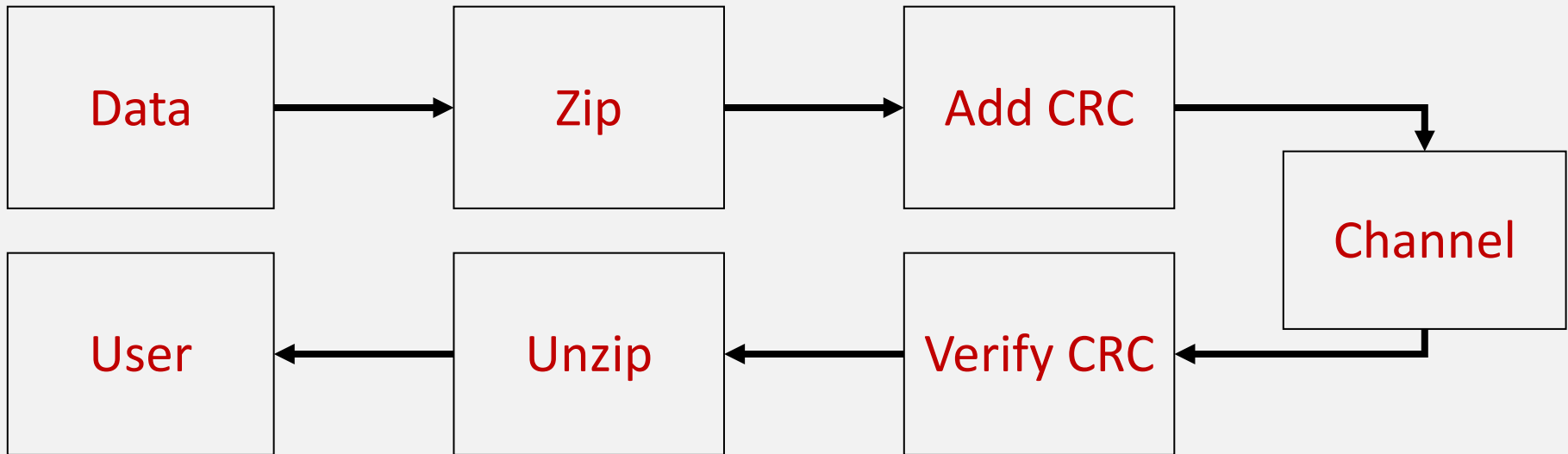




Shannon's Vision



Example: Disk Storage



CRC = Cyclic redundancy check – used for checking for errors introduced during data transfer



Information Theory Terminology

Zip	=	Source Encoding	Data Compression
Unzip	=	Source Decoding	Data Decompression
Add CRC	=	Channel Encoding	Error Protection
Verify CRC	=	Channel Decoding	Error Correction



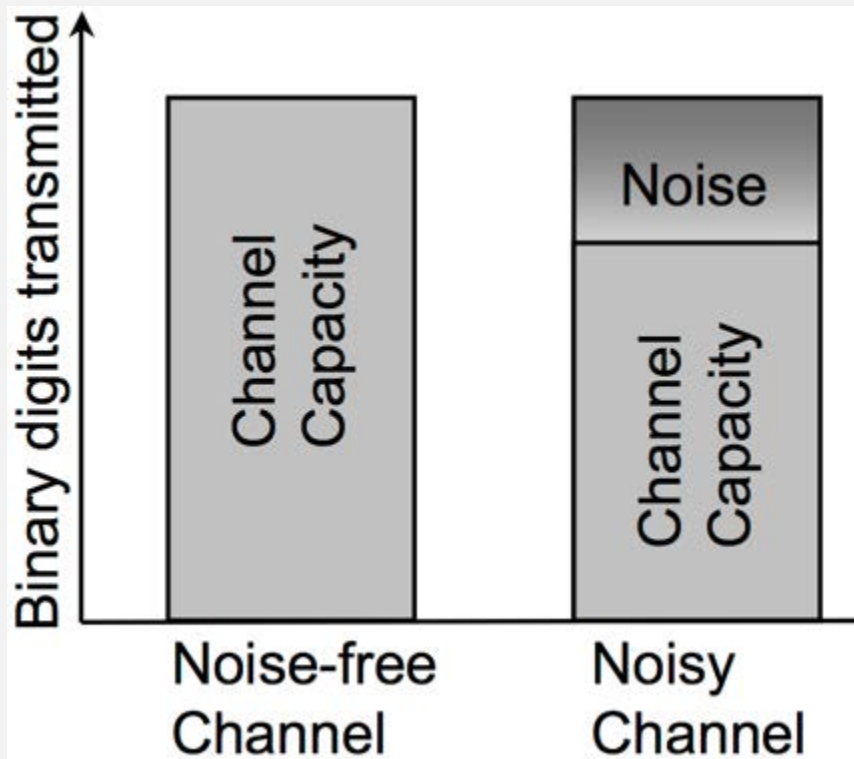
Shannon Theory

The original 1948 paper of Shannon covers:

1. Measurement of Information
2. Source Coding Theory
3. Channel Coding Theory

Shannon's Source Coding Theorem Channel Capacity

Maximum information communicated from channel input to output

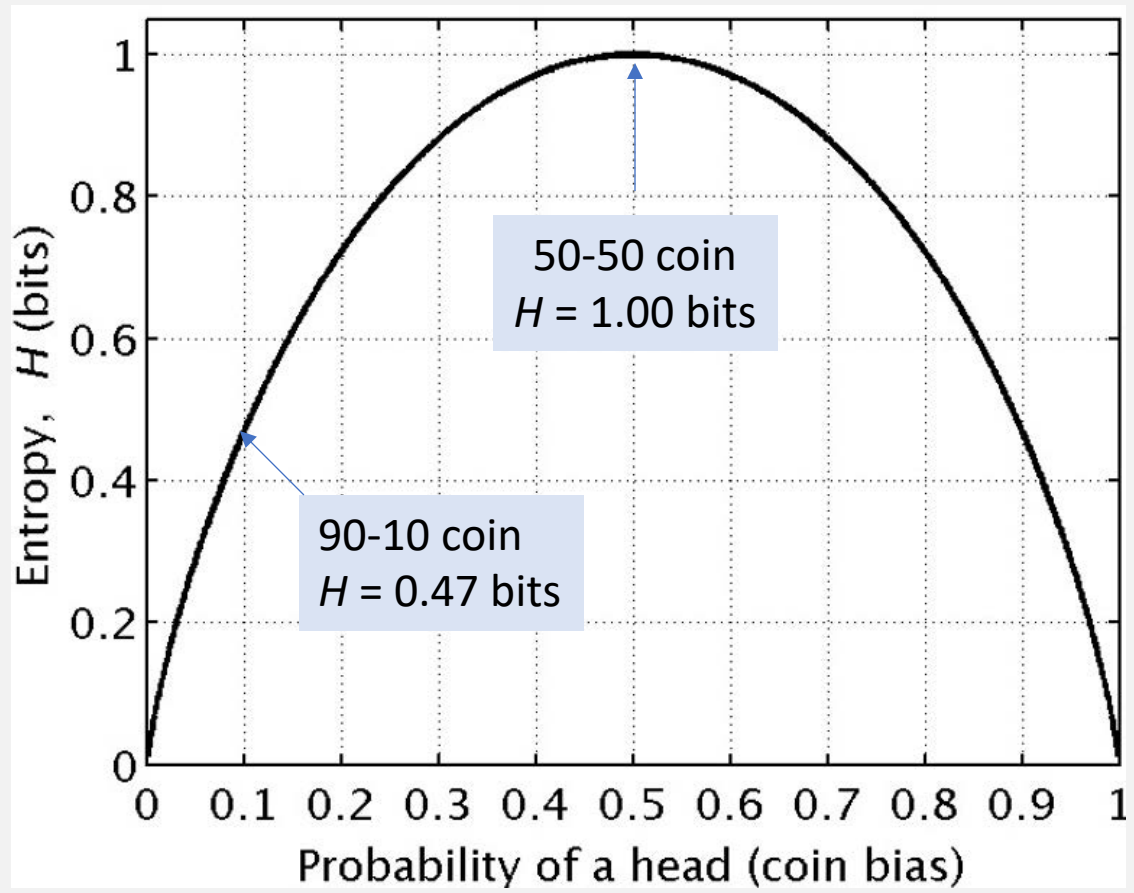


Shannon's Source Coding Theorem on Channel Capacity

Shannon's 1948 "A Mathematical Theory of Communication" studied the theoretical limits on the channel transmission of information from source to receiver, where the source is encoded before transmission

- He considered two types of channels – continuous and discrete
- And he dealt with added noise
- Only the discrete channel without noise is described in this brief presentation

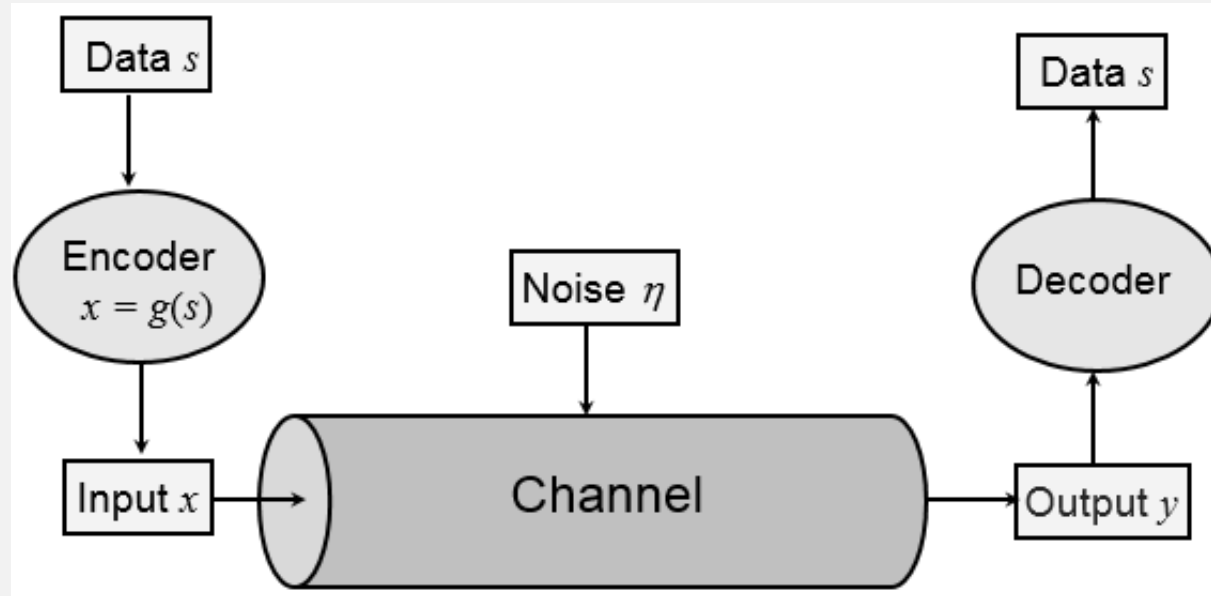
Shannon Entropy $H(X)$ versus coin bias (probability of a head)





Shannon's Source Coding Theorem

The Communication Channel



- Source generates message s – sequence of symbols
- s is encoded into channel input code words $x = g(s)$
- Channel input output leads to output y decoded to s



Shannon's First **Source Coding** Theorem



“To reliably store the information generated by some random source X , you need no more/less than, on the average, $H(X)$ bits for each outcome.”



Source coding theorem explained

- If I toss a dice 1,000,000 times and record values from each trial

1,3,4,6,2,5,2,4,5,2,4,5,6,1,....

- If we use binary representation
 - We need 3 bits for storing each outcome as 3 bits can encode each of the 6 possible outcomes (I can't manage with 2 bits).
 - We need **3,000,000** bits for storing the information.
- If we use ASCII representation
 - We need **8 bits=1 byte** for storing each outcome
 - The resulting file has size **8,000,000** bits






But Shannon's information theory implies:

- You only need $\log_2 6 = 2.585$ bits for storing each outcome (Why?)
- So, the file can be compressed to yield size
 $2.585 \times 1,000,000 = 2,585,000$ bits
- Optimal Compression Ratio is given by:

$$\frac{2,585,000}{8,000,000} = 0.3231 = 32.31\%$$



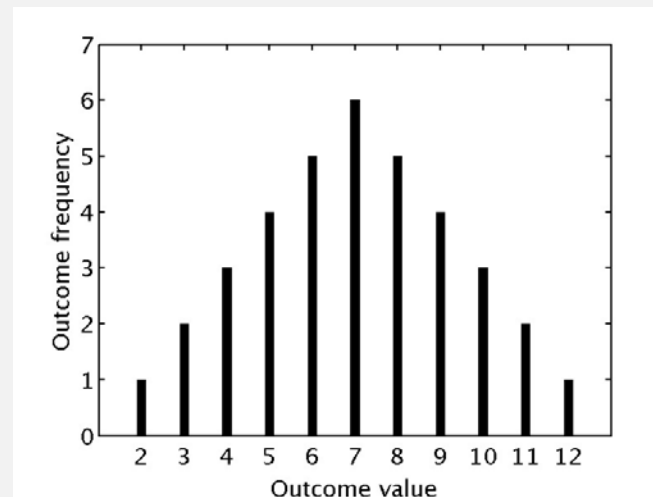
Implications of Shannon's coding theorem

		File Size	Compression Ratio
No compression		8,000,000 bits	100%
Shannon		2,585,000 bits	32.31%
Winzip		2,930,736 bits	36.63%
WinRAR		2,859,336 bits	35.74%



Shannon's Source Coding Theorem

- So far we have dealt mostly with variables having a uniform distribution, but the importance of the theorem only becomes apparent for non-uniform distributions
- Consider the numeric outcome of throwing a pair of 6-sided dice





Shannon's Source Coding Theorem

Symbol	Sum	Dice	Freq	p	h	Code x
s_1	2	1:1	1	0.03	5.17	10000
s_2	3	1:2, 2:1	2	0.06	4.17	0110
s_3	4	1:3, 3:1, 2:2	3	0.08	3.59	1001
s_4	5	2:3, 3:2, 1:4, 4:1	4	0.11	3.17	001
s_5	6	2:4, 4:2, 1:5, 5:1, 3:3	5	0.14	2.85	101
s_6	7	3:4, 4:3, 2:5, 5:2, 1:6, 6:1	6	0.17	2.59	111
s_7	8	3:5, 5:3, 2:6, 6:2, 4:4	5	0.14	2.85	110
s_8	9	3:6, 6:3, 4:5, 5:4	4	0.11	3.17	010
s_9	10	4:6, 6:4, 5:5	3	0.08	3.59	000
s_{10}	11	5:6, 6:5	2	0.06	4.17	0111
s_{11}	12	6:6	1	0.03	5.17	10001

$$\text{Shannon's optimum } H(S) = \sum_{i=1}^{m=11} p(s_i) \log \frac{1}{p(s_i)} = 3.27 \text{ bits/symbol}$$

