Summary: Knowledge Representation

Summary
Knowledge Representation

Knowledge Representation and Propositional Logics

"Definition of intelligence"

• carry out complex reasoning (solve physics problems, prove
• draw plausible inferences (diagnose cars, solve a murder
• use natural language (read stories and answer questions about them, carry out extended conversation)
• solving novel complex problems (generating plans, designing
• social activities that require a theory of mind

Two main lines of development in AI

•symbolic representations
•statistical representation

Statistical vs. symbolic AI:
very different types of applications

statistical:
- pattern recognition (images, sound, shapes)
- motor skills (robots)
- speech generation (sound)
- search engines

symbolic:
- planning (autonomous space missions)
- reasoning (diagnosis, design, decision support)
- language generation (conversations)
- search engines

Strengths & Weaknesses

	Symbolic	Connectionist
Construction	Human effort	Data hunger
Scaleable	+/- (worse with more data)	+/- (worse with less data)
Explainable	+	-
Generalisable	Performance cliff	Performance cliff

The goal of logic in KR

• To state statements which are known to be true (the knowledge base
• Some statements that describe the current state of the world (the premises
• To state statements for which we want to check if they are true (the conclusions ")
• To see if the conclusions can be derived from the knowledge base + the premises through logical reasoning
• A variety of related tasks

The ingredients of a logic in KR

• How to formulate the statements (Syntax)
• Assign meaning to the statements (Semantics)
• Assign what can be derived (Calculus)

And all of this differs from application domain to application domain and even application to application.

3) PROPOSITIONAL LOGIC

The knowledge base: simple statements

• In propositional logic, a knowledge base can have simple statements , and complex statements.
• A simple statement states one fact about the world :
  – Examples: it is hot , it is raining , it is humid
  – In our examples we often just use a single letter for a a simple statement:
  – Examples: P, Q, R.

The knowledge base: complex statements
• Complex statements are made by combining other statements . If S and T are statements (simple or complex), then the following are also complex statements:
These can be any combination

- $S \wedge T$ ("S and T") (conjunction)
- $S \vee T$ ("S or T") (disjunction)
- $\neg S$ ("not S") (negation)
- $S \rightarrow T$ ("S implies T") (implication)

- These can be any combination
- Examples: *it-is-hot* $\wedge$ *it-is-raining*
  - $\neg$ *it-is-humid*
  - *it-is-hot* $\wedge$ *it-is-raining* $\rightarrow$ *it-is-humid*
  - $(P1 \vee \neg P2) \wedge P3)$
  - $((P \vee H) \wedge \neg H) \rightarrow P$

The meaning of simple statements

• In propositional logic, the meaning of a statement is either true or false.
• We call this the " interpretation function " I :
it maps any statement p to true or false: I(p)=0 or I(p)=1 (0 means false, 1 means
• Example:
  – I(it is hot)=1 means that it is hot today
  – I(it is hot)=0 means that it is not hot today
  – I(Q)=1 means that Q is true

The meaning of complex statements
• The meaning of complex statements is calculated from the meaning of the parts:

- The meaning of P∧Q is calculated by combining the meaning of P and Q:
  – P∧Q is only true if **both** P and Q are true, otherwise P∧Q is false.
  – Mathematically: I(P∧Q)=1 only if I(P)=1 and I(Q)=1
  – We can also write this in a table, which tells us the meaning of P∧Q for each combination of P and Q:

P	Q	P∧Q
1	1	1
1	0	0
0	1	0
0	0	0

- In the same way, the meaning of P ∨ Q is calculated by combining the meaning of P and Q:
  – P ∨ Q is only true if **at least one of** P and Q are true, otherwise P ∨ Q is false.
  – Mathematically: I(P ∨ Q)=1 only if I(P)=1 or I(Q)=1, or both
  – We can also write this in a table, which tells us the meaning of P ∨ Q for each combination of P and Q:

P	Q	P∨Q
1	1	1
1	0	1
0	1	1
0	0	0

- In the same way, the meaning of ¬ P is calculated from the meaning of P
  – ¬ P is only true if P is false, otherwise ¬ P is false.
  – Mathematically: I(¬ P)=1 only if I(P)=0
  – We can also write this in a table, which tells us the meaning of ¬ P for each meaning of P:

P	¬ P
1	0
0	1

- The meaning of P → Q also computed from the meanings of P and Q, but in a bit of a strange way:

P	Q	P→Q
1	1	1
1	0	0
0	1	1
0	0	1

- (Exercise: make the truth table of ¬P ∨ Q)

We can use these tables to determine the meaning (the "truth") of **complex statements**:

To calculate the meaning of (P1 ∨ ¬P2) ∧ P3

we do the following (read the table left-to-right):

1. take all 8 combinations of the possible values for P1, P2 and P3
2. Calculate the value for ¬P2 from the values of P2
3. Calculate the value for (P1 ∨ ¬P2) by combining the values for P1 and ¬P2
4. Calculate the value for (P1 ∨ ¬P2) ∧ P3 by combining the values for (P1 ∨ ¬P2) and P3

p_1	p_2	p_3	$\neg p_2$	$(p_1 \vee \neg p_2)$	P
1	1	1	0	1	1
1	1	0	0	1	0
1	0	1	1	1	1
1	0	0	1	1	0
0	1	1	0	0	0
0	1	0	0	0	0
0	0	1	1	1	1
0	0	0	1	1	0

Special statements: tautologies

Let's look at the complex statement

((P ∨ H) ∧ ¬ H) → P.

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

There is something special with the result in the final column: this statement is **always true!**

Such statements, which are true in any world that we can possibly imagine, are called **tautologies (or: valid)**

Special statements: inconsistencies

There are also statements which are always false.
They are false for any of the possible combinations of truth values of their parts, in other words, they are false in every world we can possibly imagine.
Such statements are called inconsistent (or: invalid).
The statements which are not a tautology and also not inconsistent (in other words: they are true for some combination of truth values and false for others) are called satisfiable

Example: $(A \vee B) \wedge (\neg A \vee \neg B)$ is satisfiable,
but not a tautology.

Some terms

• The meaning or semantics of a sentence is determined by its interpretation .
• Given the truth values of all symbols in a sentence, it can be "evaluated" to determine its truth value (True or False).
• A model for a KB is a "possible world" (assignment of truth values to propositional symbols) in which each sentence in the KB is True.

More terms

• A satisfiable sentence is a sentence that can be made true in at least one world (i.e. is not inconsistent)
• P entails Q , written P |= Q, means that whenever P is True, so is Q. In other words, all models of P are also models of Q
In Propositional Logics (and in other Logics) those notions can often be reduced to each other.
• E.g. P is satisfiable iff ¬ P is not valid
• Or P |= Q iff P & ¬Q is unsatisfiable

So why are we doing all this?

1. We now have language in which we can make simple and complex statements about the world ("propositional
2. We can formulate problems that we need to solve as complex statements in propositional logic
3. We have a formal and unambiguous semantics to define if a statement can be satisfied
4. We have a method (e.g. Truth Tables) to determine which valuations for propositions satisfies the problem statements (a calculus)
5. Such a "satisfying truth assignment" is then the solution to our problem.

Practice Problem

Problem: A Diplomatic Problem

You are chief of protocol for the embassy ball. The crown prince instructs you either to invite Peru or to exclude Qatar. The queen asks you to invite either Qatar or Romania or both. The king, in a spiteful mood, wants to snub either Romania or Peru or both. Who do you invite?

- Three propositional symbols P, Q and R:

P	$\equiv$	invite Peru	$\neg P$	$\equiv$	exclude Peru
Q	$\equiv$	invite Qatar	$\neg Q$	$\equiv$	exclude Qatar
R	$\equiv$	invite Romania	$\neg R$	$\equiv$	exclude Romania

- Three operators: negation $\neg$, disjunction or,$\vee$,conjunction &,$\wedge$
- Exercise: 5 minutes to formalise the problem in PL.

Representing the Diplomatic Problem in PL

- The problem can be formalized as

prince:	invite Peru or exclude Qatar	$\equiv$	P or $\neg Q$
queen:	invite Qatar or Romania or both	$\equiv$	Q or R
king:	snub Romania or Peru or both	$\equiv$	$\neg R$ or $\neg P$

- $\Sigma = (P \text{ or } \neg Q)\ \&\ (Q \text{ or } R)\ \&\ (\neg R \text{ or } \neg P)$
- Any assignment (choice of truth or falsity of the symbols) is a solution for the diplomatic problem.
- For example, let P and Q be true, and R false, then Σ is true.
- The assignment P =T, Q=T and R =F satisfies Σ.
- Therefore: *invite Peru and Qatar, exclude Romania* is solution!

Reasoning: Solving the diplomatic problem?

- Reminder: $\Sigma = (P \text{ or } \neg Q)\ \&\ (Q \text{ or } R)\ \&\ (\neg R \text{ or } \neg P)$
- We can use truth tables

P	Q	R	P or $\neg Q$	Q or R	$\neg R$ or $\neg P$	Σ
T	T	T	T	T	F	F
T	T	F	T	T	T	T
T	F	T	T	T	F	F
T	F	F	T	F	T	F
F	T	T	F	T	T	F
F	T	F	F	T	T	F
F	F	T	T	T	T	T
F	F	F	T	F	T	F

- Problem: we need 2^n solutions, i.e. with 3 variables 8 rows, with 20 variables 1048576, and with 50 variables 1125899906842624.

Problem solving by Satisfiability testing

1. We have to answer a question
2. We formulate question as a complex statement S in propositional logic
3. We try to find a "satisfying assignment" of truth values (in other words: a combination of truth values that makes S true)
4. That combination of truth values gives us the solution to the original problem.

Problem : if S contains n symbols, we need to check 2 n combinations of truth values.
Suppose we want to solve Sudoko with 721 variables.

DPLL+Logical reasoning for Sudokus

Propositional Logic

The language of propositional logic (Syntax)

- $S \wedge T$ ("S and T") (conjunction, sometimes "and",&)
- $S \vee T$ ("S or T") (disjunction, sometimes "or")
- $\neg S$ ("not S") (negation)
- $S \rightarrow T$ ("S implies T") (implication)

One special syntax: Clausal Normal Form:

$(P_1 \vee P_2 \vee \ldots \vee P_k) \wedge (Q_1 \vee Q_2 \vee \ldots \vee Q_n) \wedge \ldots \wedge (R_1 \vee R_2 \vee \ldots \vee R_n)$
will be explained in some minutes

The meaning of propositional logic (Semantics) 1

• Propositions P and Q are interpreted to be either false or true (truth assignment, e.g. I(P)=True or I(P)=1)
• Complex Sentences S can be calculated to be true or false according to an assignment e.g. I(Rains --> Wet) is True iff I(Rains)=False (0) or I(Wet)=True (1)

The meaning of propositional logic (Semantics) 2

• Inconsistent sentence S: always false (boring) I(S)=0 for all assignment functions I
• Valid sentences S (tautology): always true (boring) I(S)=0 for all assignment functions I
• Satisfiable sentence S: interesting!
There is an assignment I such that I(S)=1. This is also called a model .
This is the task of SAT solving: finding a "model"

SAT solving for Problem Solving

• Represent your problem in Propositional Logic
• Is a problem solvable? = is the sentence satisfiable?

• Finding a solution = finding a model (= find a truth assignment I that makes the sentence S true, i.e. I(S)=1)

• Truth assignment is the solution to the problem

A calculus for propositional logic

The problem can be formalized as

prince:	invite Peru or exclude Qatar	$\equiv$ P or $\neg Q$
queen:	invite Qatar or Romania or both	$\equiv$ Q or R
king:	snub Romania or Peru or both	$\equiv$ $\neg R$ or $\neg P$

$\Sigma = (P \text{ or } \neg Q)\ \&\ (Q \text{ or } R)\ \&\ (\neg R \text{ or } \neg P)$

P	Q	R	P or $\neg Q$	Q or R	$\neg R$ or $\neg P$	Σ
T	T	T	T	T	F	F
T	T	F	T	T	T	T
T	F	T	T	T	F	F
T	F	F	T	F	T	F
F	T	T	F	T	T	F
F	T	F	F	T	T	F
F	F	T	T	T	T	T
F	F	F	T	F	T	F

A calculus for propositional logic: aka How to find a satisfying model?

•Truth table method

•But: size of the table is 2^N(N = number of variables)

•$2^{729}=10^{219}$….

And the problem can be even worse!
Luckily there are more efficient algorithms …

Clause Normal Form

A special form of statement: the clausal normal form

Complex statements can be very complex, with many combinations of ¬, ∨, ∧, →

A special kind of complex statement (the '**clausal normal form**') that is easy for Computer programmes to process

The clausal normal form looks like;

$(P_1 \vee P_2 \vee \ldots \vee P_k) \wedge (Q_1 \vee Q_2 \vee \ldots \vee Q_n) \wedge \ldots \wedge (R_1 \vee R_2 \vee \ldots \vee R_n)$

where each of the letters can be **positive** (P_k) or **negative** ($\neg P_k$) atoms, and called **Literals**.

In other words: it is a conjunction of disjunctions of literals.

Theorem: Any complex statement can be rewritten to a logically equivalent statement in clausal normal form 13

Examples of clausal normal form

- $p_1, \neg p_1, \neg p_5$ are literals. They are also in CNF
- $(p \vee q)$ is in CNF
- $((p_1 \vee p_2) \wedge (\neg p_1 \vee p_3))$ and $((p_1 \vee p_2) \wedge \neg p_1)$ are in CNF
- $((p_1 \wedge p_2) \vee (\neg p_1 \wedge p_3))$ and $(p \vee (p \wedge \neg p))$ are not in CNF.

How to rewrite a statement into Clausal Form

❶ Remove implications (p → q) = (¬p ∨ q)

❷ Move negations in front of propositional symbols
- ¬ (p ∨ q) = ¬p ∧ ¬q
- ¬ (p ∧ q) = ¬p ∨ ¬q

❸ Move conjunctions "outside" over disjunctions
- r ∨ (p ∧ q) = (r ∨ p) ∧ (r ∨ q) (result: a conjunction of clauses)

❹ You can read the separate clauses:
- $(p_1 \vee p_2) \wedge (q_1 \vee q_2) = (p_1 \vee p_2), (q_1 \vee q_2)$

❺ Sometimes we write a disjunction of clauses as $\{\{p_1, p_2\}, \{q_1, q_2\}\}$

The outer comma means conjunction ("and")
the inner comma means disjunction ("or")

This can be done for any propositional formula. It doesn't matter in which order you apply the rule, the result is always the same

15

How to rewrite a statement into Clausal Form

- You may also have to use simplification rules:
  - (p ∨ p) = (p ∧ p) = ¬ ¬ p = p
  - (p ∨ q) = (q ∨ p), (p ∧ q) = (q ∧ p)
  - (p ∨ q) ∨ r = p ∨ (q ∨ r), (p ∧ q) ∧ r = p ∧ (q ∧ r)
- Repeatedly apply rules to the result:
  - ¬(¬ p ∧ (q ∨ ¬r) = ¬¬ p ∨ ¬(q ∨ ¬r) = p ∨ (¬q ∧ ¬¬r) = p ∨ (¬q ∧ r)
- Sometimes multiple rules apply (always same result)

((p ∧ q) ∨ r) ∨ s= ((p ∨ r) ∧ (q ∨ r)) ∨ s= ((p ∨ r ∨ s) ∧ (q ∨ r ∨ s))	((p ∧ q) ∨ r) ∨ s= ((p ∧ q) ∨ (r ∨ s) = ((p ∨ r ∨ s) ∧ (q ∨ r ∨ s))

CNF alternative notation in ASCII

(¬A ∨ ¬B ∨ E) ∧ (¬E ∨ A) ∧ (¬E ∨ B) ∧ (¬C ∨ F) ∧ (¬F ∨ C) ∧ (I)

(A ' + B ' + E)(E ' + A)(E ' + B)
(C'+ F)(F'+ C)
(I)

DIMACS Format:
-1 -2 5 0
-5 1 0
-5 2 0
-3 6 0
-6 3 0
9 0

Davis Putnam (DPLL)

• SAT is NP Complete (proved by Cook in 1971, as the first NP complete problem)
• This means that any fast SAT solver could also be used to solve lots of other exponential problems
• This is widely believed to be impossible: nobody believes NP=P (but still an open problem)
• So should we give up all hope?
• Remarkably, even though they are exponential in the worst case (Cook was right in 1971….), modern SAT solver are very fast most of the time

Davis Putnam algorithm

• It was not developed by Davis & Putnam, but by Davis, Logemann & Loveland
• The original algorithm by Davis & Putnam used potentially exponential space
• The improved algorithm by DLL is now known as DP…
• It dates from the '60s, is (of course) exponential in the worst case, but performs remarkably well (if we add a few heuristics)

DPLL version 1

Davis – Putnam – Loveland – Logemann 1962

Let pa be a partial assignment α a formula

```
dpll_1(pa){
  if (pa makes α false) return false;
  if (pa makes α true) return true;

  choose P in α ;
  if (dpll_1(pa ∪ {P=false}))
      return true;
  else return dpll_1(pa ∪{P=true});
}
```

Returns true if α is satisfiable, false otherwise

We need a few new terms

- A "**variable**" is one of the simple statements (the letters) that make up the problem statement
- A **clause** is one of the disjuncts in the big disjunction in the input statement (which is in clausal normal form)
- A **literal** is a variable or its negation (both P and ¬P are literals)
- A **unit clause** is a clause of length 1 (= only one literal)
- A literal P is **pure** if ¬P appears nowhere (and vice versa)

(¬A ∨ B ∨ E) ∧ (¬E ∨ A) ∧ (¬E ∨ B) ∧ (¬C ∨ F) ∧ (¬F ∨ C) ∧ (I)

clause *literal* *pure literal* *unit clause*

- An **empty clause** is (trivially) false (disjunction: at least one of these must be true)
- An **empty set of clauses** is (trivially) true (conjunction: all of these have to be true)

23

Davis-Putnam procedure

It consists of two stages:

simplify

1. It **simplifies** the formula, and checks a few easy cases. If such an easy case exists it can assign the right value to one of the truth values and **repeats the process**

split

2. If there is no such easy case, it **picks one of the predicates** and **picks a truth value** to it, and then **repeats the process**

3. The process **stops**
   1. when all clauses have been satisfied (success!), or
   2. when one becomes inconsistent (failure)

backtrack

If there is failure, that could be because we picked the wrong truth value for a letter in step 2. So we have to "**backtrack**":

4a. for the letter from step 2 we pick the other truth value, and try again.

4b. In step 2 we pick another letter

24

Part 1: Simplication rules

1.1 **Tautology**:
if a clause contains ¬P ∨ P it can be removed
(because the clause will be satisfied in an case)

1.2 if there is a **pure literal** (occurring only positive or negative),
it can be set to true (or false)
(because this choice doesn't affect any other clause)

1.3 if there is a **unit clause** (consisting of only one literal),
that literal can be set to true
(because this is the only way the clause can be satisfied)
(of course, if the literal is of the form ¬P, that makes P false)

Part 1: Simplication rules (b)

If literal L_1 is true, then clause $(L_1 \vee L_2 \vee ...)$ is true

If clause C_1 is true, then $C_1 \wedge C_2 \wedge C_3 \wedge ...$ has the same value as $C_2 \wedge C_3 \wedge ...$

Therefore: Okay to delete clauses containing true literals!

If literal L_1 is false, then clause $(L_1 \vee L_2 \vee L_3 \vee ...)$ has the same value as $(L_2 \vee L_3 \vee ...)$

Therefore: Okay to shorten containing false literals!

If literal L_1 is false, then clause (L_1) is false

Therefore: the empty clause means false!

DPLL version 2

Davis – Putnam – Loveland – Logemann

```
dpll_2(a, literal){
  remove clauses from a containing literal
  shorten clauses from a containing ¬literal

  if (a contains no clauses) return true;
  if (a contains empty clause return false;

  choose P in a;
  if (dpll_2(a, ¬P))return true;
  return dpll_2(a, P);
}
```

Partial assignment corresponding to a node is the set of chosen literals on the path from the root to the node

Further Improvements

Formula $(L) \wedge C_2 \wedge C_3 \wedge \ldots$ is only true when literal L is true

Therefore: Branch immediately on unit literals!

If literal L does not appear negated in formula F, then setting L true preserves satisfiability of F

Therefore: Branch immediately on pure literals!

Davis-Putnam algorithm

- Input: A set Σ of clauses, i.e. sets of Literals
- Output: **Yes**, if Σ is satisfiable, **no** otherwise.

```
procedure DP(Σ)
(Sat)      if Σ = ∅ then ''sat''
(Empty)    if Σ contains the empty clause then ''unsat''
(Taut)     if Σ contains p ∨ ¬p then DP(Σ \ {p ∨ ¬p})
(Unit Pr)  if Σ has unit clause {l} then DP(Σ{l = true})
(Pure)     if Σ has pure literal l then DP(Σ{l = true})
(Split)    if DP(Σ{l = true}) is satisfiable then ''sat''
           else DP(Σ{l = false})
```

Example: The diplomatic puzzle

We have to find a satisfying assignment for

(P ∨ ¬Q) ∧ (Q ∨ R) ∧ (¬R ∨ ¬P)

P	Q	R	P or $\neg Q$	Q or R	$\neg R$ or $\neg P$	Σ
T	T	T	T	T	F	F
T	T	F	T	T	T	T
T	F	T	T	T	F	F
T	F	F	T	F	T	F
F	T	T	F	T	T	F
F	T	F	F	T	T	F
F	F	T	T	T	T	T
F	F	F	T	F	T	F

Example: The diplomatic puzzle

We have to find a satisfying assignment for

$$(P \vee \neg Q) \wedge (Q \vee R) \wedge (\neg R \vee \neg P)$$

- No easy cases, so **split**. Let's say P=1
- **Simplify**: $(Q \vee R) \wedge (\neg R)$
- **Pure literal** Q, so Q=1
- **Simplify**: $(\neg R)$
- **Pure literal** (and unit clause), so $\neg R=1$, and R=0

Solution: P=1,Q=1,R=0

You can easily check that this does indeed satisfy the input formula.

Therefore: invite **P**eru and **Q**uatar, but not **R**umania

Example: The diplomatic puzzle

We could have made another split choice in the first step:

$$(P \vee \neg Q) \wedge (Q \vee R) \wedge (\neg R \vee \neg P)$$

- No easy cases, so **split**. This time let's say R=1
- Simplify: $(P \vee \neg Q) \wedge (\neg P)$
- Pure literal $\neg Q$, so $\neg Q=1$, so Q=0
- Simplify: $(\neg P)$
- Unit clause (and pure literal), $\neg P=1$, so P=0
- Solution: P=0,Q=0,R=1
- You can easily check that this also satisfies the input formula: only invite **R**umania
- We know from the truth table method that these are the only two satisfying assignments (= the only two solutions to the puzzle)

34

Another example (continued)

- We had ((Q,R), $(\neg R,Q)$)
- Now split with Q=1
- Simplify: all clauses satisfied, success;

Thus: P=1,Q=1 satisfies the original formula.

Notice that this is independent of the value of R.

So there are two full solutions:

(P=1,Q=1,R=1) and (P=1,Q=1,R=0)

Another example:

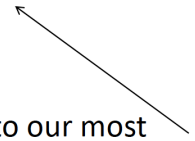
Find a satisfying truth assignment for

((P∨¬R) ∧ (P∨¬Q, R) ∧ (Q∨R∨¬P) ∧ (¬R∨¬P∨ Q))

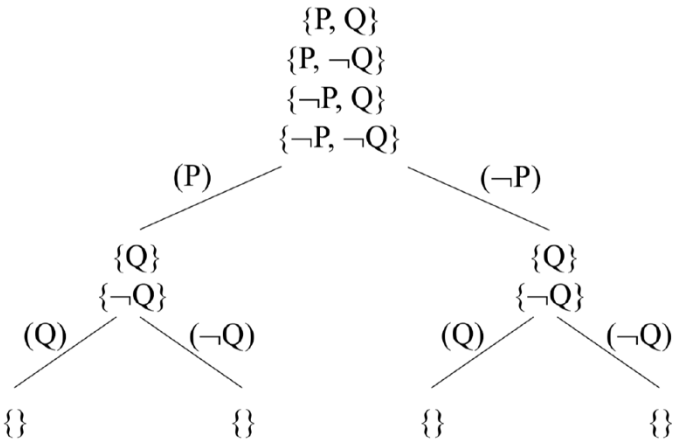
We can also write this formula as:

((P, ¬R),(P, ¬Q, R), (Q,R, ¬P), (¬R, ¬P, Q))

- No easy case, split: let's choose P=1
- Simplify: ((Q,R), (¬R,Q))
- With naïve DPLL: let's choose Q=0
- Simplify: ((R), (¬R))
- Unit clause: R=1
- Simplify: false! We must backtrack to our most recent choice, which was Q=0, and change it to Q=1



Davis-Putnam as Trees



Exercise: write DP as a tree on
((P∨¬R) ∧ (P∨¬Q, R) ∧ (Q∨R∨¬P) ∧ (¬R∨¬P∨ Q))

Variations

```
procedure DP(Σ)
(Sat)      if Σ = ∅ then ''sat''
(Empty)    if Σ contains the empty clause then ''unsat''
(Taut)     if Σ contains p ∨ ¬p then DP(Σ \ {p ∨ ¬p})
(Unit Pr)  if Σ has unit clause {l} then DP(Σ{l = true})
(Pure)     if Σ has pure literal l then DP(Σ{l = true})
(Split)    if DP(Σ{l = true}) is satisfiable then ''sat''
           else DP(Σ{l = false})
```

- The **Tautology rule** only needs to be checked once at the start
- The **Pure rule** is often deleted because the cost of using it might be more than the benefits (Question: why is it so expensive?)
- The **Unit rule** is not strictly essential (if you do a clever choice for split), but it is essential for good performance
- The **Unit rule** by itself is complete for Horn clauses! 38

Heuristics for splitting

The secret sauce: how to split?

Notation:

CP(v) is nr of occurrences of v

CN(v) is nr of occurrences of ¬v

- RAND: pick a random literal
- DLCS (Dynamic Largest Combined Sum):
  - Pick v with largest CP(v)+CN(v) (= most frequent v)
  - If CP(v)>CN(v) then v=1 else v=0
- DLIS (Dynamic Largest Individual Sum):
  - Pick v with largest CP(v) or largest CN(v)
  - If CP(v)>CN(v) then v=1 else v=0

Jeroslow-Wang Heuristics

- For a given literal l compute: $J(l) := \sum_{l \in \omega, \omega \in \varphi} 2^{-|\omega|}$

  $J(l) = \sum 2^{-|\omega|}$ Sum over 2 to the power of length of clause

  {{A,B},
  {-B,D},
  {-A,D,E}}
- One sided Jeroslow-Wang (JW-OS)
  - select the literal with the highest value of J
- Two-sided Jeroslow-Wang (JW-TS)
  - select the variable with the highest value of (J(x) + J(x'))
  - if J(x) ≥ J(x') set x true, else set x false
- Intuition:
  - Weight occurrences in small clauses higher

MOM's Heuristic

{{A,B},
{-B,D},
{-A,D,E}}

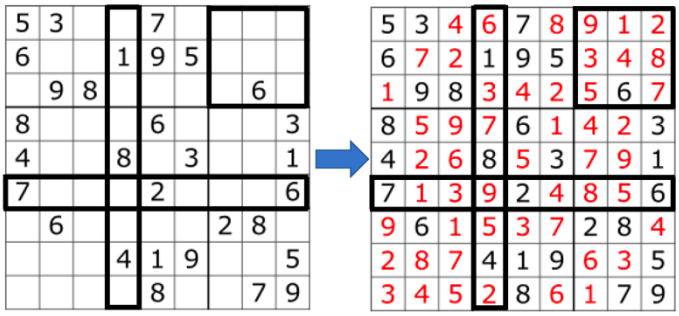
- Maximum Occurrence's in Clauses of Minimum Size
- Select the literal that maximizes the function:

  $[f^*(x) + f^*(x')]*2^k + f^*(x) * f^*(x')$
  - f*(l): Number of occurences of l in the smallest non-satisfied clauses
  - k is a tuning parameter
- Intuition: Preference is given to clauses:
  - with a large number of occurences of either x or x' in them
  - and also variables that have a large number of clauses of both phases of x in them
  - focus on the currently smallest size clauses

Sudokus in propositional logic

Checking for (in)consistency can be used to solve problems:
Many problems can be formulated as a set of constraints on the solution
These constraints can be stated in PL
Finding a solution
= asking if the set of constraints is satisfiable
= finding a satisfying truth assignment
Thus: Solving the problem = finding satisfying truth assignment
(that satisfies all the constraints

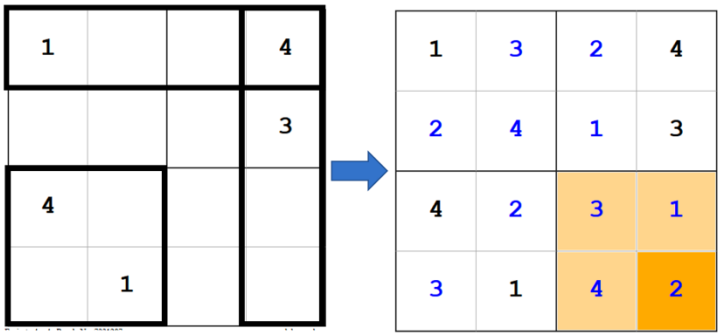
Example: solving Sudokus using PL



Formulate the problem as Constraints:

- all squares must have exactly one number from 1-9
- no number can appear twice in a row, column or square

Example: solving Sudokus using PL



ormulate the problem as Constraints:

all squares must have precisely one number from 1-**4**

no number can appear twice in a row, column or square

How to write the constraints in PL?

We need to describe "the world":
one propositional variable
for each value in each position on the board

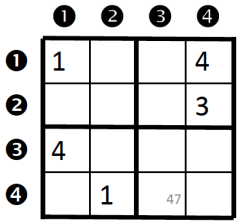
position_❶_❹_is_4,
position_❹_❷_is_1

in short: 144, 421, ….

This puzzle is described as
111 ∧ 144 ∧ 243 ∧ 314 ∧ 421

in total we need
16x4=64 variables

NB: this is propositionalisation



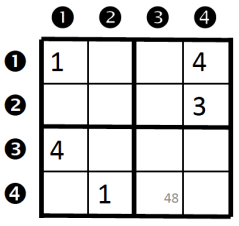
How to write the constraints in PL?

We need to describe the constraints:

At least one number in each square:
"position_❶_❶ is a 1 or a 2 or 3 or a 4"
111 ∨ 112 ∨ 113 ∨ 114 (16 statements, one for each cell)

At most one number in each square: (4x16=64 statements)
"if position_❶_❶ is a 1 it is not a 2 and it is not a 3 and …"

111 -> ¬112 ∧ ¬ 113 ∧ ¬ 114
112 -> ¬111 ∧ ¬ 113 ∧ ¬ 114
113 -> ¬111 ∧ ¬ 112 ∧ ¬ 114
114 -> ¬111 ∧ ¬ 112 ∧ ¬ 113



How to write the constraints in PL?

