Final Review Slides
CSC384

PROBABILITY + BAYES NETS

PROBABILITY: THE AXIOMS

Given U (universe of events), a probability function is a function defined over subsets of U that maps each subset onto the real numbers and that satisfies the Axioms of Probability, which are:

1. $Pr(U) = 1$
2. $Pr(A) \in [0,1]$
3. $Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$

NB: if $A \cap B = \{\}$ then $Pr(A \cup B) = Pr(A) + Pr(B)$

PROBABILITY: JOINT DISRIBUTION, CHAIN RULE

A joint distribution: $Pr(A_1 \wedge A_2 \wedge \ldots \wedge A_n)$

Decomposing the joint via the chain rule:

$Pr(A_1 \wedge A_2 \wedge \ldots \wedge A_n) =$

$Pr(A_1 | A_2 \wedge \ldots \wedge A_n) * Pr(A_2 | A_3 \wedge \ldots \wedge A_n)$

$* \ldots * Pr(A_{n-1} | A_n) * Pr(A_n)$

(Remember the Proof?)

PROBABILITY: CONDITIONAL PROBABILITY, INDEPENDENCE

Conditional Probability Definition

- $Pr(B|A) = Pr(B \cap A)/Pr(A)$

Properties of Independent Variables

- $Pr(B|A) = Pr(B)$
- Implies $Pr(A \wedge B) = Pr(B) * Pr(A)$ *(Remember the proof?)*

Properties of Dependent Variables

- $Pr(B|A) \neq Pr(B)$

Properties of Conditionally Independent Variables

- $Pr(B \wedge C|A) = Pr(B|A) * Pr(C|A)$

PROBABILITY: SUMMING OUT A VARIABLE, MARGINALIZING

Given joint distribution $Pr(A,B)$. We can sum out B to create a distribution over A alone:

$Pr(A) = Pr(A \cap B_1) + Pr(A \cap B_2) + \ldots + Pr(A \cap B_k)$

Or

$Pr(A) = Pr(A|B_1)Pr(B_1) + Pr(A|B_2)Pr(B_2) + \ldots + Pr(A|B_k)Pr(B_k)$

This is called marginalizing the distribution, as it creates a marginal distribution over A.

PROBABILITY: BAYES RULE

$$\Pr(Y|X) = \Pr(X|Y)\Pr(Y)/\Pr(X)$$

(Remember how to derive this?)

CONDITIONAL INDEPENDENCE: BENEFITS

Conditional independence allows us to break up our computation onto distinct parts

$$P(B \wedge C|A) = P(B|A) * P(C|A)$$

And it also allows us to ignore certain pieces of information

$$P(B|A \wedge C) = P(B|A)$$

This yields computational savings.

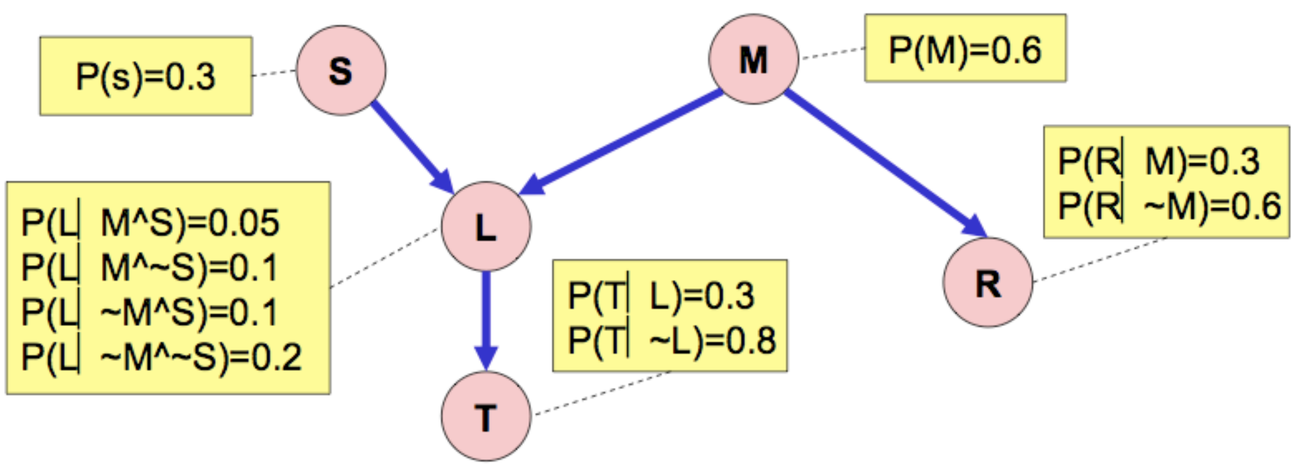
BAYESIAN NETWORKS CAPITALIZE ON BENEFITS

A BN over variables $\{X_1, X_2, \ldots, X_n\}$ consists of:

- a directed acyclic graph (DAG) whose nodes are the variables
- a set of conditional probability tables (CPTs) that specify $Pr(X_i \mid Parents(X_i))$ for each X_i

Key definitions

- parents of a node: $Parents(X_i)$
- children of node
- descendants of a node
- ancestors of a node
- family of a node (consists of X_i and its parents)
- CPTs are defined over families in the BN



BUILDING A BAYESIAN NETWORK

YOU MAY USE ANY ORDERING OF THE VARIABLES.