With 11 symbols, simple 4-bit binary code = 4.00 bits/symbol

$$\text{Huffman coding } L(X) = \sum_{i=1}^{m=11} p(x_i) L(x_i) = 3.31 \text{ bits/symbol}$$

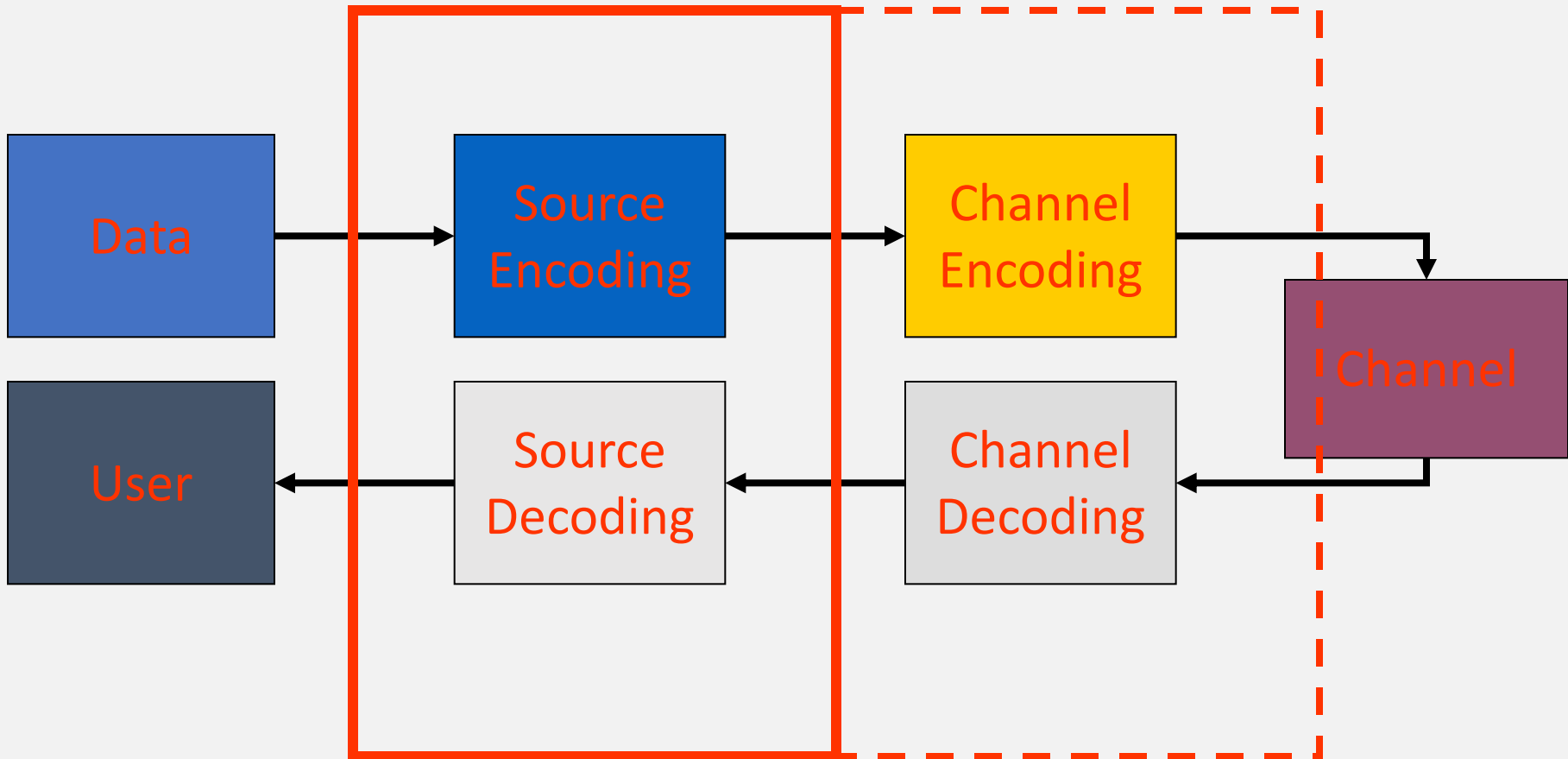
Achieving optimal compression

- Later in 1952, **David Huffman**, while was a graduate student at MIT, presented a systematic method to achieve the optimal compression ratio guaranteed by Shannon.
- The coding technique is therefore called “**Huffman code**”.
- Huffman codes are used in nearly every application that involves the compression and transmission of digital data, such as fax machines, modems, computer networks, and high-definition television (HDTV), to name a few.



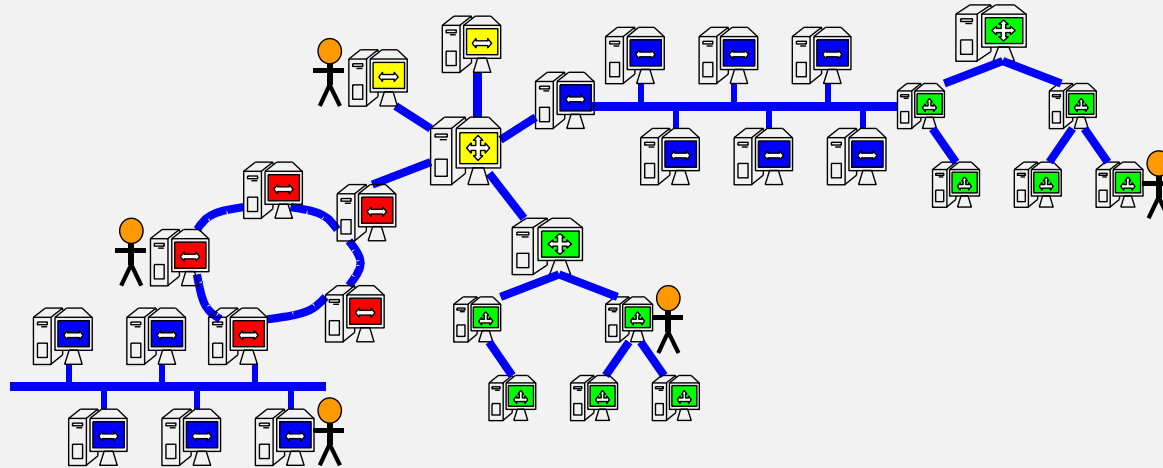
(1925-1999)

What about decoding?





Example: Communicating over the Internet

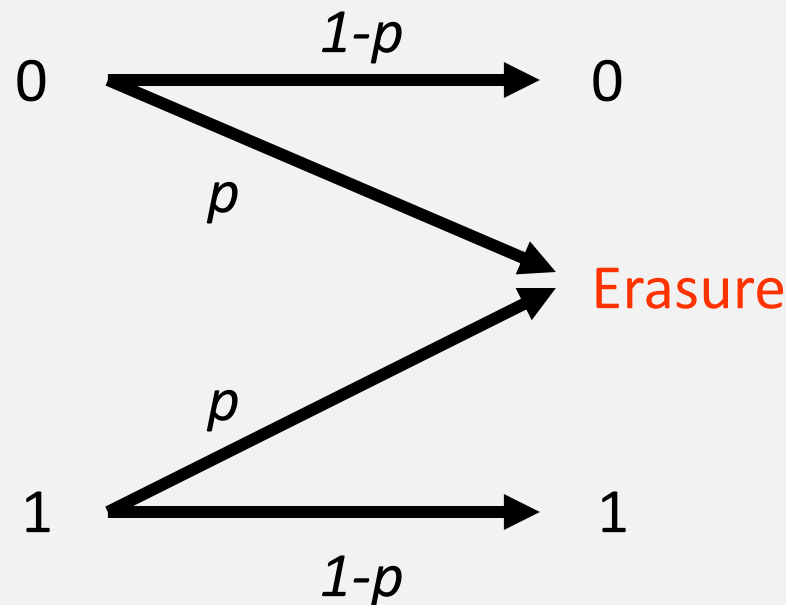


The major challenge here is **Packet Loss in the channel!**



Binary Erasure Channel

- “Packet loss” can be viewed as **Erasures**.
- Data that are erased mean they are lost during transmission...

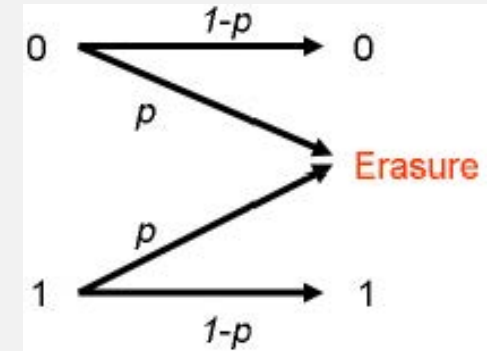


p is the packet loss rate in this network



Packet loss in a channel

- Once a binary symbol is erased, it can not be recovered...
 - Say, Alice sends **0,1,0,1,0,0** to Bob
 - But the network was so poor that Bob only receives **0,?,0,?,0,0**
 - So, Bob asks Alice to resend the message
 - Only this time Bob receives **0,?,?,1,0,0**
 - **Bob goes crazy**
 - What can Alice do?
 - **What if Alice repeats each transmission four times 0000,1111,0000,1111,0000,0000**



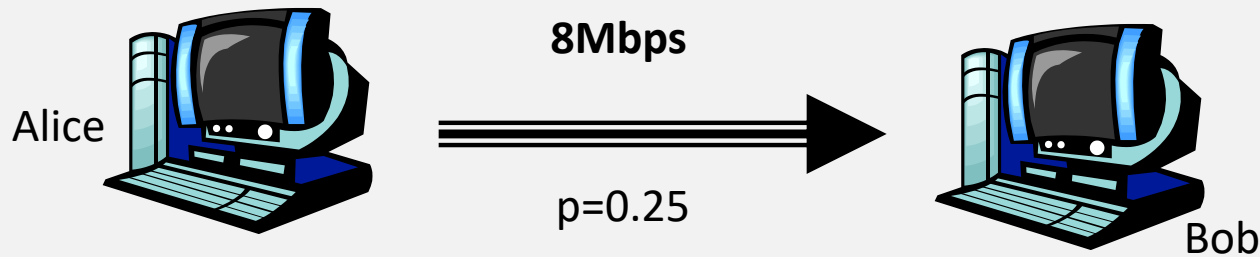


What is this redundancy good for?

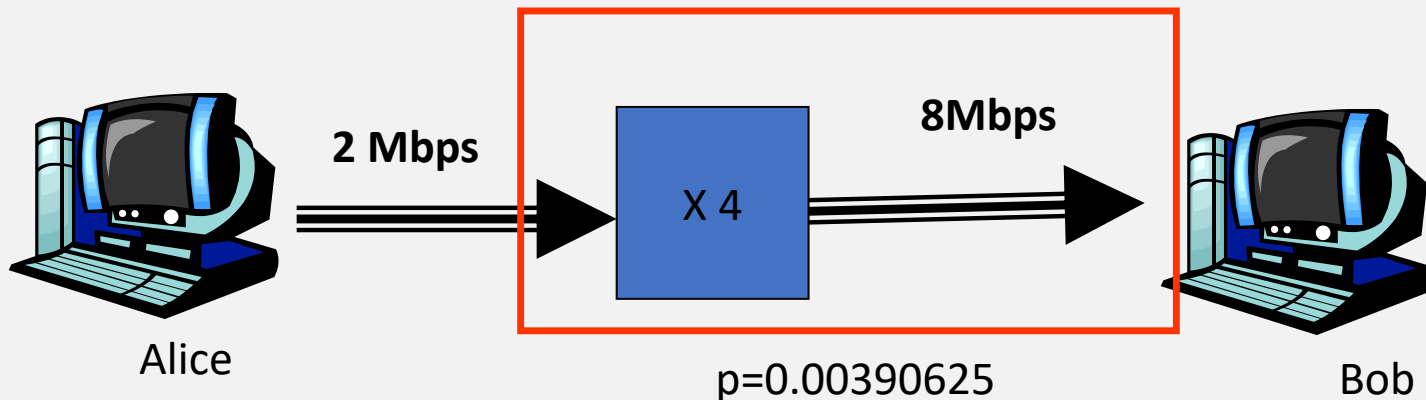
- If Alice sends **0000,1111,0000,1111,0000,0000**
- The only cases Bob can not read Alice's message is when all four repetitions are erased, for example,
- **????,1111,0000,1111,0000,0000**
all the four symbols are erased.
- But this happens at probability p^4
- Thus if the original network has packet loss rate $p=0.25$, by repeating each symbol 4 times, the resulting system has packet loss rate **$p^4=0.00390625$**



- But if the data rate in the original network is 8M bits per second



With repetition, Alice can only transmit at 2 M bps





Is repetition the best Alice can do?

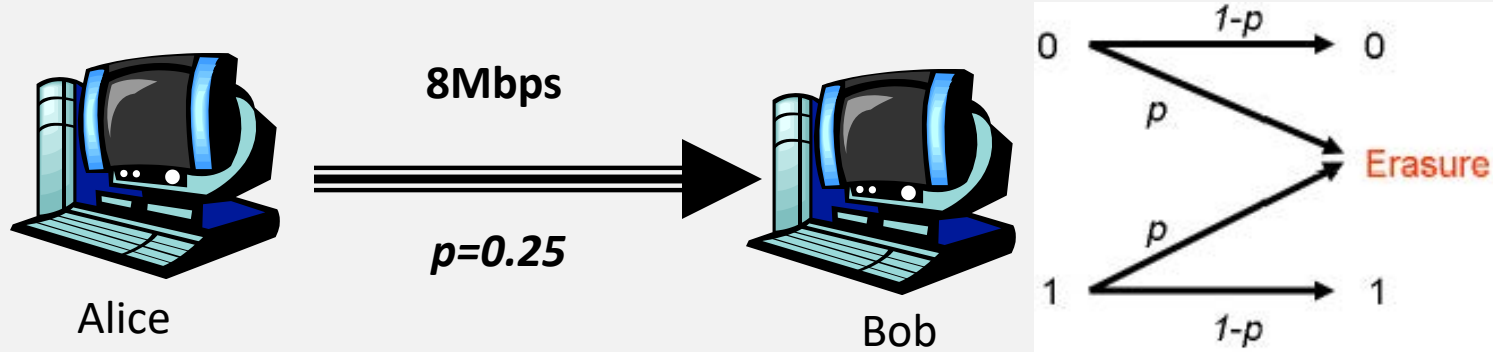
Shannon's Channel Coding Theorem



“Give me a channel and I can compute a quantity called capacity, C for that channel. Then reliable communication is possible only if your data rate stays below C .”

