Limitations of First-Order Logic

- FOL is very expressive, but...consider how to translate these:
  - "*most* students graduate in 4 years"
    - $\forall x$ student(x) $\rightarrow$ duration(undergrad(x))$\leq$years(4) (all???)
  - "*only a few* students switch majors"
    - $\exists$s,m1,m2,t1,t2 student(s)^major(s,m1,t1)$\wedge$major(s,m2,t2) $\wedge$m1$\neq$m2 $\wedge$ t1$\neq$t2 (exists???)
  - "all birds can fly, *except* penguins, stuffed birds, plastic birds, birds with broken wings..."

- The problem(s) with FOL involve expressing:
  - default rules & exceptions
  - degrees of truth
  - strength of rules

Limitation of First-Order Logic

- FOL is not good at handling exceptions
  - universal quantifier means ALL; can't say "most" birds fly
  - $\forall x\ bird(x) \rightarrow flies(x)$
  - asserting $bird(opus) \wedge \neg flies(opus)$ in the KB would cause it to be inconsistent
  - FOL is *monotonic*: if $\alpha \models \beta$, then $\alpha \wedge \omega \models \beta$
    - adding new facts does not undo conclusions
- we could say: $\forall x\ bird(x) \wedge \neg penguin(x) \rightarrow flies(x)$
- but we can't enumerate all possible exceptions
  - what about a robin with a broken wing?
  - what about birds that are made out of plastic?
  - what about Big Bird?

- Uncertainty in reasoning about actions:
  - If a gun is loaded and you pull the trigger, the gun will fire, right?
  - ...unless it is a toy gun
  - ...unless it is defective
  - ...unless it is underwater
  - ...unless the barrel is filled with concrete

Possible Solutions

- Add rule *strengths* or *priorities*
  - common in early Expert Systems
  - ...an old ad-hoc approach (with unclear semantics)
  - penguin(x) $\rightarrow_{0.9}$ $\neg$flies(x)
  - bird(x) $\rightarrow_{0.5}$ flies(x)

Solutions

- Default Logic/Non-monotonic logics
- Closed-World Assumption and Negation-as-failure in PROLOG
- Semantic Networks
- Fuzzy Logic
- Bayesian Probability

Non-monotonic Logics

- allow retractions later (popular for truth-maintenance systems)
- "birds fly", "penguins are birds that don't fly"
  - $\forall x\ bird(x) \rightarrow fly(x)$
  - $\forall x\ penguin(x) \rightarrow bird(x)$, $\forall x\ penguin(x) \rightarrow \neg fly(x)$
  - {bird(tweety), bird(opus)} |= fly(opus)
  - later, add that opus is a penguin, change inference
  - penguin(opus) |= ¬fly(opus)
- Definition: A logic is *monotonic* if everything that is entailed by a set of sentences α is entailed by any superset of sentences $\alpha \wedge \beta$
  - opus example is *non-monotonic*

Default Logic

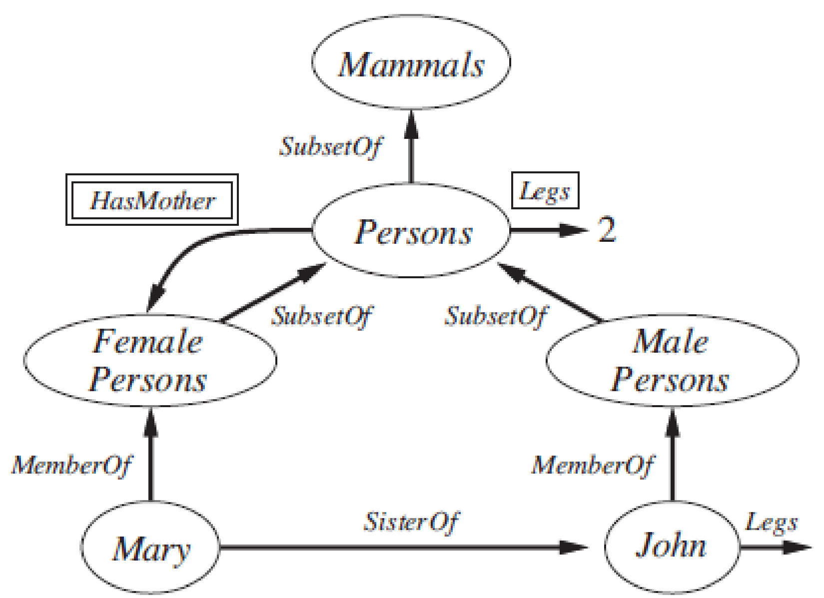
- example syntax of a default rule:
  - bird(x): fly(x) / fly(x)   or   bird(x) ≻ fly(x)
  - analogous to $\forall$x bird(x) → fly(x) , but allows exceptions
  - meaning: "if PREMISE is satisfied and it is not inconsistent to believe CONSEQUENT, then CONSEQUENT"
  - {bird(tweety),bird(opus),¬fly(opus), bird(x): fly(x) / fly(x) } |={fly(tweet),¬fly(opus)}
- requires *fixed-point semantics* (different model theory and inference procedures)