From the chain rule we obtain.

$$Pr(X_1,\ldots,X_n) = Pr(X_n|X_1,\ldots,X_{n-1})Pr(X_{n-1}|X_1,\ldots,X_{n-2})\ldots Pr(X_1)$$

Now for each X_i go through its conditioning set $X_1,\ldots,X_{i-1}$, and iteratively remove all variables X_j such that X_i is conditionally independent of X_j given the remaining variables. Do this until no more variables can be removed.

The final product will specify a Bayes net.

BUT note that not all Bayes Nets are equal!

Remember the benefit of Causal Intuitions ….

BAYESIAN NETWORK: INFERENCE

Given a Bayes net

$P(X_1, X_2,\ldots, X_n) = P(X_n \mid P(\text{Parents}(X_n))) *$

$P(X_{n-1} \mid P(\text{Parents}(X_{n-1}))) * \ldots * P(X_1 \mid P(\text{Parents}(X_1)))$

And some evidence

E = {a set of values for some of the variables}

Compute the new probability distribution

$P(X_k \mid E)$

That is, we want to figure our

$P(X\_k = d \mid E)$ for all $d \in \text{Dom}[X_k]$

VARIABLE ELIMINATION

Variable elimination is a technique that uses the product decomposition that defines a Bayes Net and the summing out rule to compute posterior probabilities from information in the network (CPTs).

= P(a)P(b) P(d|a,b) $\sum_C$ P(C|a) $\sum_E$ P(E|C)
$\sum_F$ P(F|d) P(h|E,F)$\sum_G$ P(G) P(-i|F,G)
$\sum_J$ P(J|h,-i)
$\sum_K$ P(K|-i)
+
P(a)P(-b) P(d|a,-b) $\sum_C$ P(C|a) $\sum_E$ P(E|C)
$\sum_F$ P(F|d) P(h|E,F)$\sum_G$ P(G) P(-i|F,G)
$\sum_J$ P(J|h,-i)
$\sum_K$ P(K|-i)
+
P(-a)P(b) P(d|-a,b) $\sum_C$ P(C|-a) $\sum_E$ P(E|C)
$\sum_F$ P(F|d) P(h|E,F)$\sum_G$ P(G) P(-i|F,G)
$\sum_J$ P(J|h,-i)
$\sum_K$ P(K|-i)
+
P(-a)P(-b) P(d|-a,-b) $\sum_C$ P(C|-a) $\sum_E$ P(E|C)
$\sum_F$ P(F|d) P(h|E,F)$\sum_G$ P(G) P(-i|F,G)
$\sum_J$ P(J|h,-i)
$\sum_K$ P(K|-i)

Repeated subterm

Repeated subterm

Capitalizes on repetition in sub-terms. Note that remembering “smaller” computations is a core idea of dynamic programming; VE is a dynamic programming technique.

VARIABLE ELIMINATION: ALGORITHM

Given query var **Q**, evidence vars **E** (set of variables observed to have values **e**), and remaining vars **Z**. Let **F** be factors in original CPTs.