No number appears twice in a row: (4x16=64 statements

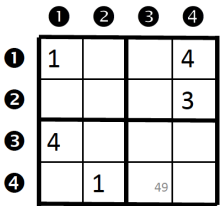
111 -> ¬121 ∧ ¬ 131 ∧ ¬ 141
112 -> ¬122 ∧ ¬ 132 ∧ ¬ 142
113 -> ¬123 ∧ ¬ 133 ∧ ¬ 143
114 -> ¬124 ∧ ¬ 134 ∧ ¬ 144

No number appears twice in a column (again 64 statements)

111 -> ¬211 ∧ ¬ 311 ∧ ¬ 411
112 -> ¬212 ∧ ¬ 312 ∧ ¬ 412
113 -> ¬213 ∧ ¬ 313 ∧ ¬ 413
114 -> ¬214 ∧ ¬ 314 ∧ ¬ 414

No number appears twice in a box (64x)

111 -> ¬121 ∧ ¬ 211 ∧ ¬ 221
112 -> ¬122 ∧ ¬ 212 ∧ ¬ 222
113 -> ¬123 ∧ ¬ 213 ∧ ¬ 223
114 -> ¬124 ∧ ¬ 214 ∧ ¬ 224



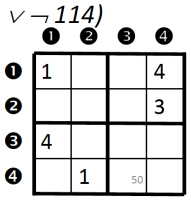
How to write the constraints in PL?

In total: 64 variables, 4x64+16=272 statements

We have to write the given numbers:
111 ∧ 144 ∧ 243 ∧ 314 ∧ 422

All of these have to be put in clausal form:
111 -> ¬112 ∧ ¬ 113 ∧ ¬ 114
= (¬111 ∨ ¬112) ∧ (¬111 ∨ ¬ 113) ∧ (¬111 ∨ ¬ 114)

There are multiple encodings.
Read:
http://anytime.cs.umass.edu/aimath06/proceedings/P34.pdf



So now our puzzle is:

111 ∧ 144 ∧ 243 ∧ 314 ∧ 422
(¬111 ∨ ¬112) ∧ (¬111 ∨ ¬ 113) ∧ (¬111 ∨ ¬ 114)
….
…..

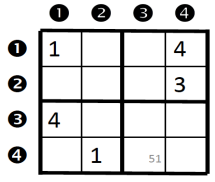
272

And we can ask:
"can we find a truth-assignment to all variables which makes all these 272 statements true"

How do we find such a truth-assignment?

Two methods: truth-tables & Davis-Putnam:

Truth tables = 2^{64} rows
= 18,446,744,073,709,551,616
= 18 million trillion…
Question: what would this by for 9x9?
So let's try Davis-Putnam!



Logic, Hardness, GSAT, MaxSat

Knowledge Representation as Logic Engineering

• Previously , you might have learned that Logics exist , how they are defined , what their theoretical properties are, etc.
• Knowledge Representation is
• The field of using the right logic for the right AI task
• Evaluating logics w.r.t. a task or in general
• Analytically (e.g. soundess , completeness , decidability , complexity , etc
• Empirically (practical complexity , practical completeness
• Adapt existing logics for a task (we do that in the course)
• Develop new logics if needed

KR in a nutshell: (syntax, semantics, calculus)

- A possible practical problem: Politics or parties in the news
- Syntax: $\mathcal{PPP} = (PN)*$ e.g. $PNPP \in \mathcal{PPP}$
- Semantics: (intended meaning)
  - Interpretation:
    - $(\boldsymbol{Pstr})^{\mathcal{I}} = (\boldsymbol{str})^{\mathcal{I}} + \mathbf{1}$
    - $(\boldsymbol{Nstr})^{\mathcal{I}} = (\boldsymbol{str})^{\mathcal{I}} - \mathbf{2}$
    - $(\varnothing)^{\mathcal{I}} = \mathbf{0}$
  - Truth (of equality of terms): $\models \boldsymbol{str_1} = \boldsymbol{str_2}$ iff, $\boldsymbol{str_1^{\mathcal{I}}} = \boldsymbol{str_2^{\mathcal{I}}}$
- Calculus: $\vdash \boldsymbol{str_1} = \boldsymbol{str_2}$
  - iff, $\#\boldsymbol{N}(\boldsymbol{str_1}) = \#\boldsymbol{N}(\boldsymbol{str_2})$ and $\#\boldsymbol{P}(\boldsymbol{str_1}) = \#\boldsymbol{P}(\boldsymbol{str_2})$

KR in a nutshell 2 (Theory about Logics)

- Properties of a logic:
  - Soundness: $\vdash \boldsymbol{str_1} = \boldsymbol{str_2}$ implies $\models \boldsymbol{str_1} = \boldsymbol{str_2}$
  - Completeness: $\models \boldsymbol{str_1} = \boldsymbol{str_2}$ implies $\vdash \boldsymbol{str_1} = \boldsymbol{str_2}$
- Our algorithm is sound but incomplete: why?
  - $\models \boldsymbol{PPNP} = \boldsymbol{P}$, but not $\vdash \boldsymbol{PPNP} = \boldsymbol{P}$
  - There are several sound and complete calculi for $\mathcal{PPP}$
- Decision problem, and decidability:
  - A terminating, sound and complete calculus (for $\mathcal{PPP}$) decides a decision problem of the logic (whether $\models \boldsymbol{str_1} = \boldsymbol{str_2}$).
  - In decidable logics, such algorithms exist, otherwise a logic is called undecidable.
  - Finally, we are interested in the complexity of the logic, i.e. how difficult it is to solve the satisfiability problem.

Reasoning Tasks and Properties of Logics

Propositional Logic (reminder)

- Syntax
  - Propositions P, Q
  - SentencesS1, S2, … basedo n complex operators v, -
- Semantics
  - Assignments I
    - I(P) = True or False
  - Assignment
    - I(S1 v S2)=True iff I(S1)=True or I(S2)=True
    - I(-S)=True iff I(S)=False
- Calculus and their properties
  - Check with Truth Tables
  - Check with DPLL

Defining Syntax of Propositional Logic

- Context-free grammars written in Bachus-Naur form (BNF) have 4 components
  - A set of terminal symbols (Words, Propositions,"True", "False", etc)
  - A set of non-terminal symbols (<sentence>,<op>,<formulas>)
  - A start symbol (*Sentence, Expr, Program*)
  - A set of rewrite rules of the form LHS ::= RHS
- Propositional Logic in BNF
  - <sentence> ::= *p* | T | F where *p*∈Propositions, and T and F be used for True and False
  - <sentence > ::= ¬ <sentence >
  - <sentence> ::= (<sentence> <op> <sentence>)
  - <op> ::= ∧ | ∨ |⟹|⟺ alternatively <op> ::= & | ∨ | -> | ⇔

Reminder: Semantics
Entailment, Validity and satisfiability

- A sentence α is entailed by a KB K *iff* α holds/is true in all models of KB. We write KB $\models \alpha$
- 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

Semantics versus Calculus

- A sentence S is *satisfiable* if it is true in *some* model (we write $\models$ S)
- DPLL(S) returns SAT or UNSAT (we write $\vdash$ S)

Intention is that these two things deliver the same things!

DPLL complexity

- DPLL determines **satisfiability** of formulas in CNF, i.e.
  - dpll(KB)=sat iff Knowledge Base KB is satisfiable

- The worst case complexity of the DP algorithm $O(1,618^n)$.
- This is an improvement!. . . Notice that, for example,

$$\begin{aligned} 2^{50} &= 1.125.899.906.842.624 \\ 1,618^{50} &= 28.114.208.661 \end{aligned}$$

DEFINITION: Properties of inference

- If an algorithm ⊢ proves **only satisfiabile** sentences S it is ***sound*, i.e.** *if*
  ⊢ S *also* ⊨ S
- It is **complete** if it derives **all satisfiable** sentence, i.e.
  ⊨ S implies ⊢ S
- A calculus **terminates** if it finds a model in finite time
- A logic is **decidable** if there is a sound and complete calculus that terminates

Properties of DPLL based SAT solving

Our calculus

1. ⊢ S : DPLL(S) is **sound**
2. ⊢ S : DPLL(S) is **complete**
3. ⊢ S : DPLL(S) **terminates**

Which implies that KB ⊢ α is a decision procedure for SAT solving of propositional logic.

Proof(sketch) for DPLL based Entailment

Our calculus

1. $\vdash$ S : DPLL(S) is **sound**
2. $\vdash$ S : DPLL(S) is **complete**
3. $\vdash$ S : DPLL(S) **terminates**

Proof sketch: Show that:

1. If $\vdash$ **S** then $\models$S. $\vdash$ **S** means that DPLL returns SAT, i.e. it found a partial assignment. This assignment is a model for S.
2. If $\models$ S, S is satisfiabile, i.e. has a model M. This model M will be an assignment found by DPLL, as DPLL tries all combinations.
3. $\vdash$ **S** assignes, in the worst case, truth values True and False to all propositional variables in S. There are only finitely many variables.

Proof of Entailment by Refutation

We define a procedure **KB** $\vdash$ **a** {

1. Translate KB into CNF to get cnf(KB)
2. Translate $\neg$a into CNF to get cnf($\neg$a)
3. Add cnf($\neg$a) to cnf(KB)
4. Apply SAT solver (such as DPLL) until
   - either model is found (satisfiable) or
   - search exhausted (unsatisfiable)
5. If satisfiable, return "FALSE", otherwise "TRUE".

}

This works in Propositional Logic, not in all logics

Semantics versus Calculus

- **KB** $\models$ α**:** A sentence α is entailed by a KB K *iff* α holds/is true in all models of KB.

- **KB** $\vdash$ α **:** DPLL(cnf(KB) + cnf({~a})) returns UNSAT

Intention is that these two things deliver the same things!

Properties DPLL based Entailment

Our calculus

- KB $\vdash$ α : DPLL(cnf(KB) + cnf({~a})) is **sound**
- KB $\vdash$ α : DPLL(cnf(KB) + cnf({~a})) is **complete**
- KB $\vdash$ α : DPLL(cnf(KB) + cnf({~a})) **terminates**

Which implies that KB $\vdash$ α is a decision procedure (or decides) entailment of propositional logic KB $\models$ a

.

Propositionalizing

Propositional logic is a weak language

• Propositional letters only describe complete "states" of the world, we cannot talk about "individuals" (e.g., Mary, 3)
• Can't directly talk about properties of individuals or relations between individuals (e.g., "Bill is tall")
• Generalizations, patterns, regularities can't easily be represented (e.g., "all triangles have 3 sides")
• First Order Logic (abbreviated FOL or FOPC) is expressive enough to concisely represent this kind of information

FOL adds relations, variables, and quantifiers, e.g.,

- "Every elephant is x (elephant(x) gray(x))
- "There is a white x (alligator(X) ^ white(X))

Propositionalising First Order Logic over finite domains

$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

2 individuals Alice & Bob

smokesA, smokesB, cancerA, cancerB
friendsAB, friendsBA, friendsAA, friendsBB

smokesA ⇒ cancerA,
smokesB ⇒ cancerB
friendsAB ⇒ (smokesA ⇔ smokesB)
friendsBA ⇒ (smokesB ⇔ smokesA)
friendsAA ⇒ (smokesA ⇔ smokesA)
friendsBB ⇒ (smokesB ⇔ smokesB)

Problem solving with Propositionalisation

In order to solve a problem using Propositionalisation
1. formulate problem as a complex formula F in first order logic
2. propositionalise F to get a Propositional statement S
3. find a "satisfying assignment" of truth values (in other words: a model of S)
4. This model gives us the solution to the original problem.

Remember:
if S contains n symbols, we need to check ~1,681 ^{n} nodes
Complexity: if S is the result of propositionalising a FOL sentence, then n is exponential in the number of individuals i in the domain
--> DPLL may have to check doubly exponentially many nodes: $1{,}681^{2n}$
n=10 gives a truth table of size size10230

Hard Satisfiability Problems

Structured vs. Random Problems

• So far, we've been dealing with SAT problems that encode other problems
• Most not as hard as # of variables & clauses suggests
• Small crossword grid + medium sized dictionary may turn into a big formula … but still a small puzzle at some level
• Unit propagation does a lot of work for you
• Clause learning picks up on the structure of the encoding
• But some random SAT problems really are hard!
• Fix the length (often 3)
• Vary the ratio between number of clauses l and number of variables n

The magic ratio 4.26

Complexity peak is very stable …
across problem sizes
across solver types
systematic (last lecture)
stochastic (this lecture)

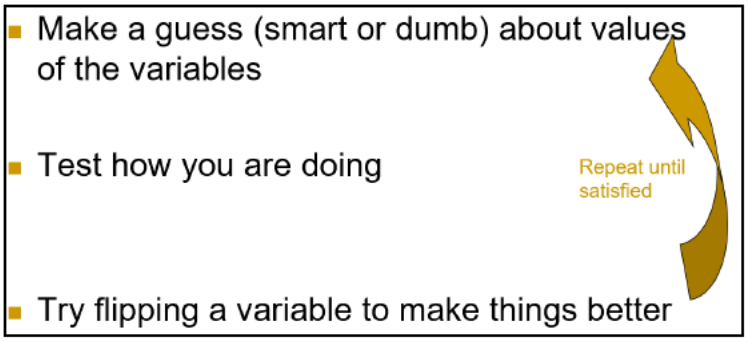
That's called a "phase transition"

• Problems < 0˚C are like ice; > 0˚C are like water
• Similar "phase transitions" for other NP hard problems
• job shop scheduling
• traveling salesperson
• exam timetables
• Boolean circuit synthesis
• Latin squares (alias sports scheduling)

Incomplete SAT methods (GSAT)

Local search for SAT

- Make a guess (smart or dumb) about values of the variables
- Test how you are doing
- Try flipping a variable to make things better
- Algorithms differ on which variable to flip



- Flip a randomly chosen variable?
- No, blundering around blindly takes exponential time
- Ought to pick a variable that improves …
- Increase the # of satisfied clauses as much as possible
- Break ties randomly
- Note: Flipping a var will repair some clauses and break others
- This is the "GSAT" (Greedy SAT) algorithm (almost)

- Flip most-improving variable. Guaranteed to work?

A v C
A v B
~C v ~B
~B v ~A

- What if first guess is A=1, B=1, C=1?
  - 2 clauses satisfied
  - Flip A to 0 ➔ 3 clauses satisfied
  - Flip B to 0 ➔ all 4 clauses satisfied (pick this!)
  - Flip C to 0 ➔ 3 clauses satisfied

- Flip most-improving variable. Guaranteed to work?

A v C
A v B
~C v ~B
~B v ~A

- But what if first guess is A=0, B=1, C=1?
  - 3 clauses satisfied
  - Flip A to 1 ➔ 3 clauses satisfied
  - Flip B to 0 ➔ 3 clauses satisfied
  - Flip C to 0 ➔ 3 clauses satisfied
  - Pick one anyway … (picking A wins on next step)

- Flip most-improving variable. Guaranteed to work?
- **Yes** for 2-SAT: probability → 1 within $O(n^2)$ steps
- **No** in general: can & usually does get locally stuck
- Therefore, GSAT just restarts periodically
  - New random initial guess

The GSAT Procedure

- This algorithm is due to Selman, Levesque and Mitchell
  - Adds restarts to the simple "greedy" algorithm, and also allows sideways flips (not necessarily increasing the "cost function.")