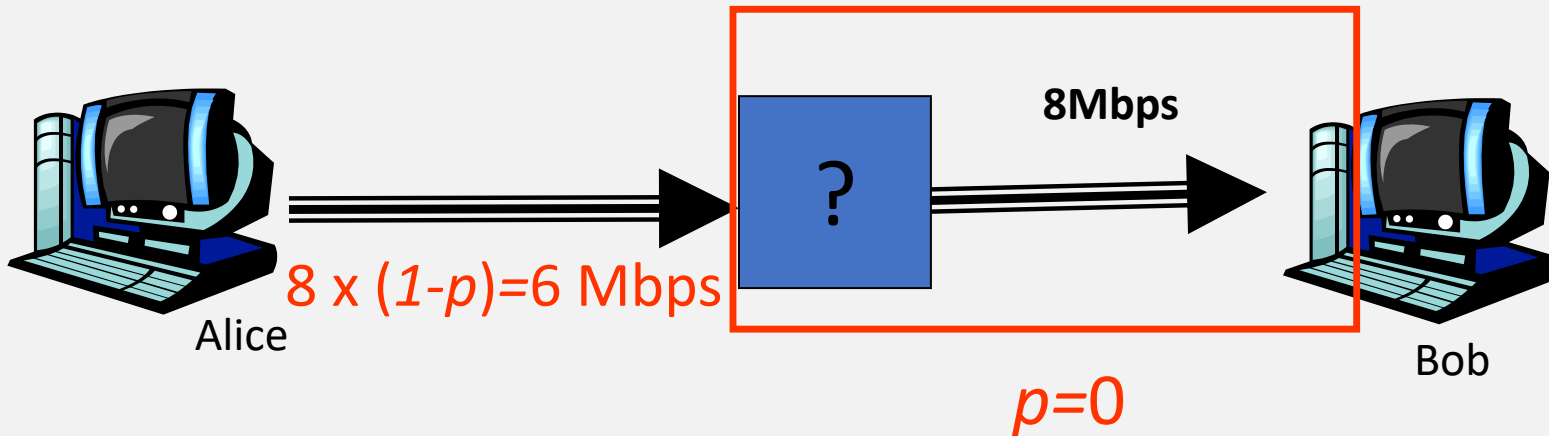


What does the channel coding theorem mean?



Channel capacity $C=1-p=0.75$

There exists coding scheme such that:





How do we find such an optimal coding scheme?

- 50 years of work on coding theory has led to a zoo of codes
 - Hamming codes
 - Convolutional codes,
 - Concatenated codes,
 - Low density parity check (LDPC) codes
 - Reed-Muller codes
 - Reed-Solomon codes,
 - BCH codes,
 - Finite Geometry codes,
 - Cyclic codes,
 - Golay codes,
 - Goppa codes
 - Algebraic Geometry codes,
 - Turbo codes
 - Zig-Zag codes,
 - Accumulate codes and Product-accumulate codes,
 - ...
- Bringing us close to realizing Shannon's optimal code



But With 50 Years of Hard Work

- **We have discovered a lot of good codes:**
 - Hamming codes
 - Convolutional codes,
 - Concatenated codes,
 - Low density parity check (LDPC) codes
 - Reed-Muller codes
 - Reed-Solomon codes,
 - BCH codes,
 - Finite Geometry codes,
 - Cyclic codes,
 - Golay codes,
 - Goppa codes
 - Algebraic Geometry codes,
 - Turbo codes
 - Zig-Zag codes,
 - Accumulate codes and Product-accumulate codes,
 - ...

We now come very close to the dream Shannon had

50 years ago! 😊
Communications Foundations of Informatics



Nowadays...

Source Coding Theorem has applied to

JPEG
2000

Image
Compression

MPEG

Audio/Video
Compression



Data
Compression

MP3

Audio
Compression

Channel Coding Theorem has applied to

- VCD/DVD – Reed-Solomon Codes
- Wireless Communication – Convolutional Codes
- Optical Communication – Reed-Solomon Codes
- Computer Network – LT codes, Raptor Codes
- Space Communication



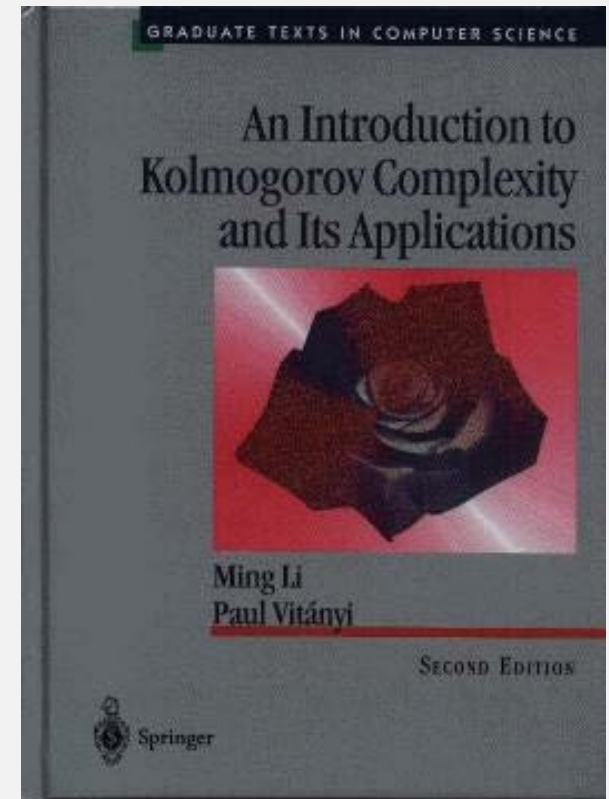
Kolmogorov complexity





Kolmogorov complexity

- Reference: Li and Vitányi: An introduction to Kolmogorov complexity and its applications.





What is the information content of an individual string?

- 111 1 (n 1's)
- $\pi = 3.1415926 \dots$
- $n = 2^{1024}$
- Champernowne's number:
0.1234567891011121314 ...
is normal in scale 10 (every block of size k has same frequency)
- All these numbers share one commonality: there are "small" programs to generate them.
- Shannon's information theory does not help here. Why not?
- Youtube video:
<http://www.youtube.com/watch?v=KyB13PD-UME>



1903: An interesting year



Kolmogorov



Church



von Neumann



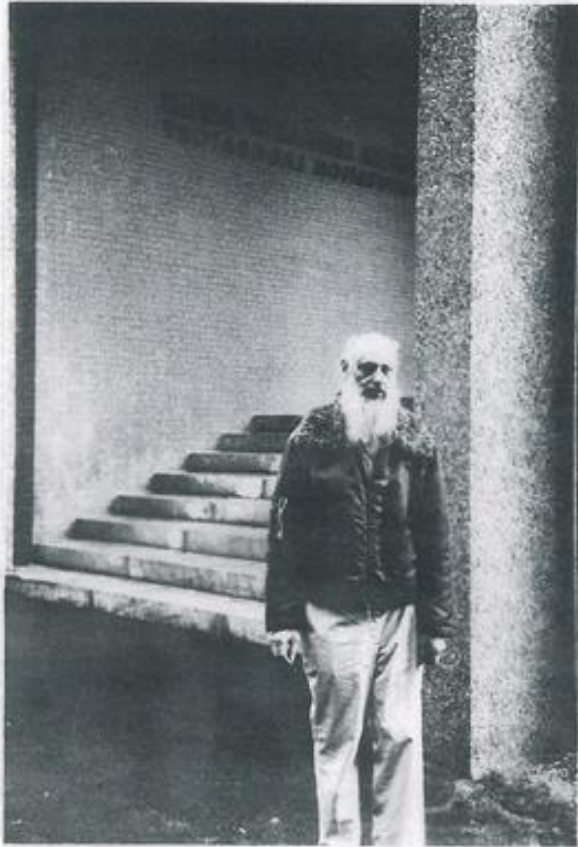


Andrey Nikolaevich Kolmogorov (1903-1987, Tambov, Russia)

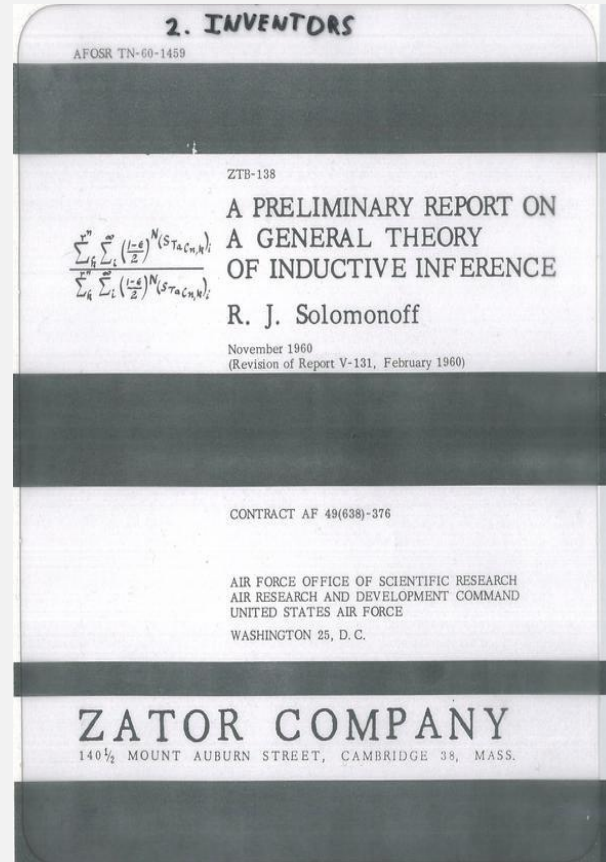


- Measure Theory
- Probability
- Analysis
- Intuitionistic Logic
- Cohomology
- Dynamical Systems
- Hydrodynamics
- Kolmogorov complexity



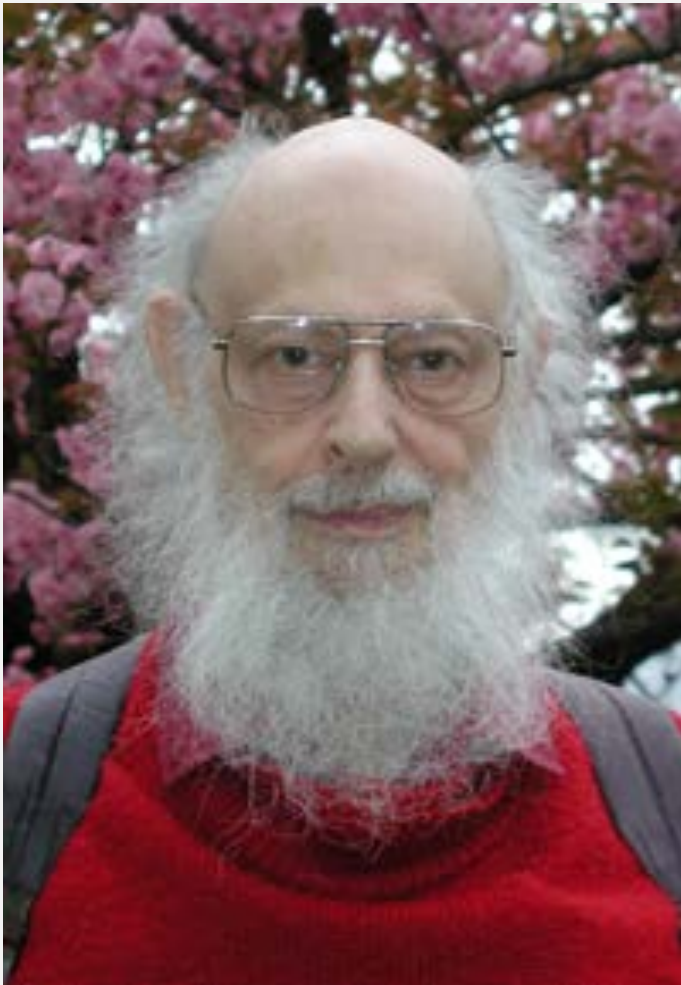


R. J. SOLOMONOFF
1960, 1964





Ray Solomonoff: 1926 -- 2009





G.J. CHAITIN
1966, 1969





The curious case of cheating casino



Bob proposes to flip a coin with Alice:

- Alice wins a dollar if Heads;
 - Bob wins a dollar if Tails
-
- Result: TTTTTT 100 Tails in a roll.
 - Alice lost \$100. She feels being cheated.





Alice goes to the court

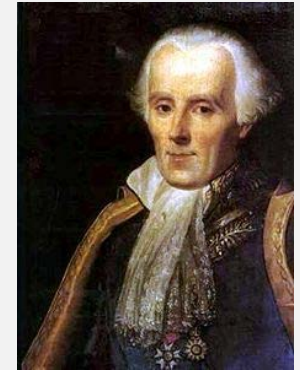
- Alice complains: T^{100} is not random.
- Bob asks Alice to produce a random coin flip sequence.
- Alice flipped her coin 100 times and got
THTTHHTHTHHHTTTTH ...
- But Bob claims Alice's sequence has probability 2^{-100} , and so does his.
- How do we define randomness?



How to define randomness has puzzled many

Laplace, von Mises, Wald, Church, Ville....

- P. Laplace: ... a sequence is extraordinary (nonrandom) because it contains rare regularity.
- ...



Laplace, 1749-1827

Measuring complexity of descriptions

Which of these two strings is more complex?