Circumscription

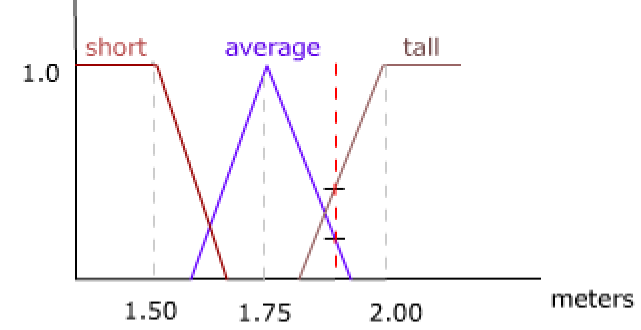
- an alternative approach to default logic
- add *abnormal* predicates to rules
    - $\forall x\ bird(x) \wedge \neg abnormal_1(x) \rightarrow fly(x)$
    - $\forall x\ penguin(x)\ \wedge \neg abnormal_2(x) \rightarrow bird(x)$
    - $\forall x\ penguin(x)\ \wedge \neg abnormal_3(x) \rightarrow \neg fly(x)$
- algorithm: minimize the number of *abnormals* needed to make the KB consistent
    - {bird(tweety),fly(tweety),bird(opus),penguin(opus), ¬fly(opus)} is INCONSISTENT
    - {bird(tweety),fly(tweety),bird(opus),penguin(opus), ¬fly(opus), $abnormal_1$(opus)} is CONSISTENT

Semantic Networks

- graphical representation of knowledge
- nodes, slots, edges, "isa" links
- procedural mechanism for answering queries
  - follow links
  - different than formal definition of "entailment"
- <u>inheritance</u>
  - can override defaults