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES ; These are the restarts
    T := random(Sigma) ; random assignment
    for j := 1 to MAX-FLIPS ; To ensure termination
      if T satisfies Sigma then return T
      else T := T with variable flipped to maximize
              number of satisfied clauses
      ; It doesn't matter if the number does
      ; not increase.  This are the sideways flips
    end
  end
```

Believe it or not, GSAT outperformed everything else when it was invented in 1992.

Decidability? Completeness?

• What does that mean for decidability?
• What does that mean for completeness?
• Remember: Completeness means that all correct solutions are found
• Soundness: If our algorithm returns a variable assignment with input KB, this variable assignment is a model of KB
• Completeness: If an assignment is a model, then our algorithm will find it.

Who cares about decidability?

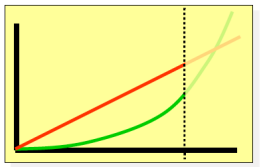
• Decidability ≈ completeness guarantee to find an answer, or tell you it doesn't exist, given enough run time & memory
• Sources of incompleteness
• incompleteness of the input data
• insufficient run time to wait for the answer
Ⓟ Completeness is unachievable in practice anyway, regardless of the completeness of the algorithm

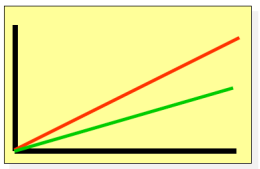
Who cares about undecidability?

• Undecidability ≠ always guaranteed not to find an answer
• Undecidability = not always guaranteed to find an answer
• Undecidability may be harmless in many cases; in all cases that matter

Who cares about complexity?

- **worst-case**: may be exponentially rare
- **Asymptotic O(n) vs O(n^2)**
- **ignores constants**





Clause Learning, Planning and Applications

Clause Learning and Dependency directed backtracking

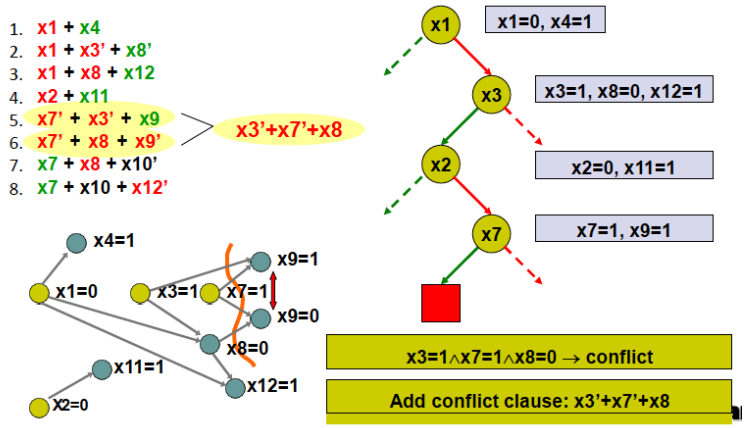
Chronological Backtracking

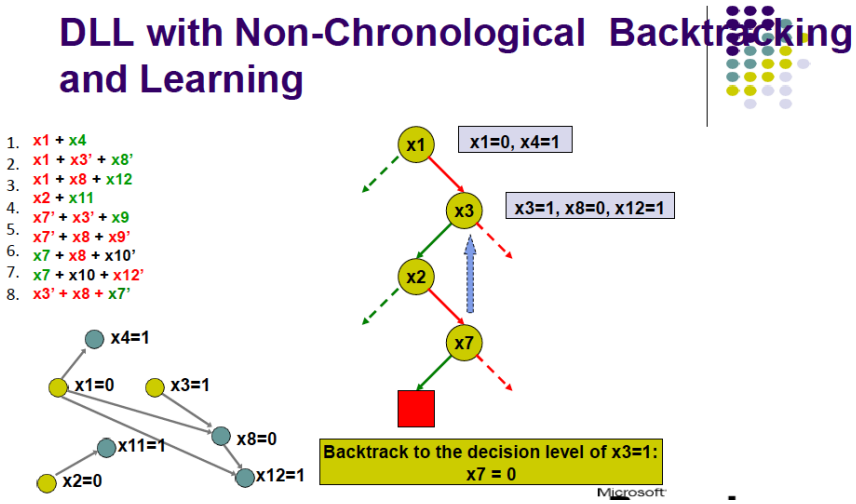
- Backtracking to the highest decision level that has not been tried with both values
- Originally proposed in the DLL paper in 1962
- OK for randomly generated instances, bad for instances generated in practical applications
- We can do better than that

Conflict Driven Learning and Non-chronological Backtracking

1. **x1 + x4**
2. **x1 + x3' + x8'**
3. **x1 + x8 + x12**
4. **x2 + x11**
5. **x7' + x3' + x9**
6. **x7' + x8 + x9'**
7. **x7 + x8 + x10'**
8. **x7 + x10 + x12'**

Notation: + is "or" (V), a ' is "not", ¬
So: x1 + x3' + x8' = x1 V ¬x3 V ¬x8





Summarizing

We have now gained two advantages:

1. We have “learned” a new clause that “ summarizes ” a potential conflict. Adding this clause to the clause base will help us to avoid making the same mistake in other parts of the search space.
2. We can make a bigger “jump back” in the search tree, replacing chronological backtracking (= just undoing the most recent choice), with dependency directed backtracking (= undoing the most recent choice that was involved in the conflict, ie the most recent choice that mattered).

Chapter 7: Propositional Satisfiability Techniques

l Propositional satisfiability: given a boolean formula

» e.g., $(P \vee Q) \wedge (\neg Q \vee R \vee S) \wedge (\neg R \vee \neg P)$,

does there exist a model

» i.e., an assignment of truth values to the propositions

that makes the formula true?

- approach:
  Try translating classical planning problems into satisfiability problems, and solving them that way

State Transition System

$\Sigma = (S,A,\gamma)$

- states $S = \{\ldots, s_i, \ldots\}$
- actions $A = \{\ldots, a_i, \ldots\}$
- State-transition function
  $\gamma: S \times A \rightarrow 2^S$
  - $S = \{s_0, \ldots, s_5\}$
  - A = {move1, move2, put, take, load, unload}
  - γ: see the arrows

Planning Problem

$P = (\Sigma, s_0, s_G)$

- Description of Σ
- Initial state or set of states

Initial state = s_0

- Objective

Goal state, set of goal states, set of tasks, "trajectory" of states, objective function, …

Goal state = s_5

Plans

- **Classical plan**: a sequence of actions
  $\pi = \langle a_0, a_1, \ldots, a_{n-1} \rangle$

⟨take, move1, load, move2⟩

Policy: partial function from S into A

{(s_0, take),
(s_1, move1),
(s_3, load),
(s_4, move2)}

Solution length = 4

Overall Approach

- A *bounded planning problem* is a pair (P,n):
  - P is a planning problem; n is a positive integer
  - Any solution for P of length n is a solution for (P,n)
- Planning algorithm: Do iterative deepening
  - for $n = 0, 1, 2, \ldots,$
    - » encode (P,n) as a satisfiability problem Φ
    - » if Φ is satisfiable, then

- From the set of truth values that satisfies Φ, a solution plan can be constructed, so return it and exit

Notation

- For satisfiability problems we use propositional logic
- Need to encode ground atoms into propositions
  - » Atom: at(r1,loc1)
  - » Proposition: at-r1-loc1
  - But we won't bother to do a syntactic rewrite
    Just use at(r1,loc1) itself as the proposition
- Also, we'll write plans starting at a_0
  - $\pi = \langle a_0, a_1, \ldots, a_{n-1} \rangle$

Fluents

- If $\pi = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ is a solution for (P,n), it generates these states:

s_0, $s_1 = \gamma(s_0,a_0)$, $s_2 = \gamma(s_1,a_1)$, …, $s_n = \gamma(s_{n-1}, a_{n-1})$

- *Fluent*: proposition saying a particular atom is true in a particular state
  - at(r1,loc1,i) is a fluent that's true iff at(r1,loc1) is in s_i
  - We'll use l_i to denote the fluent for literal l in state s_i
    - » e.g., if l = at(r1,loc1)

  then l_i = at(r1,loc1,i)
  - a_i is a fluent saying that a is the i'th step of π
    - » e.g., if a = move(r1,loc2,loc1)

  then a_i = move(r1,loc2,loc1,i)

Encoding Planning Problems

- Encode (P,n) as a formula Φ such that

$\pi = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ is a solution for (P,n) if and only if
Φ can be satisfied in a way that makes the fluents $a_0, \ldots, a_{n-1}$ true

- Let
  - A = {all actions in the planning domain}
  - S = {all states in the planning domain}
  - L = {all literals in the language}
- Φ is the conjunction of many formulas …

Formulas in Φ

- Formula describing the initial state:

$\bigwedge\{l_0 \mid l \in s_0\} \ \bigwedge \ \bigwedge\{\neg l_0 \mid l \in L - s_0\}$

- Formula describing the goal:

$\bigwedge\{l_n \mid l \in g^+\} \bigwedge \bigwedge\{\neg l_n \mid l \in g^-\}$

- For every action a in A, formulas describing what changes a would make if it were the i'th step of the plan:
  - $a_i \Rightarrow \bigwedge\{p_i \mid p \in \text{Precond}(a)\} \bigwedge \bigwedge\{e_{i+1} \mid e \in \text{Effects}(a)\}$
- *Complete exclusion* axiom:

- For all actions a and b, formulas saying they can't occur at the same time $\neg a_i \vee \neg b_i$
- this guarantees there can be only one action at a time

- Is this enough?

Frame Axioms

- *Frame axioms*:
  - Formulas describing what *doesn't* change between steps i and i+1
- Several ways to write these
- One way: *explanatory frame axioms*
  - One axiom for every literal l
  - Says that if l changes between s_i and s_{i+1}, then the action at step i must be responsible:

$$(\neg l_i \wedge l_{i+1} \Rightarrow \mathrm{V}_{a \text{ in } A}\{a_i \mid l \in \text{effects}^+(a)\}) \wedge (l_i \wedge \neg l_{i+1} \Rightarrow \mathrm{V}_{a \text{ in } A}\{a_i \mid l \in \text{effects}^-(a)\})$$

Example

- Planning domain:
  - one robot r1
  - two adjacent locations l1, l2
  - one operator (move the robot)
- Encode (P,n) where $n = 1$
  - Initial state: {at(r1,l1)}

  Encoding: at(r1,l1,0) ∧ ¬at(r1,l2,0)
  - Goal: {at(r1,l2)}

  Encoding: at(r1,l2,1) ∧ ¬at(r1,l1,1)
  - Operator: see next slide

- Operator: move(r,l,l')
  precond: at(r,l)
  effects: at(r,l'), ¬at(r,l)

  Encoding:
  move(r1,l1,l2,0) ⇒ at(r1,l1,0) ∧ at(r1,l2,1) ∧ ¬at(r1,l1,1)
  move(r1,l2,l1,0) ⇒ at(r1,l2,0) ∧ at(r1,l1,1) ∧ ¬at(r1,l2,1)
  move(r1,l1,l1,0) ⇒ at(r1,l1,0) ∧ at(r1,l1,1) ∧ ¬at(r1,l1,1){contradictions-
  move(r1,l2,l2,0) ⇒ at(r1,l2,0) ∧ at(r1,l2,1) ∧ ¬at(r1,l2,1){easy to detect}
  move(l1,r1,l2,0) ⇒ …{
  move(l2,l1,r1,0) ⇒ … non-essential
  move(l1,l2,r1,0) ⇒ …
  move(l2,l1,r1,0) ⇒ … }
- How to avoid generating the last four actions?
  - Assign data types to the constant symbols: order-sorted logic

- Locations: l1, l2
- Robots: r1

- Operator: move(r : robot, l : location, l' : location)
  precond: at(r,l)
  effects: at(r,l'), ¬at(r,l)

  Encoding:
  move(r1,l1,l2,0) ⇒ at(r1,l1,0) ⋀ at(r1,l2,1) ⋀ ¬at(r1,l1,1)
  move(r1,l2,l1,0) ⇒ at(r1,l2,0) ⋀ at(r1,l1,1) ⋀ ¬at(r1,l2,1)

- Complete-exclusion axiom:

¬move(r1,l1,l2,0) ⋁ ¬move(r1,l2,l1,0)

- Explanatory frame axioms:

¬at(r1,l1,0) ⋀ at(r1,l1,1) ⇒ move(r1,l2,l1,0)
¬at(r1,l2,0) ⋀ at(r1,l2,1) ⇒ move(r1,l1,l2,0)
at(r1,l1,0) ⋀ ¬at(r1,l1,1) ⇒ move(r1,l1,l2,0)
at(r1,l2,0) ⋀ ¬at(r1,l2,1) ⇒ move(r1,l2,l1,0)

Extracting a Plan

- Suppose we find an assignment of truth values that satisfies Φ.
  - This means *P* has a solution of length *n (as used in the encoding)*
- For *i*=1,…,*n*, there will be exactly one action *a* such that a_i = *true*
  - This is the *i*'th action of the plan.
- Example (from the previous slides):
  - Φ can be satisfied with move(r1,l1,l2,0) = *true*
  - Thus ⟨move(r1,l1,l2,0)⟩ is a solution for (*P*,0)
    - » It's the only solution - no other way to satisfy Φ

Planning

- How to find an assignment of truth values that satisfies Φ?
  - Use a satisfiability algorithm:
  - Davis-Putname
  - Local Search
  - GSAT
  - …

Discussion

- Recall the overall approach:
  - for *n* = 0, 1, 2, …,
    - » encode (*P,n*) as a satisfiability problem Φ
    - » if Φ is satisfiable, then
      - From the set of truth values that satisfies Φ, extract a solution plan and return it
      - How well does this work?
  - By itself, not very practical (takes too much memory and time)
  - But it can be combined with other techniques
    - » e.g., planning graphs

SatPlan

- SatPlan combines planning-graph expansion and satisfiability checking, roughly as follows:

- for k = 0, 1, 2, …
  - Create a planning graph that contains k levels
  - Encode the planning graph as a satisfiability problem
  - Try to solve it using a SAT solver
    - If the SAT solver finds a solution within some time limit,
      - Remove some unnecessary actions
      - Return the solution
      - Memory requirement still is combinatorially large
- but less than what's needed by a direct translation into satisfiability

- BlackBox (predecessor to SatPlan) was one of the best planners in the 1998 planning competition
- SatPlan was one of the best planners in the 2004 and 2006 planning competitions

Exponential Complexity Growth: The Challenge of Complex Domains

SAT Encoding (automatically generated from problem specification)

Applications of SAT (1)
Model Checking (hardware/software verification)
– Bounded model checking (BMC)
– Enhancement techniques: Abstraction and refinement
– Major groups at Intel, IBM, Microsoft, and universities such as CMU, Cornell, and Princeton
– **SAT has become a dominant back-end technology**

Key questions:
– Safety property: can system ever reach state S?
– Liveness property: will system always reach T after it gets to S?

Main idea:
– Encode "reachability" in a finite transition graph
– CNF encoding: in simple terms, "similar" to planning

Classical Planning
– Parallel step optimal plans
– E.g., SATPLAN-06 fastest in this category at IPC-06 planning competition
Main idea:
– Create planning graph, unfolded up to T steps
– Encode legal transitions as CNF
• Variables: action vars, fact vars for each time step t
• Clauses encode action preconditions, effects, mutual exclusion, etc.
e.g., (actionA_(t+1) preA1_t AND preA2_t AND preA3_t),
(NOT actionA_t OR NOT actionB_t), …

Combinatorial Design
– Complex mathematical structures with desirable properties
• Latin squares

• Quasi groups
• Ramsey numbers
• Steiner systems
• …
– Highly useful for
• design of experiments
• coding theory, cryptography
• drug design and testing
• crop rotation schedules

Has been applied to solve sub-problems in many domains!
– Test pattern generation
– Scheduling
– Optimal Control
– Protocol Design (e.g., for networks)
– Multi-agent systems
– E-Commerce (E-auctions and electronic trading agents), etc.

Propositional Logic, MaxSat + Exam Preparation

The MAXSAT Problem

• The well-known SAT problem is to determine if a boolean formula in Conjunctive Normal Form (CNF) has a satisfying truth assignment
• A CNF formula is a conjunction of clauses
• A clause is a disjunction of literals (variables or their negations)
• 2 denotes the empty clause (falsified by every truth assignment)
• A satisfying assignment assigns true to at least one literal of every clause.
• MAXSAT is an optimization extension of SAT that asks what is the maximum number of clauses that can be simultaneously satisfied

Example
F = (¬x)1(x 2y)1(¬y)1(z 2w)
F is unsatisfiable, so no truth assignment can satisfy all 4 clauses.
Truth assignment u = {x,¬y, z,¬w} satisfies all clauses except (¬x).)
U(pie) is a solution to the MAXSAT problem F.

Some clauses may be more important to satisfy than others
• This can be modeled by associating a positive cost with each clause C that will be incurred if C is falsified
If it is mandatory to satisfy C, its cost is / and C is called hard
• Otherwise, C is called soft
Example
F = (¬x,/(infinity))1(x 2y, 4)1(¬y, 1)1(z 2w,/(infinity))
In F, (¬x,/) is a hard clause, and (x 2y, 4) is soft with cost 4.

A truth assignment u has cost equal to the sum of the
costs of the clauses it falsifies

• Goal: find an optimal feasible truth assignment, i.e., a truth assignment of minimum finite cost mincost(F)
Example
F = (¬x,/(infinity))1(x 2y, 4)1(¬y, 1)1(z 2w,/(infinity))
u = {¬x, y, z,¬w} satisfies all clauses except (¬y, 1)
u is optimal: mincost(F)=cost(u)=1.

We use cost(C) to denote the cost of clause C.
• A solution must satisfy all hard clauses (else its cost will be infinite).
• A solution also satisfies a maximum total cost of soft clauses.
• Casting as minimization problem more closely corresponds to how most MAXSAT solvers work.
• Many solutions might exist—typically we are only interested in finding one, sometimes only interested in finding out the cost of a solution.

MAXSAT (ms) (standard MAXSAT): no hard clauses and all
clause have weight 1.
• Solution maximizes the number of satisfied clauses.
• Weighted MAXSAT (wms): no hard clauses.
• Partial MAXSAT (pms): have hard clauses but all soft clauses have weight 1.
• Weighted Partial MAXSAT (wpms): the version we have defined here (subsumes all other versions).
• Standard MAXSAT is most interesting for theory: it already has sufficient structure for theoretical insights.
• Other versions mostly an artifact of the limitations of earlier MAXSAT solvers.

Exam Preparation (I tried to answer them, emphasis on tried...)

1)

In a proof by contradiction, such as DPLL, I can prove that a formula F is entailed by a knowledge base KB by showing that

A the knowledge base KB and the formula F are together unsatisfiable.

B the knowledge base is unsatisfiable, which implies that the formula F must be entailed.

C the knowledge base KB and the negation of the formula are unsatisfiable.

D the formula F is unsatisfiable, which implies that it must be entailed by the KB.

2)

Logic Engineering

Consider a language L defined as follows
•Syntax

{A,B,C} L
If F1, F2, F3 L then "(F1*F2*F3)" L

•Semantics: let I be an interpretation
I(A)=1, I(B)=2, l(C)=3
I(F1*F2*F3) = I(F3).I(F1).I(F2)

What is I(A*B*(C*A*B))?

3)

Semantics: the meaning of expressions

Fill in the truth value of the following formulas under the assumption that A,B,C and D are all false. Fill in the truth values of the formulas (use the words "True, False").

(A & B) v -C v -D is : True
Conjunction, or, disjunction

_____
(A & B) v (-C & D) is True
Conjunction, or, conjunction

_____
(A ->B) v (C ->D) is True
Implies, or, implies _____
(-A ->B) v (-C ->D) is False
_____

4)

Semantics 2

Which of the following is true? The propositional statement
(PvQ) --> (P&Q)
To be either false or true (truth assignment)
contingency: neither a tautology (true and valid) nor a contradiction (false)

A is satisfiable, but not valid

B is valid

C is a contradiction

D neither valid, satisfiable nor a contradiction

5)

Automated Reasoning as Search

Automated reasoning can often be seen as search procedure, that tries to find an order of rules to formulas in a knowledge base.

Which of the following statements about the DPLL (Davis Putnam Logemann Loveland) Algorithm and Search is correct?

A DPLL recursively searches through all possible variable assignments for a model, i.e. an interpretation that satisfies all the clauses.

B DPLL iteratively searches through the set of all clauses for one that is satisfied by a given interpretation.

C DPLL exhaustively applies a set of transformation rules to produce a contradiction

D DPLL cannot be seen as a search procedure, it only randomly assigns values to propositions until it finds an assignment that satisfies all the clauses

6)

DPLL

Consider the following knowledge base (set of clauses):

Definition 2.1.1. A tautology is a proposition that is always true.
Example 2.1.1. $p \vee \neg p$

Definition 2.1.2. A contradiction is a proposition that is always false.
Example 2.1.2. $p \wedge \neg p$

Definition 2.1.3. A contingency is a proposition that is neither a tautology nor a contradiction.
Example 2.1.3. $p \vee q \rightarrow \neg r$

1: $\neg p \vee q \vee r$
2: $p \vee \neg q \vee \neg r$
3: $\neg p \vee \neg q \vee \neg r$
4: $p \vee q$
5: $p \vee r$
6: $\neg p \vee \neg q \vee r$

and show whether the knowledge base is satisfiable or not using DPLL. Use the following order: unit, pure, split alphabetic with a positive value.

Assign p =true
2: $p \vee \neg q \vee \neg r$
4: $p \vee q$
5: $p \vee r$

Assign q=true
1: ¬ p v q v r
4: p v q

Find unit clause 6: ¬ p v ¬ q v r, so r = true
3: ¬ p v ¬ q v ¬ r

7)

Properties of GSAT

GSAT is complete w.r.t Propositional Logic satisfiability. True or false?

GSAT: Local greedy search/algorithm of picking the next variable to flip which increases the number of satisfied clauses the most (ties are broken randomly, note that flips will break also some clauses).

A True

B False

Probabilistic Graphical Models 1

Foundations

Introduction

Probabilistic Graphical Models is a major field in AI at the crossroads of KR, Statistics and Learning.

- It has vast amount of applications:
- Decision Making
- Image Segmentation
- Medical Diagnoses
- Speech Recognition
- Natural Language Processing

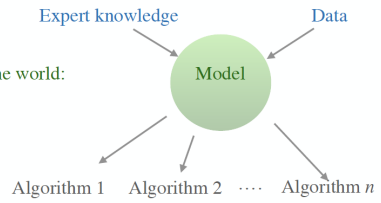
Motivation

- Probabilistic Graphical Models is a major field in AI at the crossroads of KR, Statistics and Learning.

Choice of representing uncertainty: Probability theory with declarative formal semantics.

Representing a potentially complex system compactly: a business of Computer Science.

Stand alone-model, representing the world:



Degrees of Beliefs

- (Classical) Knowledge base: a coarse (binary!) classification of the world:
  - Possible
  - Not Possible

- We can do better(!): a finer classification of sentences through a finer classification of worlds.
  - A *degree of belief* or *probability* in $[0, 1]$ assigned to each world ω : $\Pr(\omega)$

- Then, the probability/belief of/in, for a given sentence α $\Pr(\alpha) \stackrel{def}{=} \sum_{\omega \models \alpha} \Pr(\omega)$ (B1)

  which is the sum of probabilities assigned to worlds that α is true.

Degrees of Belief

Example:

world	Earthquake	Burglary	Alarm	Pr(.)
ω_1	true	true	true	.0190
ω_2	true	true	false	.0010
ω_3	true	false	true	.0560
ω_4	true	false	false	.0240
ω_5	false	true	true	.1620
ω_6	false	true	false	.0180
ω_7	false	false	true	.0072
ω_8	false	false	false	.7128

A state of belief, is also called a joint probability distribution.

$$\begin{aligned}
\Pr(\text{Earthquake}) &= \Pr(\omega_1) + \Pr(\omega_2) + \Pr(\omega_3) + \Pr(\omega_4) = .1 \\
\Pr(\text{Burglary}) &= .2 \\
\Pr(\neg\text{Burglary}) &= .8 \\
\Pr(\text{Alarm}) &= .2442
\end{aligned}$$

Properties of Belief

- $0 \leq \Pr(\alpha) \leq 1$ for any sentence α.
- $\Pr(\alpha) = 0$ when α is inconsistent.
- $\Pr(\alpha) = 1$ when α is valid.
- $\Pr(\alpha) + \Pr(\neg\alpha) = 1$.
- $\Pr(\alpha \vee \beta) = \Pr(\alpha) + \Pr(\beta) - \Pr(\alpha \wedge \beta)$.

Properties of Belief
(Partitioned Worlds)

$$\Pr(\alpha) + \Pr(\neg\alpha) = 1.$$

Example:

world	Earthquake	Burglary	Alarm	Pr(.)
ω_1	true	true	true	.0190
ω_2	true	true	false	.0010
ω_3	true	false	true	.0560
ω_4	true	false	false	.0240
ω_5	false	true	true	.1620
ω_6	false	true	false	.0180
ω_7	false	false	true	.0072
ω_8	false	false	false	.7128

Observation: The et of all worlds.

$$\Pr(\text{Burglary}) = \Pr(\omega_1) + \Pr(\omega_2) + \Pr(\omega_5) + \Pr(\omega_6) = .2$$
$$\Pr(\neg\text{Burglary}) = \Pr(\omega_3) + \Pr(\omega_4) + \Pr(\omega_7) + \Pr(\omega_8) = .8$$

VU VRIJE UNIVERSITEIT AMSTERDAM

Properties of Belief
(Partitioned Worlds)

$$\Pr(\alpha \vee \beta) = \Pr(\alpha) + \Pr(\beta) - \Pr(\alpha \wedge \beta).$$

Example:

world	Earthquake	Burglary	Alarm	Pr(.)
ω_1	true	true	true	.0190
ω_2	true	true	false	.0010
ω_3	true	false	true	.0560
ω_4	true	false	false	.0240
ω_5	false	true	true	.1620
ω_6	false	true	false	.0180
ω_7	false	false	true	.0072
ω_8	false	false	false	.7128

Observation: Th those satisfying $\alpha \wedge \neg\beta$,

$$\Pr(\text{Earthquake}) = \Pr(\omega_1) + \Pr(\omega_2) + \Pr(\omega_3) + \Pr(\omega_4) = .1$$
$$\Pr(\text{Burglary}) = \Pr(\omega_1) + \Pr(\omega_2) + \Pr(\omega_5) + \Pr(\omega_6) = .2$$
$$\Pr(\text{Earthquake} \wedge \text{Burglary}) = \Pr(\omega_1) + \Pr(\omega_2) = .02$$
$$\Pr(\text{Earthquake} \vee \text{Burglary}) = .1 + .2 - .02 = .28$$

VU VRIJE UNIVERSITEIT AMSTERDAM

Properties of Belief
(Partitioned Worlds)

$$\Pr(\alpha \vee \beta) = \Pr(\alpha) + \Pr(\beta) - \Pr(\alpha \wedge \beta).$$

Observations:

- $\Pr(\alpha \vee \beta) = \Pr(\alpha) + \Pr(\beta)$ when α and β are mutually exclusive.
- In this case, there is no world that satisfies both α and β. Hence, $\alpha \wedge \beta$ is inconsistent and $\Pr(\alpha \wedge \beta) = 0$.

Belief Dynamics

Updating Beliefs

- Recall that:

world	Earthquake	Burglary	Alarm	Pr(.)
ω_1	true	true	true	.0190
ω_2	true	true	false	.0010
ω_3	true	false	true	.0560
ω_4	true	false	false	.0240
ω_5	false	true	true	.1620
ω_6	false	true	false	.0180
ω_7	false	false	true	.0072
ω_8	false	false	false	.7128

- Now assume that we know alarm is triggered i.e., Alarm = true (also called evidence).
- Yet, this knowledge is not compatible with the belief state above (i.e., 0.2442) alarm being true.

- Given evidence β, our goal is to update the state of belief $\Pr(.)$ to $\Pr(.|\beta)$

- Hence we expect $\Pr(.|\beta)$ to assign a belief of 1 to β: $\Pr(\beta|\beta) = 1$.

- This implies $\Pr(\neg\beta|\beta) = 0$ and, hence,

  every world ω that satisfies $\neg\beta$ must be assigned the belief 0:

  $$\Pr(\omega|\beta) = 0 \qquad \text{for all } \omega \models \neg\beta.$$

- To define $\Pr(.|\beta)$, we need to define it in every world ω that satisfies β. But how?

- We already know that the sum of all such beliefs must be 1: $\sum_{\omega \models \beta} \Pr(\omega|\beta) = 1.$ (C1)

- But this leaves us with many options for $\Pr(\omega|\beta)$ when world ω satisfies β.

  What should we do?

The Idea: Minimal Change

- Impossible worlds (0- probability worlds) stay 0:

  $$\Pr(\omega|\beta) = 0 \qquad \text{for all } \omega \text{ where } \Pr(\omega) = 0.$$ (C2)

- Relative probability of positive probability worlds, also stay the same:

  $$\frac{\Pr(\omega)}{\Pr(\omega')} = \frac{\Pr(\omega|\beta)}{\Pr(\omega'|\beta)} \qquad \text{for all } \omega, \omega' \models \beta, \Pr(\omega) > 0, \Pr(\omega') > 0.$$ (C3)

Having those constraints…

- We already know that the sum of all such beliefs must be 1: $\sum_{\omega \models \beta} \Pr(\omega|\beta) = 1.$ (C1)
- Impossible worlds (0- probability worlds) stay 0:
$$\Pr(\omega|\beta) = 0 \qquad \text{for all } \omega \text{ where } \Pr(\omega) = 0.$$ (C2)
- Relative probability of positive probability worlds, also stay the same:
$$\frac{\Pr(\omega)}{\Pr(\omega')} = \frac{\Pr(\omega|\beta)}{\Pr(\omega'|\beta)} \qquad \text{for all } \omega, \omega' \models \beta, \Pr(\omega) > 0, \Pr(\omega') > 0.$$ (C3)

…leaves us with only one option:

$$\Pr(\omega|\beta) = \frac{\Pr(\omega)}{\Pr(\beta)} \qquad \text{for all } \omega \models \beta. \quad \text{(B2)}$$

which is a result of normalising the old belief w.r.t. β

Now, our new state of belief is completely defined:

$$\Pr(\omega|\beta) \stackrel{def}{=} \begin{cases} 0, & \text{if } \omega \models \neg\beta \\ \dfrac{\Pr(\omega)}{\Pr(\beta)} & \text{if } \omega \models \beta. \end{cases}$$

The new state of belief $\Pr(.|\beta)$ is obtained via *conditioning* the old state Pr on evidence β.

Consider the earlier example with the evidence Alarm = true.

world	Earthquake	Burglary	Alarm	Pr(.)	Pr(.|Alarm)
ω_1	true	true	true	.0190	.0190/.2442
ω_2	true	true	false	.0010	0
ω_3	true	false	true	.0560	.0560/.2442
ω_4	true	false	false	.0240	0
ω_5	false	true	true	.1620	.1620/.2442
ω_6	false	true	false	.0180	0
ω_7	false	false	true	.0072	.0072/.2442
ω_8	false	false	false	.7128	0

Pr(Burglary) = .2 Pr(Earthquake) = .1
Pr(Burglary|Alarm) ≈ .741 ↑ Pr(Earthquake|Alarm) ≈ .307 ↑

$$
\begin{aligned}
&\Pr(\alpha|\beta) \\
&= \sum_{\omega \models \alpha} \Pr(\omega|\beta) \quad \text{by (B1)} \\
&= \sum_{\omega \models \alpha,\, \omega \models \beta} \Pr(\omega|\beta) + \sum_{\omega \models \alpha,\, \omega \models \neg\beta} \Pr(\omega|\beta) \quad \text{since } \omega \text{ satisfies } \beta \text{ or } \neg\beta \text{ but not both} \\
&= \sum_{\omega \models \alpha,\, \omega \models \beta} \Pr(\omega|\beta) \quad \text{by (B2)} \\
&= \sum_{\omega \models \alpha \wedge \beta} \Pr(\omega|\beta) \quad \text{by properties of } \models \\
&= \sum_{\omega \models \alpha \wedge \beta} \Pr(\omega)/\Pr(\beta) \quad \text{by (B2)} \\
&= \frac{1}{\Pr(\beta)} \sum_{\omega \models \alpha \wedge \beta} \Pr(\omega) \\
&= \frac{\Pr(\alpha \wedge \beta)}{\Pr(\beta)} \quad \text{by (B1)}
\end{aligned}
$$

$$
\Pr(\alpha) \stackrel{def}{=} \sum_{\omega \models \alpha} \Pr(\omega) \qquad \text{(B1)}
$$

$$
\Pr(\omega|\beta) \stackrel{def}{=} \begin{cases} 0, & \text{if } \omega \models \neg\beta \\ \dfrac{\Pr(\omega)}{\Pr(\beta)} & \text{if } \omega \models \beta. \end{cases} \qquad \text{(B2)}
$$

Hence,

$$
\Pr(\alpha|\beta) = \frac{\Pr(\alpha \wedge \beta)}{\Pr(\beta)} \qquad \textit{(Bayes Conditioning)} \quad \text{(B3)}
$$

Remark: Note that defined only when $\Pr(\beta) \neq 0$.

Summary

- Worlds that contradict the evidence β will have zero probability.
- Worlds that have zero probability continue to have zero probability.
- Worlds that are consistent with evidence β and have positive probability will maintain their relative beliefs.

Scenario I

Evidence: Earthquake = true

Pr(Burglary)	=	.2
Pr(Burglary|Earthquake)	=	.2
Pr(Alarm)	=	.2442
Pr(Alarm|Earthquake)	≈	.75 ↑

Observation: Belief in Burglary does not change while Alarm increases.

Scenario II

Evidence: Burglary = true

Pr(Alarm)	=	.2442
Pr(Alarm|Burglary)	≈	.905 ↑
Pr(Earthquake)	=	.1
Pr(Earthquake|Burglary)	=	.1

Observation: Belief in Alarm increases while Earthquake stays the same.

Scenario I

We know: Pr(Burglary) ↑ if Pr(Alarm) ↑ and Alarm = true.

Question: How does our belief (on Burglary) change with the new evidence of an Earthquake?

Answer:

Pr(Burglary|Alarm)	≈	.741
Pr(Burglary|Alarm ∧ Earthquake)	≈	.253 ↓

Observation: Our belief in Burglary decreases, when we have an explanation for the Alarm.

Scenario II

We know: Pr(Burglary) ↑ if Pr(Alarm) ↑ and Alarm = true.

Question: How does our belief (on Burglary) change with the confirmation of no-Earthquake?

Answer:

Pr(Burglary|Alarm)	≈	.741
Pr(Burglary|Alarm ∧ ¬Earthquake)	≈	.957 ↑

Observation: The new evidence of no-Earthquake strengthens our belief on Burglary (as an explanation for the Alarm).

Recall the original state of belief:

world	Earthquake	Burglary	Alarm	Pr(.)
ω_1	true	true	true	.0190
ω_2	true	true	false	.0010
ω_3	true	false	true	.0560
ω_4	true	false	false	.0240
ω_5	false	true	true	.1620
ω_6	false	true	false	.0180
ω_7	false	false	true	.0072
ω_8	false	false	false	.7128

Observation:

Pr(Earthquake)	=	.1
Pr(Earthquake|Burglary)	=	.1

Conclusion:

According to Pr, evidence of Burglary does not change the belief in Earthquake, hence two events are *independent*.

VU VRIJE UNIVERSITEIT AMSTERDAM

Independence

More generally, given a distribution Pr, the events α and β are independent in Pr *iff*

$$\Pr(\alpha|\beta) = \Pr(\alpha) \quad \text{or } \Pr(\beta) = 0.$$

Observe that Burglary is also independent of Earthquake, according to Pr:

$$\begin{aligned} \Pr(\text{Burglary}) &= .2 \\ \Pr(\text{Burglary}|\text{Earthquake}) &= .2 \end{aligned}$$

Joint probability of independent events

Given a Pr, the event α is independent from the event β w.r.t Pr *iff* $\Pr(\alpha \wedge \beta) = \Pr(\alpha)\Pr(\beta)$.

Remark: Do not mix it up with logical disjointness.

Independence is a dynamic notion: two events which are independent can be dependent after a new evidence.

Example:

$$\begin{aligned} \Pr(\text{Burglary}|\text{Alarm}) &\approx .741 \\ \Pr(\text{Burglary}|\text{Alarm} \wedge \text{Earthquake}) &\approx .253 \end{aligned}$$

Observation: Note that initially independent Burglary and Earthquake became dependent after the evidence Alarm. Why?

Explanation: These are two competing explanations for Alarm confirming one decreases the belief in the other.

world	Temp	Sensor1	Sensor2	Pr(.)
ω_1	normal	normal	normal	.576
ω_2	normal	normal	extreme	.144
ω_3	normal	extreme	normal	.064
ω_4	normal	extreme	extreme	.016
ω_5	extreme	normal	normal	.008
ω_6	extreme	normal	extreme	.032
ω_7	extreme	extreme	normal	.032
ω_8	extreme	extreme	extreme	.128

We have two different sensors which have noise and different reliabilities.

It follows,

$$\begin{aligned} \Pr(\text{Temp} = \text{normal}) &= .80 \\ \Pr(\text{Sensor1} = \text{normal}) &= .76 \\ \Pr(\text{Sensor2} = \text{normal}) &= .68 \end{aligned}$$

Initial Setting:

$$\Pr(\text{Temp} = \text{normal}) = .80$$
$$\Pr(\text{Sensor1} = \text{normal}) = .76$$
$$\Pr(\text{Sensor2} = \text{normal}) = .68$$

world	Temp	Sensor1	Sensor2	Pr(.)
ω_1	normal	normal	normal	.576
ω_2	normal	normal	extreme	.144
ω_3	normal	extreme	normal	.064
ω_4	normal	extreme	extreme	.016
ω_5	extreme	normal	normal	.008
ω_6	extreme	normal	extreme	.032
ω_7	extreme	extreme	normal	.032
ω_8	extreme	extreme	extreme	.128

Follow up:

Suppose, we checked the first sensor and it reads normal. Our belief in the second sensor reading normal as well, would increase (as expected):

$$\Pr(\text{Sensor2} = \text{normal} | \text{Sensor1} = \text{normal}) \approx .768 \uparrow$$

Hence, they are dependent!

Initial Setting:

$$\Pr(\text{Temp} = \text{normal}) = .80$$
$$\Pr(\text{Sensor1} = \text{normal}) = .76$$
$$\Pr(\text{Sensor2} = \text{normal}) = .68$$
$$\Pr(\text{Sensor2} = \text{normal} | \text{Sensor1} = \text{normal}) \approx .768 \uparrow$$

world	Temp	Sensor1	Sensor2	Pr(.)