➤ A = 0101010101010101010101010101010101

➤ B = 1110010110100011101010000111

Kolmogorov Complexity

- The minimal description of x , written $d(x)$, is the shortest string $\langle M, w \rangle$ such that a Turing machine M on input w halts with x on its tape. The descriptive complexity (Kolmogorov complexity) of x ,

$K(x)$ is given by
$$K(x) = |d(x)|$$

- Kolmogorov complexity is unique modulo an additive constant

- Chain rule
$$K(x, y) \approx K(x) + K(y | x)$$



Kolmogorov complexity: applications

- Mathematics --- probability theory, logic.
- Physics --- chaos, thermodynamics.
- Computer Science – average case analysis, inductive inference and learning, shared information between documents, machine learning, algorithmic complexity,
- Philosophy, biology etc. – randomness, inference, complex systems, sequence similarity
- Information theory – information in individual objects, information distance
 - Classifying objects: documents, genomes
 - Approximating semantics



Properties of Kolmogorov complexity

- Intuitively: $C(x)$ = length of shortest description of x
- Define conditional Kolmogorov complexity similarly, $C(x|y)$ = length of shortest description of x given y .
- Examples
 - $C(xx) = C(x) + O(1)$
 - $C(xy) \leq C(x) + C(y) + O(\log(\min\{C(x), C(y)\}))$
 - $C(1^n) \leq O(\log n)$
 - $C(\pi_{1:n}) \leq O(\log n)$
 - For all x , $C(x) \leq |x| + O(1)$
 - $C(x|x) = O(1)$
 - $C(x|\epsilon) = C(x)$





Properties

Theorem (Kolmogorov) $C(x)$ is not partially recursive. That is, there is no Turing machine M s.t. M accepts (x,k) if $C(x) \geq k$ and undefined otherwise. However, there is $H(t,x)$ such that

$$\lim_{t \rightarrow \infty} H(t,x) = C(x)$$

where $H(t,x)$, for each fixed t , is total recursive.

Proof. If such M exists, then design M' as follows. Choose $n \gg |M'|$. M' simulates M on input (x,n) , for all $|x|=n$ in “parallel” (one step each), and outputs the first x such that M says yes. Thus we have a contradiction: $C(x) \geq n$ by M , but $|M'|$ outputs x hence

$$|x|=n \gg |M'| \geq C(x) \geq n.$$

QED



Godel's Theorem

Theorem. The statement “x is random” is not provable.

Proof (G. Chaitin). Let F be an axiomatic theory. $C(F) = C$. If the theorem is false and statement “x is random” is provable in F, then we can enumerate all proofs in F to find an $|x| \gg C$ and a proof of “x is random”, we output (first) such x. We have only used $C + O(1)$ bits in this proof to generate x, thus:

$$C(x) < C + O(1).$$

But the proof for “x is random” implies by definition:

$$C(x) \geq |x| \gg C.$$

Contradiction. QED



Transformative role of computing

Pre-Turing, Shannon, Kolmogorov

- Focus on the **physical basis of the universe**
- Explaining natural phenomena in terms of physical processes

Post-Turing, Shannon, Kolmogorov

- Focus on **informational basis of the universe**
- Explaining phenomena in algorithmic terms
 - processes that acquire, store, retrieve, organize, abstract, and use information



Summary

- Church-Turing thesis
 - Anything that is computable is computable by a Turing Machine!
 - Anything that is describable is describable by a computer program!
- Turing machines are universal computers!
- Computation is the best formalism we have for describing how information is processed in
 - Computers
 - Brains
 - Genomes
 - Organizations
 - societies ...



Algorithmic Abstractions of Intelligence



Artificial Intelligence: Driving Questions

Working hypothesis of Artificial Intelligence

Computation: Cognition: : Calculus : Physics

- How do we perceive?
- How do we learn?
- How do we reason?
- How do we make decisions?
- How do we plan and act?
- How do we create?
- How do we communicate?



How do we reason?

- Let us start with a simple case of reasoning
- Reasoning is essentially about substituting the potentially dangerous task of interacting directly with the physical world by a lot less dangerous process of “thinking”
 - Coffee is hot
 - You can find out that hot coffee burns your tongue by drinking it
 - Or... you can represent the key facts about the world and reason about the consequences of particular facts
- We understand reasoning via knowledge representation

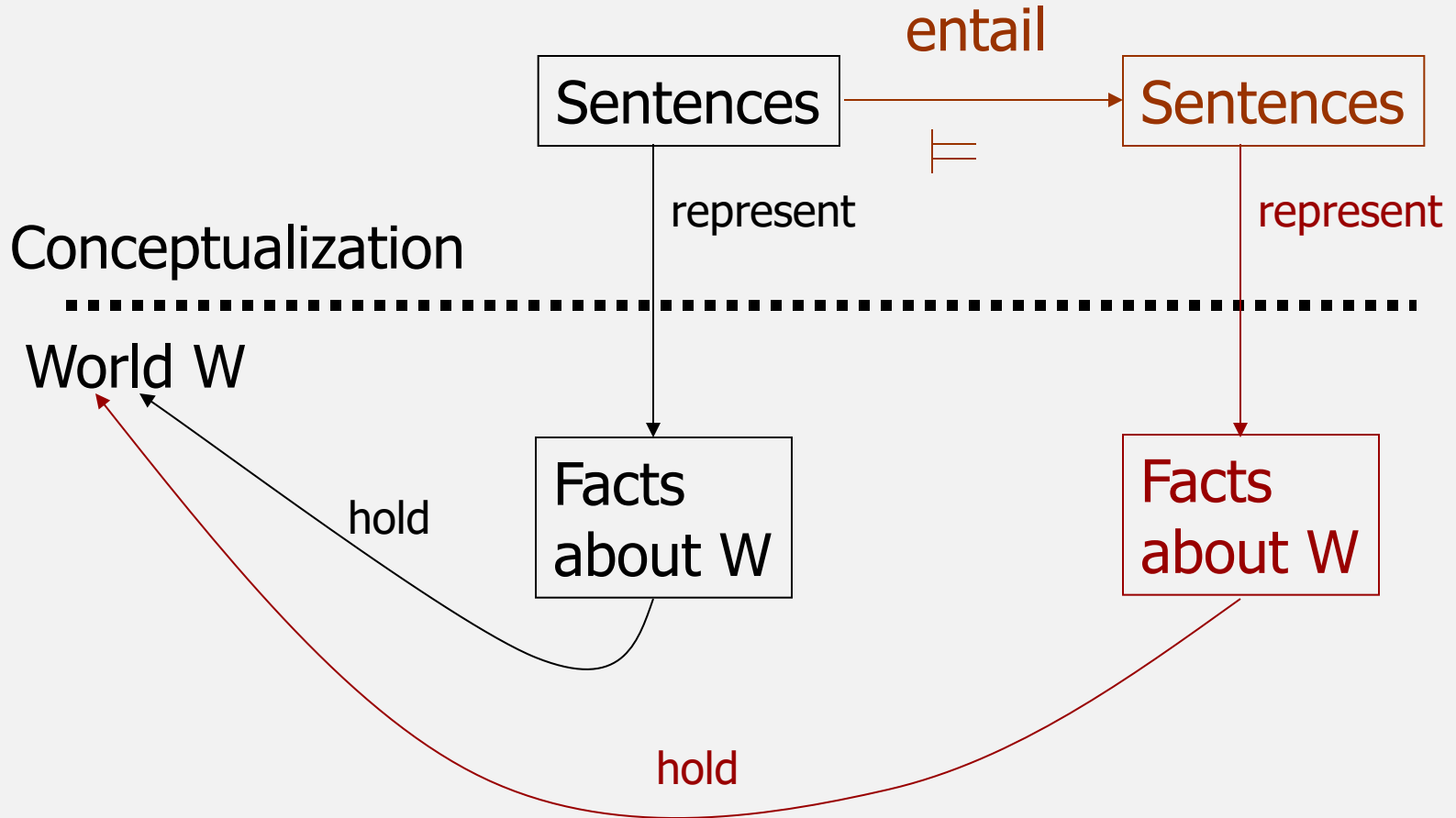


Knowledge representation (KR) is a **surrogate**

A **declarative** knowledge representation

- Encodes **facts that are true in the world** into **sentences**
- **Reasoning** is performed by manipulating **sentences** according to **sound** rules of **inference**
- The results of inference are **sentences** that **correspond** to **facts that are true in the world**
- The correspondence between facts that hold in the world and sentences that describe the world ensures **semantic grounding** of the representation
- Allows agents to **substitute thinking for acting** in the world
 - **Known facts:** The coffee is hot; coffee is a liquid; a hot liquid will burn your tongue;
 - **Inferred fact:** Coffee will burn your tongue

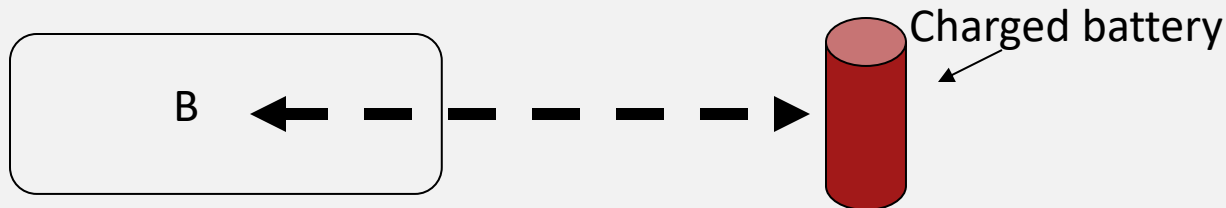
The nature of representation



Propositional Logic - Semantics

A proposition (sentence)

- **does not** have intrinsic meaning
- **gets its meaning from** correspondence with statements about the world (**interpretation**)



- Proposition **B** denotes the fact that **battery is charged**
- Is **True** or **False** in a chosen **interpretation**
- Logical connectives have no intrinsic meaning – need **interpretation**



KR is a set of ontological commitments

- What does an agent care about?
 - ✓ Entities
 - coffee, liquid, tongue
 - ✓ Properties
 - being hot, being able to burn
 - ✓ Relationships
 - Coffee **is a** liquid
- KR involves abstraction, simplification
 - A representation is
 - a (logical) model of the world
 - like a cartoon
 - **All models are wrong, but some are useful**



KR involves a set of epistemological commitments

- What can we know?
 - Propositional logic
 - Is a proposition *true* or *false*?
 - Probability theory
 - What is the *probability* that a given proposition true?
 - Decision theory
 - Which choice among a set of candidate choices is the most *rational*?
 - Logic of Knowledge
 - What does John *know* that Jim does not?





KR is a theory of intelligent reasoning

- How can knowledge be encoded ?
 - Syntax
- What does the encoded knowledge mean?
 - Semantics (entailment)
 - Inferences that are sanctioned by the semantics
- What can we infer from what we know?
 - Inferences that can be performed (by rules of inference)
 - Soundness, completeness, efficiency
- How can we manage inference?
 - What should we infer from among the things we can infer?



KR formalisms

KR formalisms provide provision for describing

- Individuals
- Sets of individuals (classes)
- Properties of individuals
- Properties of classes
- Relationships between individuals
- Relationships between classes
- Actions and their effects
- Locations and events in space and time
- Uncertainty
- Knowledge
- Beliefs
- Preferences
-



KR formalisms

- Logical
 - e.g., Propositional Logic, First order logic, Description logic
- Probabilistic
 - e.g., Bayesian networks
- Grammatical
 - e.g., Context free grammars
- Structured
 - e.g., frames – as in object-oriented programming
- Decision theoretic
- ...



KR is a medium for efficient computation

- Reasoning = computation
- Anticipated by Leibnitz, Hilbert
 - Can all truths be reduced to calculation?
 - Is there an effective procedure for determining whether or not a conclusion is a logical consequence of a set of facts?
- KR involves tradeoffs between
 - Expressivity and tractability (decidability, efficiency) tradeoff
 - The more you can say (using a representational formalism), the less you can effectively do (within the formalism)
 - General purpose reasoning versus special-purpose, domain-specific inference
 - Declarative versus procedural knowledge



KR is a medium of expression and communication

- If we assume shared
 - Ontological and epistemological commitments
 - KR formalism (syntax, semantics, reasoning)
- Then KR is a medium for
 - Expression
 - How general is it?
 - How precise is it?
 - Is it expressive enough?
 - Communication
 - Can we talk or think in the language?
 - Can we communicate the things we want to communicate?
 - What things are difficult to communicate?



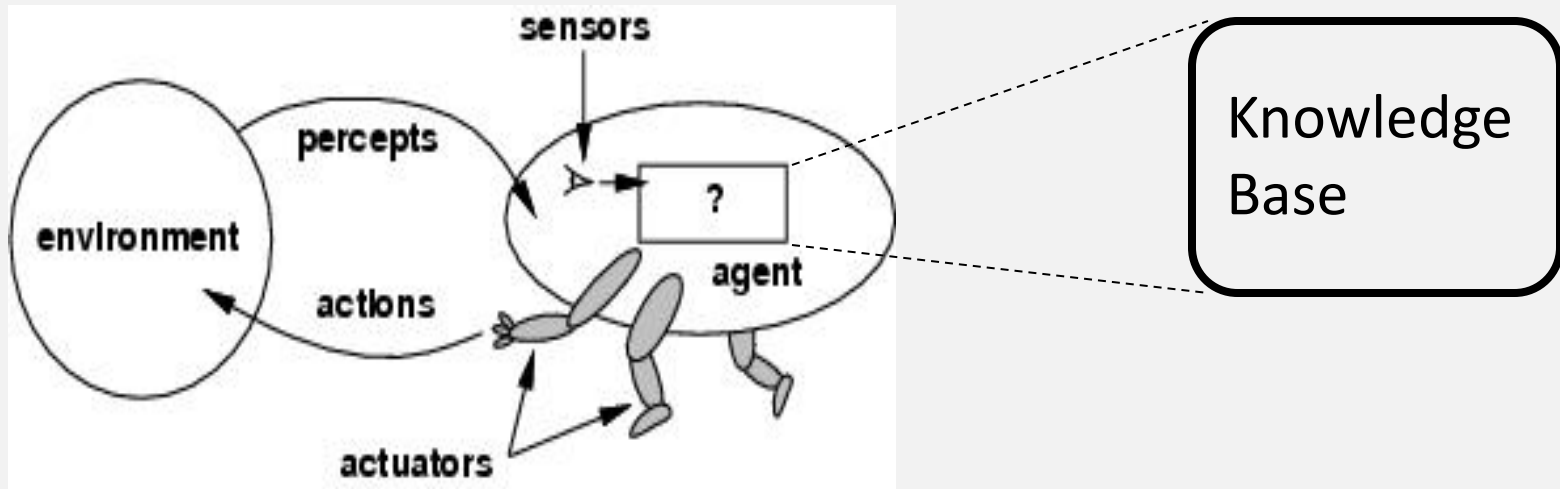
Logical Agents I – Propositional Logic

- Knowledge-based agents
- Logic in general - models and entailment
- Propositional (Boolean) logic
 - Syntax
 - Semantics
 - Equivalence, Validity, Satisfiability, Decidability
- Inference rules
 - Soundness, completeness
 - Resolution
- Inference procedures for automated theorem proving
 - Soundness, completeness, efficiency



Knowledge-Based Agents

- Knowledge-based agents represent and use knowledge about the world





A simple knowledge-based agent

Knowledge based agents

- Are not arbitrary programs
- Are amenable to **knowledge level description**
 - You can specify an agent's capability in terms of what it knows



Logic as a Knowledge Representation Formalism

Logic is a **declarative** language to:

- Assert sentences representing **facts that hold** in a world W (these sentences are given the value **true**)
- Deduce the **true/false** values to sentences representing other aspects of W



Examples of Logics

- Propositional logic $A \wedge B \rightarrow C$
- First-order predicate logic $\forall x \exists y \text{Mother}(y, x)$
- Logic of Knowledge $\mathbf{K}(\text{John, Father}(\text{Zeus, Cronus}))$
- Logic of Belief $\mathbf{B}(\text{John, Father}(\text{Zeus, Cronus}))$

Knowledge versus belief:

- You can *believe* things that are false
- You cannot *know* things that are false





Components of Logical KR

- A logical formalism
 - Syntax for well-formed-formulae (wff)
 - Vocabulary of logical symbols (**and**, **or**, **not**, **implies** etc.)
 - Interpretation semantics for logical symbols
 - e.g. **A or B holds** in the world whenever **A holds** in the world **or B holds** in the world
- An ontology
 - Vocabulary of non logical symbols
 - objects, properties (e.g., **A** above), etc.
 - definitions of non-primitive symbols (e.g., iff)
 - axioms restricting the interpretation of primitive symbols (more on this later)
- Proof theory – sound rules of inference



Propositional Logic

- **Atomic sentences** - propositions
 - Propositions can be **true** or **false**
- Statements can be combined using *logical connectives*.