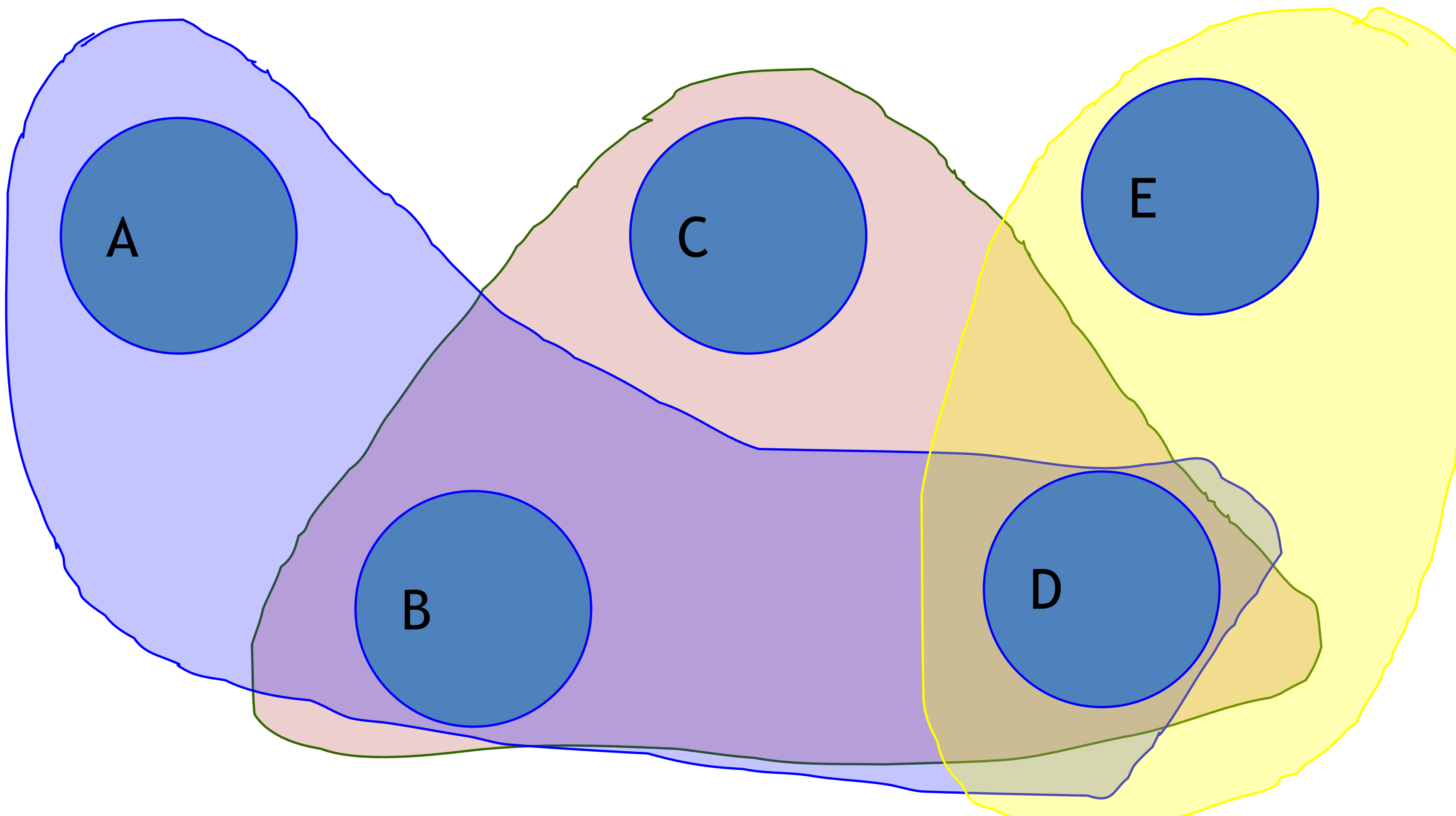
1. Replace each factor $f \in \mathbf{F}$ that mentions a variable(s) in **E** with its restriction $f_{\mathbf{E=e}}$ (this might yield a "constant" factor)
2. For each $\mathbf{Z_j}$ - in the order given - eliminate $\mathbf{Z_j} \in \mathbf{Z}$ as follows:

   (a) Compute new factor $g_j = \sum_{Z_j} f_1 \times f_2 \times \ldots \times f_k$, where the f_i are the factors in **F** that include $\mathbf{Z_j}$

   (b) Remove the factors f_i (that mention $\mathbf{Z_j}$) from **F** and add new factor g_j to **F**
3. The remaining factors at the end of this process will refer only to the query variable **Q**. Take their product and normalize to produce **P(Q|E)**.

VARIABLE ELIMINATION: COMPLEXITY

Complexity depends on the hypergraph and hyperedges of the Bayes Net in question.



{A,B,D}
{B,C,D}
{E,D}

The maximum size (number of variables) of any hyperedge in any of the hypergraphs that are created during variable elimination determines the complexity of the process.

This size is called the elimination width.

VARIABLE ELIMINATION: COMPLEXITY

Different orderings = different elimination width.