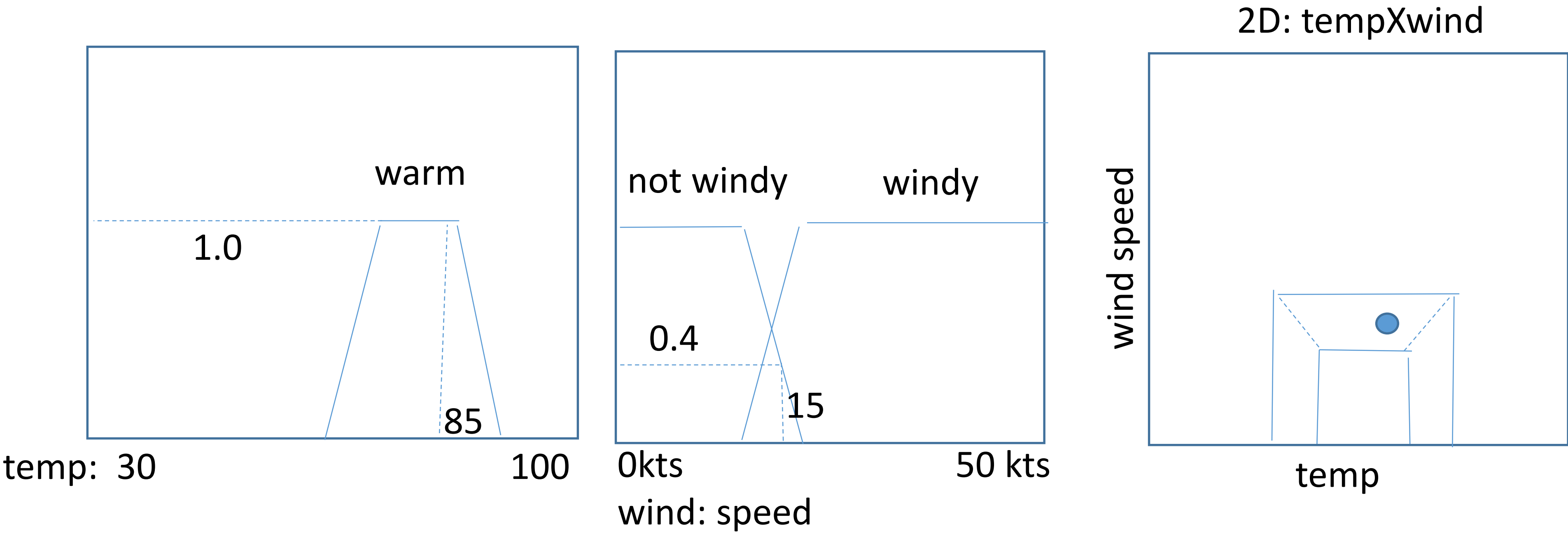
Fuzzy Logic



- some expressions involve "degrees" of truth, like "John is tall"
- membership functions
- "most students with high SATs have high GPAs"
- inference by computing with membership funcs.
  - "only days that are warm and not windy are good for playing frisbee"
  - suppose today is 85 and the wind is 15 kts NE
  - T(A^B) = min(T(A),T(B))
  - T(AvB) = max(T(A),T(B))
- popular for control applications (like thermostats...)

Fuzzy Logic

- doing inference in FL involve computing truncation (min) and intersection with membership functions
    - i.e. to evaluate satisfaction of antecedents of a rule
    - (temp is warm) and (wind is not-windy) -> playFrisbee



Handling Defaults in Prolog

- $\forall x\ bird(x) \land \neg penguin(x) \rightarrow flies(x)$
  - bird(tweety)
  - bird(woodstock)
  - bird(opus) penguin(opus)

initial KB has 4 facts

Handling Defaults in Prolog

- $\forall x\ bird(x) \wedge \neg penguin(x) \rightarrow flies(x)$
  - bird(tweety)
  - bird(woodstock)
  - bird(opus) penguin(opus)

initial KB has 4 facts

- $\forall x\ bird(x) \wedge \neg penguin(x) \rightarrow flies(x)$
  - bird(tweety) ¬penguin(tweety)
  - bird(woodstock) ¬penguin(woodstock)
  - bird(opus), penguin(opus)

the problem here is that, if you add a qualifying condition like ¬*penguin* to a rule in FOL, then you have to explicitly say whether
every individual is a penguin or not (which is not scalable to large KBs)

Handling Defaults in Prolog

- $\forall x$ bird(x)∧¬penguin(x) →flies(x)
  - bird(tweety)
  - bird(woodstock)
  - bird(opus)   penguin(opus)

initial KB has 4 facts

- $\forall x$ bird(x)∧¬penguin(x) →flies(x)
  - bird(tweety)   ¬penguin(tweety)
  - bird(woodstock)   ¬penguin(woodstock)
  - bird(opus),   penguin(opus)

the problem here is that, if you add a qualifying condition like ¬penguin to a rule in FOL, then you have to explicitly say whether every individual is a penguin or not (which is not scalable to large KBs)

- $\forall x$ bird(x)∧¬penguin(x)^ ¬emu(x) →flies(x)
  - bird(tweety)   ¬penguin(tweety)   ¬emu(tweety)
  - bird(woodstock)   ¬penguin(woodstock)   ¬emu(woodstock)
  - bird(opus),   penguin(opus)   ¬emu(opus)

if we add another condition like ¬*emu*, then we have explicitly identify all the non-emus

- other examples:
  - a football player is eligible to play in a game, *unless* they have not passed a physical, or are on academic probation
  - an item is on sale (50% off), *unless* it is already discounted
  - a house can be sold, as long as it does not have a lien on it
  - fish is a healthy option for protein, *unless* it has high mercury levels (shark, swordfish, orange roughy...)