ω_1	normal	normal	normal	.576
ω_2	normal	normal	extreme	.144
ω_3	normal	extreme	normal	.064
ω_4	normal	extreme	extreme	.016
ω_5	extreme	normal	normal	.008
ω_6	extreme	normal	extreme	.032
ω_7	extreme	extreme	normal	.032
ω_8	extreme	extreme	extreme	.128

They are dependent.

Now, assume you observed a new evidence: Temp = normal

$$\Pr(\text{Sensor2} = \text{normal} | \text{Temp} = \text{normal}) = .80$$
$$\Pr(\text{Sensor2} = \text{normal} | \text{Temp} = \text{normal}, \text{Sensor1} = \text{normal}) = .80$$

DYNAMIC NOTION!

Now, they have become independent!

This urges us to look into the more general notion of conditional independence.

Conditional Independence

- Given a state of belief Pr, event α is *conditionally independent* from the event β given γ *iff*

$$\Pr(\alpha|\beta \wedge \gamma) = \Pr(\alpha|\gamma) \quad \text{or } \Pr(\beta \wedge \gamma) = 0. \tag{B4}$$

- Realise that this is symmetric (i.e., β is *conditionally independent* from the event α given γ), hence, alternatively:

$$\Pr(\alpha \wedge \beta|\gamma) = \Pr(\alpha|\gamma)\Pr(\beta|\gamma) \quad \text{or } \Pr(\gamma) = 0. \tag{B4`}$$

Bayes' Rule

Recall: $\Pr(\alpha|\beta) = \dfrac{\Pr(\alpha \wedge \beta)}{\Pr(\beta)}$ (*Bayes Conditioning*) (B3)

By repeatedly applying *Bayes conditioning* (B3), we get the following:

Chain rule

$$\Pr(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) = \Pr(\alpha_1|\alpha_2 \wedge \ldots \wedge \alpha_n)\Pr(\alpha_2|\alpha_3 \wedge \ldots \wedge \alpha_n)\ldots\Pr(\alpha_n). \tag{B5}$$

…which will be very important for *Bayesian Networks* later on.

A set of events $\beta_1, \ldots, \beta_n$ is called:

- *logically disjoint* (or *mutually exclusive*) *iff* $Mods(\beta_j) \cap Mods(\beta_k) = \emptyset$ for $j \neq k$ where $j, k \in \{1,\ldots,n\}$.
- *exhaustive iff* $\bigcup_{i=1}^{n} Mods(\beta_i) = \Omega$, where Ω is the set of all worlds.

Marginalisation (also called ***case analysis*** or ***law of total probability***)

Assume a set of events $\beta_1, \ldots, \beta_n$ which are mutually exclusive and exhaustive, then

$$\Pr(\alpha) = \sum_{i=1}^{n} \Pr(\alpha \wedge \beta_i) \tag{B6}$$

$$\Pr(\alpha) = \sum_{i=1}^{n} \Pr(\alpha|\beta_i)\Pr(\beta_i) \tag{B6'}$$

Why does it work?

Examples: Consider the following two simple examples of case analysis:

- $\Pr(\alpha) = \Pr(\alpha \wedge \beta) + \Pr(\alpha \wedge \neg\beta)$
- $\Pr(\alpha) = \Pr(\alpha|\beta)\Pr(\beta) + \Pr(\alpha|\neg\beta)\Pr(\neg\beta)$.

Observe:

These expressions hold because β and $\neg\beta$ are mutually exclusive and exhaustive.

Value:

This is useful when, as in many situations, computing the belief in cases is easier than the whole α.

Bayes' rule or Bayes' theorem

$$\Pr(\alpha|\beta) = \frac{\Pr(\beta|\alpha)\Pr(\alpha)}{\Pr(\beta)}$$

Classical Use:

- When an event α is perceived to be a cause of an event β e.g., α is a disease, β is a symptom.
- The belief in an effect given its cause, $\Pr(\beta\,|\,\alpha)$, is usually more readily available than the belief in a cause given one of its effects, $\Pr(\alpha\,|\,\beta)$.
- Bayes' theorem helps us to compute the latter from the former.

Exercise: False Positive

Problem Setting: Suppose that we have a patient who was just tested for a particular disease and the test came out positive. We know that one in every thousand people has this disease. We also know that the test is not completely reliable: it has a false positive rate of 2% and a false negative rate of 5%.

Question: What should be our belief in the patient having the disease given that the test came out positive?

Hint: Let the propositional variable D stand for "the patient has the disease" and the propositional variable T stand for "the test came out positive," our goal is then to compute $\Pr(D|T)$.

Problem Statement

Suppose that we have a patient who was just tested for a particular disease and the test came out positive. We know that one in every thousand people has this disease. We also know that the test is not reliable: it has a false positive rate of 2% and a false negative rate of 5%. What should be our belief in the patient having the disease given that the test came out positive? In particular, let the propositional variable D stand for "the patient has the disease" and the propositional variable T stand for "the test came out positive," our goal is then to compute $\Pr(D|T)$.

Our prior belief in the patient having the disease before we run any tests: $\Pr(D) = \frac{1}{1{,}000}$

The false positive rate of the test is $\Pr(T|\neg D) = \frac{2}{100}$ hence, $\Pr(\neg T|\neg D) = \frac{98}{100}$

Similarly, the false negative rate is $\Pr(\neg T|D) = \frac{5}{100}$ hence, $\Pr(T|D) = \frac{95}{100}$

Our prior belief in the patient having the disease before we run any tests: $\Pr(D) = \frac{1}{1{,}000}$

The false positive rate of the test is $\Pr(T|\neg D) = \frac{2}{100}$ hence, $\Pr(\neg T|\neg D) = \frac{98}{100}$

Similarly, the false negative rate is $\Pr(\neg T|D) = \frac{5}{100}$ hence, $\Pr(T|D) = \frac{95}{100}$

Using Bayes rule, we get $\Pr(D|T) = \frac{\frac{95}{100} \times \frac{1}{1{,}000}}{\Pr(T)}$

$\Pr(T)$ is not readily available, but can be obtained by case analysis:

$$\begin{aligned}\Pr(T) &= \Pr(T|D)\Pr(D) + \Pr(T|\neg D)\Pr(\neg D) \\ &= \frac{95}{100} \times \frac{1}{1{,}000} + \frac{2}{100} \times \frac{999}{1{,}000} = \frac{2{,}093}{100{,}000}\end{aligned}$$

which yields,

$$\Pr(D|T) = \frac{95}{2{,}093} \approx 4.5\%$$

VU

2nd Solution: Constructing the Belief State

Because we have only two events of interest, T and D, leading to only four worlds, this solution is feasible.

world	*D*	*T*	
ω_1	true	true	has disease, test positive
ω_2	true	false	has disease, test negative
ω_3	false	true	has no disease, test positive
ω_4	false	false	has no disease, test negative

which gives rise to …

$$\begin{aligned}\Pr(\omega_1) &= \Pr(T \wedge D) = \Pr(T|D)\Pr(D) \\ \Pr(\omega_2) &= \Pr(\neg T \wedge D) = \Pr(\neg T|D)\Pr(D) \\ \Pr(\omega_3) &= \Pr(T \wedge \neg D) = \Pr(T|\neg D)\Pr(\neg D) \\ \Pr(\omega_4) &= \Pr(\neg T \wedge \neg D) = \Pr(\neg T|\neg D)\Pr(\neg D).\end{aligned}$$

$$\Pr(\omega_1) = \Pr(T \wedge D) = \Pr(T|D)\Pr(D)$$
$$\Pr(\omega_2) = \Pr(\neg T \wedge D) = \Pr(\neg T|D)\Pr(D)$$
$$\Pr(\omega_3) = \Pr(T \wedge \neg D) = \Pr(T|\neg D)\Pr(\neg D)$$
$$\Pr(\omega_4) = \Pr(\neg T \wedge \neg D) = \Pr(\neg T|\neg D)\Pr(\neg D).$$

..whose values are readily available in the problem setting, which yields:

world	D	T	Pr(.)			
ω_1	true	true	95/100	×	1/1,000	= .00095
ω_2	true	false	5/100	×	1/1,000	= .00005
ω_3	false	true	2/100	×	999/1,000	= .01998
ω_4	false	false	98/100	×	999/1,000	= .97902

$$\frac{Pr(\omega_1)}{Pr(\omega_1) + Pr(\omega_3)} \approx 4.5\,\%$$

VL

Probabilistic Graphical Models 2

Introduction

We have seen that joint probability distribution is exponential in the size of variables.

i.e., Pr($X_1, \ldots, X_n$) needs a state of belief/distribution, specified for 2^n cases/possible worlds.

$n = 3$ variables

world	Earthquake	Burglary	Alarm	Pr(.)
ω_1	true	true	true	.0190
ω_2	true	true	false	.0010
ω_3	true	false	true	.0560
ω_4	true	false	false	.0240
ω_5	false	true	true	.1620
ω_6	false	true	false	.0180
ω_7	false	false	true	.0072
ω_8	false	false	false	.7128

$2^n = 8$ cases/possible worlds

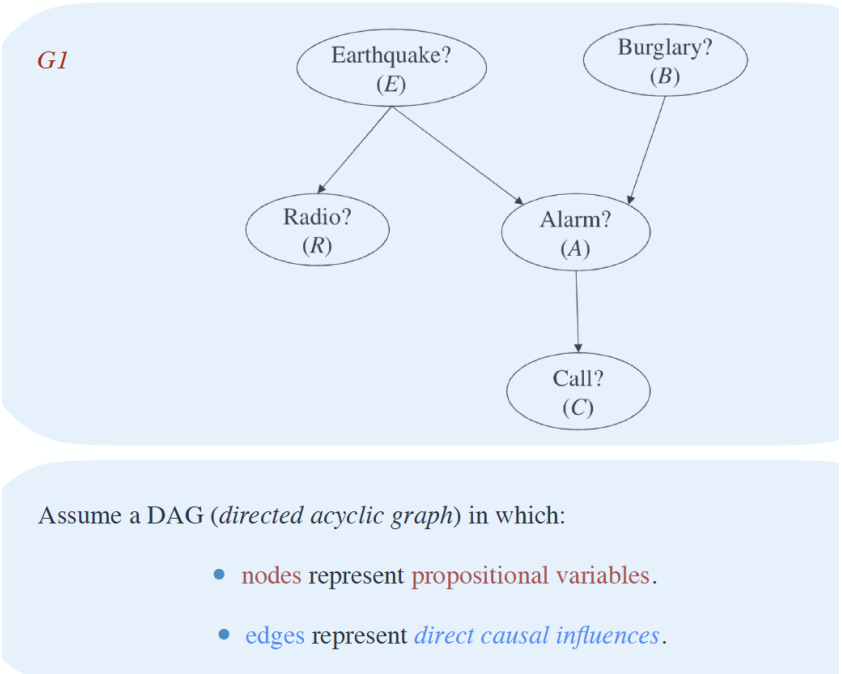
Even if this is addressed, one still needs to ensure that the synthesized distribution is *faithful* i.e., matches the beliefs held about a given situation.

Representing an expert knowledge in some domain, one needs to ensure some correspondence between the independencies held by the distribution and those believed by the expert.

Ex: Pr(Earthquake | Burglary) = Pr(Earthquake) or Pr(Aids | Covid-19) = Pr(Aids).

The *Bayesian network (BN)* is a modelling tool to specify probability distribution which can, in principle, address these challenges. A BN relies on the basic insight that independence forms a significant aspect of beliefs and that it can be elicited relatively easily using the language of graphs.

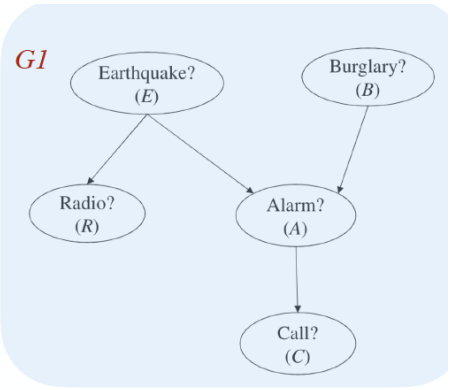
A Graphical Tool for capturing Independence



Assume a DAG (*directed acyclic graph*) in which:

- nodes represent propositional variables.
- edges represent *direct causal influences*.

Assuming that the links are causal, what can be said on the dependencies in *G1*:



- Alarm triggering (*A*) is a direct cause of a call from the neighbour (*C*).
- If we get a radio report (*R*) (that an earthquake (*E*) took place in our neighbourhood), then our belief in A could increase, which in turn increase our belief in *C*.
- Yet, if we know already that $\neg A$, then our belief in *C* stays unchanged.
  i.e., *C* is independent of R given $\neg A$.

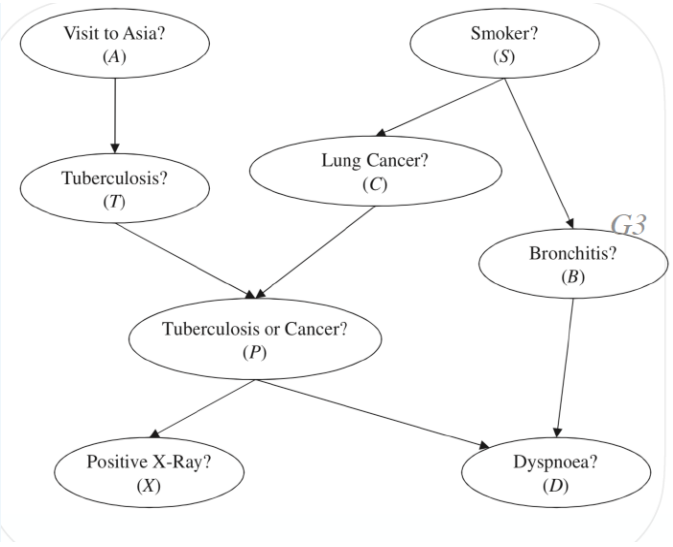
Example:

In a limited medical domain,

we could find a visit to Asia relevant to our belief in the x-ray test coming out positive.

we would find the visit irrelevant if we know for sure that the patient does not have tuberculosis

That is, X is dependent on A but is independent of A given ¬T .



First introducing some notions:
Given a DAG G, and a variable V in G,

- Parents(V) are the parent nodes of V i.e., a set of variables N which has a directed edge to V.

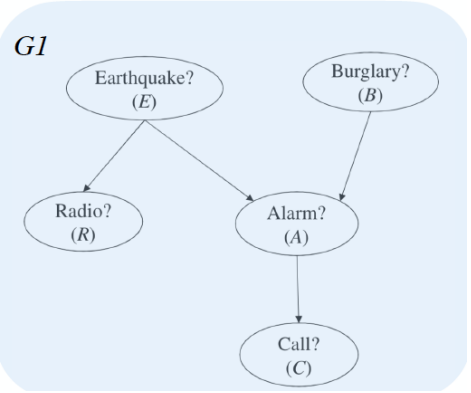
Ex.: Parents(A) = $\{E, B\}$

- Descendants(V) are the descendant nodes of V i.e., a set of variables N s.t. there is a directed path from V.

Ex.: Descendants(B) =$\{A, C\}$

- Non_Descendants(V) are the nodes that are neither in Descendants(V) nor Parents(V) other than V.

Ex.: Non_Descendants(B) =$\{E, R\}$

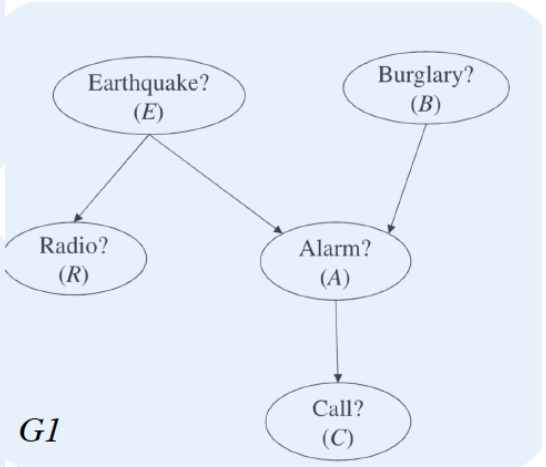


Then, each DAG G corresponds to a compact representation of the following independence statements:

(also called) Markovian assumptions or Markov(G)

$I(V, \text{Parents}(V), \text{Non\_Descendants}(V))$ for all variables V in DAG G.

which says every variable is conditionally independent of its nondescendants given its parents.



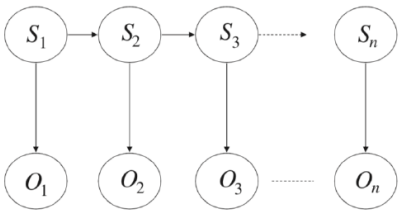
Example

Markov($G1$): $I(C, A, \{B, E, R\})$
$I(R, E, \{A, B, C\})$
$I(A, \{B, E\}, R)$
$I(B, \emptyset, \{E, R\})$
$I(E, \emptyset, B)$

Markov(G):

$I(V, \text{Parents}(V), \text{Non\_Descendants}(V))$ for all variables V in DAG G.

which says every variable is conditionally independent of its nondescendants given its parents.



HMM

Example

This is the famous *Hidden Markov Model* (*HMM*) *

variables $S_1, S_2, \ldots, S_n$ represent the state of a dynamic system and

variables $O_1, O_2, \ldots, O_n$ represent sensors that measure the system state at time points $1, 2, \ldots, n$, respectively.

For every S_i, $I(S_i, \{S_{i-1}\}, \{S_1, \ldots, S_{i-2}, O_1, \ldots, O_{i-1}\}) \in \text{Markov}(HMM)$.

VU VRIJE UNIVERSITEIT AMSTERDAM

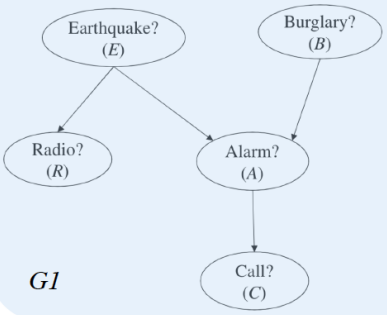
*: Vast amount of applications in *reinforcement learning*, *temporal pattern recognition* and etc.

A GRAPHICAL TOOL FOR CAPTURING INDEPENDENCE

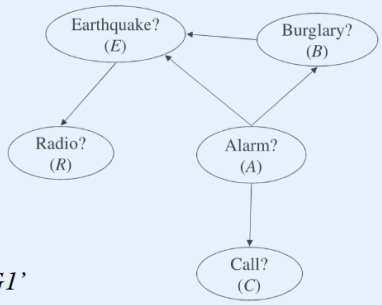
Remark 1:

If one views the DAG as a causal structure, then

- Parents(V) are the *direct causes* of V,
- Descendants(V) are the *effects* of V.



G1



G1'

Remark 2:

Although we used causality to motivate the interpretation, the formal interpretation makes no reference to the notion of *causality*.

Markov($G1$) $\Longrightarrow$ Markov($G1'$)

Bayesian Networks

Parameterising the Independence Structure

Suppose, we would like to construct a probability distribution Pr that captures state of our belief in a given domain.

1st step: Construct a DAG
To construct a DAG G whose independence structure is consistent with our belief state about that domain.

This, indeed, constrains the possible choices, yet does not uniquely defines it.

2nd step: Add conditional probability tables
For every variable X in the DAG, and its parents **U**, For every value x of variable X, and every instantiation **u** of U, define Pr(x | **u**).

The second step corresponds to constructing conditional probability table (CPT). This ensures that Pr is unique.

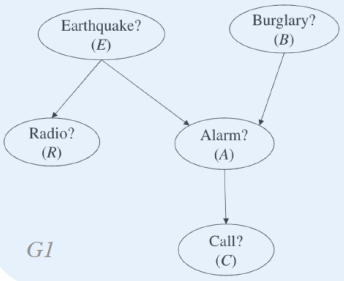
Conditional Probability Tables

Example:

For *G1*, we need to provide the following conditional probabilities:

$$\Pr(c|a),\ \Pr(r|e),\ \Pr(a|b,e),\ \Pr(e),\ \Pr(b)$$

where a, b, c, e, and r are values of variables A, B, C, E, R, respectively.



Example:

A	C	$\Pr(c\|a)$
true	true	.80
true	false	.20
false	true	.001
false	false	.999

Such a table is called CPT (for C).

Observe: $\Pr(c|a) + \Pr(\bar{c}|a) = 1$ and $\Pr(c|\bar{a}) + \Pr(\bar{c}|\bar{a}) = 1$.

Two entries in the CPT are redundant i.e., can be inferred from one another, hence can be omitted.

Bayesian Networks

Finally, we can formally define a Bayesian network [Pearl, 1985]:

Given a set $\mathbf{Z}$ of variables, a Bayesian network (BN) for $\mathbf{Z}$ is a pair (G, Θ) where:

- G is a DAG over $\mathbf{Z}$, called the network *structure*.
- Θ is a set of CPTs, one for each element in $\mathbf{Z}$, called the network *parameterisation*.

Notation

- $\Theta_{X|\mathbf{U}}$ denotes the CPT for variable X and its parents $\mathbf{U}$.
- $\theta_{x|\mathbf{u}}$, the network *parameter*, denotes the value assigned by $\Theta_{X|\mathbf{U}}$ to the $\Pr(x|\mathbf{u})$.

Remark: $\sum_x \theta_{x|\mathbf{u}} = 1$ for every parent instantiation $\mathbf{u}$.

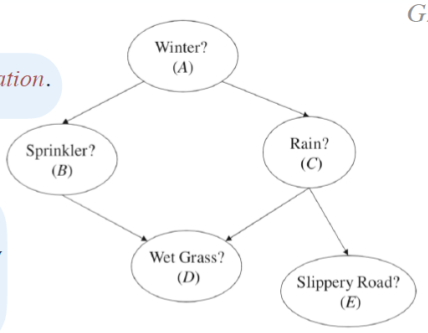
Auxilliary notions:

An instantiation of all network variables will be called a *network instantiation*.

A network parameter $\theta_{x|\mathbf{u}}$ is *compatible* with a network instantiation $\mathbf{z}$, denoted $\theta_{x|\mathbf{u}} \sim z$, iff the instantiations $\mathbf{xu}$ and $\mathbf{z}$ are compatible (i.e., they agree on the values they assign to their common variables).

$\theta_a, \theta_{b|a}, \theta_{\bar{c}|a}, \theta_{d|b,\bar{c}}$ and $\theta_{\bar{e}|\bar{c}}$ are the network parameters compatible with network instantiation $a, b, \bar{c}, d, \bar{e}$ in $G2$.

G2



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|B,C}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Chain Rule for Bayesian networks:

$$\Pr(\mathbf{z}) \stackrel{def}{=} \prod_{\theta_{x|\mathbf{u}} \sim \mathbf{z}} \theta_{x|\mathbf{u}}.$$

Intuitively:

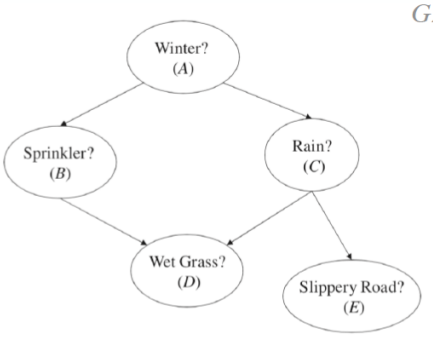
The probability assigned to a network instantiation $\mathbf{z}$ is simply the product of all network parameters compatible with $\mathbf{z}$.

Examples:

$$\begin{aligned}\Pr(a, b, \bar{c}, d, \bar{e}) &= \theta_a \ \theta_{b|a} \ \theta_{\bar{c}|a} \ \theta_{d|b,\bar{c}} \ \theta_{\bar{e}|\bar{c}} \\ &= (.6)(.2)(.2)(.9)(1) \\ &= .0216\end{aligned}$$

$$\begin{aligned}\Pr(\bar{a}, \bar{b}, \bar{c}, \bar{d}, \bar{e}) &= \theta_{\bar{a}} \ \theta_{\bar{b}|\bar{a}} \ \theta_{\bar{c}|\bar{a}} \ \theta_{\bar{d}|\bar{b},\bar{c}} \ \theta_{\bar{e}|\bar{c}} \\ &= (.4)(.25)(.9)(1)(1) \\ &= .09\end{aligned}$$

G2



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|B,C}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

VU VRIJE UNIVERSITEIT AMSTERDAM

Remark! Every Bayesian network implicitly represents a unique Pr.

The distribution Pr specified by a Bayesian network (G, Θ) is guaranteed to satisfy every independence assumption in Markov(G):

$$I_{\Pr}(X, \text{Parents}(X), \text{Non Descendants}(X))$$

for every variable X in the network.

Recall:

$I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ means distribution Pr finds variables $\mathbf{X}$ independent of variables $\mathbf{Y}$ given variables $\mathbf{Z}$:

$$\Pr(\mathbf{x}|\mathbf{z}, \mathbf{y}) = \Pr(\mathbf{x}|\mathbf{z}) \text{ or } \Pr(\mathbf{y}, \mathbf{z}) = 0,$$

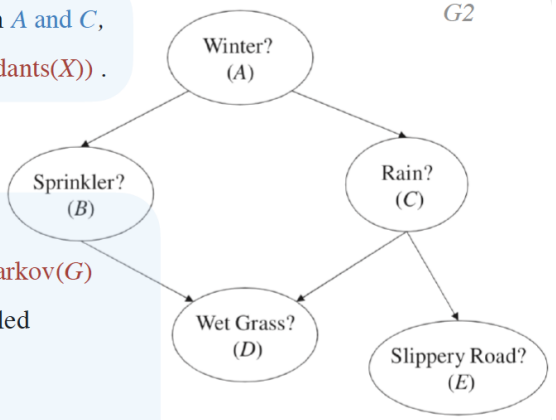
for all instantiations $\mathbf{x}, \mathbf{y}, \mathbf{z}$ of variables $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$, respectively.

Properties of Probabilistic Independence

In principle, a Bayesian network G might imply more independence statement than the ones in Markov(G).

Observe: According to *G2*, *D* and *E* are independent given *A* and *C*,

Yet they are not included in $I_{\Pr}(X, \text{Parents}(X), \text{Non Descendants}(X))$.



This one and the additional ones follow from the ones in Markov(G)

Using a set of properties for probabilistic independence, called

Graphoid axioms:

- Symmetry
- Decomposition
- Weak Union
- Contraction

VU VRIJE UNIVERSITEIT AMSTERDAM

Properties of Probabilistic Independence: Symmetry

The first and the simplest property of probabilistic independence we consider is symmetry:

Symmetry:

$I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ if and only if $I_{\Pr}(\mathbf{Y}, \mathbf{Z}, \mathbf{X})$.

Intuition (Symmetry):

If learning **y** does not influence our belief in **x**, then learning **x** does not influence our belief in **y**, either.

Example:

We know that $I_{\Pr}(A, \{B, E\}, R) \in \text{Markov}(G1)$.

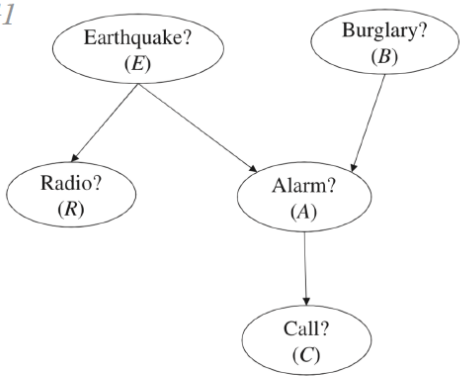
Using symmetry, we can conclude that

$I_{\Pr}(R, \{B, E\}, A)$ which is **not** in Markov($G1$).

Little reminder: "only if" means "$\Longrightarrow$"

"if" means "$\Longleftarrow$"

$G1$



Markov($G1$): $I(C, A, \{B, E, R\})$
$I(R, E, \{A, B, C\})$
$I(A, \{B, E\}, R)$
$I(B, \emptyset, \{E, R\})$
$I(E, \emptyset, B)$

Decomposition

Decomposition:

$$I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W}) \Longrightarrow \text{only if } I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \text{ and } I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{W}).$$

Intuition (Decomposition):

If learning **yw** does not influence our belief in **x**, then learning **y** alone or **w** alone, will not influence our belief in **x**.

Remark

Note that the opposite, say, *composition* is not true in general:

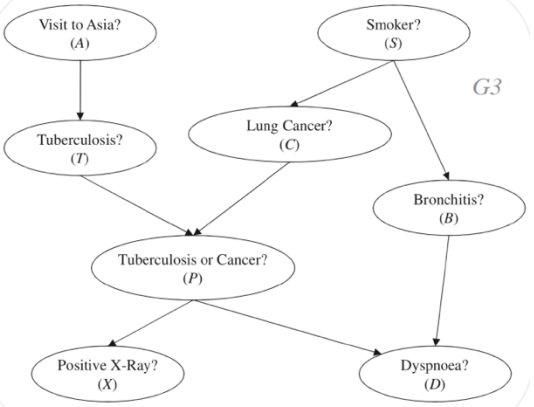
$$I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \text{ and } I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{W}) \text{ only if } I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$$

Example:

We know that $I_{\Pr}(B, S, \{A. C, P, T, X\}) \in \text{Markov}(G3)$.

Using decomposition, we can conclude that

$I_{\Pr}(B, S, C)$: Once we know that the person is a smoker, our belief in developing bronchitis is not influenced by information about developing cancer.



More generally, decomposition, let us to state

$I_{\Pr}(X, \text{Parents}(X), \mathbf{W})$ for every $\mathbf{W} \subseteq \text{Non Descendants}(X)$

which can be seen as a strengthening of Markov(G):

$I_{\Pr}(X, \text{Parents(X)}, \text{Non Descendants}(X))$

which is a special case when **W** contains all nondescendants of X.

Another application:
Decomposition can simplify the initial chain rule.

Example:

Assume, we want to calculate $\Pr(r, c, a, e, b)$.

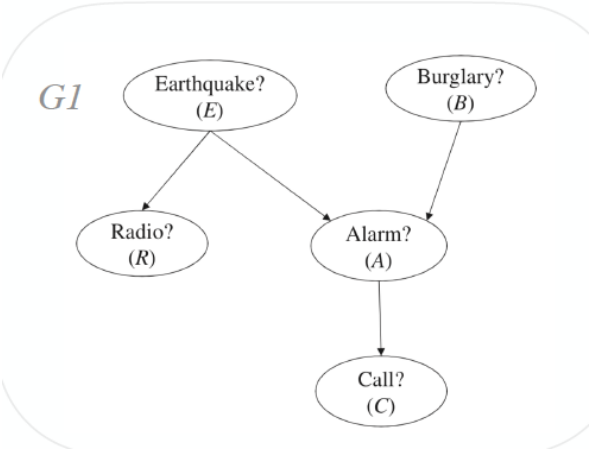
Then, by $\Pr(\alpha_1 \wedge \alpha_2 \wedge \ldots \wedge \alpha_n) = \Pr(\alpha_1 | \alpha_2 \wedge \ldots \wedge \alpha_n)\Pr(\alpha_2 | \alpha_3 \wedge \ldots \wedge \alpha_n) \ldots \Pr(\alpha_n)$. (B5)

$\Pr(r, c, a, e, b) = \Pr(r|c, a, e, b)\Pr(c|a, e, b)\Pr(a|e, b)\Pr(e|b)\Pr(b)$.

By, $I_{\Pr}(X, \text{Parents}(X), \mathbf{W})$ for every $\mathbf{W} \subseteq$ Non Descendants(X), we get:

$$\Pr(r|c, a, e, b) = \Pr(r|e)$$
$$\Pr(c|a, e, b) = \Pr(c|a)$$
$$\Pr(e|b) = \Pr(e).$$

$$\Pr(r, c, a, e, b) = \Pr(r|e)\Pr(c|a)\Pr(a|e, b)\Pr(e)\Pr(b)$$
$$= \theta_{r|e}\, \theta_{c|a}\, \theta_{a|e,b}\, \theta_e\, \theta_b$$



Weak Union

Weak Union:

$I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$ only if $I_{\Pr}(\mathbf{X}, \mathbf{Z} \cup \mathbf{Y}, \mathbf{W})$.

Intuition (Weak Union):

If the information **yw** is not relevant to our belief in **x**, then the partial information **y** will not make the rest of the information **w**, relevant.

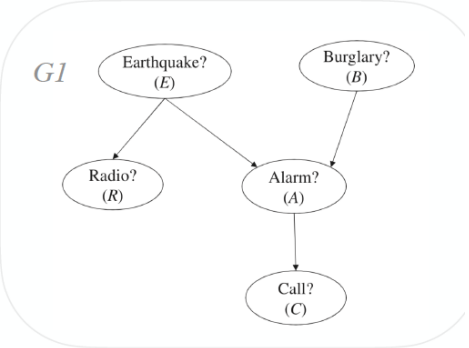
Example:

Assume a Θ on $G1$ and also assume that Pr is generated by $(G1, \Theta)$.

We have $I_{\Pr}(C,A,\{B,E,R\}) \in \text{Markov}(G1)$

By weak-union, we can conclude

$I_{\Pr}(C,\{A,E,B\},R)$ which was initially not in Markov($G1$)



More generally:

$I_{\Pr}(X, \text{Parents}(X) \cup \mathbf{W}, \text{Non Descendants}(X) \setminus \mathbf{W})$, for any $\mathbf{W} \subseteq \text{Non Descendants}(X)$.

Contraction

Contraction:

$I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ and $I_{\Pr}(\mathbf{X}, \mathbf{Z} \cup \mathbf{Y}, \mathbf{W})$ only if $I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$.

Intuition (Contraction):

If after learning the irrelevant information **y** the information **w** is found to be irrelevant to our belief in **x**, then the combined information **yw** must have been irrelevant from the beginning.

Remark

Realize that contraction is a weaker version of composition:

$I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ and $I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{W})$ only if $I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y} \cup \mathbf{W})$

d-Separation

We have seen that deriving new independencies from Markov(G) is a non-trivial task.

Good news is that there is an easy graphical test called d-seperation which captures the inferential power of graphoid axioms.

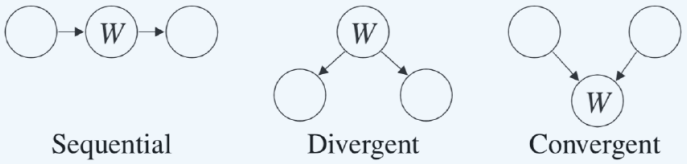
This means $\text{dsep}_G(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ (which reads **X** and **Y** is d-seperated by **Z**) implies $I\Pr(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$ for every probability distribution Pr induced by G.

The intuition is similar to a water flow along a pipe (a path in the network) controlled by a set of valves:
Two variables are independent if all the paths between them are blocked by a closed valve (to be defined).

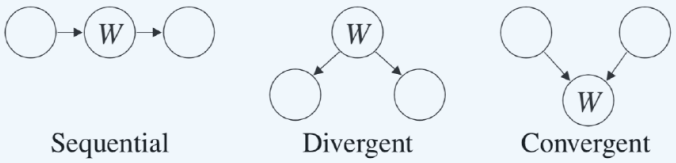
We will need only three types of valves:

- *A sequential valve* ($\rightarrow W \rightarrow$) arises when W is a parent of one of its neighbors and a child of the other.
- *A divergent valve* ($\leftarrow W \rightarrow$) arises when W is a parent of both neighbors.
- *A convergent valve* ($\rightarrow W \leftarrow$) arises when W is a child of both neighbors.

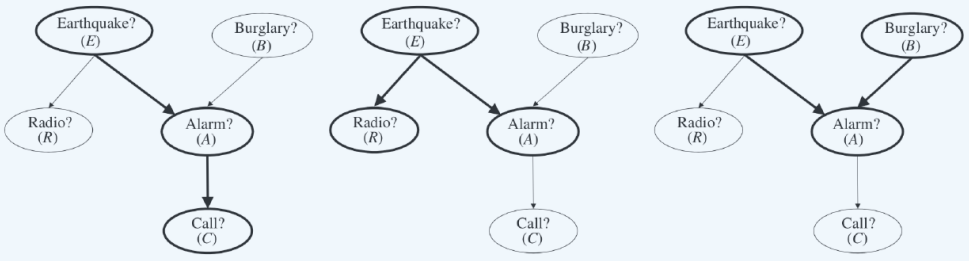
Types of Valves (visually)



Types of Valves

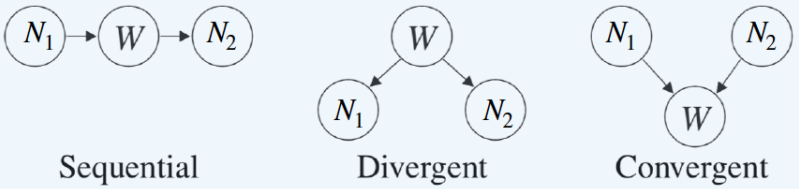


Examples:



Sequential valve | Divergent valve | Convergent valve

Types of Valves:



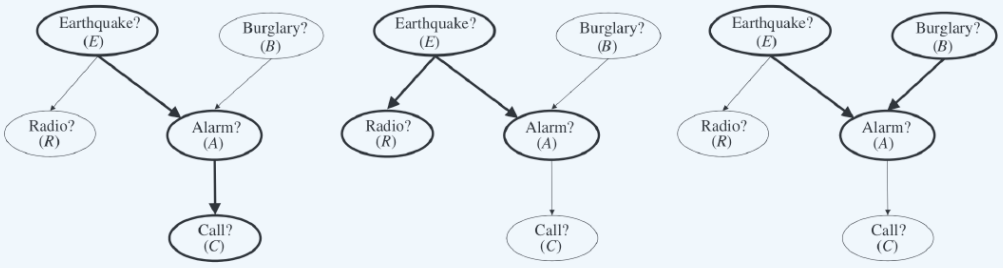
A Causal Interpretation to Help Intuition:

A *sequential valve* $N_1 \rightarrow W \rightarrow N_2$ declares variable W as an intermediary between a cause N_1 and its effect N_2.

A *divergent valve* $N_1 \leftarrow W \rightarrow N_2$ declares variable W as a common cause of two effects N_1 and N_2.

A *convergent valve* $N_1 \rightarrow W \leftarrow N_2$ declares variable W as a common effect of two causes N_1 and N_2.

Examples:



Closedness:

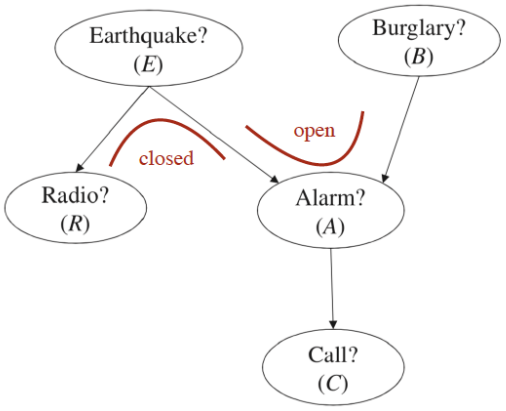
A sequential valve ($\rightarrow W \rightarrow$) is closed iff variable W appears in $\mathbf{Z}$.

A divergent valve ($\leftarrow W \rightarrow$) is closed iff variable W appears in $\mathbf{Z}$.

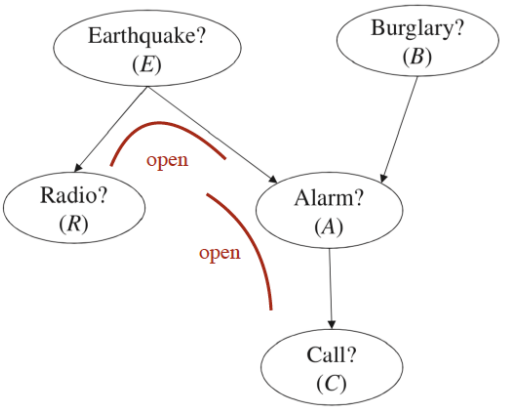
A convergent valve ($\rightarrow W \leftarrow$) is closed iff neither variable W nor any of its descendants appears in $\mathbf{Z}$.

d-Seperation:

Given disjoint sets **X**, **Y**, and **Z** of nodes in a DAG G, **X** and **Y** are *d-separated* by **Z**, denoted $\text{dsep}_G(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$, iff every path between a node in **X** and a node in **Y** is blocked by **Z** (i.e., has at least one closed valve $W \in \mathbf{Z}$).

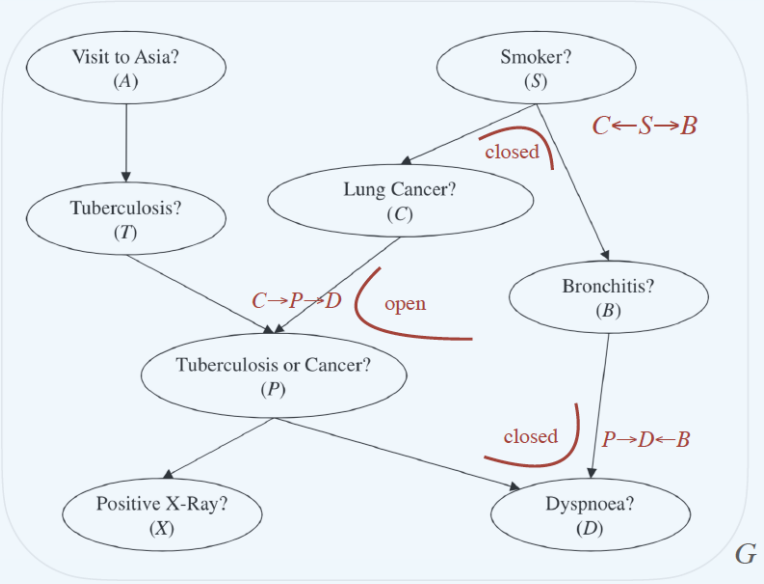


Are R and B d-seperated by E, C? Yes!



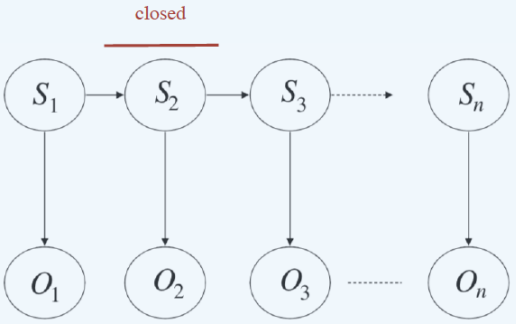
Are R and C d-seperated? No!

More example:



Does $\text{dsep}_G(B, S, C)$ hold? Yes!

More example:



$\text{dsep}_G(S1, S2, \{S3, S4\})$? Yes!

Any path between $S1$ and $\{S3, S4\}$ must have the valve $S1 \rightarrow S2 \rightarrow S3$ on it, which is closed given $S2$.

Pruning Method

Paths between sets of nodes can be exponentially many. The following method guarantees it can be decided in linear time/space in the size of the graph.

Pruning method:

Given a DAG G and disjoint sets of nodes $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ in G, a pruned G' w.r.t. $\mathbf{Z}$ is obtained by:

Deleting every leaf node $W \notin \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$,

Deleting all edges outgoing from nodes in $\mathbf{Z}$,

iteratively until both rules do not apply anymore.

Theorem:

$\mathbf{X}$ and $\mathbf{Y}$ are d-separated by $\mathbf{Z}$ in G iff $\mathbf{X}$ and $\mathbf{Y}$ are disconnected in the pruned G' w.r.t. $\mathbf{Z}$.

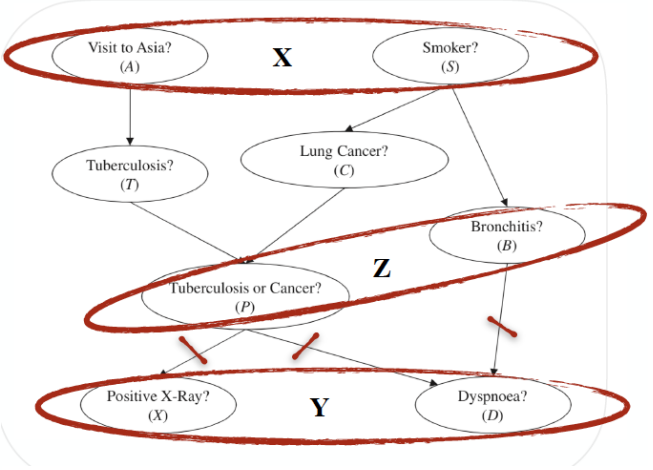
Recall the Pruning rules (iteratively):

Delete every leaf node $W \notin \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$.

Delete all edges outgoing from nodes in $\mathbf{Z}$.

Example:

Is $\mathbf{X} = \{A, S\}$ d-separated from $\mathbf{Y} = \{D, X\}$ by $\mathbf{Z} = \{B, P\}$?

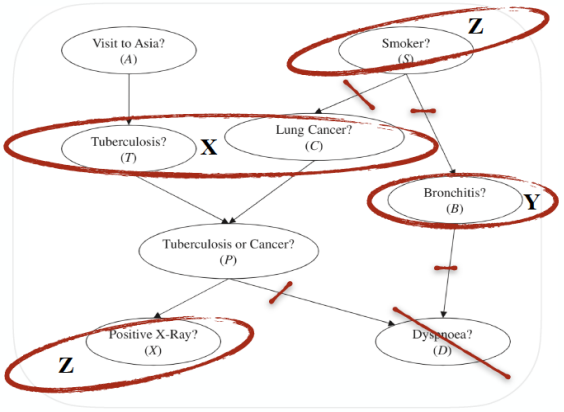


Yes!

Recall the Pruning rules:

Delete every leaf node $W \notin \mathbf{X} \cup \mathbf{Y} \cup \mathbf{Z}$.

Delete all edges outgoing from nodes in $\mathbf{Z}$.



Example:

Is $\mathbf{X} = \{T, C\}$ d-separated from $\mathbf{Y} = \{B\}$ by $\mathbf{Z} = \{S, X\}$?

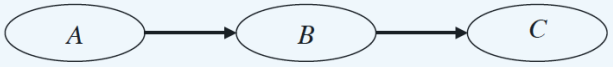
Yes!

Theorem

Let Pr be a probability distribution induced by a BN (G, Θ), then $\text{dsep}_G(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \Longrightarrow I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$.

Theorem $\text{dsep}_G(\mathbf{X}, \mathbf{Z}, \mathbf{Y}) \not\Longleftarrow I_{\Pr}(\mathbf{X}, \mathbf{Z}, \mathbf{Y})$.

Proof by contradiction:



Assume a CPT for variable B is with $\theta b|a = \theta b|\bar{a}$. Then, although there is an unblocked edge between A and B, B is independent from A, hence C is independent from A.

However, C is not d-separated from A.

Variable Elimination:
A tool for Inference in Bayesian Networks

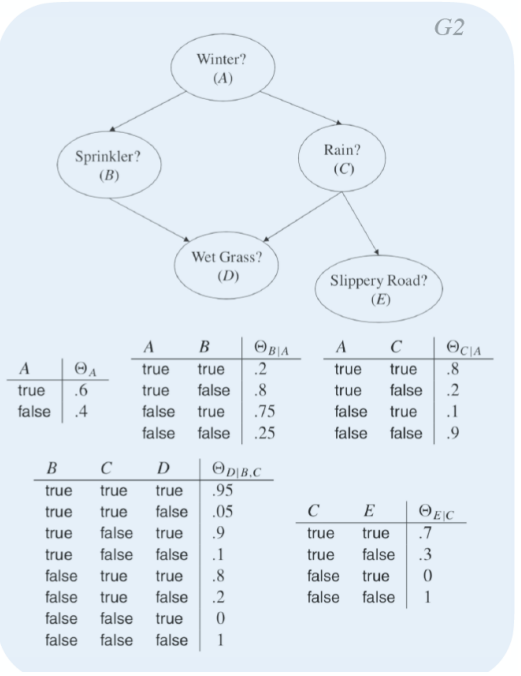
The Idea of Variable Elimination

Consider the Bayesian network *G2* from the previous lecture:

And suppose we are interested in computing the marginal Pr(*D*, *E*), that will look like:

D	*E*	Pr(*D*, *E*)
true	true	.30443
true	false	.39507
false	true	.05957
false	false	.24093

How can we do that?



A	Θ_A
true	.6
false	.4

A	*B*	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	*C*	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	*C*	*D*	$\Theta_{D\|B,C}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	*E*	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Suppose we are interested in computing the marginal Pr(*D*,*E*)

How can we do that? One way to do that:

First, compute the joint probability distribution (recall the chain rule):

And then summing out each variable (i.e., *A*, *B*, *C*) one by one:

For instance, for *A* the following rows

A	*B*	*C*	*D*	*E*	Pr(.)
true	true	true	true	true	.06384
false	true	true	true	true	.01995

Would merge into:

B	*C*	*D*	*E*	Pr(.)
true	true	true	true	.08379 = .06384 + .01995

A	*B*	*C*	*D*	*E*	Pr(.)
true	true	true	true	true	.06384
true	true	true	true	false	.02736
true	true	true	false	true	.00336
true	true	true	false	false	.00144
true	true	false	true	true	0
true	true	false	true	false	.02160
true	true	false	false	true	0
true	true	false	false	false	.00240
true	false	true	true	true	.21504
true	false	true	true	false	.09216
true	false	true	false	true	.05376
true	false	true	false	false	.02304
true	false	false	true	true	0
true	false	false	true	false	0
true	false	false	false	true	0
true	false	false	false	false	.09600
false	true	true	true	true	.01995
false	true	true	true	false	.00855
false	true	true	false	true	.00105
false	true	true	false	false	.00045
false	true	false	true	true	0
false	true	false	true	false	.24300
false	true	false	false	true	0
false	true	false	false	false	.02700
false	false	true	true	true	.00560
false	false	true	true	false	.00240
false	false	true	false	true	.00140
false	false	true	false	false	.00060
false	false	false	true	true	0
false	false	false	true	false	0
false	false	false	false	true	0
false	false	false	false	false	.0900

It will work: Resulting new distribution Pr' we have is as good as Pr i.e., $Pr'(\alpha) = Pr(\alpha)$ for any α that does not include A.

But it will necessarily require an exponential effort in the number of variables in the BN.

The value of variable elimination method is that it can *sometimes* avoid such complexity (i.e., having to construct the joint distribution explicitly by brute-force.)

An auxiliary notion: Factor

Given a set of variables $\mathbf{X}$, a *factor* f is a (total) function* mapping each instantiation to a non-negative number i.e., $f = \mathbf{X} \rightarrow \mathbb{R}_+$.

Hence, a factor does not necessarily represent a probability distribuiton.

Some notation:

Denote the variables that f is defined for, by vars(f).

I.e., vars(f) = $X_1, \ldots, X_n$ where $f(X_1, \ldots, X_n) \in \mathbb{R}_+$.

We will also allow f to be defined over an empty set of variables.

Such factors called *trivial* as they assign a single number to the trivial instantiation $\top$.

Example:

B	C	D	f_1
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

$f_1(b, c, d) = \Pr(d|b, c)$

D	E	f_2
true	true	.448
true	false	.192
false	true	.112
false	false	.248

$f_2(d, e) = \Pr(d, e)$

We will have two key operations on factors:
summing-out and multiplication.

We will have two key operations on factors:

summing-out and *multiplication*.

Summing-out (*or Marginalising*)

Given a factor f over variables $\mathbf{X}$, summing-out variable $X \in \mathbf{X}$, from factor f results in another factor $\sum_X f$ over variables $\mathbf{X} \setminus \{X\}$: and defined as

$$\left(\sum_X f\right)(\mathbf{y}) \stackrel{def}{=} \sum_x f(x, \mathbf{y})$$

Example:

B	C	D	f_1
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

$f_1(b, c, d) = \Pr(d|b, c)$

Example (cont'd):

The result of summing-out D from f_1.

B	C	$\sum_D f_1$
true	true	1
true	false	1
false	true	1
false	false	1

Example (cont'd):

If we sum-out all variables, we end up with the trivial factor:

	$\sum_B \sum_C \sum_D f_1$
$\top$	4

Remark: Summing-out is commutative.

$$\sum_Y \sum_X f = \sum_X \sum_Y f$$

Summing-out

Given a factor f over variables $\mathbf{X}$, summing-out variable $X \in \mathbf{X}$, from factor f results in another factor $\sum_X f$ over variables $\mathbf{X} \setminus \{X\}$: and defined as

$$\left(\sum_X f\right)(\mathbf{y}) \stackrel{def}{=} \sum_x f(x, \mathbf{y})$$

Remark: Summing-out is commutative.

$$\sum_Y \sum_X f = \sum_X \sum_Y f$$

Complexity: $O(\exp(|\mathbf{X}|))$ of time & space.

Notation: Summing-out multiple variables:

$$\sum_{\mathbf{X}} f$$

where $\mathbf{X}$ is a set of variables.

If $\mathbf{Y}$ are other variables of f, then $\sum_{\mathbf{X}} f$

is also called projection of f on $\mathbf{Y}$.

Algorithm 1 `SumOutVars`$(f(\mathbf{X}), \mathbf{Z})$

input:
- $f(\mathbf{X})$: factor over variables $\mathbf{X}$
- $\mathbf{Z}$: a subset of variables $\mathbf{X}$

output: a factor corresponding to $\sum_{\mathbf{Z}} f$

main:

```
Y ← X − Z
f' ← a factor over variables Y where f'(y) = 0 for all y
for each instantiation y do
    for each instantiation z do
        f'(y) ← f'(y) + f(yz)
    end for
end for
return f'
```

Multiplication of Factors

Given factors f_1 and f_2 over variables $\mathbf{X}$ and $\mathbf{Y}$, respectively. Multiplying $f_1 f_2$ is another factor over variables $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$ s.t.

$$(f_1 f_2)(\mathbf{z}) \stackrel{def}{=} f_1(\mathbf{x}) f_2(\mathbf{y}),$$

where $\mathbf{x}$ and $\mathbf{y}$ are compatible with $\mathbf{z}$; that is, $\mathbf{x} \sim \mathbf{z}$ and $\mathbf{y} \sim \mathbf{z}$.

Example:

B	C	D	f_1
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

$f_1(b, c, d) = \Pr(d|b, c)$

D	E	f_2
true	true	.448
true	false	.192
false	true	.112
false	false	.248

$f_2(d, e) = \Pr(d, e)$

Example (cont'd):

B	C	D	E	$f_1(B, C, D) f_2(D, E)$
true	true	true	true	$.4256 = (.95)(.448)$
true	true	true	false	$.1824 = (.95)(.192)$
true	true	false	true	$.0056 = (.05)(.112)$
⋮	⋮	⋮	⋮	⋮
false	false	false	false	$.2480 = (1)(.248)$

Remark: Factor multiplication is commutative and associative.

VU VRIJE UNIVERSITEIT AMSTERDAM

Multiplication of Factors

Given factors f_1 and f_2 over variables $\mathbf{X}$ and $\mathbf{Y}$, respectively. Multiplying $f_1 f_2$ is another factor over variables $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$ s.t.

$$(f_1 f_2)(\mathbf{z}) \stackrel{def}{=} f_1(\mathbf{x}) f_2(\mathbf{y}),$$

where $\mathbf{x}$ and $\mathbf{y}$ are compatible with $\mathbf{z}$; that is, $\mathbf{x} \sim \mathbf{z}$ and $\mathbf{y} \sim \mathbf{z}$.

Remark: Factor multiplication is commutative and associative.

Algorithm 2 `MultiplyFactors`$(f_1(\mathbf{X}_1), \ldots, f_m(\mathbf{X}_m))$

input:
$f_1(\mathbf{X}_1), \ldots, f_m(\mathbf{X}_m)$: factors

output: a factor corresponding to the product $\prod_{i=1}^{m} f_i$

main:

```
1: Z ← ⋃_{i=1}^{m} X_i
2: f ← a factor over variables Z where f(z) = 1 for all z
3: for each instantiation z do
4:    for i = 1 to m do
5:       x_i ← instantiation of variables X_i consistent with z
6:       f(z) ← f(z) f_i(x_i)
7:    end for
8: end for
9: return f
```

Complexity: $O(m \exp(|\mathbf{Z}|))$ of time & space.

The Process of Elimination

Consider again the Bayesian network *G2*:

And suppose we are interested in computing the joint probability distribution.

We can do this in two ways:

- Either use chain rule for BN for each (compatible) instantiation e.g., a, b, c, d, e

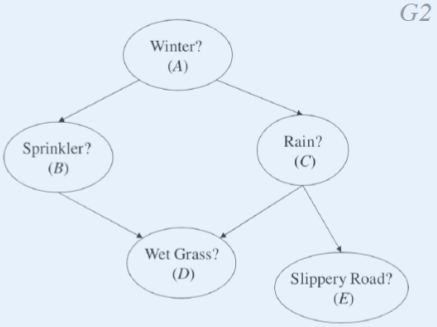
$$\Pr(a, b, c, d, e) = \theta_{e|c}\, \theta_{d|bc}\, \theta_{c|a}\, \theta_{b|a}\, \theta_a.$$

- Or interpreting each CPT as a factor and multiplying them getting the following factor:

$$\Theta_{E|C}\, \Theta_{D|BC}\, \Theta_{C|A}\, \Theta_{B|A}\, \Theta_A$$

which is indeed the joint probability distribution induced by *G2*.

G2



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|B,C}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Remark
This shows one of the key applications of factor multiplication: Express the joint distribution of any BN as a product of its CPTs.

VU VRIJE UNIVERSITEIT AMSTERDAM

Now, suppose we are interested in computing the marginal distribution on variables D and E.

As a combination of summing-out and multiplication, and we get:

$$\Pr(D, E) = \sum_{A,B,C} \Theta_{E|C}\, \Theta_{D|BC}\, \Theta_{C|A}\, \Theta_{B|A}\, \Theta_A.$$

In fact, this can be done for any set of marginals.

Exponential complexity (in the size of the variables due to multpilying CPTs) is still on the table.

The following result shows that such multiplication is not necessary in general.

Theorem 3.1 *If f_1 and f_2 are factors and if variable X appears only in f_2, then*

$$\sum_X f_1 f_2 = f_1 \sum_X f_2.$$

Why is this cool?

It implies that, if $f_1, \ldots, f_n$ are CPTs of a BN, to sum-out X, it may not be necessary to multiply these factors, first.

That is, if X only appears in f_n, we don't need to multiply any of these factors before one sums-out X since

$$\sum_X f_1 \ldots f_n = f_1 \ldots f_{n-1} \sum_X f_n.$$

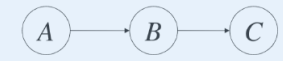
Or, if X only appears in f_{n-1} and f_n, then one only needs to multiply these two factors before summing-out X since

$$\sum_X f_1 \ldots f_n = f_1 \ldots f_{n-2} \sum_X f_{n-1} f_n.$$

In general, to sum-out X from the product $f_1, \ldots, f_n$, we need to just multiply the factors f_k that include X before summing out X from the resulting factor $\Pi_k f_k$.

Remember the idea: To sum-out X from the product $f_1, \ldots, f_n$, we need to just multiply the factors f_k that include X before summing out X from the resulting factor $\Pi_k f_k$.

Example: Consider the following BN *G4*



G4

A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.9
true	false	.1
false	true	.2
false	false	.8

B	C	$\Theta_{C\|B}$
true	true	.3
true	false	.7
false	true	.5
false	false	.5

Let's say we want to compute Pr(C) (*the prior marginal on C*), by first eliminating A and then B.

Note that there are two factors that mention variable A: Θ_A and $\Theta_{B|A}$. We must multiply them first, then sum out variable A.

Eliminating A

Multiplying Θ_A and $\Theta_{B|A}$, we get:

A	B	$\Theta_A \Theta_{B\|A}$
true	true	.54
true	false	.06
false	true	.08
false	false	.32

Summing out variable A, we get:

B	$\sum_A \Theta_A \Theta_{B\|A}$
true	$.62 = .54 + .08$
false	$.38 = .06 + .32$

Eliminating B

Since both factors have B, we multiply them and get:

B	C	$\Theta_{C\|B} \sum_A \Theta_A \Theta_{B\|A}$
true	true	.186
true	false	.434
false	true	.190
false	false	.190

Summing out:

C	$\sum_B \Theta_{C\|B} \sum_A \Theta_A \Theta_{B\|A}$
true	.376
false	.624

$\Pr(C)$

We now have factors the $\sum_A \Theta_A \Theta_{B|A}$ and $\Theta_{C|B}$, and we want to eliminate B.

VU VRIJE UNIVERSITEIT AMSTERDAM

Computing Prior Marginals

Observe: All the crucial work is on Line 3 and 4.

Recall that in the previous examples, we have eliminated first A and then B, yielded single variable factors:

$$\underbrace{\sum_{B} \Theta_{C|B} \overbrace{\sum_{A} \Theta_{A} \Theta_{B|A}}^{1}}_{1}.$$

Algorithm 3 VE_PR1($\mathcal{N}$, $\mathbf{Q}$, π)

input:
- $\mathcal{N}$: Bayesian network
- $\mathbf{Q}$: variables in network $\mathcal{N}$
- π: ordering of network variables not in $\mathbf{Q}$

output: the prior marginal Pr($\mathbf{Q}$)

main:

store CPTs
1: $\mathcal{S} \leftarrow$ CPTs of network $\mathcal{N}$
2: **for** $i = 1$ to length of order π **do**
eliminate variable $\pi(i)$
3: $f \leftarrow \prod_k f_k$, where f_k belongs to $\mathcal{S}$ and mentions variable $\pi(i)$
4: $f_i \leftarrow \sum_{\pi(i)} f$
5: replace all factors f_k in $\mathcal{S}$ by factor f_i
6: **end for**
Pr($\mathbf{Q}$)
7: **return** $\prod_{f \in \mathcal{S}} f$

Assume instead we eliminated B first and A afterwards. This yields:

$$\underbrace{\sum_{A} \Theta_{A} \overbrace{\sum_{B} \Theta_{B|A} \Theta_{C|B}}^{2}}_{1}$$

Now, inner-part having two variables, costing significantly more computation (both in terms of space and time).

Bottom-line: Order matters!

Choosing An Elimination Order

Computing Prior Marginals

Remember: All the crucial work is on Line 3 and 4.

Theorem 3.2: Given a BN, if the largest built factor on Line 4 has w variables, then the complexity of Lines 3-5 is $O(n \exp(w))$ where n is the number of variables in the BN.

Quality of order

The number w is known as the *width* of used order π and is taken as a measure of the order quality.

Idea: Choose an order that has the smallest possible width.

Remark: Note that total time complexity of the algorithm:

$O(n \exp(w) + n \exp(|\mathbf{Q}|))$.

Algorithm 3 VE_PR1($\mathcal{N}$, $\mathbf{Q}$, π)

input:
- $\mathcal{N}$: Bayesian network
- $\mathbf{Q}$: variables in network $\mathcal{N}$
- π: ordering of network variables not in $\mathbf{Q}$

output: the prior marginal Pr($\mathbf{Q}$)

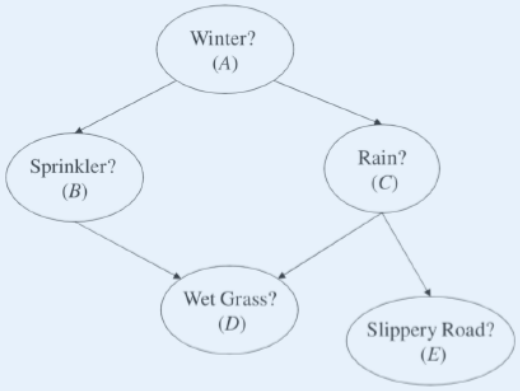
main:

1: $\mathcal{S} \leftarrow$ CPTs of network $\mathcal{N}$
2: **for** $i = 1$ to length of order π **do**
3: $f \leftarrow \prod_k f_k$, where f_k belongs to $\mathcal{S}$ and mentions variable $\pi(i)$
4: $f_i \leftarrow \sum_{\pi(i)} f$
5: replace all factors f_k in $\mathcal{S}$ by factor f_i
6: **end for**
7: **return** $\prod_{f \in \mathcal{S}} f$

Choosing an Order of Elimination

Observe: A straightforward idea is to track the degrees of each order.

i	$\pi(i)$	$\mathcal{S}$	f_i	w
		$\Theta_A \ \Theta_{B\|A} \ \Theta_{C\|A} \ \Theta_{D\|BC} \ \Theta_{E\|C}$		
1	B	$\Theta_A \ \Theta_{C\|A} \ \Theta_{E\|C} \ f_1(A, C, D)$	$f_1 = \sum_B \Theta_{B\|A} \ \Theta_{D\|BC}$	3
2	C	$\Theta_A \ f_2(A, D, E)$	$f_2 = \sum_C \Theta_{C\|A} \ \Theta_{E\|C} \ f_1(A, C, D)$	3
3	A	$f_3(D, E)$	$f_3 = \sum_A \Theta_A \ f_2(A, D, E)$	2
4	D	$f_4(E)$	$f_4 = \sum_D f_3(D, E)$	1



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	C	D	$\Theta_{D\|B,C}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	E	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Interaction Graphs

It is simpler to visualize such width with interaction graphs which expresses the interactions between CPTs of a BN.

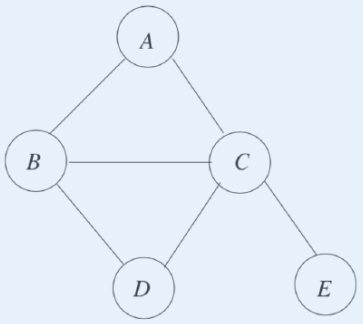
The interaction graph G of the given factors , is an undirected graph constructed as follows: $f1,\ldots, fn$

• The nodes of G are the variables that appear in factors . $f1,\ldots, fn$

• Edges connect the variables which appear in the same factor.

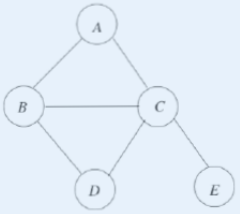
Example:

$\mathcal{S}_1 : \Theta_A \quad \Theta_{B|A} \quad \Theta_{C|A} \quad \Theta_{D|BC} \quad \Theta_{E|C}$

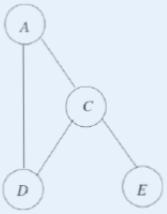


Example:

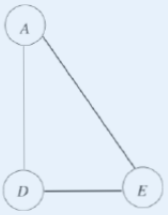
$\mathcal{S}_1 : \Theta_A \quad \Theta_{B|A} \quad \Theta_{C|A} \quad \Theta_{D|BC} \quad \Theta_{E|C}$



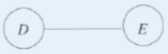
$\mathcal{S}_2 : \Theta_A \quad \Theta_{C|A} \quad \Theta_{E|C} \quad f_1(A, C, D)$



$\mathcal{S}_3 : \Theta_A \quad f_2(A, D, E)$



$\mathcal{S}_4 : f_3(D, E)$



$\mathcal{S}_5 : f_4(E)$

Observations:

- Eliminating a variable $\pi(i)$ from S leads to constructing a factor over the neighbours of $\pi(i)$.

- For S $\xrightarrow{\text{eliminating } \pi(i)}$ S' which corresponds to G $\xrightarrow{\text{eliminating } \pi(i)}$ G'

G' can be obtained by G as follows:

1. Add an edge to G between every pair of neighbours of variable $\pi(i)$ that are not already connected by an edge.
2. Delete variable $\pi(i)$ from G.

Multiplying all factors containing $\pi(i)$ in S.

Summing out variable $\pi(i)$ from the resulting factor.

Also observe that each set of variables per factor forms a *clique*.

Algorithm 4 `OrderWidth`$(\mathcal{N}, \pi)$

input:
$\mathcal{N}$: Bayesian network
π: ordering of the variables in network $\mathcal{N}$

output: the width of elimination order π

main:

```
G ← interaction graph of the CPTs in network N
w ← 0
for i = 1 to length of elimination order π do
    w ← max(w, d), where d is the number of π(i)'s neighbors in G
    add an edge between every pair of non-adjacent neighbors of π(i) in G
    delete variable π(i) from G
end for
return w
```

A simple heuristic: Always choose the node with the smallest degree.

Algorithm 5 `MinDegreeOrder`$(\mathcal{N}, \mathbf{X})$

input:
$\mathcal{N}$: Bayesian network
$\mathbf{X}$: variables in network $\mathcal{N}$

output: an ordering π of variables $\mathbf{X}$

main:

```
G ← interaction graph of the CPTs in network N
for i = 1 to number of variables in X do
    π(i) ← a variable in X with smallest number of neighbors in G
    add an edge between every pair of non-adjacent neighbors of π(i) in G
    delete variable π(i) from G and from X
end for
return π
```

Another heuristic: Always choose the node whose elimination adds smallest number of edges.

Algorithm 6 `MinFillOrder`($\mathcal{N}$, $\mathbf{X}$)

input:
- $\mathcal{N}$: Bayesian network
- $\mathbf{X}$: variables in network $\mathcal{N}$

output: an ordering π of variables $\mathbf{X}$

main:

1: $G \leftarrow$ interaction graph of the CPTs in network $\mathcal{N}$
2: **for** $i = 1$ to number of variables in $\mathbf{X}$ **do**
3: $\quad \pi(i) \leftarrow$ a variable in $\mathbf{X}$ that adds the smallest number of edges on Line 4
4: $\quad$ add an edge between every pair of non-adjacent neighbors of $\pi(i)$
5: $\quad$ delete variable $\pi(i)$ from G and from $\mathbf{X}$
6: **end for**
7: **return** π

Computing Posterior Marginals

Now assume that given a BN , N, a set of variables Q, and an instantiation e, we want to compute the posterior marginal Pr(Q|e) for variables Q.

Example:

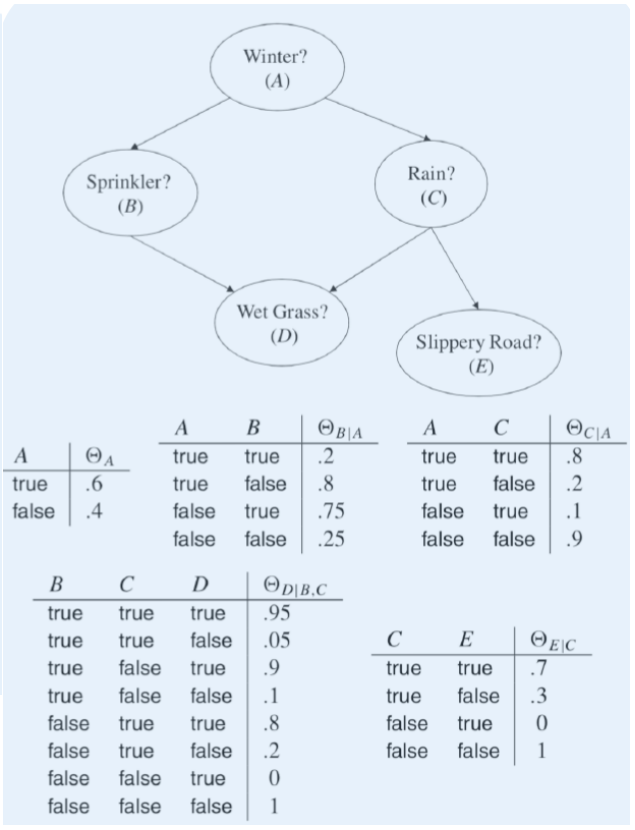
Now considering *G2*, assume that we have **Q** = {*D*, *E*} and evidence for **e**: A= true, *B*= *false*.

We would like to compute the following factor:

D	*E*	Pr(**Q**|**e**)
true	true	.448
true	false	.192
false	true	.112
false	false	.248

For instance, 3rd row corresponds to:

$$\Pr(D = \text{false}, E = \text{true} | A = \text{true}, B = \text{false}) = .112$$



A	Θ_A
true	.6
false	.4

A	*B*	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

A	*C*	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	*C*	*D*	$\Theta_{D\|B,C}$
true	true	true	.95
true	true	false	.05
true	false	true	.9
true	false	false	.1
false	true	true	.8
false	true	false	.2
false	false	true	0
false	false	false	1

C	*E*	$\Theta_{E\|C}$
true	true	.7
true	false	.3
false	true	0
false	false	1

Remark: Note that prior marginal is a special case of posterior marginal where e is empty.

Example (cont'd):

It might be more straightforward to compute a variation on posterior marginals called *joint marginals*, Pr(**Q**,**e**).

That is, instead of computing, say, Pr(**q**|**e**), we compute Pr(**q**, **e**)

D	E	Pr(**Q**, **e**)
true	true	.21504
true	false	.09216
false	true	.05376
false	false	.11904

For instance, 3rd row corresponds to:

$$\Pr(D=\text{false}, E=\text{true}, A=\text{true}, B=\text{false}) = .05376$$

Note that if we keep on summing up, we end up with the probability of evidence **e**: A=true, B=false (since case analysis).

This means, we can compute the posterior marginal Pr(**Q**|**e**) simply by normalising Pr(**Q**, **e**) by **e** which is obtained for free.

Goal: Computing Pr(**Q**, **e**) and normalising by **e**.

In doing so, we can start by zeroing out the rows that are inconsistent with **e**.

Factor given evidence: Given a factor f over **X**, and an evidence **e**, reduced factor $f^{\mathbf{e}}$ over **X** is defined as follows:

$$f^{\mathbf{e}}(\mathbf{x}) \stackrel{def}{=} \begin{cases} f(\mathbf{x}), & \text{if } \mathbf{x} \sim \mathbf{e} \\ 0, & \text{otherwise.} \end{cases}$$

Example:

D	E	f
true	true	.448
true	false	.192
false	true	.112
false	false	.248

→

D	E	$f^{\mathbf{e}}$
true	true	.448
true	false	0
false	true	.112
false	false	0

We can omit zeroed rows:

D	E	$f^{\mathbf{e}}$
true	true	.448
false	true	.112

Example (con'td):

Q = {D, E} and **e** : A = true, B = false.

$$\Pr(\mathbf{Q}, \mathbf{e}) = \sum_{A,B,C} \left(\Theta_{E|C}\ \Theta_{D|BC}\ \Theta_{C|A}\ \Theta_{B|A}\ \Theta_{A}\right)^{\mathbf{e}}.$$

Realise that we have to multiply all CPTs before we start eliminating variables, which is undesirable.

The following result helps.

Theorem 3.3: If f_1 and f_2 are two factors and **e** is an instantiation, then $(f_1 f_2)^{\mathbf{e}} = f_1^{\mathbf{e}} f_2^{\mathbf{e}}$.

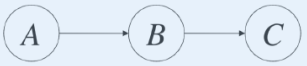
Hence, by the theorem it reduces to:

$$\Pr(\mathbf{Q} = \{D, E\}, \mathbf{e}) = \sum_{A,B,C} \Theta^{\mathbf{e}}_{E|C}\ \Theta^{\mathbf{e}}_{D|BC}\ \Theta^{\mathbf{e}}_{C|A}\ \Theta^{\mathbf{e}}_{B|A}\ \Theta^{\mathbf{e}}_{A}$$

hence, *variable elimination algorithm* can be used.

Example: Consider G4 once again.

Suppose Q = {C}, e : A=true,
Goal: Pr(Q, e) by eliminating A first, B second.



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.9
true	false	.1
false	true	.2
false	false	.8

B	C	$\Theta_{C\|B}$
true	true	.3
true	false	.7
false	true	.5
false	false	.5

Reducing the CPTs given evidence **e**:

A	$\Theta^{\mathbf{e}}_A$
true	.6

A	B	$\Theta^{\mathbf{e}}_{B\|A}$
true	true	.9
true	false	.1

B	C	$\Theta^{\mathbf{e}}_{C\|B}$
true	true	.3
true	false	.7
false	true	.5
false	false	.5

The formula we need to evaluate then becomes

$$\Pr(\mathbf{Q}, \mathbf{e}) = \sum_B \sum_A \Theta^{\mathbf{e}}_A \ \Theta^{\mathbf{e}}_{B|A} \ \Theta^{\mathbf{e}}_{C|B}$$
$$= \sum_B \Theta^{\mathbf{e}}_{C|B} \ \sum_A \Theta^{\mathbf{e}}_A \ \Theta^{\mathbf{e}}_{B|A}.$$

Example (cont'd): Consider G4 once again.
Suppose Q = {C}, e : A=true,
Goal: Pr(Q | e) by eliminating A first, B second.

$$\Pr(\mathbf{Q}, \mathbf{e}) = \sum_B \Theta^{\mathbf{e}}_{C|B} \ \sum_A \Theta^{\mathbf{e}}_A \ \Theta^{\mathbf{e}}_{B|A}.$$

All intermediate factors needed to evaluate this formula are shown here:

A	B	$\Theta^{\mathbf{e}}_A \Theta^{\mathbf{e}}_{B\|A}$
true	true	.54
true	false	.06

B	$\sum_A \Theta^{\mathbf{e}}_A \Theta^{\mathbf{e}}_{B\|A}$
true	.54
false	.06

B	C	$\Theta^{\mathbf{e}}_{C\|B} \sum_A \Theta^{\mathbf{e}}_A \Theta^{\mathbf{e}}_{B\|A}$
true	true	.162
true	false	.378
false	true	.030
false	false	.030

C	$\sum_B \Theta^{\mathbf{e}}_{C\|B} \sum_A \Theta^{\mathbf{e}}_A \Theta^{\mathbf{e}}_{B\|A}$
true	.192
false	.408

Therefore,

$\Pr(C = \text{true}, \ A = \text{true}) = .192$
$\Pr(C = \text{false}, \ A = \text{true}) = .408$
$\Pr(A = \text{true}) = .600$

To compute, $\Pr(C \mid A = \text{true})$, we normalise by $\Pr(A = \text{true})$.

C	$\Pr(C \mid A = \text{true})$
true	.32
false	.68

All we explained so far (except normalisation by e):

Algorithm 7 VE_PR2($\mathcal{N}$, **Q**, **e**, π)

input:
- $\mathcal{N}$: Bayesian network
- **Q**: variables in network $\mathcal{N}$
- **e**: instantiation of some variables in network $\mathcal{N}$
- π: an ordering of network variables not in **Q**

output: the joint marginal Pr(**Q**, **e**)

main:

1: $\mathcal{S} \leftarrow \{f^{\mathbf{e}} : f \text{ is a CPT of network } \mathcal{N}\}$
2: **for** $i = 1$ to length of order π **do**
3: $f \leftarrow \prod_k f_k$, where f_k belongs to $\mathcal{S}$ and mentions variable $\pi(i)$
4: $f_i \leftarrow \sum_{\pi(i)} f$
5: replace all factors f_k in $\mathcal{S}$ by factor f_i
6: **end for**
7: **return** $\prod_{f \in \mathcal{S}} f$

The only difference to to the previous algorithm (i.e., VE_PR1).

Canvas Quiz with Answers

1.

LAS8 In a proof by contradiction, such as DPLL or tableau, I can prove that a formula F is entailed by a knowledge base KB by showing that

A: the knowledge base KB and the negation of the formula are unsatisfiable.

the formula F is unsatisfiable, which implies that it must be entailed by the KB.

the knowledge base is unsatisfiable, which implies that the formula F must be entailed.

the knowledge base KB and the formula F are together unsatisfiable

2.
LAS5 A calculus is called complete w.r.t. the semantics of a logic if, and only, if

all the formulas it proves are semantically entailed.
A: it can prove all the semantically entailed formulas.
it proves all the correct formulas in finite time.
all the formulas it proves are tautologies.

3.
LAS4 A formula is in clause normal form if it is a A: conjunction of A: disjunctions of A: literals

CNF

4.
AS6 The semantics (meaning) of a formula in Propositional Logic is determined as:

multiple choice
a numeric value
set membership
A: a truth value

5.

Consider the sentence

((- A v -B) -> (A -> -B)) & (A v B)

Which of the following statements are true?

The sentence is neither valid, nor satisfiable, nor a contradiction
A: The sentence is satisfiable, but not valid
The sentence is valid, but not satisfiable
The sentence is not valid, and thus also not satisfiable
The sentence is not valid, and thus a contradiction

6.
LAC2 In the following truth table X1, X2 and X3 stand for possible truth values:

A	B	B->A	A -> (B -> A)
True	True	X1(T)	T
True	False	T	X2 (T)
False	True	F	X3 (T)
False	False	T	T

Which of the following statements is correct (multiple answers possible):

A -> (B -> A) is valid because there is a model that interprets it as true
X1 =True
X1 = False
A -> (B -> A) is not valid but satisfiable
A -> (B -> A) is valid because all possible models interpret it as true

X2 = True
X2 = False

7.
LAC1 Match formulas that are logically equivalent.

A -> (B & C)

A: (-A v B) & (-A v C)

A v (B & C)

A: (A v B) & (A v C)

A -> -(B & C)

A: -A v -B v -C

A & - (B & C)

A: A & (-B v -C)

A & (B -> C)

A: (A & -B) v (A & C)

A -> (B & C) is equivalent to -A v (B & C) by definition of ->, which is equivalent to (-A v B) & (-A v C) by rule 2 of slide 23

A v (B & C) is (AvB) &(AvC) by rule 2 on slide 23.
A-> -(B&C) is equivalent to -A v -(B&C) by definition of ->. -(B&C) is equivalent to -B v -C by rule 1.
A v -(B&C) applies rule 1 again.

8.

Weighted partial MAXSAT formulas

both hard and soft clauses are allowed. Subsumes all previously mentioned variations.

Consider the following statements about geese

1. all geese are white X
2. geese often have two legs Y
3. It is very likely that a goose is either white or has two legs or both (X v Y) v (X^Y)

4. if a goose does not have wings, it cannot fly. (-W Z)

and the following variables:

W stands for goose has wings

X stands for goose is white

Y stands for goose has two legs

Z stands for goose can not fly

Which of the following statements is a faithful representation of the knowledge described above?

F= (X,-∞) & (X v Y, 0.4) & (Y,-5) & (Z v W,-∞)
F= (X,∞) & (-(X v Y), 0.4) & (Y,5) & (-(Z v W),∞)
A: F= (-X,∞) & (X v Y, 0.4) & (-Y,5) & (Z v W,∞)
F= X & (X v Y, 0.4) & (-Y,5) & (Z v W)

9.

Use DPLL procedure to prove or disprove satisfiability of the formula

(X v Y v Z) & (X v -Y) & (Y v -Z) & (Z v -X) & (-X v -Y v -Z)

Label each step which part of the algorithm you have used

1. Simplification

Tautology: remove tautologies like P v -P from knowledge base (once in the beginning)

(X v Y v Z)

2. Split

3. pick a predicate and assume a truth value
4. – Heuristics of which literal to pick next can improve the efficiency of DPLL a lot
5. – DLCS (Dynamic Largest Combined Sum): Pick v with the largest count of positive and negative
6. occurrences: CP(v) + CN(v). If CP(v) > CN(v), choose v = 1, else v = 0
7. – DLIS (Dynamic Largest Individual Sum): Pick v with either largest CP or CN. Same truth
8. assignment as for DLCS.
9. – Jeroslow-Wang: weight of literal depends on the length of clauses it occurs in. Thereby, we
10. prefer small clauses. The score of a literal is J(v) =
11. P
12. c2Cv
13. 2 jcj, and we pick the highest value.
14. One-sided JW looks at v and :v independently, whereas the two-sided approach looks at sum
15. of v and :v, and just picks the truth value based on J(v) J(:v)) v = 1, else v = 0.
16. – MOMs (Maximum Occurrences in Clauses of Minimum Size): similar to JW, but we only look at

17. the smallest clauses in the knowledge base. The number of occurrences in those is indicates by
18. the function f. We choose the literal that maximizes [f(v) + f(:v)] 2_k + f(v) f(:v). k is
19. a tuning parameter for the trade-off between the balanced distribution of v and :v and their
20. individual ones.

– Pure literals: set predicates that solely occur in their positive or negative form to the corresponding truth value
– Unit clauses: set literals for which the knowledge base contain a unit clause to true (or the

predicate to false respectively)

10.
Give a pseudocode description of GSAT

Note that this algorithm tends to get stuck in local minimum (flipping a single variable does not increase score). Thus, we perform random restarts to start new. That's also why GSAT spends most time on plateaus where score is not improved