Try E,C,A,B,G,H,F vs. A,F,H,G,B,C,E

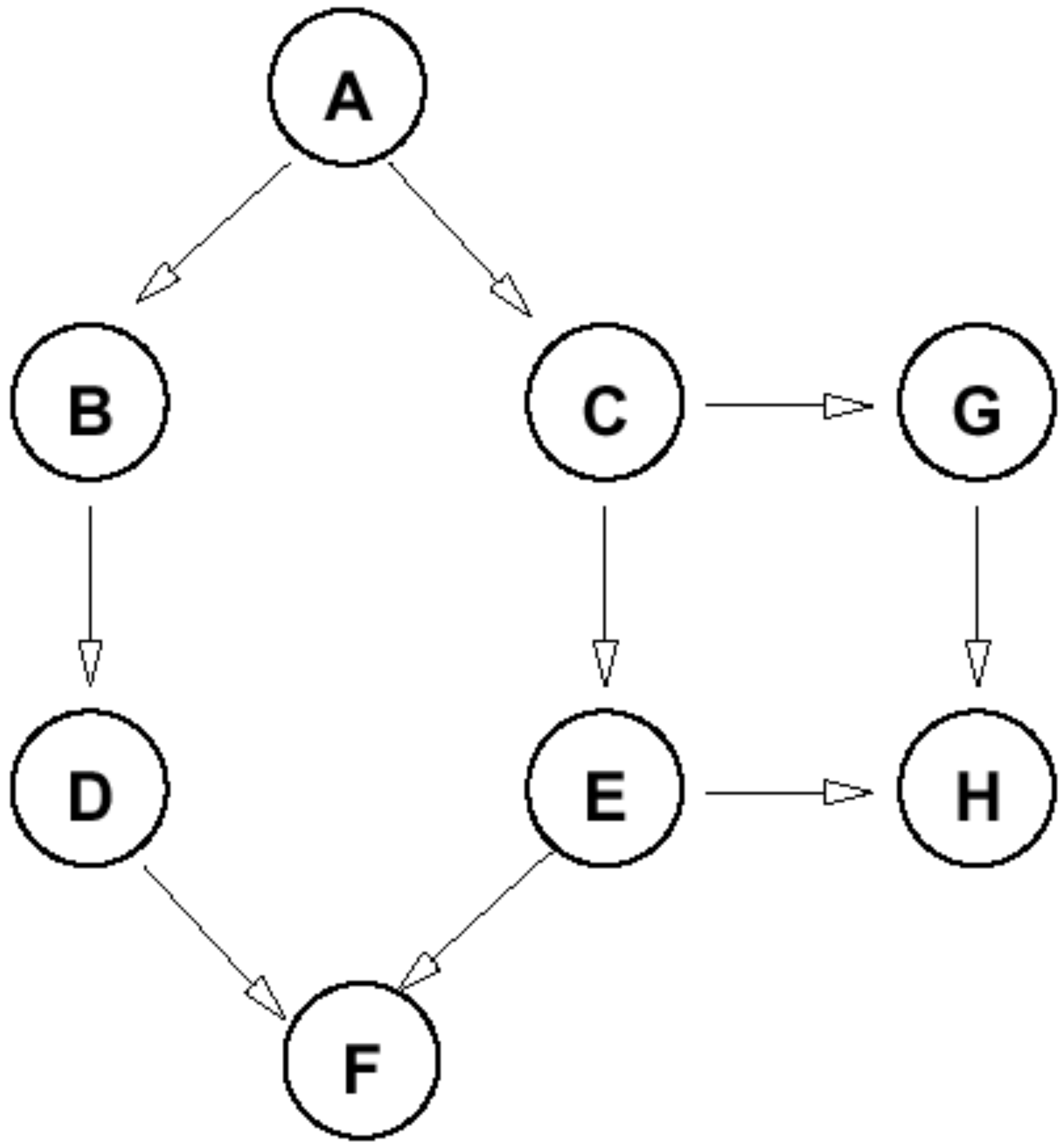
Best elimination width?

- Tree width (ω) is the MINIMUM elimination width of <u>any of the n!</u> different orderings of the variables minus 1.
- Best case elimination complexity of $2^{O(\omega)}$ where ω is the tree width.

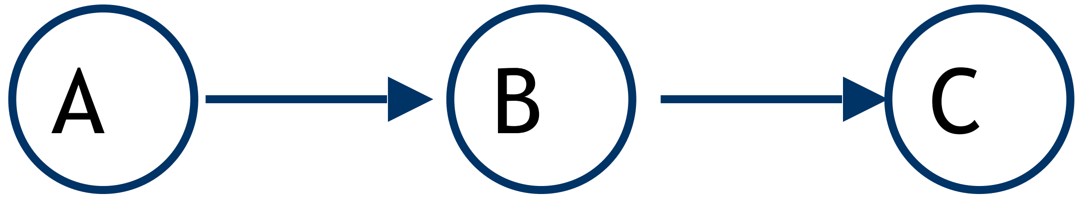
Worst case complexity?

Definition and complexity of VE for polytrees?

Definition of the Min-Fill heuristic.