  - in all these cases, you would have to add a negative antecedent to a FOL rule, but then have to assert things like ¬academic_probation(<player>) for all players, or ¬highMg(trout), ¬highMg(bass), ¬highMg(catfish)...

A^B -> C ... -Av-BvC

A^ -B -> C ... -A v B vC

Handling Defaults in Prolog

- Potential problems:
  - 1) can't assert negative facts, e.g. ¬penguin(tweety)
  - 2) can't have negative literals as antecedents in definite clauses

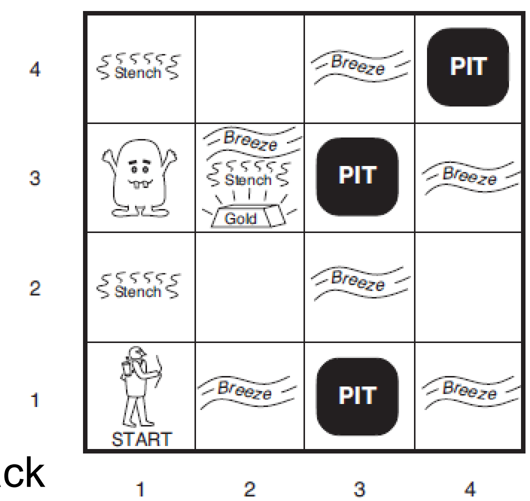
```
dog(fido).
dog(snoopy).
canary(tweety).
canary(woodstock).
penguin(opus).
animal(X) :- mammal(X).
animal(X) :- bird(X).
dangerous_animal(X) :- animal(X),has_sharp_teeth(X),aggressive(X).
```

Closed-World Assumption (CWA) in PROLOG

- every fact that is not explicitly asserted (or provable) is *assumed to be false*
- can include negated antecedents in rules ("\+" = not)

```
stench(1,2). // col,row
stench(2,3).
stench(1,4).
wumpus_free(X,Y) :- room(X,Y),adjacent(X,Y,P,Q),\+ stench(P,Q).

?- wumpus_free(1,3).
No
?- wumpus_free(2,2).
Yes (because no stench in 2,1)
```



this prolog rule is equivalent to:

$\forall x,y,p,q\ room(x,y) \wedge adjacent(x,y,p,q) \wedge \neg stench(p,q) \rightarrow wumpus\_free(x,y)$

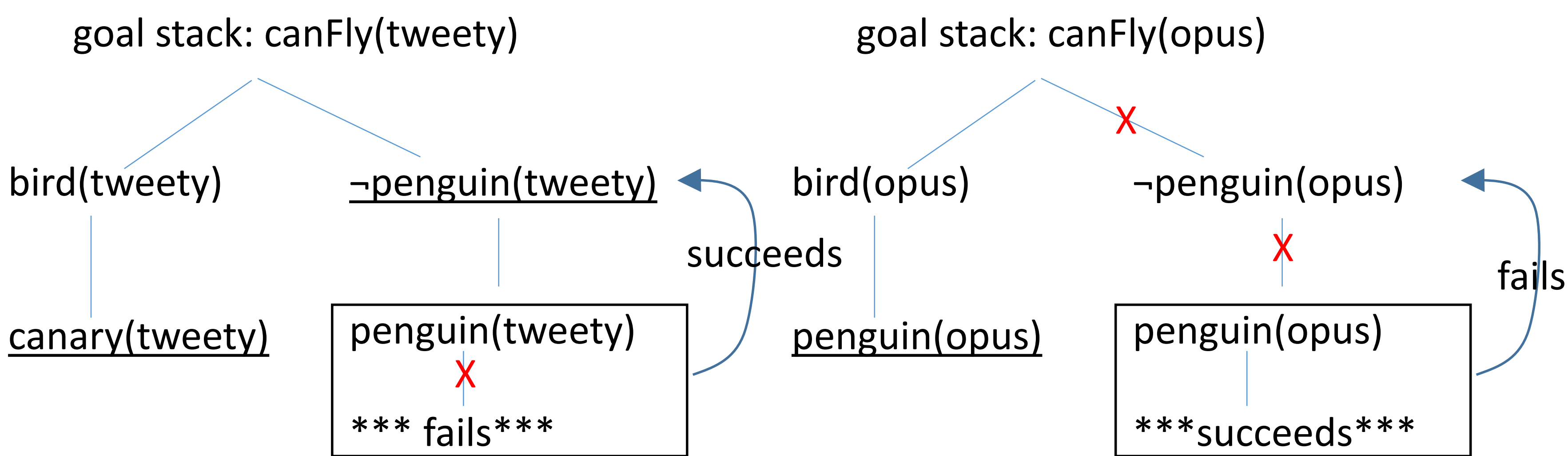
which is not a definite clause (because CNF has 2 positive literals), so technically, we could not do back-chaining; can't put ¬stench on goal stack