```
procedure GSAT(Sigma)
  for i := 1 to MAX-TRIES ; These are the restarts
    T := random(Sigma) ; random assignment
    for j := 1 to MAX-FLIPS ; To ensure termination
      if T satisfies Sigma then return T
      else T := T with variable flipped to maximize
                number of satisfied clauses
      ; It doesn't matter if the number does
      ; not increase.  This are the sideways flips
    end
  end
```

11.

What are advantages of a short restart and a long restart?

More than 1 answer might be correct.

Advantage long restart: if we've randomly jumped to a part of the search space without solutions, we will waste too much time on local walks

Advantage short restart: if we've randomly jumped to a part of the search space close to a solution, we might still miss it because we didn't do enough local exploration

A: Advantage short restart: if we've randomly jumped to a part of the search space without solutions, we won't waste to much time on local walks

A: Advantage long restart:if we've randomly jumped to a part of the search space close to a solution, we are likely to actually find the solution because we'll do enough local exploration

12.

In Description Logics, a model is an interpretation function that assigns a value true or false to each concept name.

- ontologies
-

True or false?

True
False

13.

In Description Logics a concept is called unsatisfiable if there is no model.

True or false?

True
False

14.

Description Logics are truly more expressive than Propositional Logics (i.e. everything that can be said in PL can be said in all Description Logics).

True or false?

True
False

15.
Suppose that the concept names to use are Kid and Ice-Cream, and the role name is likes.

Translate the following sentence from DL ALC to English

- ∃ (Links to an external site.) likes.Ice-Cream ⊑ Kid

A: Everybody who likes ice-cream is a kid
there is a kid that likes ice-cream
every kid likes ice-cream
everybody who only likes ice-cream is a kid

16.

Use a tableau to prove whether (∀r.A ⊓ ∀r.B) ⊑ ∀r.(A⊓B)

Explain the individual steps.

17.

1. Every Bayesian network implicitly represents a unique probability distributions.
2. True or false?

A: True
False

18.

Given a Bayesian Network G, if two random variables X and Y are independent (w.r.t. to the probability distribution induced by G) given a third random variable Z, then it follows that X and Y are d-separated by Z.

True or false?

True
A: False

19.

A divergent valve ($\leftarrow W \rightarrow$) is closed iff neither variable W nor any of its descendants appears in Z.

True or false?

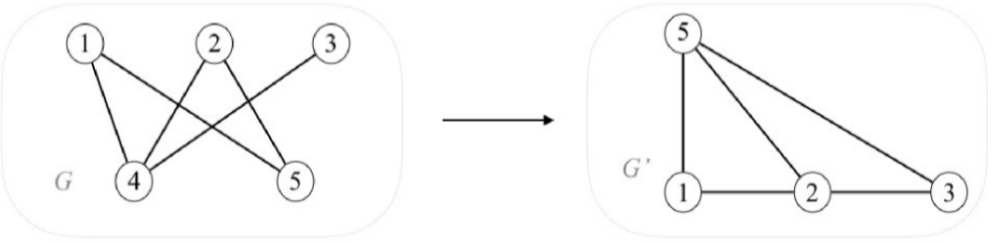
True
False

20.

Q5 - Check out the statements below whether they are true or not, and label them accordingly with either "true" or "false".

a) If $\alpha \models \beta$ and $\Pr(\alpha)=0$, then $\Pr(\beta)=0$.
b) If $\alpha \models \beta \models \gamma$, then $\Pr(\alpha \mid \beta) \geq \Pr(\alpha \mid \gamma)$.

c) In the interaction graph G below, if we eliminate 4, we get G'

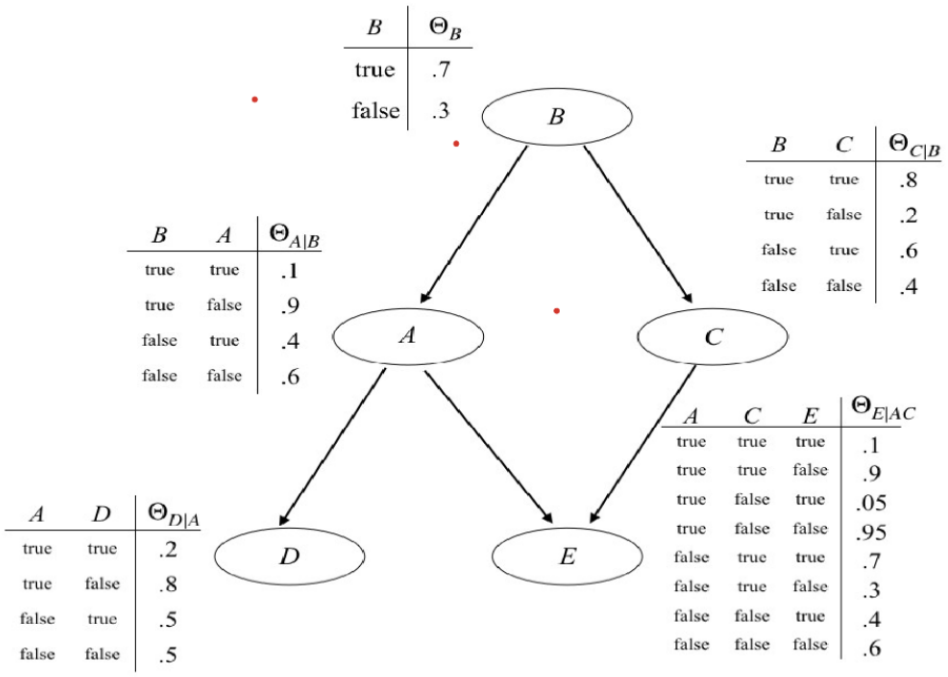


d) Assume G' above, according to the order elimination heuristic of "choosing the node with smallest degree", one possible elimination order is: 1, 5, 3 ,2.

Group of answer choices
A: c) false
a) true
c) true
b) false
A: b) true
A: d) true
A: a) false
d) false

21.

Q6 - Consider the Bayesian network in the figure, and answer the questions below accordingly.

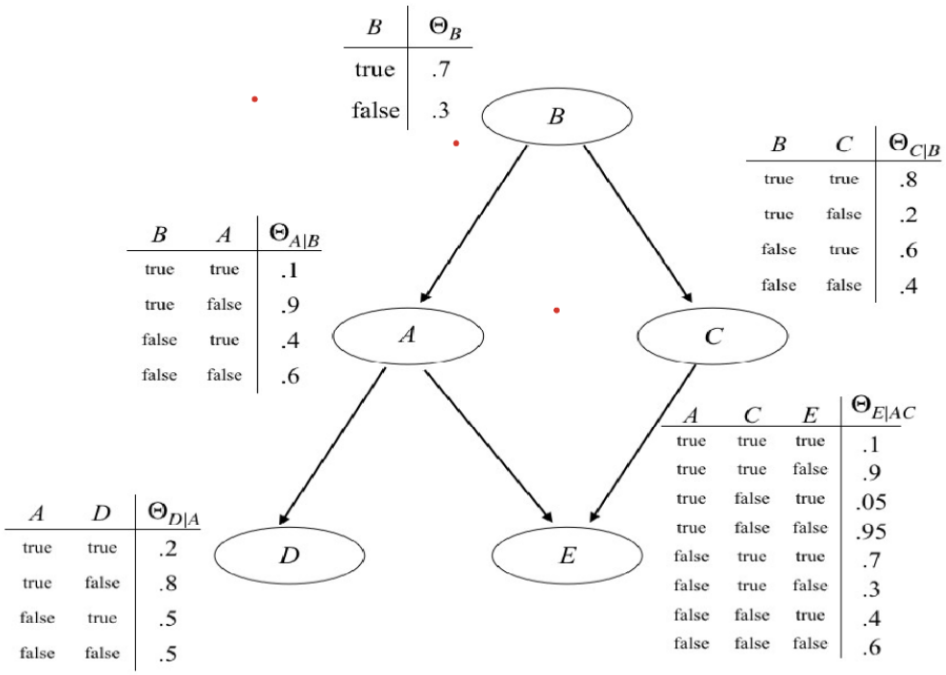


Are the following statements true or false?

IPr(D,emptyset,C) is True
A: IPr(C,{B,E},A) is False
A: IPr(E,{A},B) is False
IPr(E,{A},B) is True
A: IPr(D,emptyset,C) is False
IPr(C,{B,E},A) is True
IPr({BD},{A,c},E) is False
A: IPr({BD},{A,C},E) is True

22.

Q6 - Consider the Bayesian network in the figure, and answer the questions below accordingly.

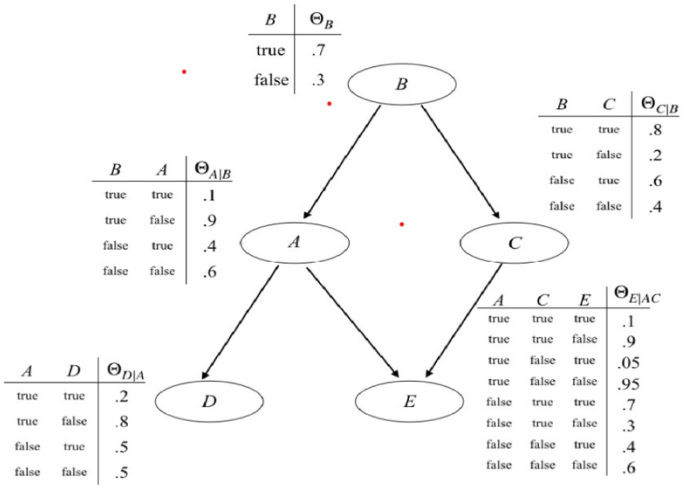


For this BN (as before), calculate the values for Pr(A=true,B=false,E=true)

A: 0.0096

23.

Q6 - Consider the Bayesian network in the figure, and answer the questions below accordingly.



Calculate the value for Pr(A=false, B=true | C= true)

A: 0.681

Probabilistic Graphical Models 4

Probabilistic Graphical Models 4

In this lecture, we will take a look at the problem of finding variable instantiations that have maximal probability under some given evidence.

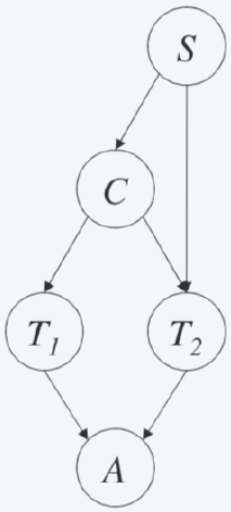
Consider the following examplary scenario:

• A society (S) that is 55% male and 45% female.
• Members of this population can suffer from a medical condition (C) that is more likely in males.
• Two diagnostic tests are available for detecting this condition, $T1$ and $T2$.
• $T2$ is more effective on females while both tests are equally effective on males.

Example Scenario:

- A society (S) that is 55% male and 45% female.
- Members of this population can suffer from a medical condition (C) that is more likely in males.
- Two diagnostic tests are available for detecting this condition, $T1$ and $T2$.
- $T2$ is more effective on females while both tests are equally effective on males.

One possible modelling of this scenario:



A: T_1 and T_2 agrees.

S	θ_s
male	.55
female	.45

S	C	$\theta_{c\|s}$
male	yes	.05
male	no	.95
female	yes	.01
female	no	.99

C	T_1	$\theta_{t_1\|c}$
yes	+ve	.80
yes	−ve	.20
no	+ve	.20
no	−ve	.80

S	C	T_2	$\theta_{t_2\|c,s}$
male	yes	+ve	.80
male	yes	−ve	.20
male	no	+ve	.20
male	no	−ve	.80
female	yes	+ve	.95
female	yes	−ve	.05
female	no	+ve	.05
female	no	−ve	.95

T_1	T_2	A	$\theta_{a\|t_1,t_2}$
+ve	+ve	yes	1
+ve	+ve	no	0
+ve	−ve	yes	0
+ve	−ve	no	1
−ve	+ve	yes	0
−ve	+ve	no	1
−ve	−ve	yes	1
−ve	−ve	no	0

One can partition this population into four different groups:

whether they are male or female,
and whether having the condition or not.

• Suppose that all we know is *a person took both tests and the test results agree* (i.e., A = true).
Then we may ask: *What is the most likely group that this person belong to?*

Network Pruning

A Tool of Simplification: Network Pruning

Bayesian Network Simplification: Network Pruning

Network structure has a major impact on the performance of variable elimination (and also on the performance of most algorithms).

Intuition: The complexity of inference can be very much affected by the number and location of query and evidence variables within the network structure.

Network Pruning is done in two parts: Node Pruning and Edge Pruning.

Node Pruning

Idea: Given a BN N and a query ($\mathbf{Q}$, $\mathbf{e}$), one can remove any leaf node (with its CPT) from the network as long as it does not belong to variables $\mathbf{Q} \cup \mathbf{E}$.

This does not affect the ability of the network to answer the query correctly, thanks to the following result.

Theorem: Let N be a Bayesian network and let ($\mathbf{Q}$, $\mathbf{e}$) be a corresponding query. If N' = pruneNodes(N, $\mathbf{Q} \cup \mathbf{E}$), then Pr($\mathbf{Q}$, $\mathbf{e}$) = Pr($\mathbf{Q}$, $\mathbf{e}$) where Pr and Pr' are the probability distributions induced by networks N and N' , respectively.

Remark that this can be done iteratively, possibly leading to further simplifications.

Edge Pruning

Idea: Given a Bayesian network N and a query (**Q**, **e**), one can eliminate some of the network edges and reduce some of its CPTs without affecting its ability to compute the joint marginal Pr(**Q**, **e**) correctly.

Procedure: For each edge $U \rightarrow X$ that originates from a node U in **E**

1. Remove the edge $U \rightarrow X$ from the network.
2. Replace the CPT $\Theta_{X|\mathbf{U}}$ for node X by a smaller CPT, which is obtained from $\Theta_{X|\mathbf{U}}$ by assuming the value u of parent U given in evidence **e**. This new CPT corresponds to $\sum_U \Theta^u_{X|\mathbf{U}}$.

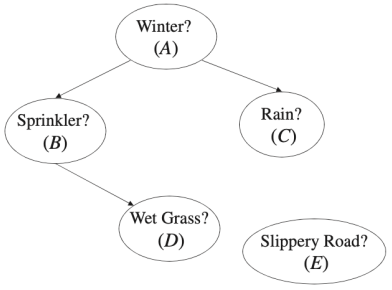
Theorem: Let N be a BN network and let **e** be an instantiation. If N′ = pruneEdges(N, **e**), then Pr(**Q**, **e**) = Pr′(**Q**, **e**) where Pr and Pr′ are the probability distributions induced by networks N and N , respectively

Example: Pruning the network G w.r.t. evidence C=false

The two edges originating from node C were deleted and CPTs and were modified. $\Theta_{D|B,C}\Theta_{E|C}$

G

Remark the pruned network is only good for answering queries of the form Pr(**q**, C = false) Otherwise pruned network and the original network may disagree.



A	Θ_A
true	.6
false	.4

A	B	$\Theta_{B\|A}$
true	true	.2
true	false	.8
false	true	.75
false	false	.25

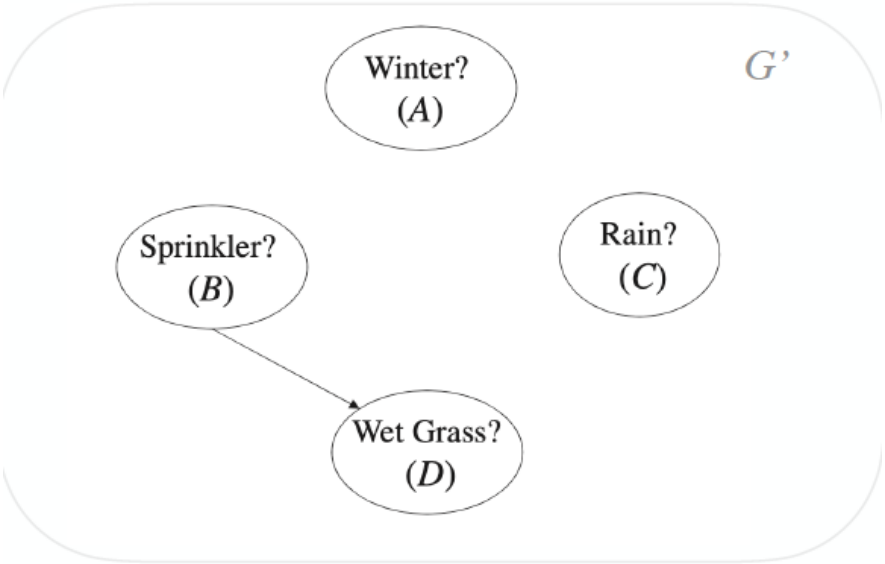
A	C	$\Theta_{C\|A}$
true	true	.8
true	false	.2
false	true	.1
false	false	.9

B	D	$\sum_C \Theta^{C=\text{false}}_{D\|BC}$
true	true	.9
true	false	.1
false	true	0
false	false	1

E	$\sum_C \Theta^{C=\text{false}}_{E\|C}$
true	0
false	1

Network Pruning: Example

Example: Pruning the network given the query **Q** = $\{D\}$ and **e** : A=true, C = false Pruning leads to removing node E and the edges originating from nodes A and C, and modifying the CPTs, , and . $\Theta_{B|A}$ $\Theta_{C|A}$ $\Theta_{D|BC}$



A	Θ_A
true	.6
false	.4

B	$\Theta'_B = \sum_A \Theta^{A=\text{true}}_{B\|A}$
true	.2
false	.8

C	$\Theta'_C = \sum_A \Theta^{A=\text{true}}_{C\|A}$
true	.8
false	.2

B	D	$\Theta'_{D\|B} = \sum_C \Theta^{C=\text{false}}_{D\|BC}$
true	true	.9
true	false	.1
false	true	0
false	false	1

Most Likely Instantiations

MAP Queries

One can partition this society into four different groups:

Whether they are male or female,
and having the condition or not.

Suppose that all we know is:

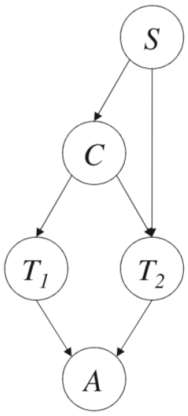
A person took both tests and the test results agree (i.e., A = yes).

Then we may ask things like:

What is the most likely group that this person belong to?

=

What are the most likely instantiation of variables S and C given evidence A=yes



S	θ_s
male	.55
female	.45

S	C	$\theta_{c\|s}$
male	yes	.05
male	no	.95
female	yes	.01
female	no	.99

C	T_1	$\theta_{t_1\|c}$
yes	+ve	.80
yes	−ve	.20
no	+ve	.20
no	−ve	.80

S	C	T_2	$\theta_{t_2\|c,s}$
male	yes	+ve	.80
male	yes	−ve	.20
male	no	+ve	.20
male	no	−ve	.80
female	yes	+ve	.95
female	yes	−ve	.05
female	no	+ve	.05
female	no	−ve	.95

T_1	T_2	A	$\theta_{a\|t_1,t_2}$
+ve	+ve	yes	1
+ve	+ve	no	0
+ve	−ve	yes	0
+ve	−ve	no	1
−ve	+ve	yes	0
−ve	+ve	no	1
−ve	−ve	yes	1
−ve	−ve	no	0

What is the most likely group that this person belong to?

=

What are the most likely instantiation of variables S and C given evidence A=yes

S=male and C=no with a posterior probability of about 0.49.

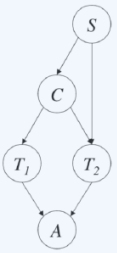
Such queries are known as *MAP (most a posteriori estimate) queries*,

And the answer to those queries are known as *MAP instantiations*.

MAP variables.

What are the most likely instantiation of variables S and C given evidence A=yes

$\Pr(S=\text{male}, C=\text{no} \mid A=\text{yes}) \approx 0.49$



S	θ_s
male	.55
female	.45

S	C	$\theta_{c\mid s}$
male	yes	.05
male	no	.95
female	yes	.01
female	no	.99

C	T_1	$\theta_{t_1\mid c}$
yes	+ve	.80
yes	−ve	.20
no	+ve	.20
no	−ve	.80

S	C	T_2	$\theta_{t_2\mid c,s}$
male	yes	+ve	.80
male	yes	−ve	.20
male	no	+ve	.20
male	no	−ve	.80
female	yes	+ve	.95
female	yes	−ve	.05
female	no	+ve	.05
female	no	−ve	.95

T_1	T_2	A	$\theta_{a\mid t_1,t_2}$
+ve	+ve	yes	1
+ve	+ve	no	0
+ve	−ve	yes	0
+ve	−ve	no	1
−ve	+ve	yes	0
−ve	+ve	no	1
−ve	−ve	yes	1
−ve	−ve	no	0

To calculate that, we can simply compute the posterior marginal (previous lecture) over MAP variables and then select the instantiation that has a maximum one.

A Special Case: MPE Queries

One can also consider the most likely instantiations of variables excluding the evidence (i.e., S, C, T_1, T_2):

S=female, C=no, T_1=−ve, T_2=−ve, with the probability value of 0.47.

Such queries are called MPE queries which can be considered as a special type of MAP queries.

Computing MPE instantiations is easier than computing MAP instantiations in general. (Hence, they have their own name.)

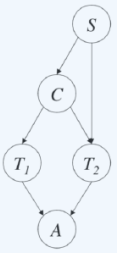
Remark:

Note that if we map the MPE to the previous variable, we likely to get a different answer

S=female, C=no

which is different than the earlier answer:

S=male, C=no



S	θ_s
male	.55
female	.45

S	C	$\theta_{c\mid s}$
male	yes	.05
male	no	.95
female	yes	.01
female	no	.99

C	T_1	$\theta_{t_1\mid c}$
yes	+ve	.80
yes	−ve	.20
no	+ve	.20
no	−ve	.80

S	C	T_2	$\theta_{t_2\mid c,s}$
male	yes	+ve	.80
male	yes	−ve	.20
male	no	+ve	.20
male	no	−ve	.80
female	yes	+ve	.95
female	yes	−ve	.05
female	no	+ve	.05
female	no	−ve	.95

T_1	T_2	A	$\theta_{a\mid t_1,t_2}$
+ve	+ve	yes	1
+ve	+ve	no	0
+ve	−ve	yes	0
+ve	−ve	no	1
−ve	+ve	yes	0
−ve	+ve	no	1
−ve	−ve	yes	1
−ve	−ve	no	0

Remark:
Although likely to be different, sometimes this technique is used to approximate MAP instances.

MAP & MPE Queries

Formal Definition Given a Bayesian network with variables **M/Q** and evidence **e,** **MAP/MPE** queries (values and instantiations)are defined as follows:

MAP

Value: $\text{MAP}_P(\mathbf{M}, \mathbf{e}) \stackrel{def}{=} \max_{\mathbf{m}} \Pr(\mathbf{m}, \mathbf{e})$.

Instantiation: $\text{MAP}(\mathbf{M}, \mathbf{e}) \stackrel{def}{=} \operatorname{argmax}_{\mathbf{m}} \Pr(\mathbf{m}, \mathbf{e})$.

MPE

Value: $\text{MPE}_P(\mathbf{e}) \stackrel{def}{=} \max_{\mathbf{q}} \Pr(\mathbf{q}, \mathbf{e})$.

Instantiation: $\text{MPE}(\mathbf{e}) \stackrel{def}{=} \operatorname{argmax}_{\mathbf{q}} \Pr(\mathbf{q}, \mathbf{e})$.

Remark:

Note that both MAP and MPE instantiations can also be characterised by $\Pr(\mathbf{q}|\mathbf{e})$ since, $\Pr(\mathbf{q}|\mathbf{e}) = \Pr(\mathbf{q}, \mathbf{e})/\Pr(\mathbf{e})$ and $\Pr(\mathbf{e})$ is independent of instantiation $\mathbf{q}$.

Computing MPE & MAP (by Variable Elimination)

Lets assume no evidence whatsoever, then
MPE instantiation:

	S	C	T_1	T_2	A	Pr(.)
31	female	no	−ve	−ve	yes	.338580

We can calculate this by variable elimination:
Idea: Instead of summing-out, we maximise-out.
Example: To maximise S out from the factor, $f(S,C, T1, T2, A)$ we construct a factor on the remaining variables : $C, T1, T2, A$

	S	C	T_1	T_2	A	Pr(.)
1	male	yes	+ve	+ve	yes	.017600
3	male	yes	+ve	−ve	no	.004400
5	male	yes	−ve	+ve	no	.004400
7	male	yes	−ve	−ve	yes	.001100
9	male	no	+ve	+ve	yes	.020900
11	male	no	+ve	−ve	no	.083600
13	male	no	−ve	+ve	no	.083600
15	male	no	−ve	−ve	yes	.334400
17	female	yes	+ve	+ve	yes	.003420
19	female	yes	+ve	−ve	no	.000180
21	female	yes	−ve	+ve	no	.000855
23	female	yes	−ve	−ve	yes	.000045
25	female	no	+ve	+ve	yes	.004455
27	female	no	+ve	−ve	no	.084645
29	female	no	−ve	+ve	no	.017820
31	female	no	−ve	−ve	yes	.338580

Even rows are omitted since they are all zeros e.g., $T_1 \neq T_2$ with A = yes.

	S	C	T_1	T_2	A	Pr(.)
1	male	yes	+ve	+ve	yes	.017600
17	female	yes	+ve	+ve	yes	.003420

→

C	T_1	T_2	A	
yes	+ve	+ve	yes	.017600 = max(.017600, .003420)

When we eliminate S, we call the new factor $\max_S f$.

Note that $\max_S f$ and f agree on MPE, hence it is as good as f in calculating this probability.

We continue until we are left with the trivial factor, and this gives us the MPE.

	S	C	T_1	T_2	A	Pr(.)
1	male	yes	+ve	+ve	yes	.017600
3	male	yes	+ve	−ve	no	.004400
5	male	yes	−ve	+ve	no	.004400
7	male	yes	−ve	−ve	yes	.001100
9	male	no	+ve	+ve	yes	.020900
11	male	no	+ve	−ve	no	.083600
13	male	no	−ve	+ve	no	.083600
15	male	no	−ve	−ve	yes	.334400
17	female	yes	+ve	+ve	yes	.003420
19	female	yes	+ve	−ve	no	.000180
21	female	yes	−ve	+ve	no	.000855
23	female	yes	−ve	−ve	yes	.000045
25	female	no	+ve	+ve	yes	.004455
27	female	no	+ve	−ve	no	.084645
29	female	no	−ve	+ve	no	.017820
31	female	no	−ve	−ve	yes	.338580

Even rows are omitted since they are all zeros e.g., $T_1 \neq T_2$ with A = yes.

Just like summing-out:

Maximising-Out

Given a factor f over variables $\mathbf{X}$, maximising-out variable $X \in \mathbf{X}$, from factor f results in another factor over variables $\mathbf{X} \backslash \{X\}$ and is defined as $\left(\max_X f\right)(\mathbf{y}) \stackrel{def}{=} \max_x f(x, \mathbf{y})$.

Maximising-out

$$\left(\max_X f\right)(\mathbf{y}) \stackrel{def}{=} \max_x f(x, \mathbf{y}).$$

Recall: Elimination continues until the trivial factor.

In case of MAP: We first sum-out all the non-MAP variables, and then maximise-out the MAP variables.

Remarks:

Similar to summation, maximisation is commutative,

Hence, no need to specify the order we maximise.

Moreover, it interacts with multiplication in a similar fashion to summation.

Theorem 4.1

If f_1 and f_2 are factors and if variable X appears only in f_2, then

$$\max_X f_1 f_2 = f_1 \max_X f_2.$$

Extended Factors

We can compute MPE instantiations in addition to its probability:

Idea: The extended factors which assign to each instantiation both a number and an instantiation

Extended Factors

Let $f(\mathbf{X})$ be an extended factor, X be a variable in $\mathbf{X}$, and let $\mathbf{Y} = \mathbf{X} \setminus \{X\}$. We then define

$$\left(\max_X f\right)[\mathbf{y}] \stackrel{def}{=} x^\star f[x^\star, \mathbf{y}],$$

where $\mathbf{x}^\star = \text{argmax}_{\mathbf{X}} f(\mathbf{x}, \mathbf{y})$. Moreover, let $\mathbf{Z} = \mathbf{X} \cup \mathbf{Y}$

$$(f_1 f_2)[\mathbf{z}] \stackrel{def}{=} f_1[\mathbf{x}] f_2[\mathbf{y}],$$

where $\mathbf{x}$ and $\mathbf{y}$ are instantiation compatible with $\mathbf{z}$ ($\mathbf{x} \sim \mathbf{z}$ and $\mathbf{y} \sim \mathbf{z}$).

MPE Algorithm

Algorithm 27 VE_MPE($\mathcal{N}$, **e**)

input:
- $\mathcal{N}$: a Bayesian network
- **e**: evidence

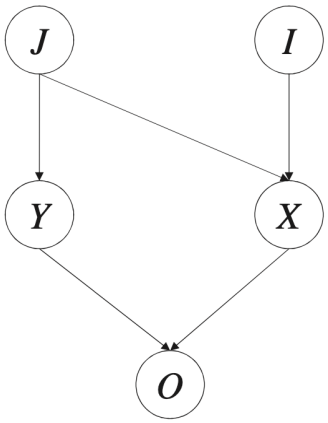
output: trivial factor f, where $f(\top)$ is the MPE probability of evidence **e**

main:

1: $\mathcal{N}' \leftarrow$ pruneEdges($\mathcal{N}$, **e**) {see Section 6.9.2}
2: $\mathbf{Q} \leftarrow$ variables in network $\mathcal{N}'$
3: $\pi \leftarrow$ elimination order of variables $\mathbf{Q}$
4: $\mathcal{S} \leftarrow \{f^{\mathbf{e}} : f$ is a CPT of network $\mathcal{N}'\}$
5: **for** $i = 1$ to $|\mathbf{Q}|$ **do**
6: $\quad f \leftarrow \prod_k f_k$, where f_k belongs to $\mathcal{S}$ and mentions variable $\pi(i)$
7: $\quad f_i \leftarrow \max_{\pi(i)} f$
8: $\quad$ replace all factors f_k in $\mathcal{S}$ by factor f_i
9: **end for**
10: **return** trivial factor $\prod_{f \in \mathcal{S}} f$

Computing MPE

Example:Assume, we want to compute MPE instantiation given evidence J=true and O = false.



I	Θ_I
true	.5
false	.5

J	Θ_J
true	.5
false	.5

J	Y	$\Theta_{Y\|J}$
true	true	.01
true	false	.99
false	true	.99
false	false	.01

I	J	X	$\Theta_{X\|IJ}$
true	true	true	.95
true	true	false	.05
true	false	true	.05
true	false	false	.95
false	true	true	.05
false	true	false	.95
false	false	true	.05
false	false	false	.95

X	Y	O	$\Theta_{O\|XY}$
true	true	true	.98
true	true	false	.02
true	false	true	.98
true	false	false	.02
false	true	true	.98
false	true	false	.02
false	false	true	.02
false	false	false	.98

Pruning edges and simplifying (omitting rows that are assigned with 0) gives the following.

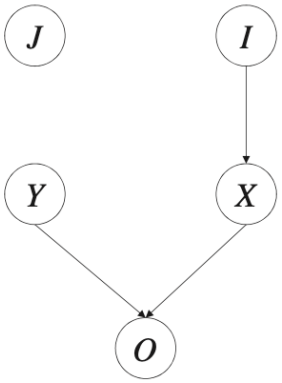
I	$\Theta^{\mathbf{e}}_I$	
true	.5	$\top$
false	.5	$\top$

J	$\Theta^{\mathbf{e}}_J$	
true	.5	$\top$

Y	$\Theta^{\mathbf{e}}_Y$	
true	.01	$\top$
false	.99	$\top$

I	X	$\Theta^{\mathbf{e}}_{X\|I}$	
true	true	.95	$\top$
true	false	.05	$\top$
false	true	.05	$\top$
false	false	.95	$\top$

X	Y	O	$\Theta^{\mathbf{e}}_{O\|XY}$	
true	true	false	.02	$\top$
true	false	false	.02	$\top$
false	true	false	.02	$\top$
false	false	false	.98	$\top$



Assuming the elimination order J, I, X, Y, O, Algorithm VE_MPE will then evaluate the following expression:

$$\max_{J,I,X,Y,O} \Theta^{\mathbf{e}}_{I}\ \Theta^{\mathbf{e}}_{J}\ \Theta^{\mathbf{e}}_{Y}\ \Theta^{\mathbf{e}}_{X|I}\ \Theta^{\mathbf{e}}_{O|XY}$$
$$= \max_{O}\left(\max_{Y}\left(\max_{X}\left(\max_{I}\Theta^{\mathbf{e}}_{I}\Theta^{\mathbf{e}}_{X|I}\right)\Theta^{\mathbf{e}}_{O|XY}\right)\Theta^{\mathbf{e}}_{Y}\right)\left(\max_{J}\Theta^{\mathbf{e}}_{J}\right).$$

Continuing the example:

$$\max_{J,I,X,Y,O} \Theta^{\mathbf{e}}_{I}\ \Theta^{\mathbf{e}}_{J}\ \Theta^{\mathbf{e}}_{Y}\ \Theta^{\mathbf{e}}_{X|I}\ \Theta^{\mathbf{e}}_{O|XY}$$
$$= \max_{O}\left(\max_{Y}\left(\max_{X}\left(\max_{I}\Theta^{\mathbf{e}}_{I}\Theta^{\mathbf{e}}_{X|I}\right)\Theta^{\mathbf{e}}_{O|XY}\right)\Theta^{\mathbf{e}}_{Y}\right)\left(\max_{J}\Theta^{\mathbf{e}}_{J}\right).$$

Intermediate steps:

1/2

	$\max_J \Theta^{\mathbf{e}}_J$	
⊤	.5	J = true

I	X	$\Theta^{\mathbf{e}}_I \Theta^{\mathbf{e}}_{X\|I}$	
true	true	.475	⊤
true	false	.025	⊤
false	true	.025	⊤
false	false	.475	⊤

X	$\max_I \Theta^{\mathbf{e}}_I \Theta^{\mathbf{e}}_{X\|I}$	
true	.475	I = true
false	.475	I = false

X	Y	O	$\left(\max_I \Theta^{\mathbf{e}}_I \Theta^{\mathbf{e}}_{X\|I}\right)\Theta^{\mathbf{e}}_{O\|XY}$	
true	true	false	.0095	I = true
true	false	false	.0095	I = true
false	true	false	.0095	I = false
false	false	false	.4655	I = false

Y	O	$\max_X\left(\max_I \Theta^{\mathbf{e}}_I \Theta^{\mathbf{e}}_{X\|I}\right)\Theta^{\mathbf{e}}_{O\|XY}$	
true	false	.0095	I = true, X = true
false	false	.4655	I = false, X = false

2/2

Y	O	$\left(\max_X\left(\max_I \Theta^{\mathbf{e}}_I \Theta^{\mathbf{e}}_{X\|I}\right)\Theta^{\mathbf{e}}_{O\|XY}\right)\Theta^{\mathbf{e}}_{Y}$	
true	false	.000095	I = true, X = true
false	false	.460845	I = false, X = false

O	$\max_Y\left(\max_X\left(\max_I \Theta^{\mathbf{e}}_I \Theta^{\mathbf{e}}_{X\|I}\right)\Theta^{\mathbf{e}}_{O\|XY}\right)\Theta^{\mathbf{e}}_{Y}$	
false	.460845	I = false, X = false, Y = false

	$\max_O\left(\max_Y\left(\max_X\left(\max_I \Theta^{\mathbf{e}}_I \Theta^{\mathbf{e}}_{X\|I}\right)\Theta^{\mathbf{e}}_{O\|XY}\right)\Theta^{\mathbf{e}}_{Y}\right)$	
⊤	.460845	I = false, X = false, Y = false, O = false

	$\max_O\left(\max_Y\left(\max_X\left(\max_I \Theta^{\mathbf{e}}_I \Theta^{\mathbf{e}}_{X\|I}\right)\Theta^{\mathbf{e}}_{O\|XY}\right)\Theta^{\mathbf{e}}_{Y}\right)\left(\max_J \Theta^{\mathbf{e}}_J\right)$	
⊤	.2304225	J = true, I = false, X = false, Y = false, O = false

MPE Instance

Recall that in case of MAP: We first sum-out all the non-MAP variables, and then maximise-out the MAP variables.

Algorithm 30 VE_MAP($\mathcal{N}$, **M**, **e**)

input:

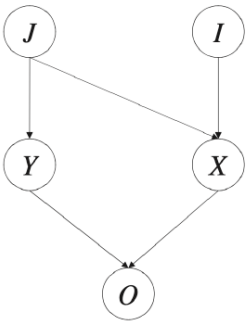
N: Bayesian network
M: some variables in the network
e: evidence ($\mathbf{E} \cap \mathbf{M} = \emptyset$)

output: trivial factor containing the MAP probability $\text{MAP}_P(\mathbf{M}, \mathbf{e})$

main:

1: $\mathcal{N}' \leftarrow$pruneNetwork($\mathcal{N}$, **M**, **e**)
2: $\pi \leftarrow$ a variable elimination order for $\mathcal{N}'$ in which variables **M** appear last
3: $\mathcal{S} \leftarrow \{f^{\mathbf{e}} : f \text{ is a CPT of network } \mathcal{N}'\}$
4: **for** $i = 1$ to length of order π **do**
5: $f \leftarrow \prod_k f_k$, where factor f_k belongs to $\mathcal{S}$ and mentions variable $\pi(i)$
6: **if** $\pi(i) \in \mathbf{M}$ **then**
7: $f_i \leftarrow \max_{\pi(i)} f$
8: **else**
9: $f_i \leftarrow \sum_{\pi(i)} f$
10: **end if**
11: replace all factors f_k in $\mathcal{S}$ by factor f_i
12: **end for**
13: **return** trivial factor $\prod_{f \in \mathcal{S}} f$

Example: Assume the same BN. MAP variables **M**={*I*, *J*}
We use the order π = *O*, *Y*, *X*, *I*, *J*



Summing out variables *O*, *Y* , and *X*, we obtain the following set of factors, which represent a factored representation of the joint marginal Pr(*I*, *J*, *O* =true) where "*O*=true" is the evidence

I	*J*	f_1	
true	true	.93248	⊤
true	false	.97088	⊤
false	true	.07712	⊤
false	false	.97088	⊤

I	f_2	
true	.5	⊤
false	.5	⊤

J	f_3	
true	.5	⊤
false	.5	⊤

Next, to eliminate *I* and *J*, we multiply their factors and then maximizing out.

First *I*,

I	*J*	$f_1\ f_2$	
true	true	.466240	⊤
true	false	.485440	⊤
false	true	.038560	⊤
false	false	.485440	⊤

J	$\max_I f_1\ f_2$	
true	.466240	*I* = true
false	.485440	*I* = true

Next *J*,

J	$(\max_I f_1\ f_2)\ f_3$	
true	.233120	*I* = true
false	.242720	*I* = true

	$\max_J (\max_I f_1\ f_2)\ f_3$	
⊤	.242720	*I* = true, *J* = false

MAP probability, Pr(*I* = true, *J* = false, *O* = true)

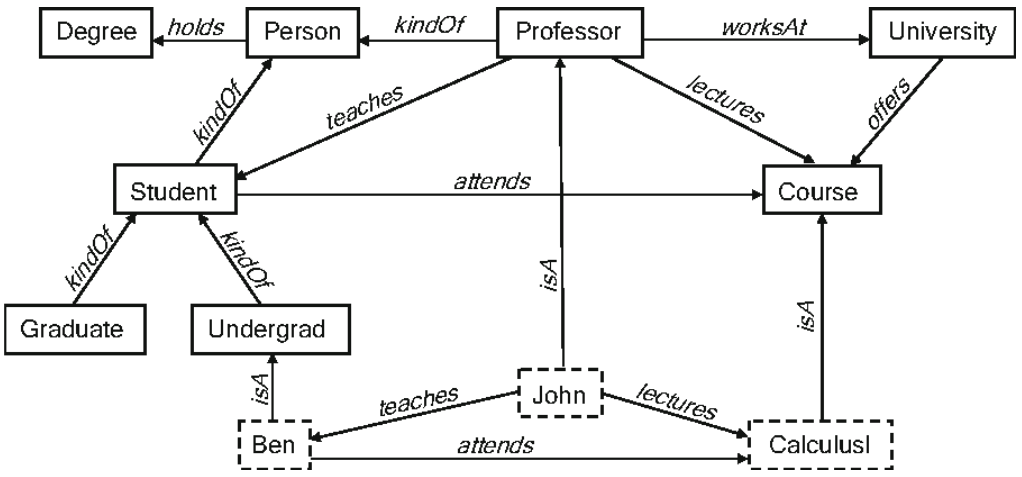
VU VRIJE UNIVERSITEIT AMSTERDAM

Ontologies

11. Ontologies: a way of organizing knowledge that is useful for *people*

What Is an ontology?
•"An ontology is a formal, explicit specification of a shared conceptualization.

Ontology/ Knowledge Graph



Definition(s) of an ontology

"An ontology defines the terms used to describe and represent an area of knowledge" -Jeff Heflin

"An ontology
-defines the basic terms and relations
-that form the vocabulary of a topic area
-as well as the rules for combining terms and relations
-to infer extensions to the vocabulary" -Neches ('91)

The goal of an Ontology = Knowledge Sharing and Reuse
•Communication between people
•Interoperability between software agent

Ontology Elements

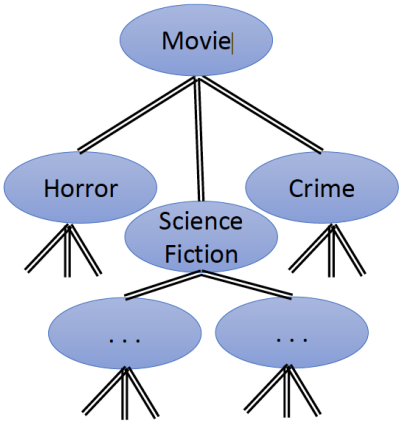
Class (concept,type)
-a name and a set of properties that describe a certain set of individuals in the domain
Example: person + name, birthdate, ID-nr, …

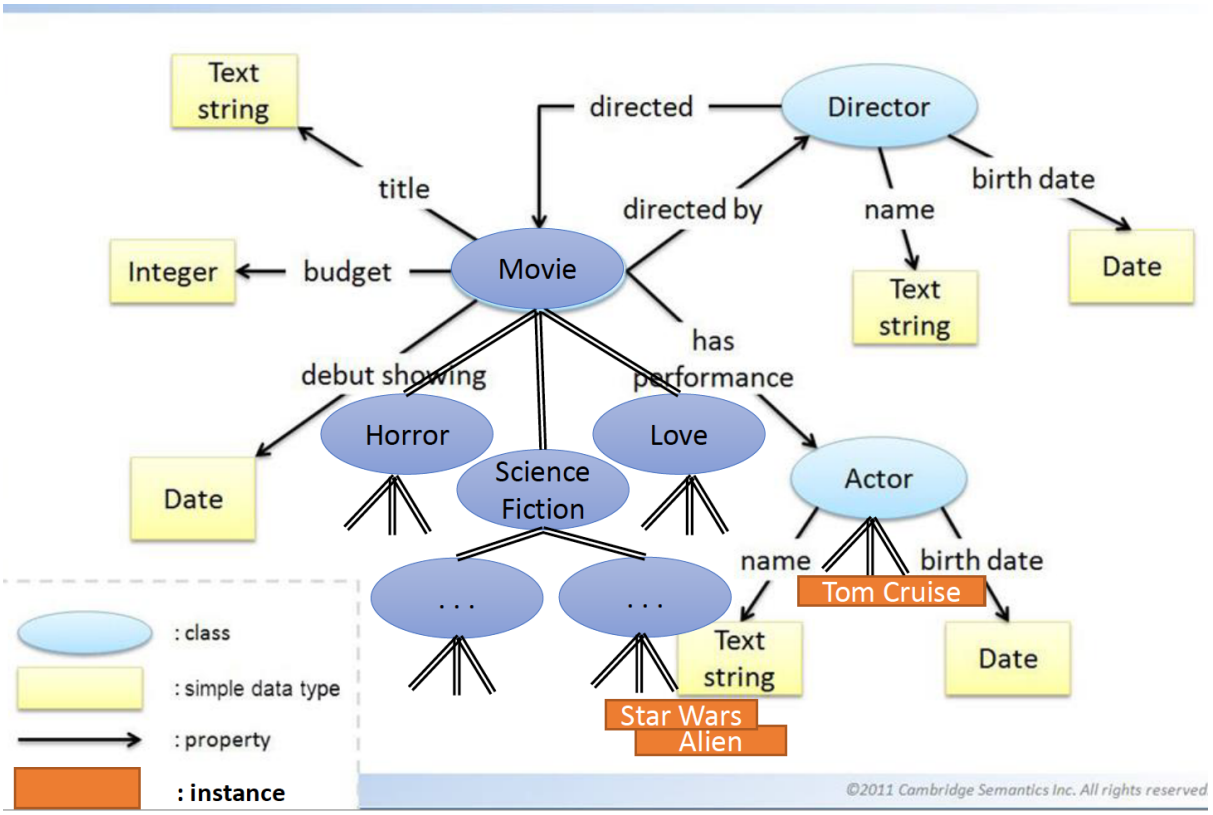
Instances
-the members of the sets defined by classes
Example: person45, name=Frank, birthdate=01-01-1960, ID-nr=12345

Property (relation)
-assert facts about the instances
Example: person45 is–father-of person256

An example of ontology

A Film Ontology





Real life examples

handcrafted
•music: Music ontology , MusicMoz(1073/7)
•biomedical: SNOMED(300k), Gene Ontology GO(43k terms)Emtree(45k+190kSystems biology
•ranging from lightweight
•UNSPC, Open directory (400k) to heavyweight (Cyc (300k))
•ranging from small (METAR) to large (UNSPC)

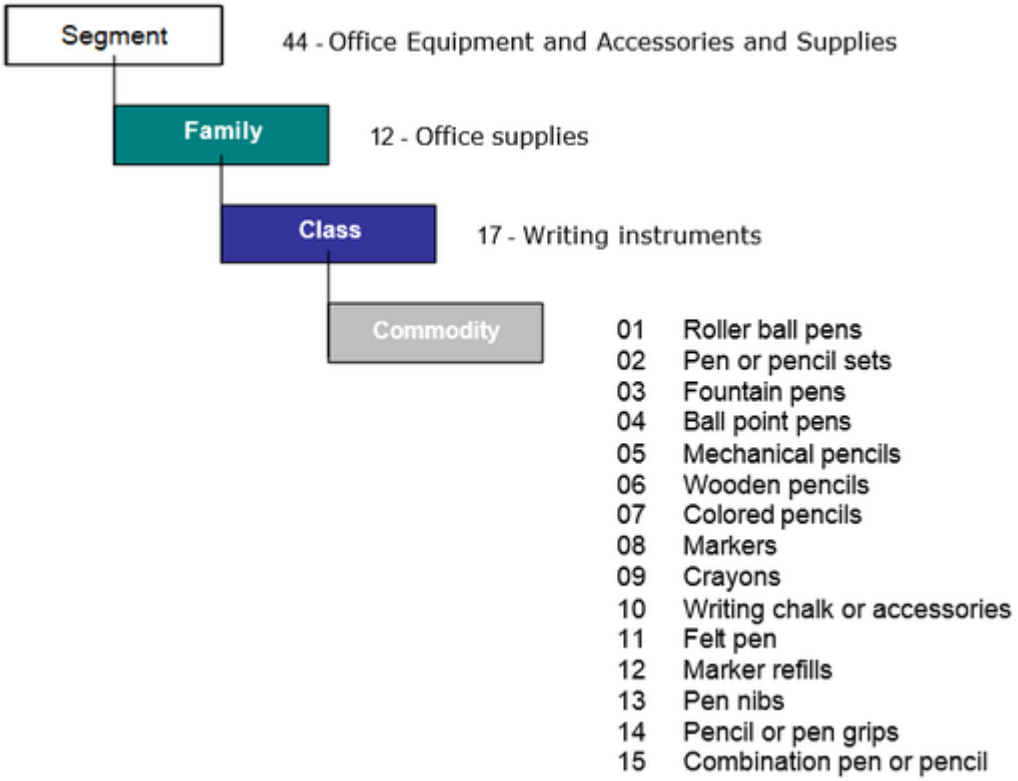
METAR is the most common format in the world for the transmission of observational weather data

METAR Class Hierarchy

- UnitAbbreviation ()
- WeatherEvent (hasWeatherQualifier)
  - CumulativeEvent ()
    - MaximumTemperature ()
      - SixHourMaximumTemp ()
      - TwentyFourHourMaximumTemp ()
    - MinimumTemperature ()
      - SixHourMinimumTemp ()
      - TwentyFourHourMinimumTemp ()
    - SixHourCumulativeEvent ()
      - SixHourMaximumTemp ()
      - SixHourMinimumTemp ()
    - ThreeHourCumulativeEvent ()
      - ThreeHourPressureTendency ()
    - TwentyFourHourCumulativeEvent ()
      - TwentyFourHourMaximumTemp ()
      - TwentyFourHourMinimumTemp ()
  - CurrentWeatherEvent ()
    - ObscurationEvent ()
      - Fog ()
      - Haze ()
      - Mist ()
      - Sand ()
      - Smoke ()
      - Spray ()
      - VolcanicAsh ()
      - WidespreadDust ()

A METAR weather report is predominantly used by aircraft pilots, and by meteorologists, who use aggregated METAR information to assist in weather forecasting.

UNSPC Product and services



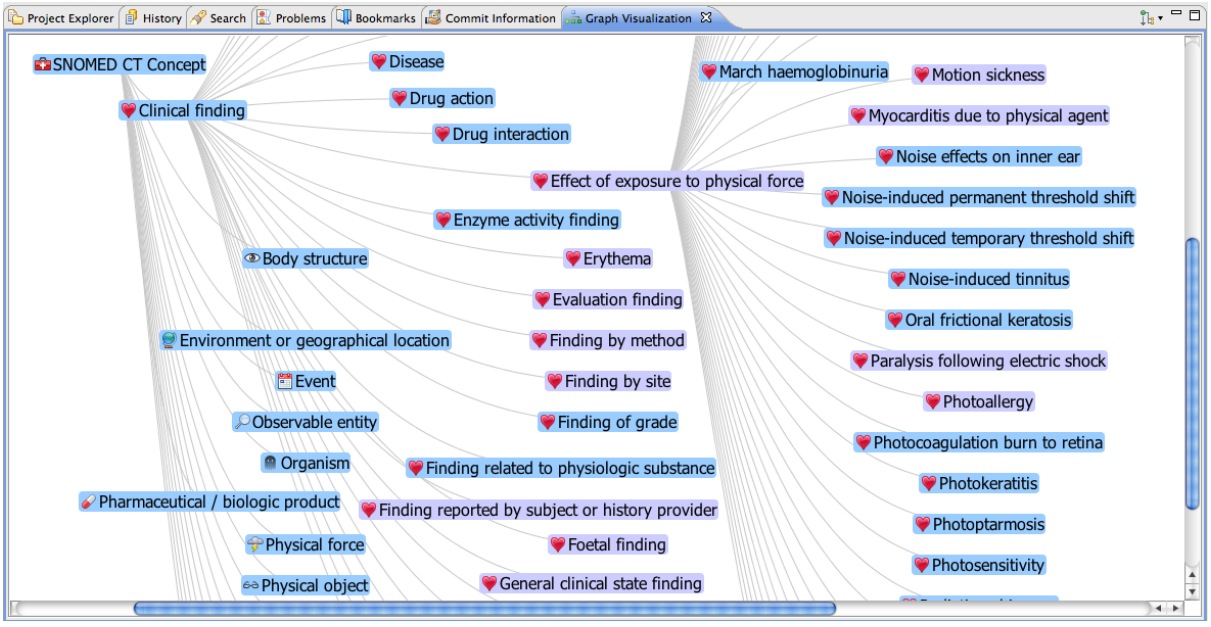
Level	Code	Description
Segment	**44**000000	Office Equipment, Accessories and Supplies
Family	44**12**0000	Office supplies
Class	4412**19**00	Ink and lead refills
Commodity	44121**903**	Pen refills

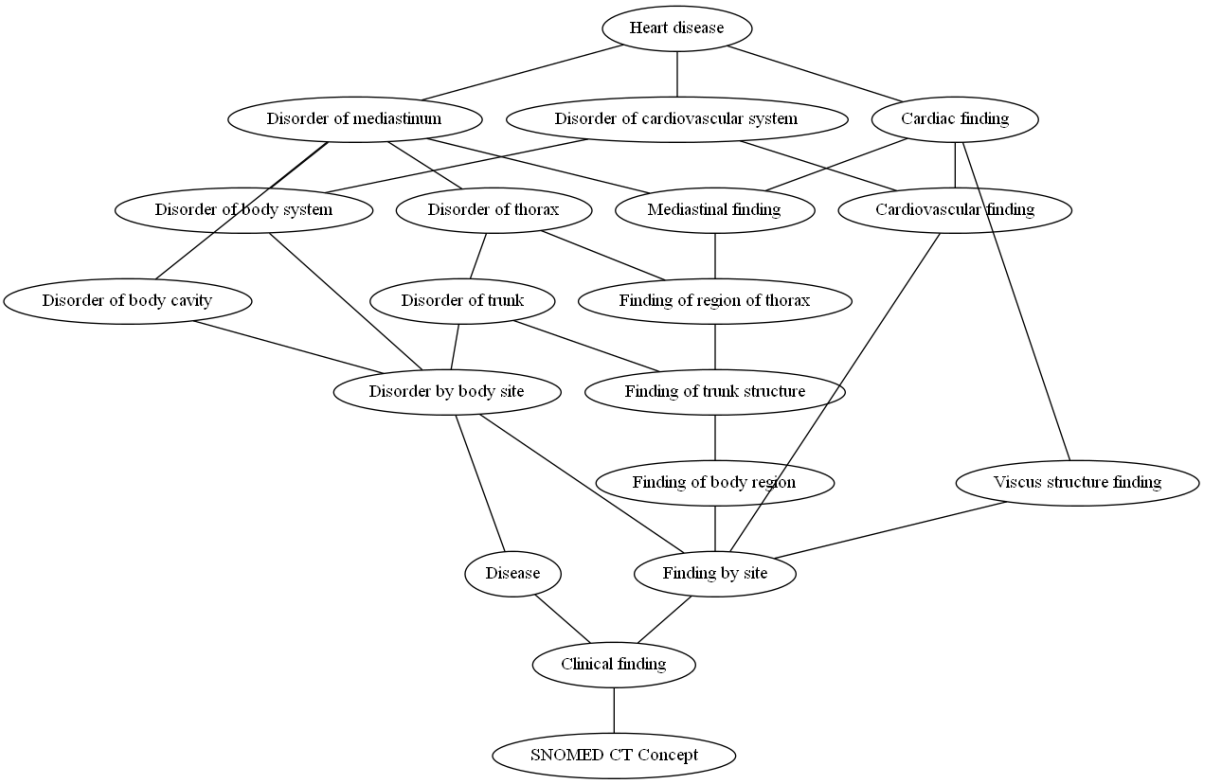
The **United Nations Standard Products and Services Code**(**UNSPSC**) is a taxonomyof products and services for use in eCommerce. It is a four-level hierarchy coded as an eight-digit number, with an optional fifth level adding two more digits.

By classifying their products & services, businesses can assist their customers with:
Finding and Purchasing
Product discovery
Facilitates expenditure analysis.
Control and uniformity across the company

UNSPC Product and services

SNOMED: used in healthcare





Biomedical ontologies (a few..)

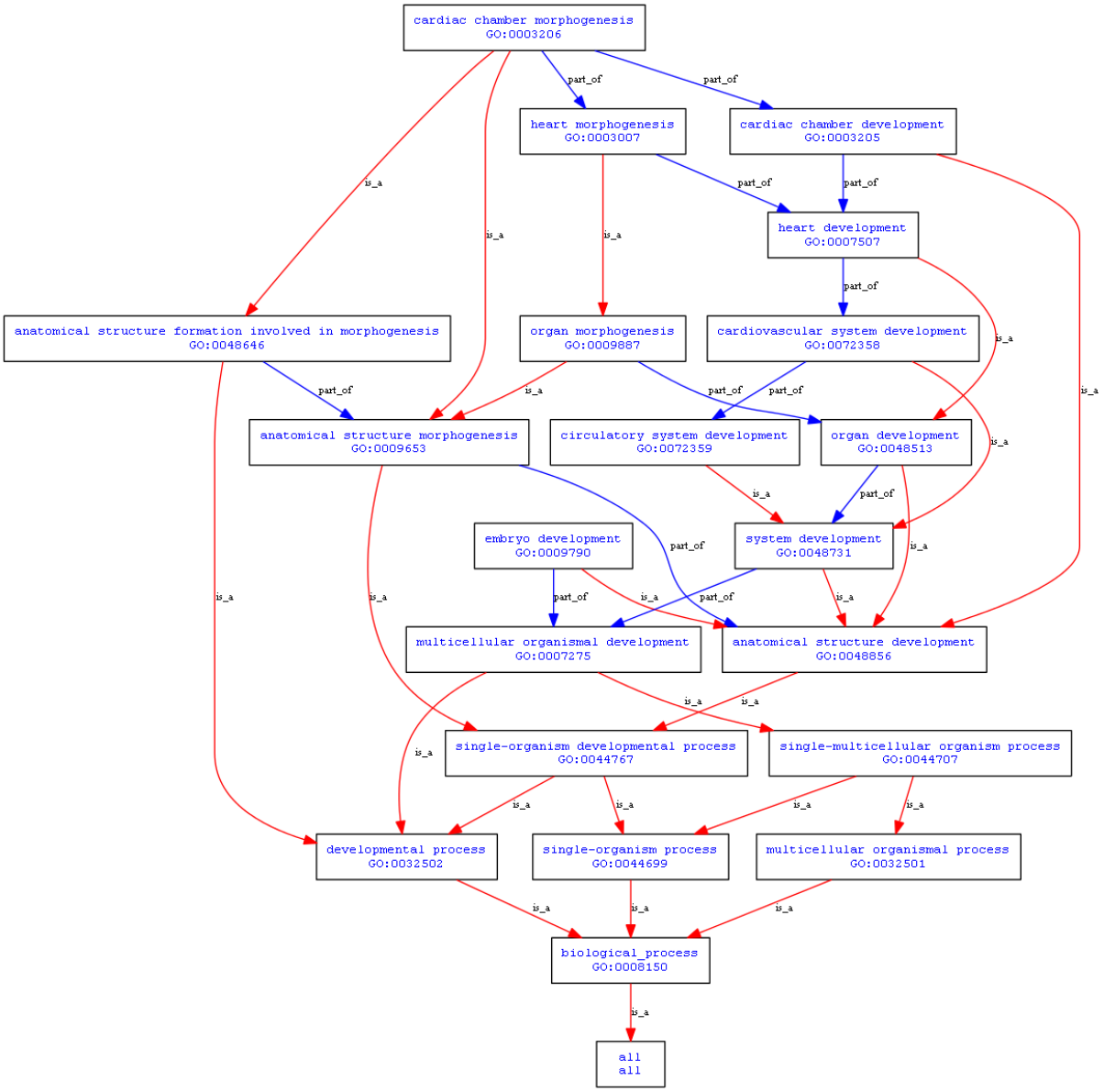
•Mesh
•Medical Subject Headings, National Library of Medicine
•22.000 descriptions
•EMTREE
•Commercial Elsevier, Drugs and diseases
•45.000 terms, 190.000 synonyms
•UMLS
•Integrates 100 different vocabularies
•SNOMED
•300.000 concepts, College of American Pathologists
•Gene Ontology
•15.000 terms in molecular biology
•NCBI Cancer Ontology:
•17,000 classes (about 1M definitions),

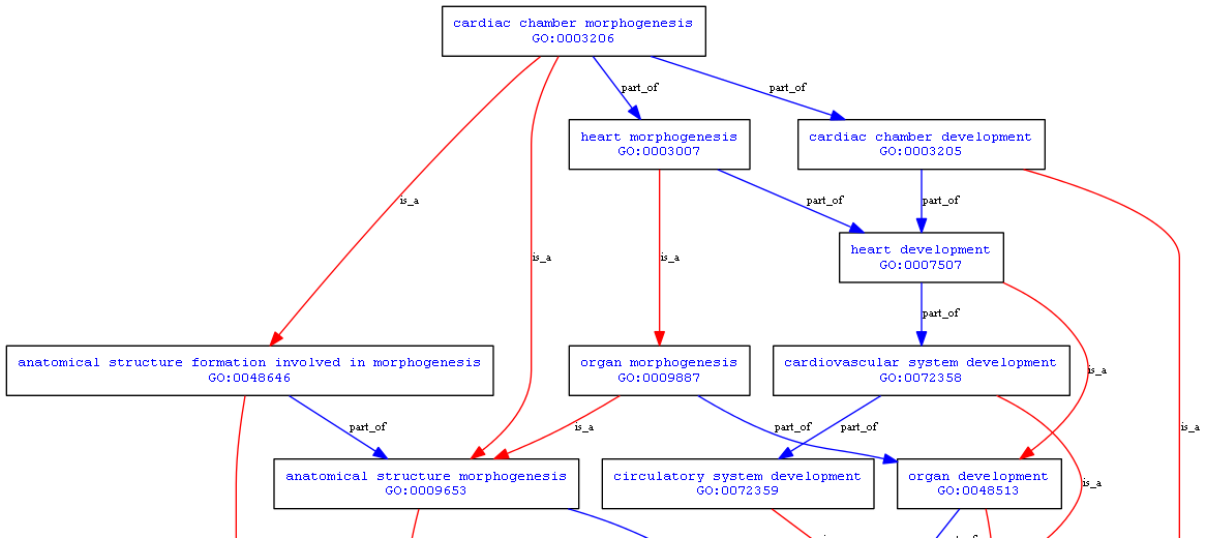
Gene Ontology

The Gene Ontology (GO) is a major bioinformatics initiative to unify the representation of gene and gene product attributes across all species

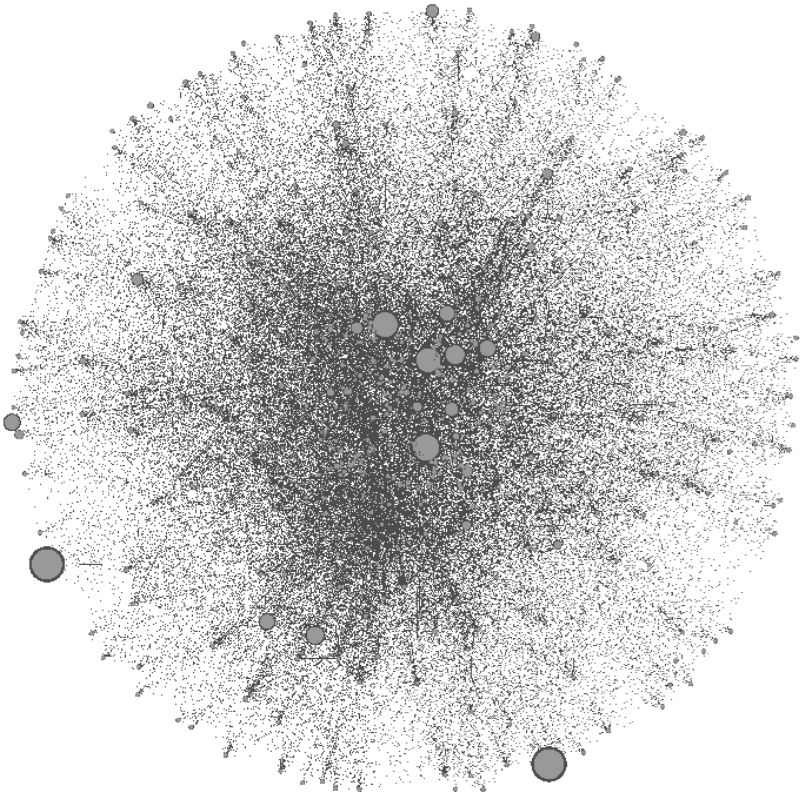
Three domains:
cellular component
molecular function
biological process

Gene product: Actin… UniProtKB:P68032
GO term: heartcontraction; GO:0060047 (biologicalprocess)
Evidencecode: InferredfromMutant Phenotype(IMP)
Reference: PMID 17611253
Assignedby: UniProtKB, June6, 2008





Bio-medical ontologies in Bio-portal at Stanford



A logic for Ontologies

A taskforLogic Engineering (KR)
•Whatlogic do we need forUMS PC?

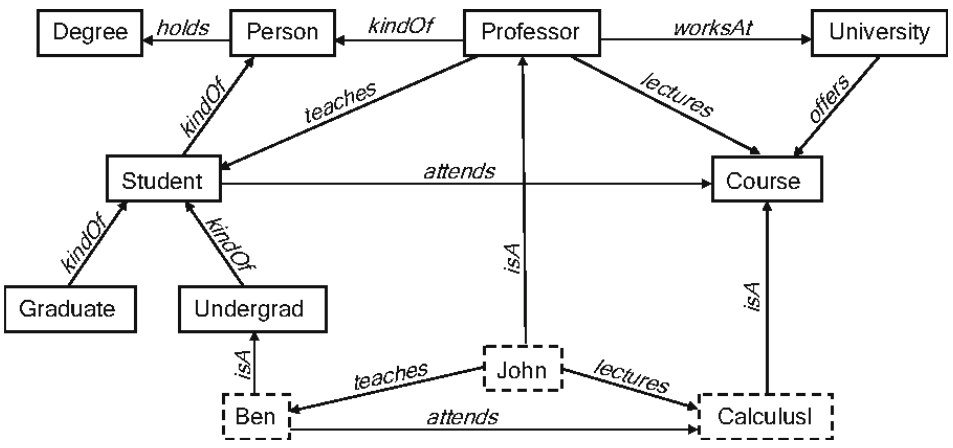
Whatlogic do we needforMesh?

•WhataboutGO?
F–PsychiatryandPsychologyF01–behavior and behavior mechanisms
F02–psychological phenomena and processes
F03–mental disorders

What's inside an ontology?

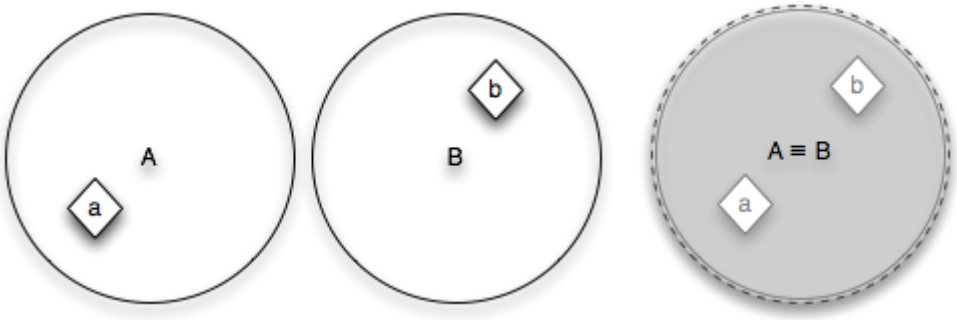
Classes + class-hierarchy
instances
slots/values
inheritance
•restrictions on slots (type, cardinality)
•properties of slots (symm., trans., …)
•relations between classes (disjoint, covers)
•reasoning tasks: classification, subsumption

Operators for ontologies?
Ontology/ Knowledge Graph

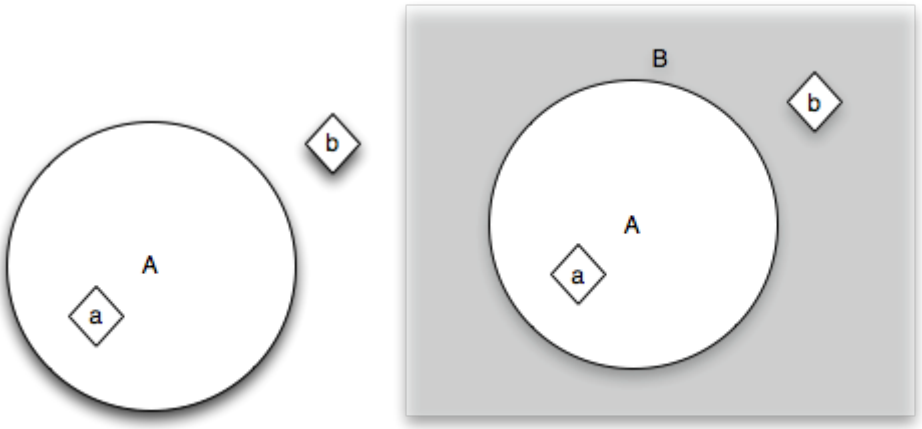


Class Axioms -Equivalence and Complement

Equivalent Classes contain the same individuals, and have the same definition

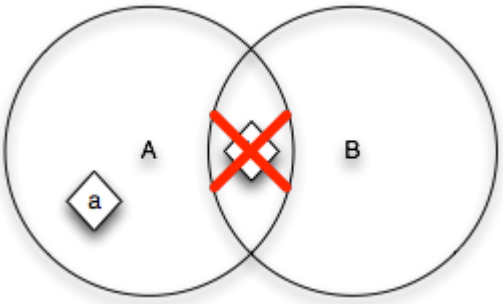


The **complement** of a class contains all individuals that are not in the class

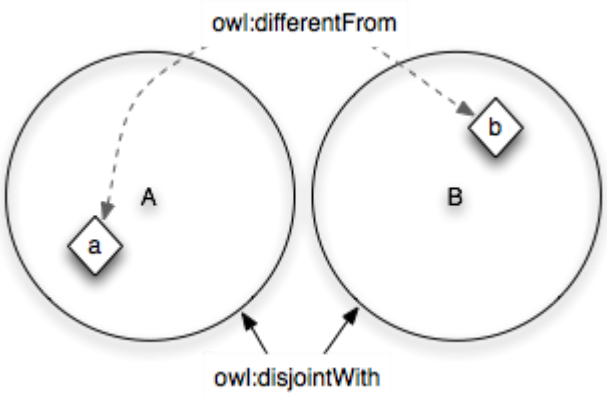


Class Axioms -Disjointness

Disjoint Classes do not contain the same individuals

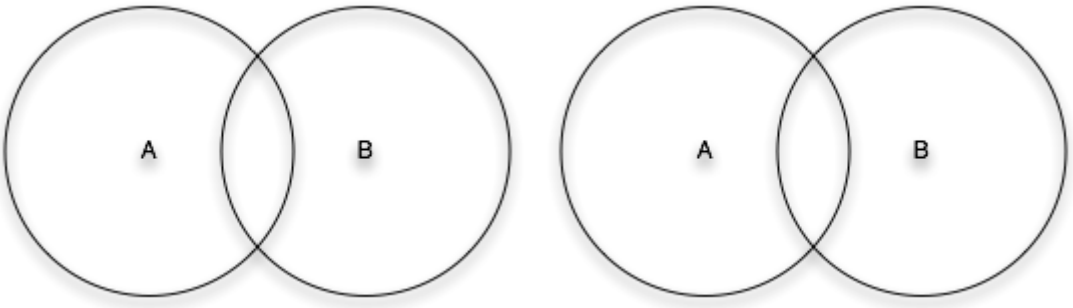


The **complement and disjointness allow** us to infer that individuals are different.

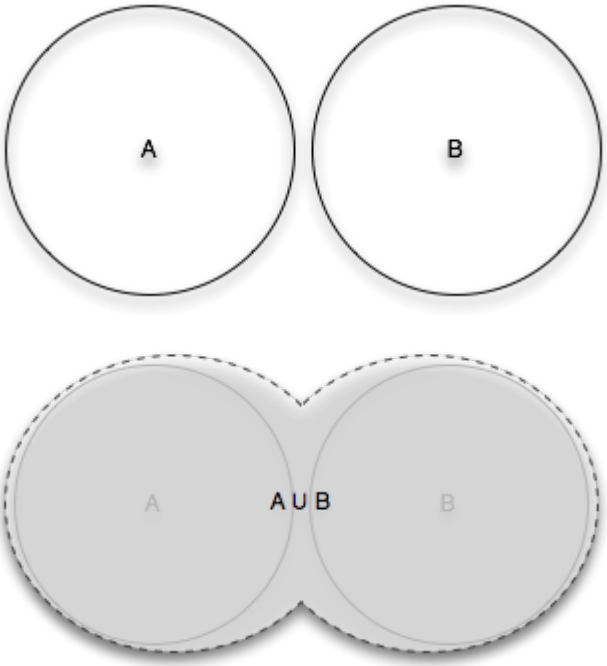


Class Axioms -Union and Disjoint Union

The Union Contains all individuals that belong to the classes of the union

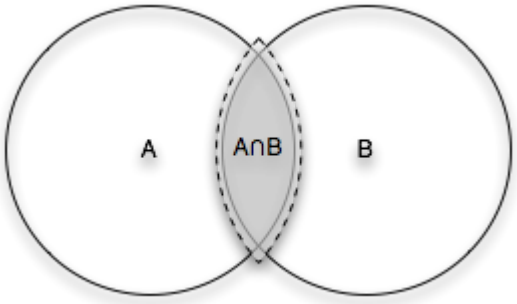


A **disjoint union** is a union of mutually **disjoint classes**

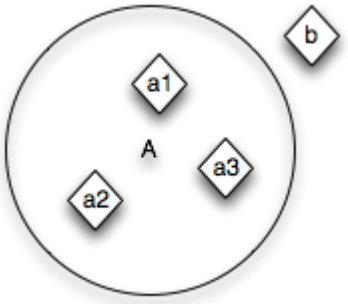


Class Axioms -Intersection and Enumeration

The Intersection contains individuals that each belong to **both classes**



You can **enumerate all** members of a class



Class Restrictions-*Some/all*(existential, universal)

All members of a class have at least **some**value from the specified class
"**NobelPrizeWinners**is the class of **Persons**who have **won**a **NobelPrize**"

All members of a class have only values from one specified class

A **NobelPrize**is theclass of **Prizes**thatonlyhave been won by**NobelPrizeWinners**

That is not the same as: Every **NobelPrizeWinner**only **won NobelPrize**s

Necessary and Sufficient Conditions

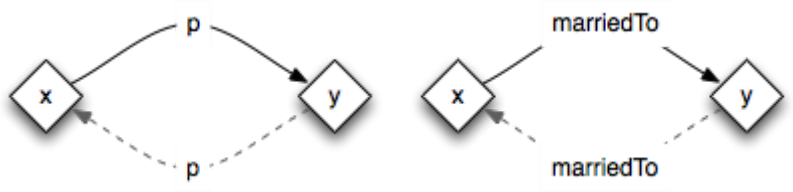
All members of a class must also be members of anotherClass (it is **necessary for a** NobelPrizeWinnertohave been nominated for Nobel Price).
"**NobelPrizeWinners**is exactly the class of all **Persons**who have **won**a **NobelPrize**"

The class is precisely specified by some other class (it is a **necessary and sufficient** condition).
"**NobelPrizeWinners**is the subset of all **Persons**who were **nominatedFor**a **NobelPrize**"

Property Types -Symmetric Property

Used to specify that a property always holds in both directions

if	type(p,SymmetricProperty) **and** p(x,y)
then	p(y,x)

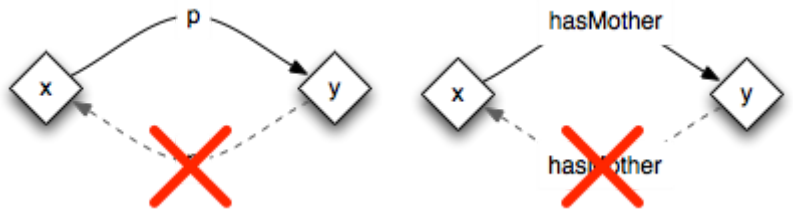


Property Types -Asymmetric Property

Used to specify that a property **never**holds in both directions

if type(p,AsymmetricProperty) **and**p(x,y)

then p(y,x) **is inconsistent**

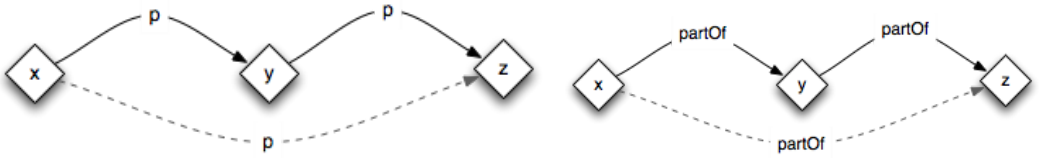


Property Types -Transitive Property

Used to specify that a property **propagates**over itself

if type(p,TransitiveProperty) **and**p(x,y) **and**p(y,z)

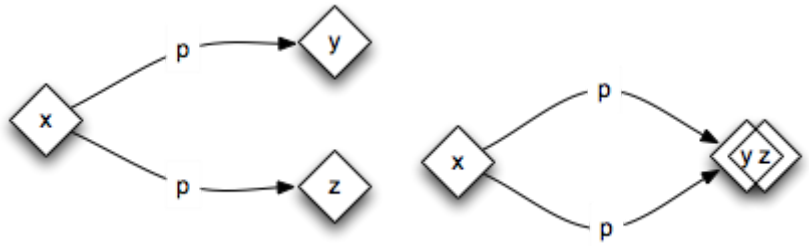
then p(x,z)



Property Types -Functional Property
Used to specify that a property has **only one**value for any particular instance

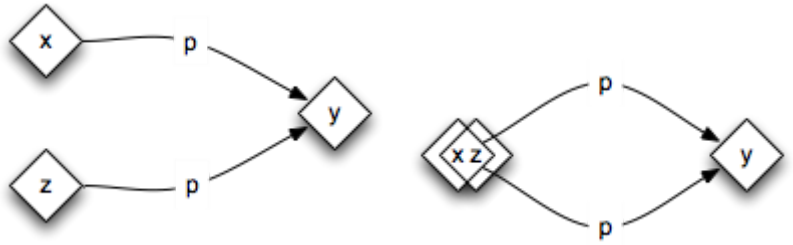
if type(p,FunctionalProperty) **and**p(x,y) **and**p(x,z)

then y = z



Property Types -Inverse Functional Property
Used to specify that a value for the property **uniquely identifies**an instance

if type(p,InverseFunctionalProperty) **and**p(x,y) **and**p(z,y)
then x = z

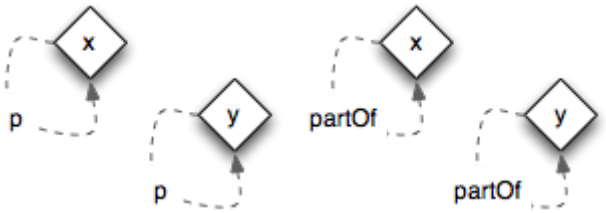


Property Types -Reflexive Property

Used to specify that **every individual**is **always**related to **itself**by that property

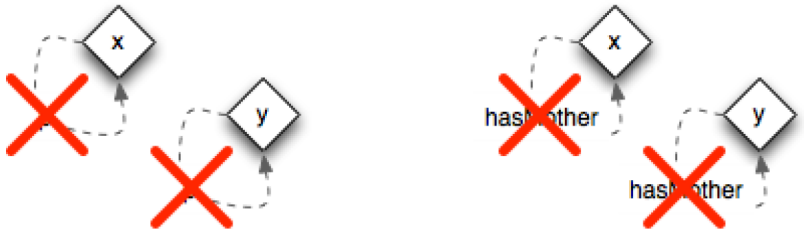
if type(p,ReflexiveProperty)

then ∀x p(x,x)



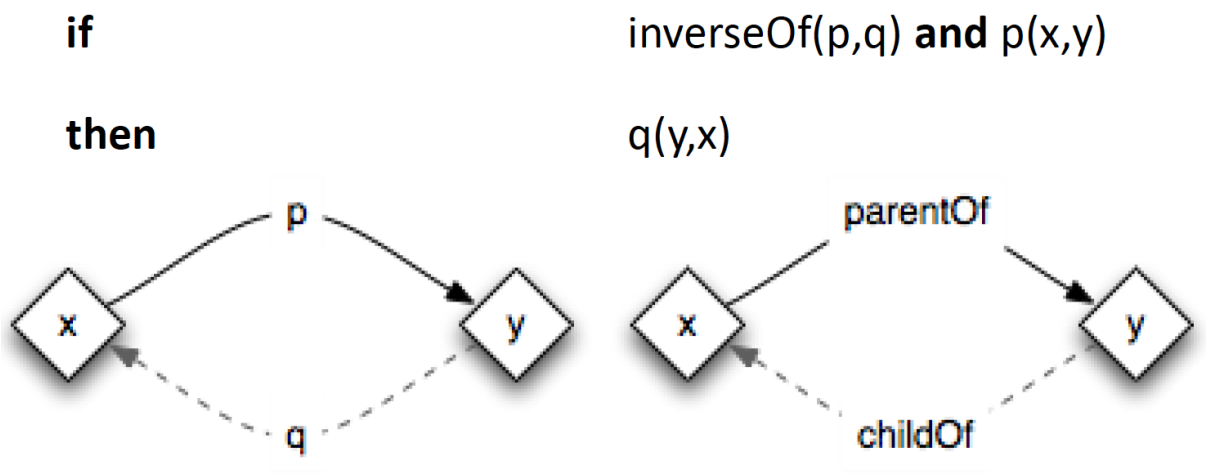
if type(p,IrreflexiveProperty) **and** p(x,x)

then p(x,x) **is inconsistent**



Property Axioms -Inverse Property

Used to specify that one property is **always**the inverse of another property



Summary & what comes now?

•We have seen

•Ontologies: a **hierarchical representation** of types ("**classes**"),with **instances**("individuals") that belong to classes,and **properties between** the individuals.

•A number of realistic examples of ontologies in use

•These ontologies are easy to read by people

•**Now:**But can we also make a **logics for** ontologies,so that computers can reason automatically?