VARIABLE ELIMINATION: RELEVANCE



- Restrict attention to the *sub-network comprising only relevant variables* when evaluating a query Q

- Given query Q, evidence E:
  - Q is relevant
  - if any node Z is relevant, its parents are relevant
  - if e∈E is a descendent of a relevant node, then E is relevant

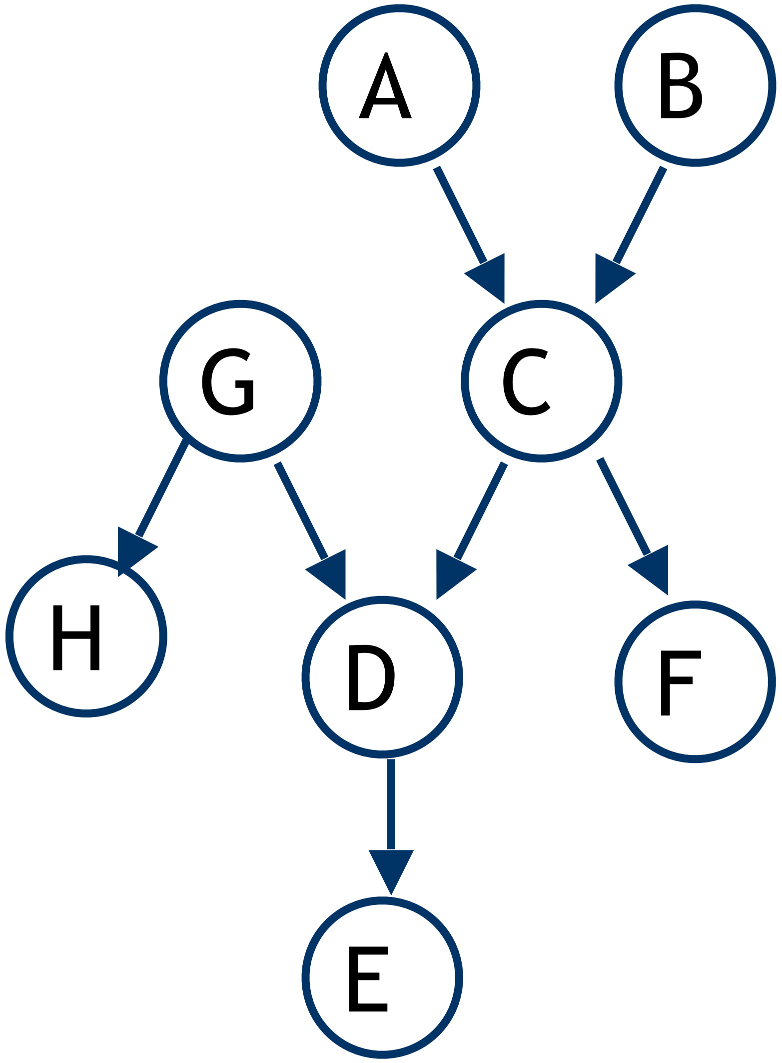
- Note this algorithm may over-estimate relevant set

RELEVANCE AND D-SEPARATION

Another piece of information we can use to assess relevance:

D-separation

A set of variables D-separates X and Y if they block every undirected path between X and Y.



D-SEPARATION: BLOCKING

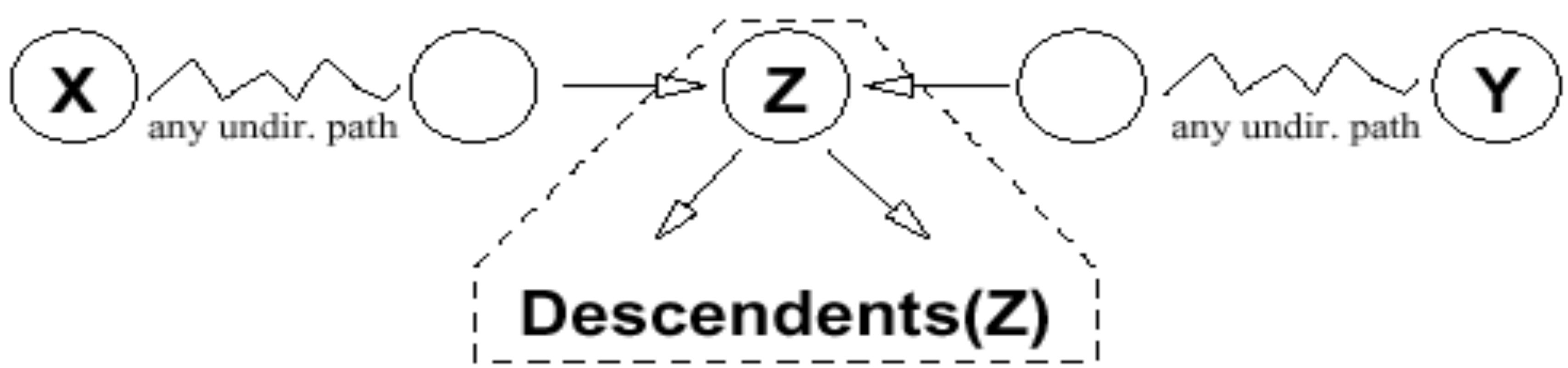
(1)

If Z in evidence, the path between X and Y blocked

(2)

If Z in evidence, the path between X and Y blocked

(3)



If Z is ***not*** in evidence and ***no*** descendent of Z is in evidence, then the path between X and Y is blocked