- using CWA for default reasoning

```
bird(X) :- canary(X).
bird(X) :- penquin(X).
canary(tweety).
penquin(opus).
canFly(X) :- bird(X),\+ penquin(X).
```

```
?- bird(tweety). Yes
?- canFly(tweety). Yes
?- bird(opus). Yes
?- canFly(opus). No
```

- how is negation-as-failure implemented?
  - modify back-chaining to handle negative antecedents
  - when trying to prove ¬P(X) on goal stack, try proving P(X) and if fail then ¬P(X) succeeds



Probability

- an alternative route to encoding default rules like "most birds fly" is to quantify it using probability, *p(fly|bird)*=0.95
- probabilistic reasoning has had a major impact on AI over the years
  - conferences and journals on UAI (Uncertainty in AI)
- probabilistic models has led to major algorithms like:
  - Hidden Markov Models (applications to speech, genomics...)
  - SLAM (simultaneous mapping and localization) for robotics
  - Bayesian networks/graphical models (as knowledge bases)
  - Kalman filters, ICA, POMPDs, ...
  - Reinforcement Learning

Probability

- encode knowledge in the form of *prior* probabilities and *conditional* probabilities
  - P(x speaks portugese)=0.012
  - P(x is from Brazil)=0.007
  - P(x speaks portugese|x is from Brazil)=0.9
  - P(x flies|x is a bird)=0.9 (?)

prior probs (the first two)

conditional probs (the last two)

- inference is done by calculating *posterior* probabilities given evidence (using Bayes' Rule)
  - compute P(cavity | toothache, flossing, dental history, recent consumption of candy...)
  - compute P(fed will raise interest rate | unemployment=5%, inflation=0.5%, GDP=2%, recent geopolitical events...)

read about axioms of probability, etc. in Ch. 12
(you can skip sec 12.6 and 12.7)

Bayes' Rule

- product rule : *joint prob* P(A,B) = P(A|B)*P(B)
  - P(A|B) is read as "probability of A *given* B"
  - *in general, P(A,B)≠P(A)*P(B) (*unless A and B are independent)
- Bayes' Rule: convert between causal and diagnostic

$$P(H \mid E) = \frac{P(E \mid H) \cdot P(H)}{P(E)}$$

H = hypothesis (cause, disease)
E = evidence (effect, symptoms)

- joint probabilities: P(E,H), priors: P(H)
- conditional probabilities play role of "rules"
  - people with a toothache are likely to have a cavity
  - p(cavity|toothache) = 0.6

Causal vs. diagnostic knowledge

- *causal*: P(x has a toothache|x has a cavity)=0.9
- *diagnostic*: P(x has a cavity|x has a toothache)=0.5

- typically it is easier to articulate knowledge in the causal direction, but we often want to use it in a diagnostic way to make inferences from observations

- Joint probability table (JPT)
  - you can calculate answer to any question from JPT
  - the problem is there are exponential # of entries (2^N, where N is the number of binary random variables)

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

P(¬*cavity* | *toothache*) = ?

- Joint probability table (JPT)
  - you can calculate answer to any question from JPT
  - the problem is there are exponential # of entries (2^N, where N is the number of binary random variables)

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

P(¬*cavity* | *toothache*) = P(¬*cavity* ∧ *toothache*) / P(*toothache*)

= 0.016+0.064 /

(0.108 + 0.012 + 0.016 + 0.064)

= 0.4

- Joint probability table (JPT)
  - you can calculate answer to any question from JPT
  - the problem is there are exponential # of entries (2^N, where N is the number of binary random variables)

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	.108	.012	.072	.008
¬ cavity	.016	.064	.144	.576

P(¬*cavity* | *toothache*) = P(¬*cavity* ∧ *toothache*) / P(*toothache*)

= 0.016+0.064 /

(0.108 + 0.012 + 0.016 + 0.064)

= 0.4

Conditional Independence

- Applying Bayes' Rule in larger domains has a <u>scalability</u> problem
  - the size of the JPT grows exponentially with the number of variables (2^n for n variables)
- Solution to reduce complexity:
  - employ the <u>Independence Assumption</u>
- Most variables are not strictly independent; most variables are at least partially correlated (but which is cause and which is effect?).
- However, many variables are *conditionally* independent.