Examples:

- C = “It’s cold outside”
 - C is a proposition
- O = “It’s October”
 - O is a proposition
- If O then C
 - if it’s October then it’s cold outside



Propositional Logic: Syntax, logical symbols

The proposition symbols P_1, P_2 etc are sentences

If S is a sentence, $\neg S$ is a sentence (negation)

If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)

If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)

If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)

If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

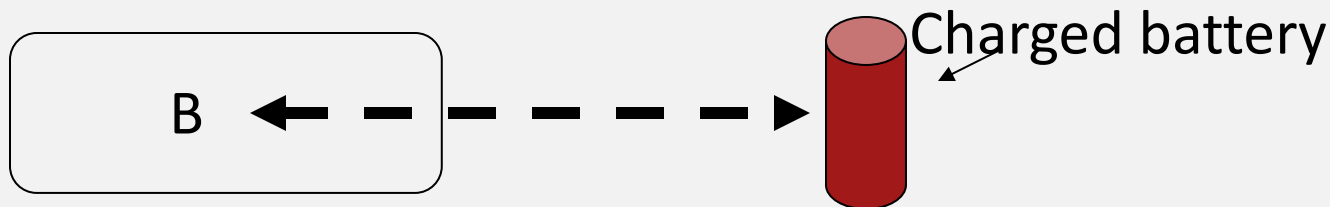
We use extra-linguistic symbols (e.g., parentheses) to group sentences to avoid ambiguity



Propositional Logic - Semantics

A Proposition

- **does not** have intrinsic meaning
- **gets its meaning from** correspondence with statements about the world (**interpretation**)



- Has a **denotation** in a given interpretation
 - e.g., proposition **B** denotes the fact that *battery is charged*
- Is **True** or **False** in a chosen **interpretation**
- Connectives do not have intrinsic meaning – need **interpretation**



Propositional Logic – Model Theoretic Semantics

- Consider a logic with only two propositions:
 - *Rich, Poor*
 - denoting *Tom is rich* and *Tom is poor* respectively
- A *model* M is a **subset** of the set A of **atomic sentences** in the language
- By a model M we mean the **state of affairs** in which
 - **every atomic sentence that is in M is *true*** and
 - **every atomic sentence that is not in M is *false***

Example

$$A = \{Rich, Poor\}$$

$$M_0 = \{ \}; M_1 = \{Rich\}; M_2 = \{Poor\}; M_3 = \{Rich, Poor\}$$



Propositional Logic: Semantics of Logical Symbols

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	false	true	false	false	true	true
false	true	true	false	true	true	false
true	false	false	false	true	false	false
true	true	false	true	true	true	true

Note that $(P \Rightarrow Q) \Leftrightarrow (\neg P \vee Q)$



Model Theoretic Semantics

$$A = \{Rich, Poor\}$$

$$M_0 = \{ \}; M_1 = \{Rich\}; M_2 = \{Poor\}; M_3 = \{Rich, Poor\}$$

Rich is True in M_1, M_3

Rich \vee *Poor* is True in M_1, M_2, M_3

Rich \wedge *Poor* is True in M_3 Hmm !!

Rich $\Rightarrow \neg$ *Poor* is True in M_0, M_1, M_2



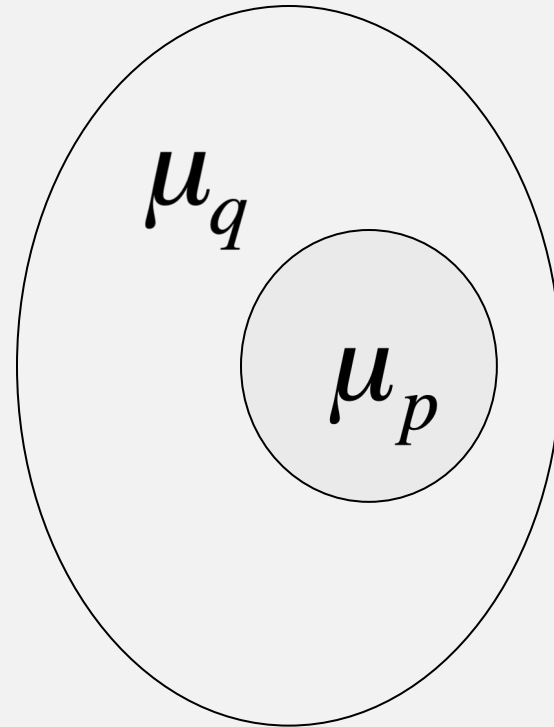
Proof Theory: Entailment

- We say that p **entails** q (written as $p \models q$) if q holds in **every** model in which p holds

μ_q = set of models in which q holds

μ_p = set of models in which p holds

$p \models q$ if it is the case that $\mu_p \subseteq \mu_q$



Entailment – Example

$$p \wedge (p \Rightarrow q) \models q$$

Proof

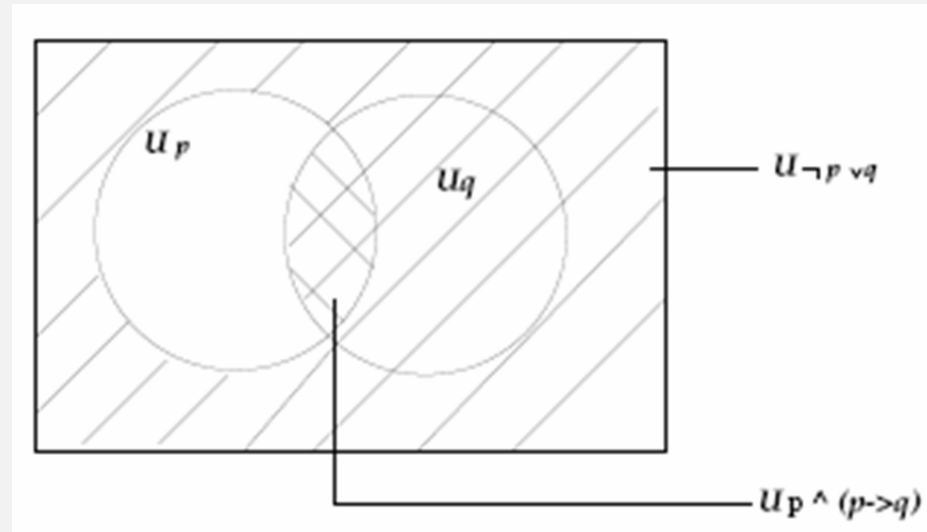
$$(p \in M) \wedge ((p \Rightarrow q) \in M)$$

$$(p \in M) \wedge ((\neg p \vee q) \in M)$$

$$((p \wedge \neg p) \vee (p \wedge q)) \in M$$

$$p \wedge q \in M$$

$$\therefore \mu_{p \wedge (p \Rightarrow q)} = \mu_{p \wedge q} \subseteq \mu_q$$





Validity and satisfiability

- A sentence is **valid** if it is true in **all** models,
 - e.g., *True*, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$
- A sentence is **satisfiable** if it is true in **some** model
 - e.g., $A \vee B$, C
- A sentence is **unsatisfiable** if it is true in **no** models
 - e.g., $A \wedge \neg A$
- Satisfiability is connected to inference via the following:
 - $KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable
 - Useful for proof by contradiction





Logical Rationality

- An propositional logic based agent A with a knowledge base KB_A is justified in inferring q if it is the case that

$$KB_A \models q$$

- How can the agent A decide whether in fact $KB_A \models q$?
 - Model checking
 - Enumerate all the models in which KB_A holds
 - Enumerate all the models in which q holds
 - Check whether $KB_A \subseteq \mu_q$
 - Inference algorithm based on inference rules





Searching for proofs: inference

- An **inference rule** $\{\xi, \psi\} \models \varphi$ consists of
 - 2 **sentence patterns** ξ and ψ called the **premises** and
 - one **sentence pattern** φ called the **conclusion**
- If ξ and ψ match two sentences of KB then
 - the corresponding φ can be inferred according to the rule
- Given a set of inference rules I and a knowledge base KB
 - **inference** is the process of successively applying inference rules from I to KB
 - each rule application adds its conclusion to KB



Inference rules

- *Modus ponens*

$$p \Rightarrow q$$

$$\frac{p}{q}$$

Modus ponens derives only inferences sanctioned by entailment

Modus ponens **is sound**

- *Loony tunes*

$$\frac{\textit{friday}}{q}$$

Loony tunes can derive inferences that are **not** sanctioned by entailment

Loony tunes **is not sound**



Example: Inference using Modus Ponens

$$\{ p \Rightarrow q, p \} \vdash q$$

$$\{ \xi, \psi \} \vdash \varphi$$

KB:

Battery-OK \wedge *Bulbs-OK* \Rightarrow *Headlights-Work*

Battery-OK \wedge *Starter-OK* \wedge \neg *Empty-Gas-Tank* \Rightarrow *Engine-Starts*

Engine-Starts \wedge \neg *Flat-Tire* \wedge *Headlights-Work* \Rightarrow *Car-OK*

Battery-OK, *Bulbs-OK*, *Starter-OK*, \neg *Empty-Gas-Tank*, \neg *Flat-Tire*

Ask

Car-OK?





Soundness and Completeness of an inference rule \vdash

- We write $p \vdash q$ to denote that that p can be **inferred from q using the inference rule \vdash**

An inference rule \vdash is said to be

- **Sound** if whenever $p \vdash q$, it is also the case that $p \models q$
- **Complete** if whenever $p \models q$, it is also the case that $p \vdash q$



Soundness and Completeness of an inference rule \vdash

- We can show that *modus ponens* is sound, but *not* complete unless the KB is *Horn* i.e., the KB can be written as a collection of sentences of the form

$$a_1 \wedge a_2 \wedge \dots \wedge a_n \Rightarrow b$$

where each a_i and b are atomic sentences



Unsound inference rules are not necessarily useless!

Abduction (Charles Peirce) is **not sound**, but useful in diagnostic reasoning or hypothesis generation

$$p \Rightarrow q$$

$$\frac{q}{\quad}$$

$$p$$

$$\textit{BlockedArtery} \Rightarrow \textit{HeartAttack}$$

$$\frac{\textit{HeartAttack}}{\quad}$$

$$\textit{BlockedArtery}$$





Logical equivalence

- Two sentences are **logically equivalent** iff they are true in same set of models or $\alpha \equiv \beta$ iff $\alpha \models \beta$ and $\beta \models \alpha$.

$$\begin{aligned}
 (\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
 (\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
 ((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
 ((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
 \neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
 (\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
 (\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
 (\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
 \neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
 \neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
 (\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
 (\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
 \end{aligned}$$



Sound Inference rules in PL

- Modus Ponens
$$\frac{\alpha \Rightarrow \beta, \alpha}{\beta}$$

- And-elimination: from a conjunction any conjunct can be inferred:

$$\frac{\alpha \wedge \beta}{\alpha}$$

- All logical equivalences can be used as inference rules

$$\frac{\alpha \Leftrightarrow \beta}{(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)}$$

- An inference procedure involves repeated application of applicable inference rules.
- An inference procedure is sound if it uses only sound inference rules.





Constructing proofs

- Finding proofs can be cast as a search problem
- Search can be
 - forward (forward chaining) to derive *goal* from *KB*
 - or backward (backward chaining) from the *goal*
- Searching for proofs
 - Is not more efficient than enumerating models in the worst case
 - Can be more efficient in practice with the use of suitable heuristics
- **Propositional logic is monotonic**
 - the set of entailed sentences can only increase as inferred facts are added to KB

for any sentences α and β :

if $KB \models \alpha$ then $KB \wedge \beta \models \alpha$





Soundness and Completeness

- An inference algorithm starts with the KB and applies applicable inference rules until the desired conclusion is reached
- An inference algorithm is sound if it uses a sound inference rule
- An inference algorithm is complete if
 - It uses a **complete inference rule** and
 - a **complete** search procedure



Soundness of Modus Ponens for Propositional Logic

- **Modus Ponens is sound for Propositional Logic**
- Modus Ponens is complete for Horn KB:
 - Whenever $KB \models q$, repeated application of Modus ponens is guaranteed to infer q if the KB consists of only Horn Clauses, i.e., only sentences of the form: $p_1 \wedge p_2 \wedge \dots \wedge p_n \Rightarrow q$
 - Complexity of inference is polynomial in the size of the Horn KB
- **Modus Ponens is not complete for arbitrary Propositional KB**



Completeness of Modus Ponens for Propositional Logic

- **Modus Ponens is not complete for Propositional Logic**
- Suppose that all classes at some university meet either Mon/Wed/Fri or Tue/Thu. The AI course meets at 2:30 PM in the afternoon, and Jane has volleyball practice Thursdays and Fridays at that time.
- Can Jane take AI?