KNOWLEDGE REPRESENTATION

KNOWLEDGE REPRESENTATION

Some Key Definitions:

- Propositional Logic
- First Order Logic
- Knowledge Base
- Syntax
- Semantics
- Proof Procedure
- Derivation
- Entailment
- Soundness
- Completeness

FIRST ORDER LOGIC: SYNTAX

1. *constants (objects)*
2. *functions*
3. *predicates (and relations)*
4. *variables*
5. *connectives* $\rightarrow$,<=>, $\vee$, $\wedge$, $\neg$
6. *equality* =
7. *quantifiers* $\exists$, $\forall$

Definitions:

Term (Variables, Constants, or Function)

Atom (Predicate)

Atomic Formula (Literal)

FIRST ORDER LOGIC: SEMANTICS

Semantics establish meaning; they map formulas onto semantic entities.

- Requires a LANGUAGE: L(Functions,Predicates,Variables)
- Requires a MODEL or INTERPRETATION: $\langle D, \Phi, \Psi, v \rangle$

D is the set of individuals in the domain of discourse.

Φ maps functions of individuals onto individuals in the domain.

Ψ maps predicates (and relations) involving individuals onto T/F

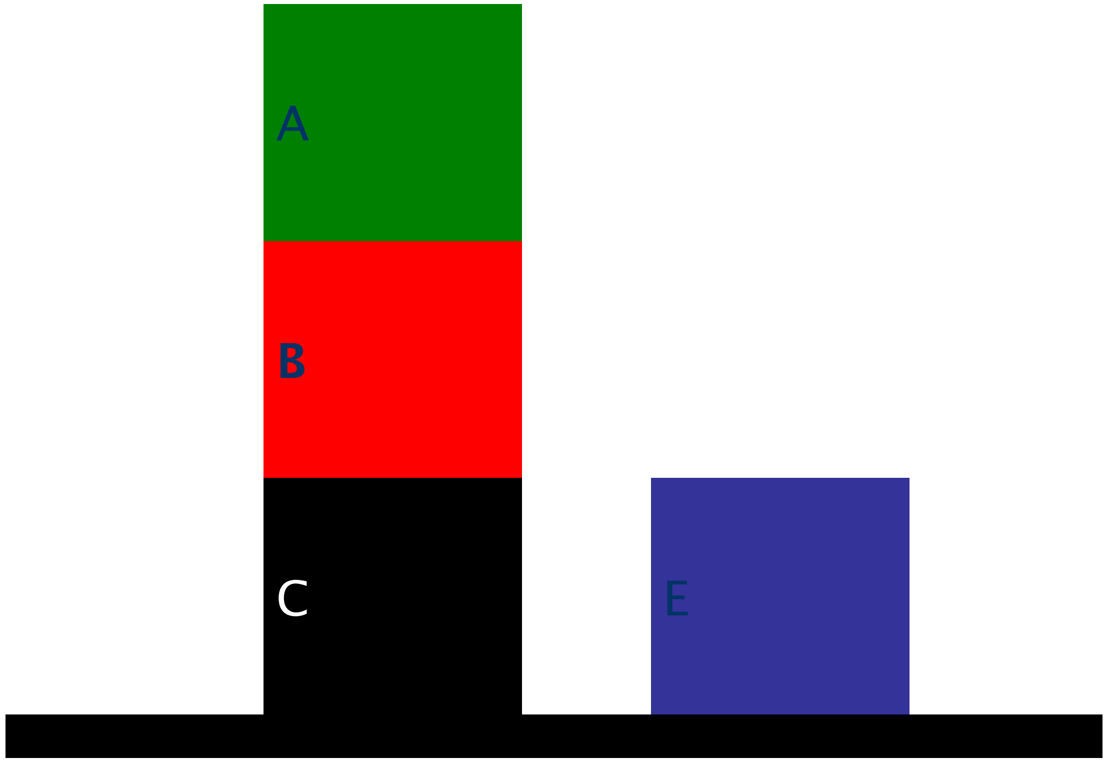
v is a variable assignment function (maps a VARIABLE onto an individual in the domain).