A and B are *conditionally independent* <u>given C</u> if:
$P(A,B|C) = P(A|C)P(B|C)$, or equivalently
$P(A|B,C) = P(A|C)$

Conditional Independence

If I have a cavity, the probability that the probe catches in it doesn't depend on whether I have a toothache:

(1) $P(catch|toothache, cavity) = P(catch|cavity)$

The same independence holds if I haven't got a cavity:

(2) $P(catch|toothache, \neg cavity) = P(catch|\neg cavity)$

$Catch$ is conditionally independent of $Toothache$ given $Cavity$:

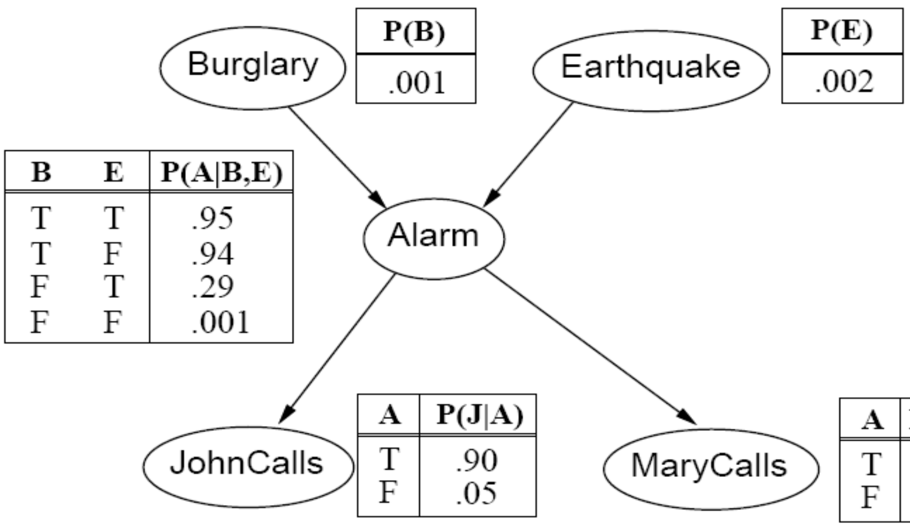
$\mathbf{P}(Catch|Toothache, Cavity) = \mathbf{P}(Catch|Cavity)$

- conditional independence gives us an efficient way to combine evidence
  - consider P(Cav|toothache,catch)
  - using Bayes' Rule:
    - P(Cav|toothache,catch) $\propto$ P(toothache^catch|Cav)*P(Cav)
    - this requires a mini JPT for all combinations of evidence
  - assuming toothache is conditionally independent of catch given Cavity:
    - P(toothache^catch|Cav) = P(toothache|Cav)*P(catch|Cav)
    - therefore...

    P(Cav|toothache,catch) $\propto$ P(toothache|Cav)*P(catch|Cav)*P(Cav)

Bayesian Networks

- graphical models where edges represent conditional probabilities
- popular for modern AI systems (expert systems)
  - important for handling uncertainty