MonWedFriAI230pm \vee TueThuAI230pm

TueThuAI230pm \wedge JaneBusyThu230pm \rightarrow JaneConflictAI

MonWedFriAI230pm \wedge JaneBusyFri230pm \rightarrow JaneConflictAI

JaneBusyThu230pm

JaneBusyFri230pm



Completeness of Modus Ponens for Propositional Logic

- Modus Ponens is not complete for Propositional Logic
- Can Jane take AI?

MonWedFriAI230pm \vee TueThuAI230pm

TueThuAI230pm \wedge JaneBusyThu230pm \rightarrow JaneConflictAI

MonWedFriAI230pm \wedge JaneBusyFri230pm \rightarrow JaneConflictAI

JaneBusyThu230pm

JaneBusyFri230pm

- Of course not!
- Try proving this using Modus Ponens
- You can't!
- Why?



Completeness of Modus Ponens for Propositional Logic

1. $MonWedFriAI230pm \vee TueThuAI230pm$

2. $TueThuAI230pm \wedge JaneBusyThu230pm \rightarrow JaneConflictAI$

3. $MonWedFriAI230pm \wedge JaneBusyFri230pm \rightarrow JaneConflictAI$

4. $JaneBusyThu230pm$

5. $JaneBusyFri230pm$

- We can use Modus Ponens to establish

2&4: $TueThuAI230pm \rightarrow JaneConflictAI$

3&4: $MonWedFriAI230pm \rightarrow JaneConflictAI$

But Modus Ponens can't take us further to conclude $JaneConflictAI$!

- Modus Ponens is not complete for Propositional Logic (except in the restricted case when the KB is Horn)
- However, we can generalize Modus Ponens to obtain an inference rule for Propositional Logic





Proof

- The **proof** of a sentence α from a set of sentences KB is the derivation of α obtained through a series of applications of sound inference rules to KB



Inference with Horn KB

- Each Horn clause has only one positive, i.e., un-negated, literal
- Inference can be done with forward or backward chaining
- Entailment decidable in time linear in the size of propositional KB
- Prolog

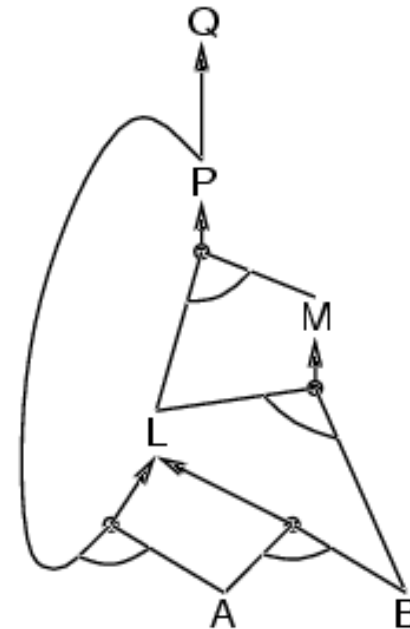
Forward chaining

- Idea: Apply any rule whose premises are satisfied in the *KB*,
 - add its conclusion to the *KB*, until query is proved.
 - Example: Query: *Q*

KB

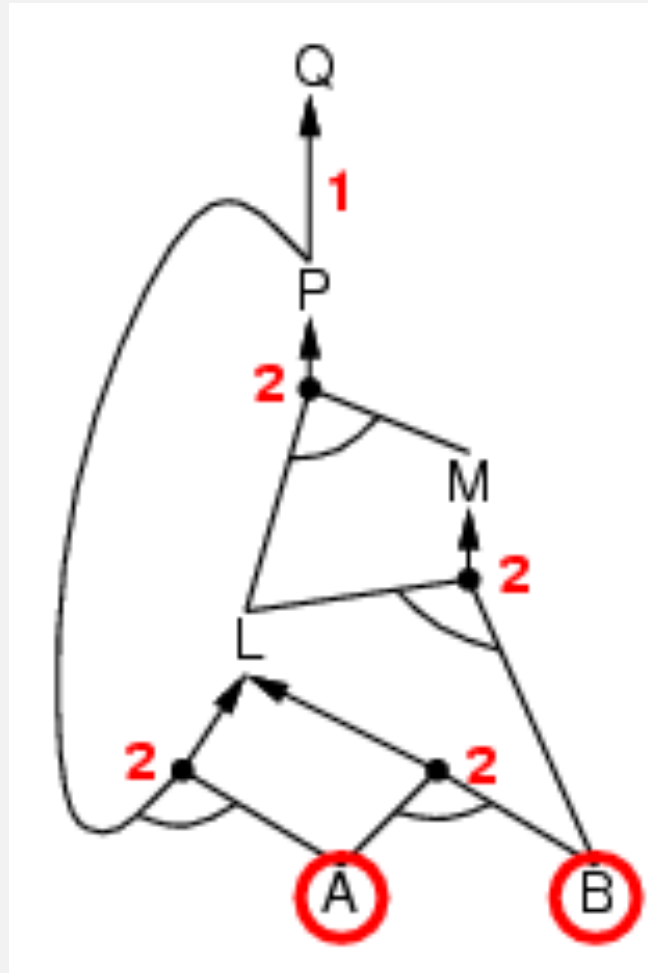
$P \Rightarrow Q$
 $L \wedge M \Rightarrow P$
 $B \wedge L \Rightarrow M$
 $A \wedge P \Rightarrow L$
 $A \wedge B \Rightarrow L$
 A
 B