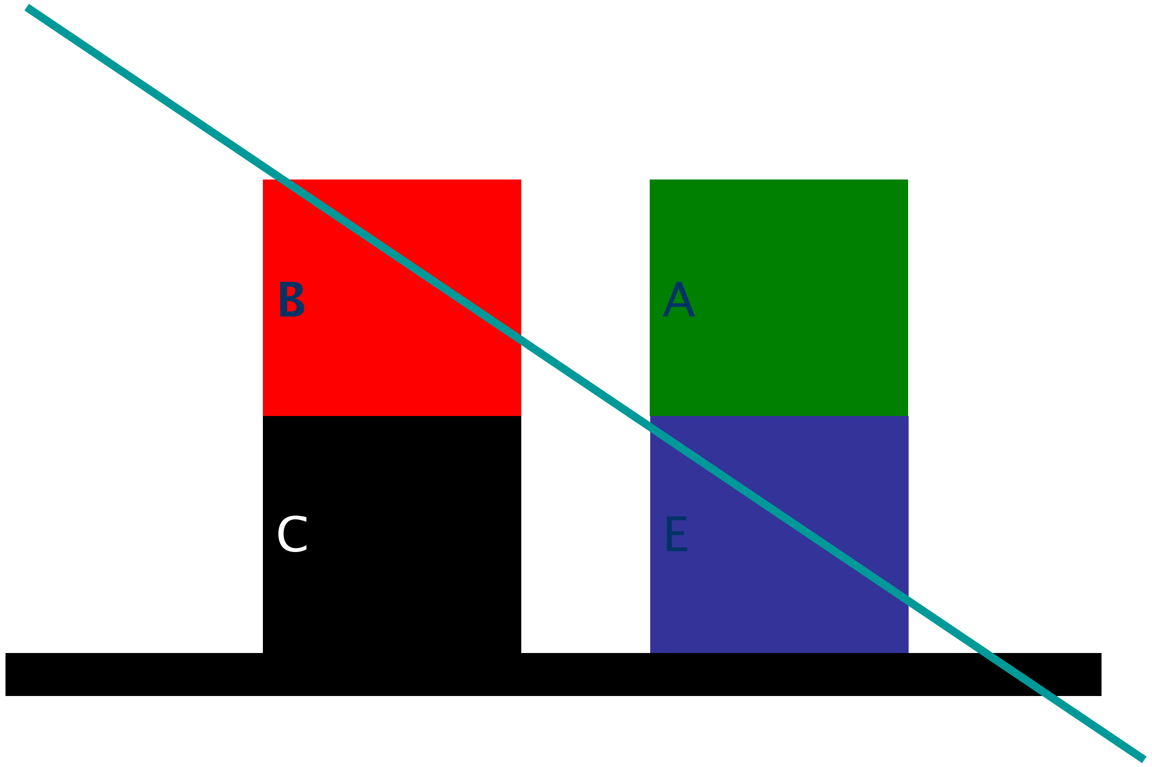
NB: There may be *many* interpretations or models of an underlying Knowledge Base.

FIRST ORDER LOGIC: SEMANTICS

The KB can support many models.

1. On(b,c)
2. Clear(e)





LOGIC: SOME USEFUL EQUIVALENCES

Implication:

$f1 \rightarrow f2$ is equivalent to $\neg f1 \vee f2$.

Remember DeMorgan's Laws:

$\neg(A \wedge B)$ is equivalent to $\neg A \vee \neg B$

$\neg(A \vee B)$ is equivalent to $\neg A \wedge \neg B$

$\neg\forall X.\ f$ is equivalent to $\exists X.\ \neg f$

$\neg\exists X.\ f$ is equivalent to $\exists X.\ \neg f$

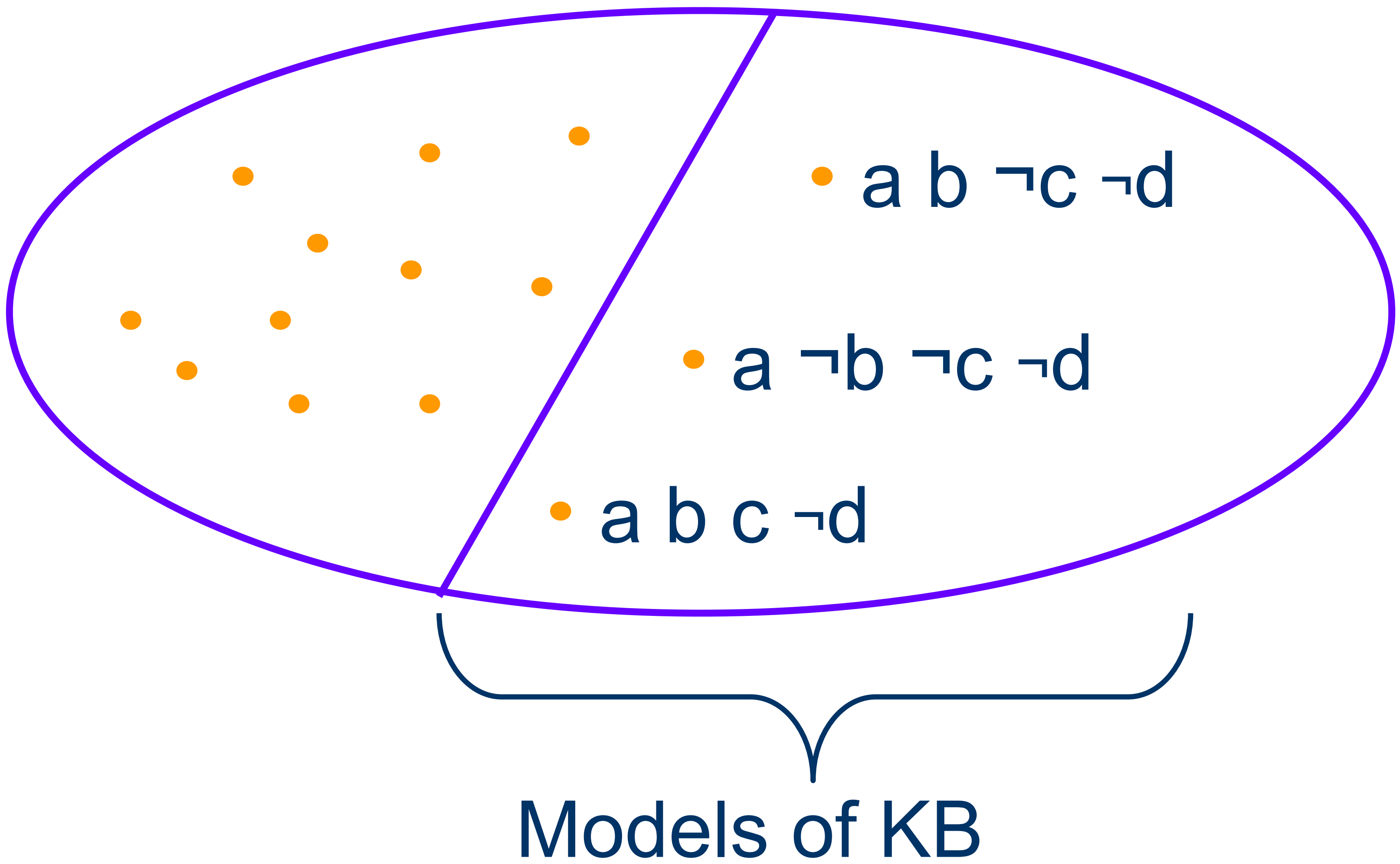
Double Negation:

$\neg\neg A$ is equivalent to A

AXIOMATIZING A DOMAIN

Propositional KB: a, c $\rightarrow$ b, b $\rightarrow$ c, d $\rightarrow$ b, ¬b $\rightarrow$ ¬c

Set of All Interpretations



Adding new sentences rules out additional unintended interpretations. This is called axiomatizing the domain.

PROOF PROCEDURES

Desirable Features of a Proof Procedure:

Soundness

$KB \vdash f \rightarrow KB \models f$

Completeness

$KB \models f \rightarrow KB \vdash f$

RESOLUTION IN FOL

Definitions and Requirements:

- Clausal Form
- Clause: A Disjunction of Atomic Formulae (Literals)
- Horn Clause
- Clausal Theory: A Conjunction of Clauses

Forward Chaining Proof Procedure:

Is it Sound? Complete?

Refutation Proof Procedure:

Is it Sound? Complete?

Advantages of Refutation v. Forward Chaining

CLAUSAL FORM: CONVERSION

To convert the KB into Clausal form we perform the following 8-step procedure:

1. Eliminate Implications.
2. Move Negations inwards (and simplify ¬¬).
3. Standardize Variables.
4. Skolemize.
5. Convert to Prenix Form.
6. Distribute conjunctions over disjunctions.
7. Flatten nested conjunctions and disjunctions.
8. Convert to Clauses.

RESOLUTION: NON-GROUND CLAUSES

Requires substitutions, e.g.

p(X,g(Y,Z))[X=Y, Y=f(a)] ➔p(Y,g(f(a),Z))

How to Compose Substitutions?

1. *Construct the composition.*
2. *Delete any identities, i.e., equations of the form V=V.*
3. *Delete any equation $Y_i=s_i$ where Y_i is equal to one of the X_j in θ.*

Definition of Unifiers and Most General Unifiers (why is the MGU preferred?)

Factoring and Answer Extraction

Required Notation, e.g.

- R[1a,2b]{Y=a} (p(a), ¬p(W))
- f[1ab]{X=Y} (p(Y))

RESOLUTION: MGU ALGORITHM

To find the MGU of two formulas f and g.

1. $k = 0$; $\sigma_0 = \{\}$; $S_0 = \{f,g\}$
2. If S_k contains an identical pair of formulas stop, and return σ_k as the MGU of f and g.
3. Else find the disagreement set $D_k=\{e_1,e_2\}$ of S_k
4. If $e_1 = V$ a variable, and $e_2 = t$ a term not containing V (or vice-versa) then let
   $\sigma_{k+1} = \sigma_k \{V=t\}$ (Compose** the additional substitution)
   $S_{k+1} = S_k\{V=t\}$ (Apply the additional substitution)
   $k = k+1$
   GOTO 2
5. Else stop, f and g cannot be unified.

*** Note that this is compose not conjoin!*