P(B)
.001

P(E)
.002

B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

$$P(x_1, \ldots, x_n) = \Pi_{i=1}^{n} P(x_i | parents(X_i))$$

$$\mathbf{P}(B, e, a, j, m) = \mathbf{P}(B)P(e)\mathbf{P}(a|B, e)P(j|a)P(m|a)$$

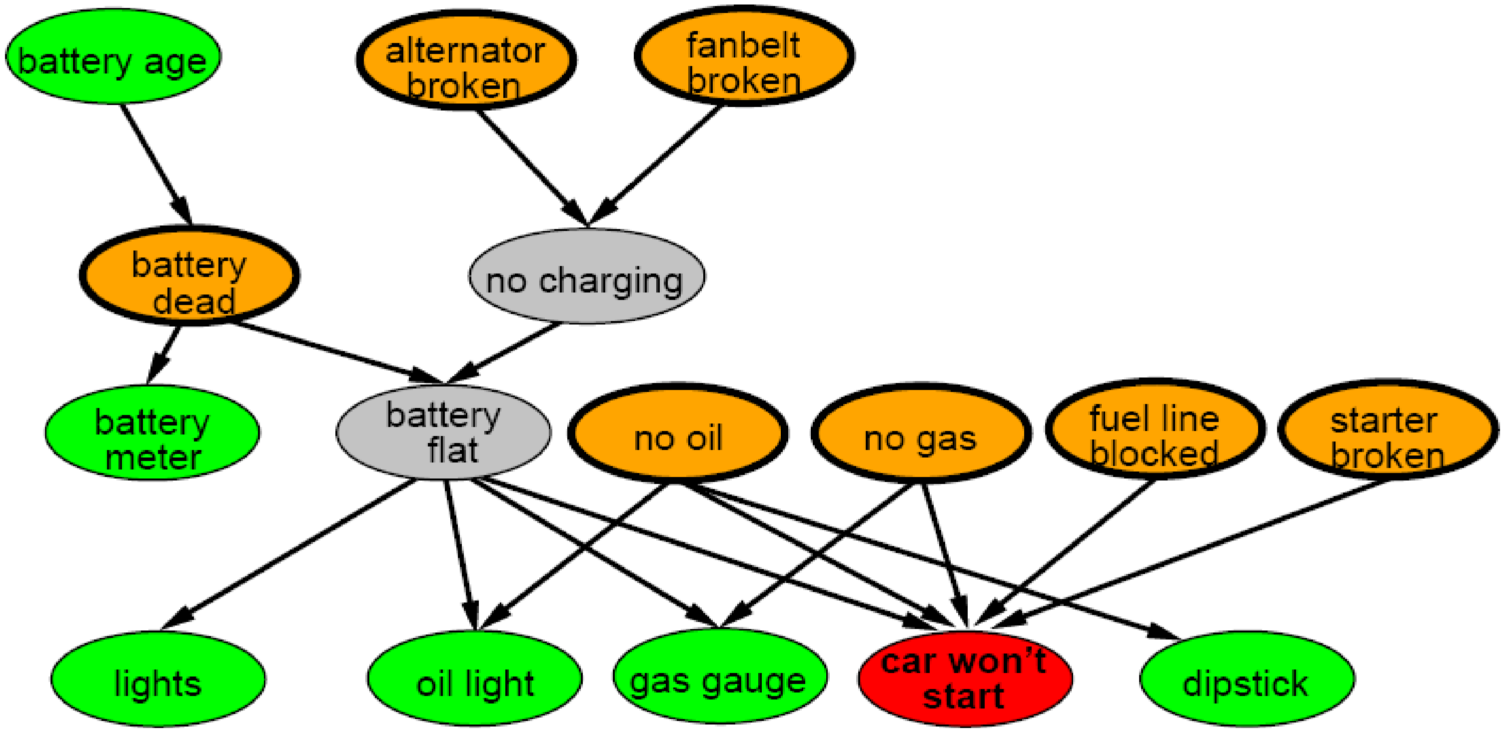
$$P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$$

$$\begin{aligned} &= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e) \\ &= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998 \\ &\approx 0.00063 \end{aligned}$$

Initial evidence: car won't start

Testable variables (green), "broken, so fix it" variables (orange)

Hidden variables (gray) ensure sparse structure, reduce parameters



- Many modern knowledge-based systems are based on probabilistic inference
  - including Bayesian networks, Hidden Markov Models, (HMMs), Markov Decision Problems (MDPs)
  - example: Bayesian networks are used for inferring user goals or help needs from actions like mouse clicks in an automated software help system (think 'Clippy')

  - Decision Theory combines utilities with probabilities of outcomes to decide actions to tak
- the challenge is capturing all the numbers needed for the prior and conditional probabilities
  - objectivists (frequentists) - probabilities represent outcomes of trials/experiments
  - subjectivists - probabilities are degrees of belief
- probability and statistics is at the core of many Machine Learning algorithms