Forward chaining proof of *Q*

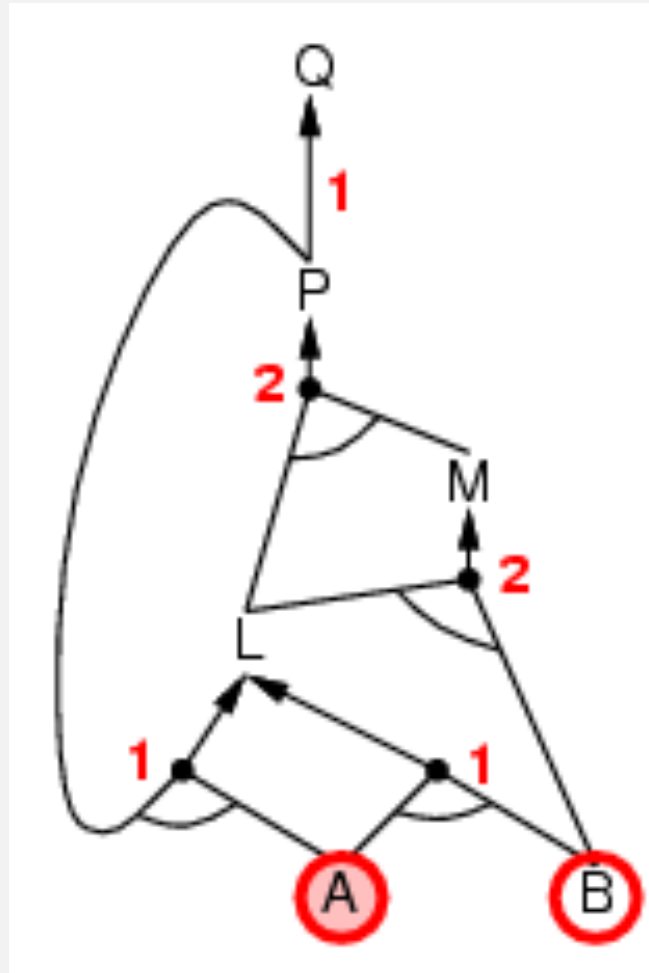


- Forward chaining is sound and complete for Horn KB

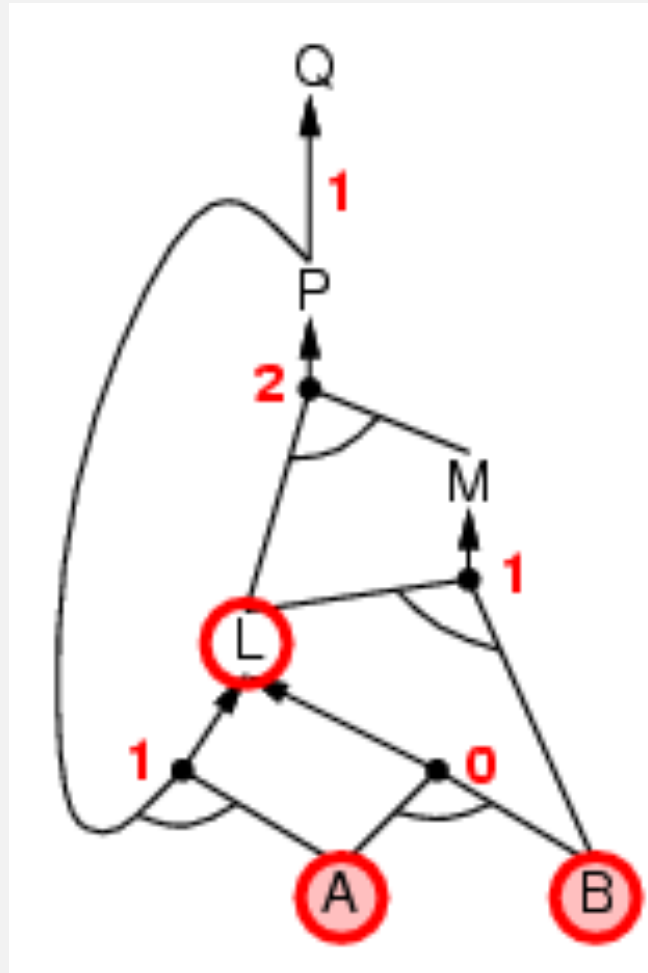
Forward chaining example



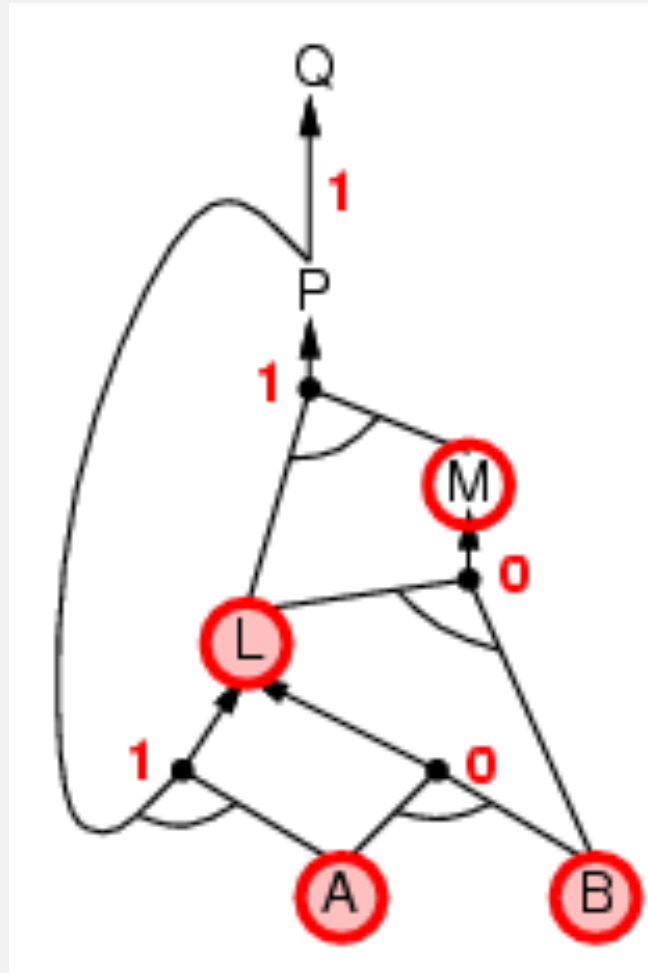
Forward chaining example



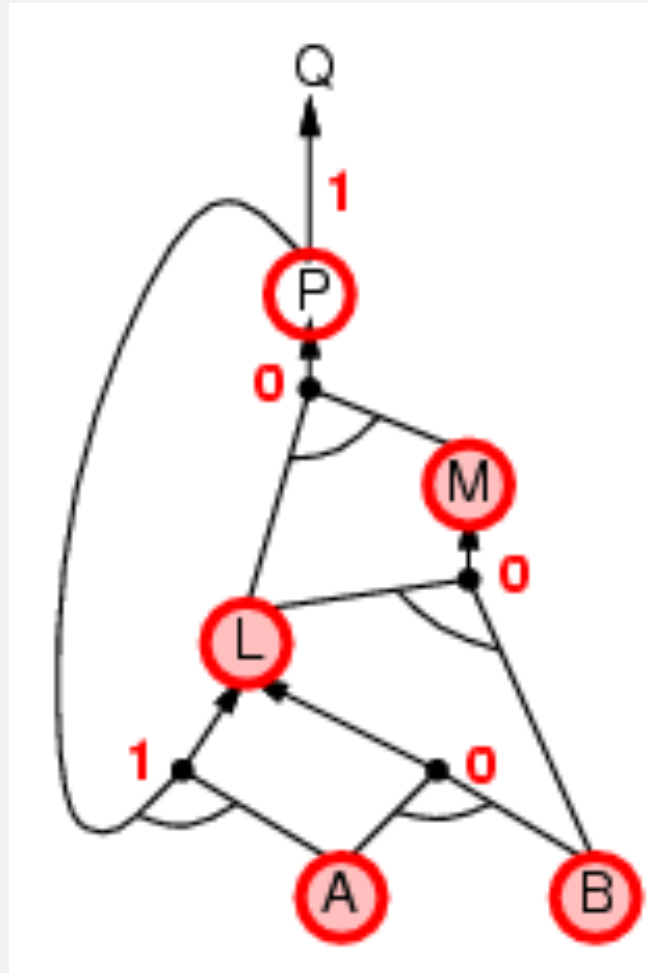
Forward chaining example



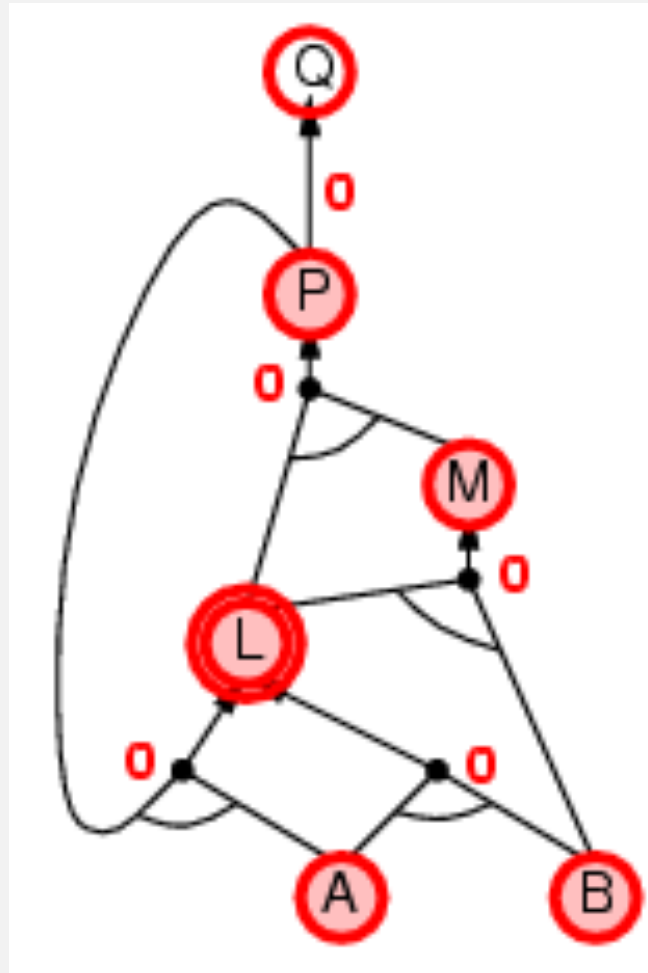
Forward chaining example



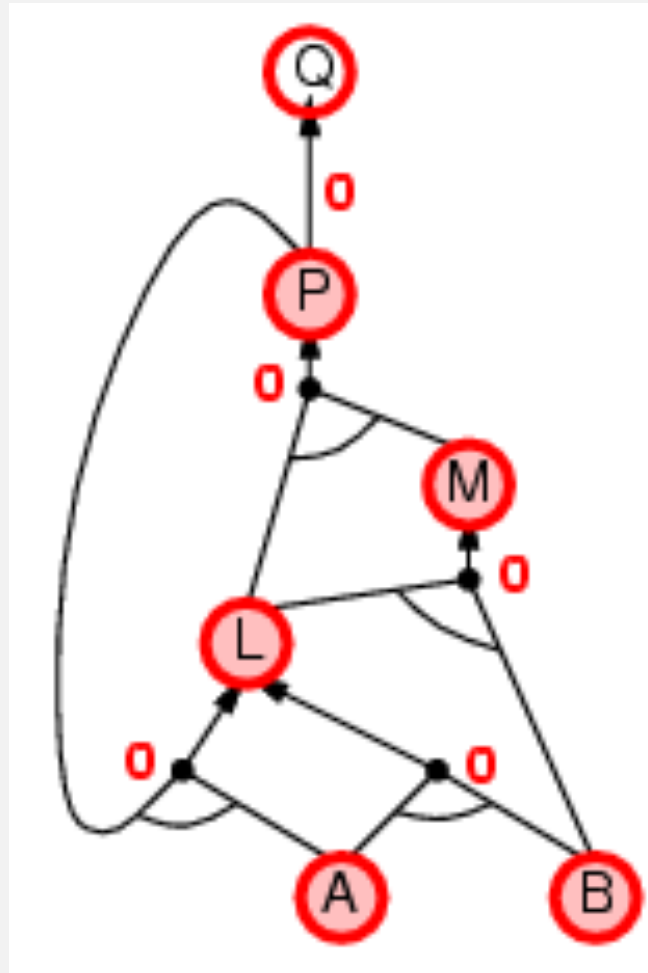
Forward chaining example



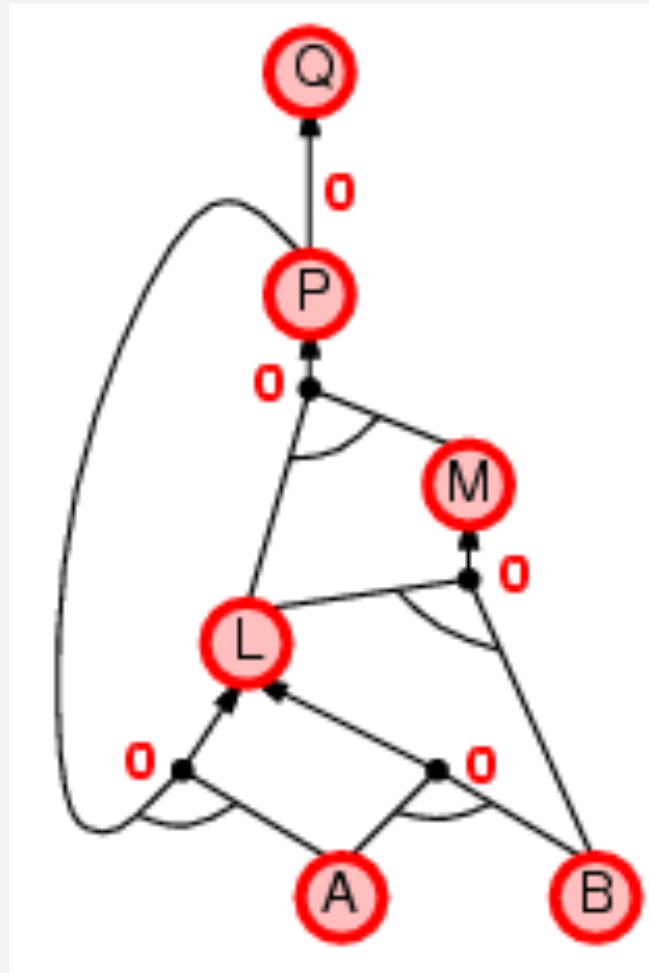
Forward chaining example



Forward chaining example



Forward chaining example





Soundness and Completeness of Forward Chaining

- An inference algorithm starts with the KB and applies applicable inference rules until the desired conclusion is reached
- An inference algorithm is sound if it uses a sound inference rule
- An inference algorithm is complete if
 - It uses a **complete inference rule** and
 - a **complete** search procedure
- Forward chaining using Modus Ponens is sound and complete for Horn knowledge bases (i.e., knowledge bases that contain only Horn clauses)



Backward chaining

Idea: work backwards from the query q :

to prove q by BC,

check if q is known already, or

prove by BC all premises of some rule concluding q

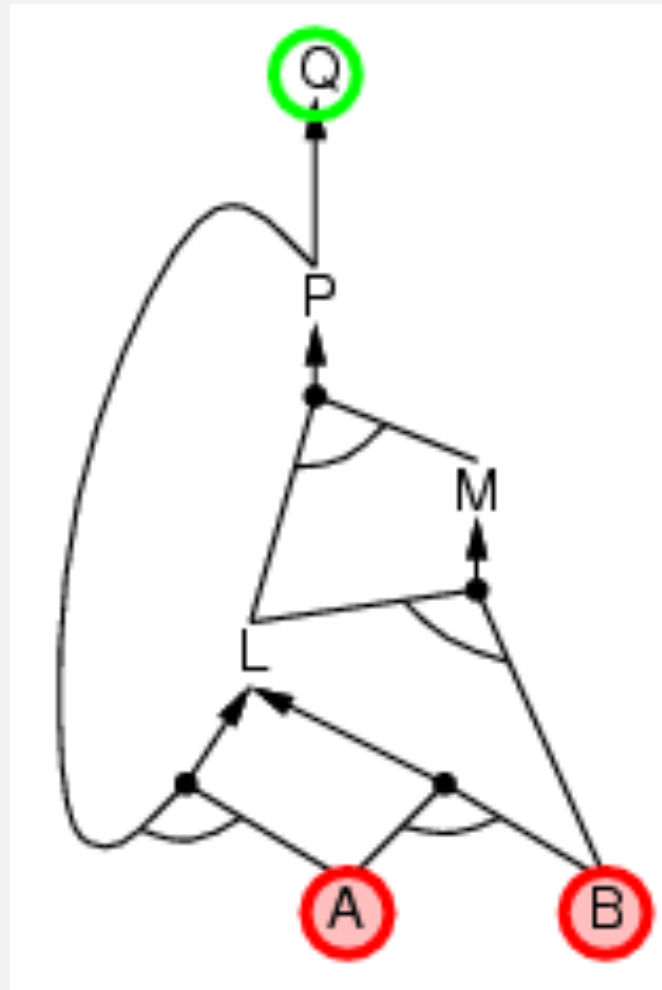
Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

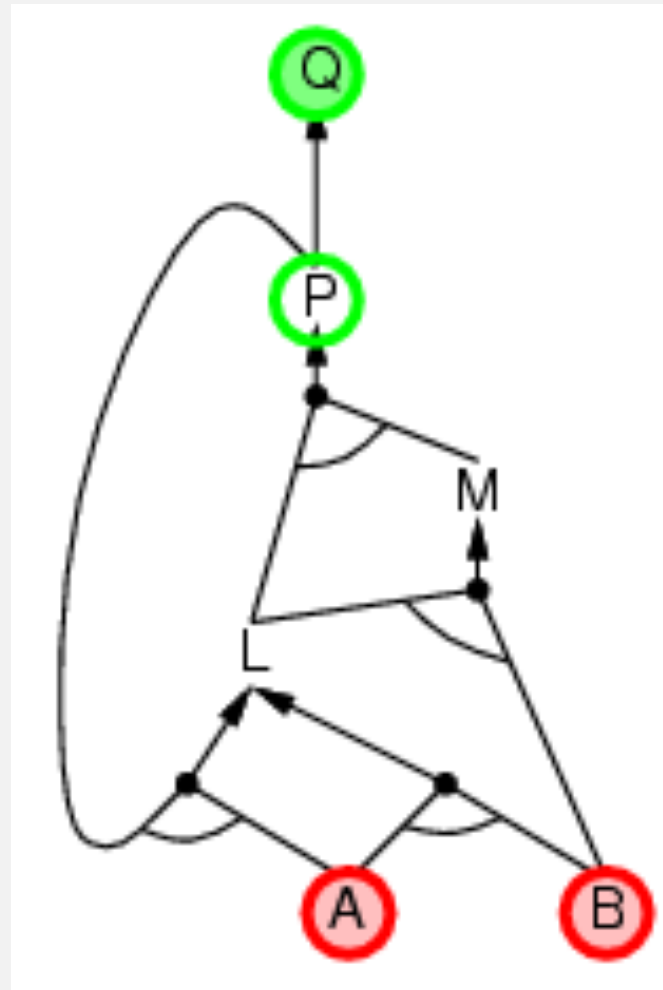
- has already been proved true, or
- has already failed



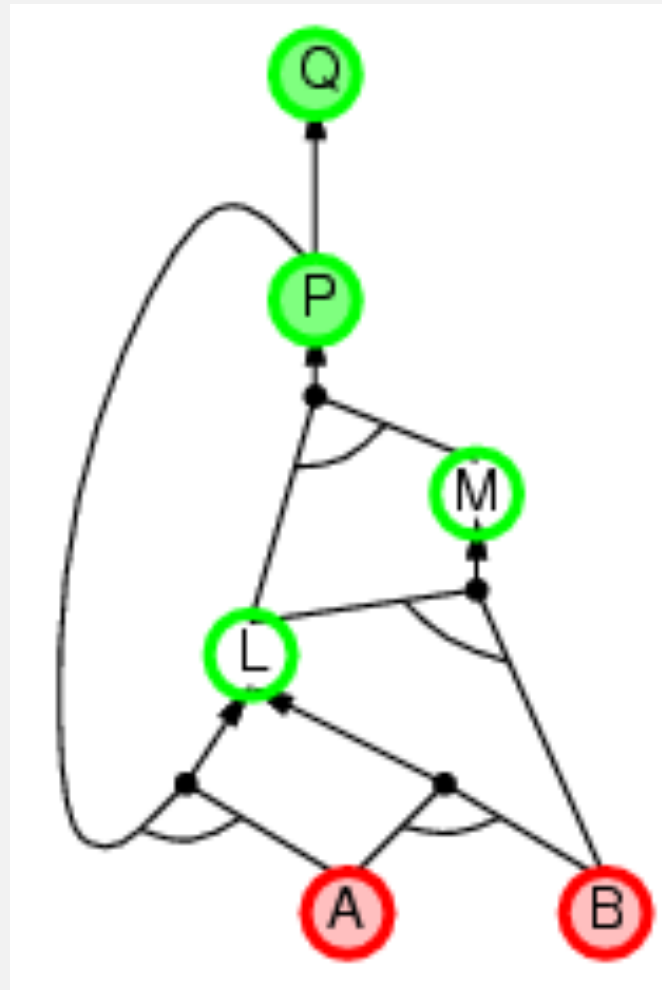
Backward chaining example



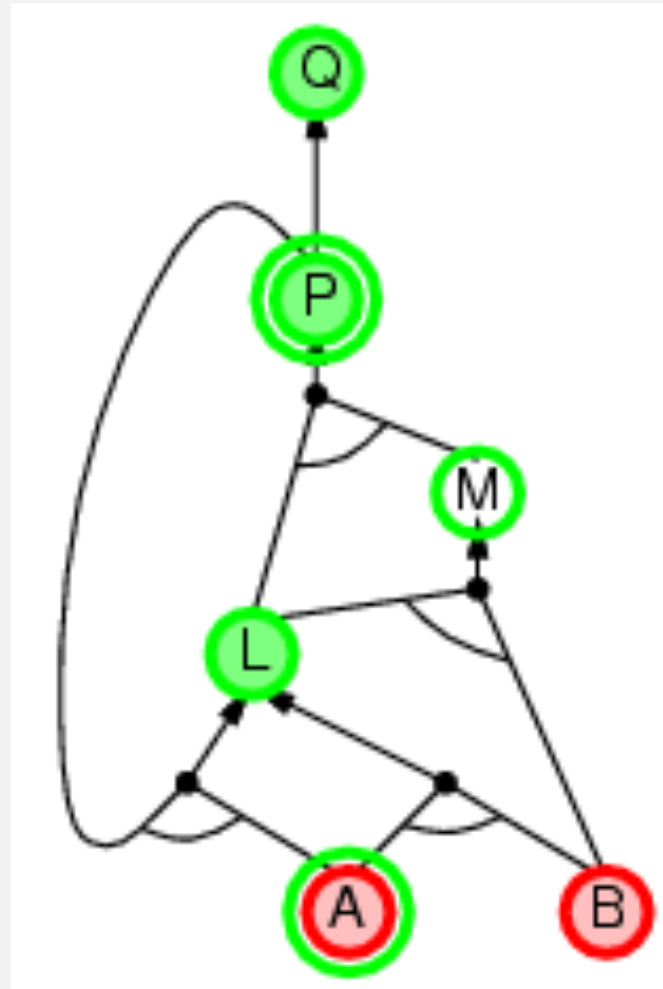
Backward chaining example



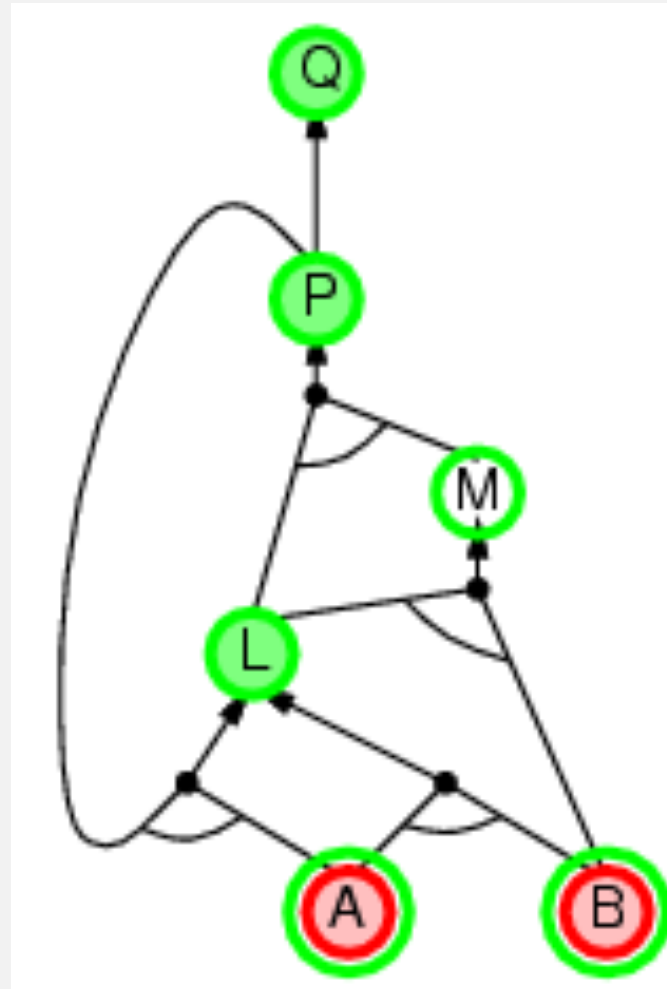
Backward chaining example



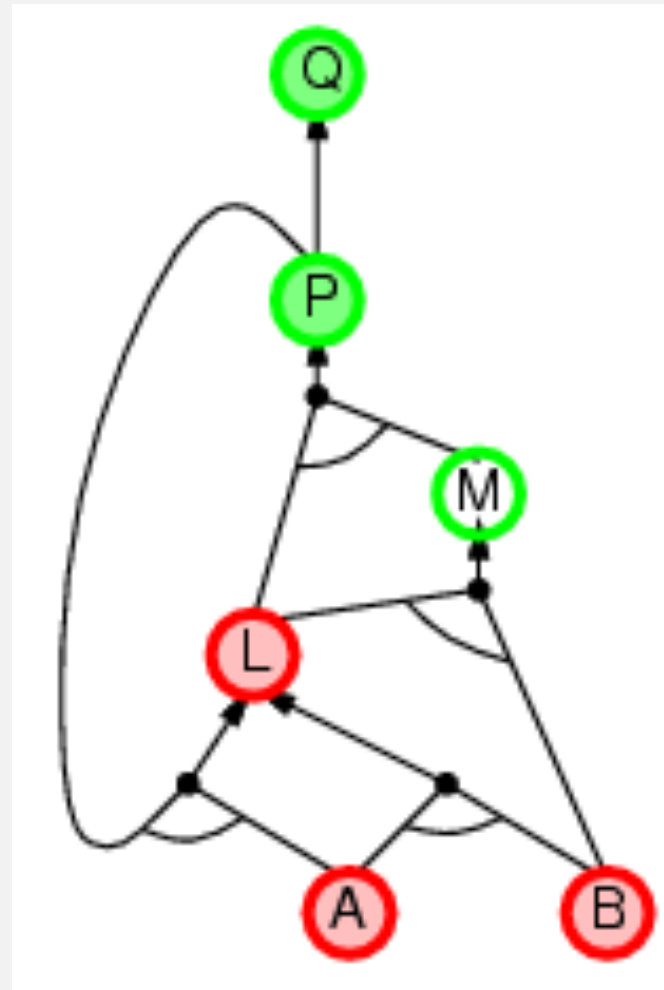
Backward chaining example



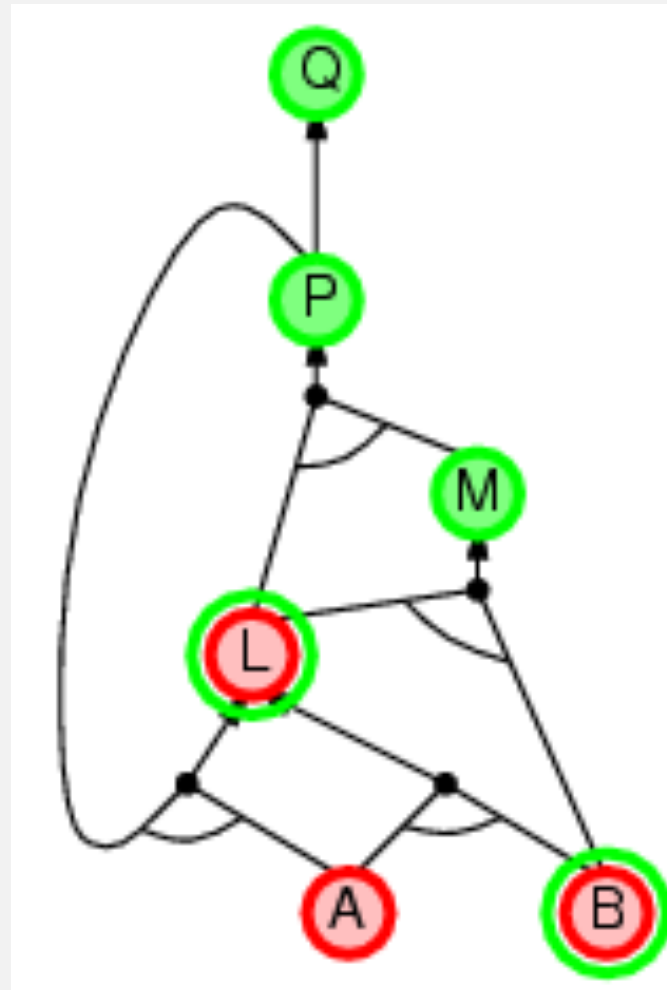
Backward chaining example



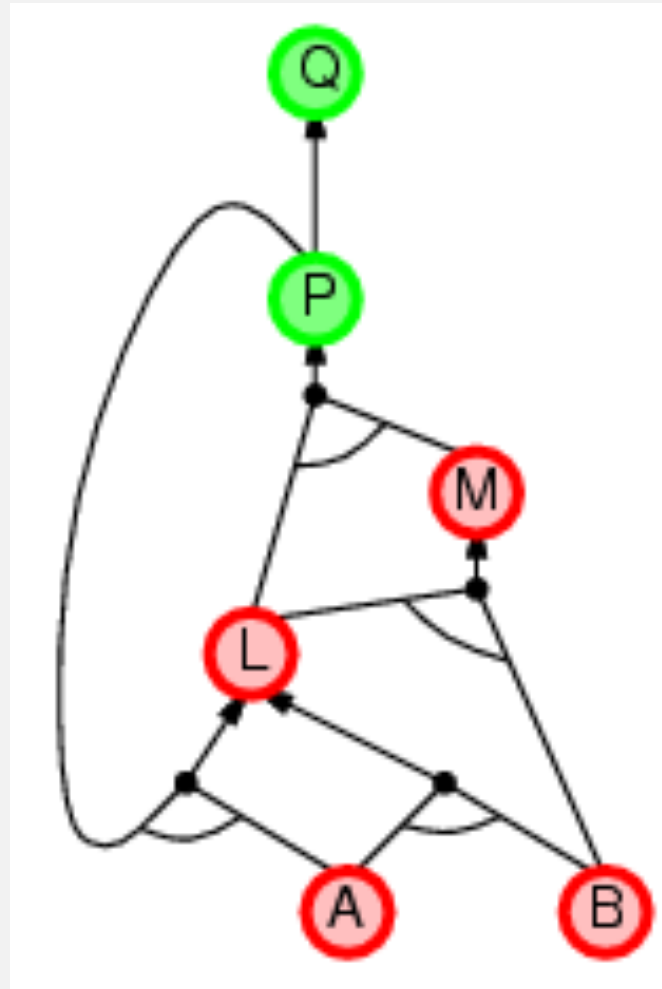
Backward chaining example



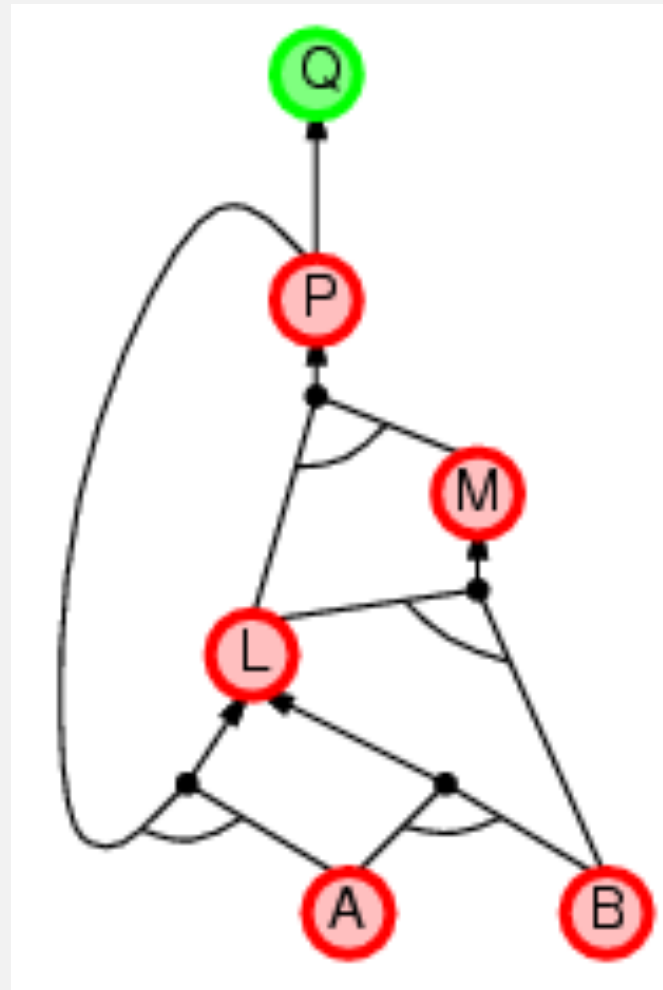
Backward chaining example



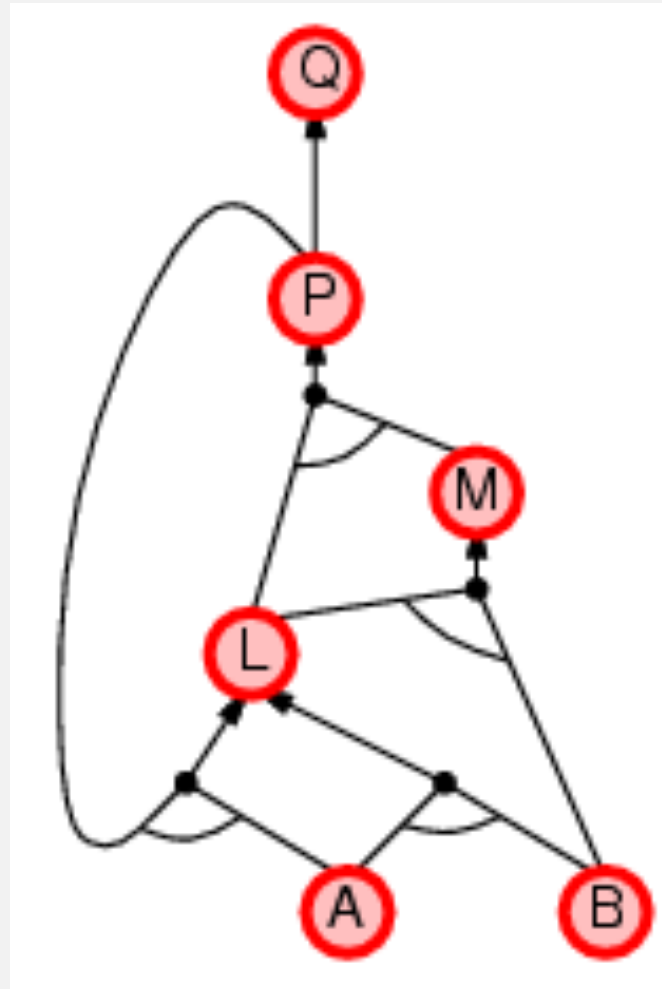
Backward chaining example



Backward chaining example



Backward chaining example





Soundness and Completeness of Forward Chaining

- An inference algorithm starts with the KB and applies applicable inference rules until the desired conclusion is reached
- An inference algorithm is sound if it uses a sound inference rule
- An inference algorithm is complete if
 - It uses a **complete inference rule** and
 - a **complete** search procedure
- Backward chaining using Modus Ponens is sound and complete for Horn knowledge bases (i.e., knowledge bases that contain only Horn clauses)



Forward vs. backward chaining

- FC is **data-driven**, automatic, unconscious processing,
 - Akin to day dreaming...
- May do lots of work that is irrelevant to the goal
- BC is **goal-driven**, appropriate for problem-solving
 - e.g., Where are my keys? How do I get into a PhD program?
- The run time of FC is linear in the size of the KB.
- The run time of BC can be, in practice, **much less** than linear in size of *KB*

