Soft Constraints: Exponential Models

Factor graphs (undirected graphical models) and their connection to constraint programming

Soft constraint problems *(e.g, MAX-SAT)*

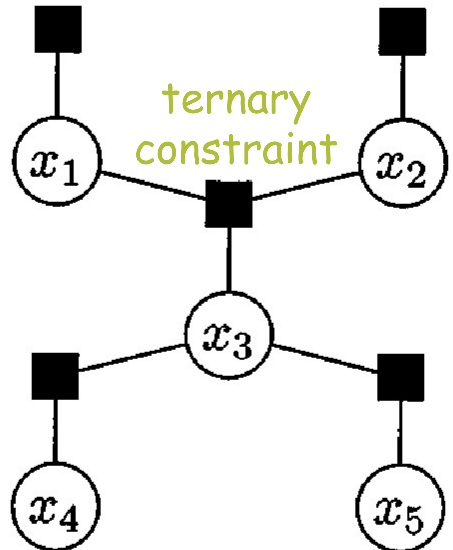
- Given
  - n variables
  - m constraints, over various subsets of variables
- Find
  - Assignment to the n variables that maximizes the number of satisfied constraints.

Soft constraint problems *(e.g, MAX-SAT)*

- Given
  - n variables
  - m constraints, over various subsets of variables
  - **m weights, one per constraint**
- Find
  - Assignment to the n variables that maximizes the **total weight** of the satisfied constraints.
    - Equivalently, minimizes total weight of violated constraints.

Draw problem structure as a "factor graph"

Each constraint ("factor") is a function of the values of its variables.



unary constraint

variable

binary constraint

variable

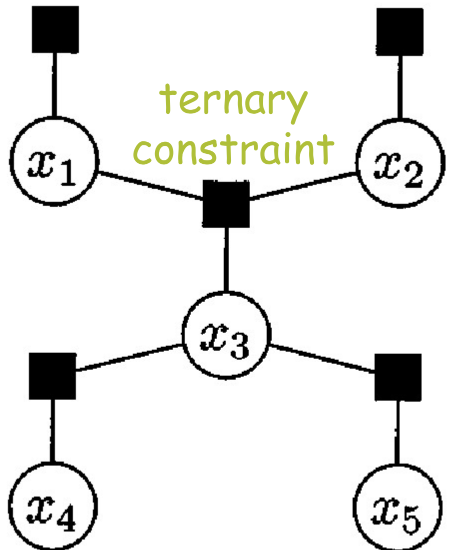
weight w ➔
if satisfied, factor=exp(w)
if violated, factor=1

- Measure goodness of an assignment by the **product of all the factors** (>= 0).
  - How can we reduce previous slide to this?
    - There, each constraint was either satisfied or not (simple case).
    - There, good score meant large total weight for satisfied constraints.

figure thanks to Brian Potetz

Draw problem structure as a "factor graph"

Each constraint ("factor") is a function of the values of its variables.



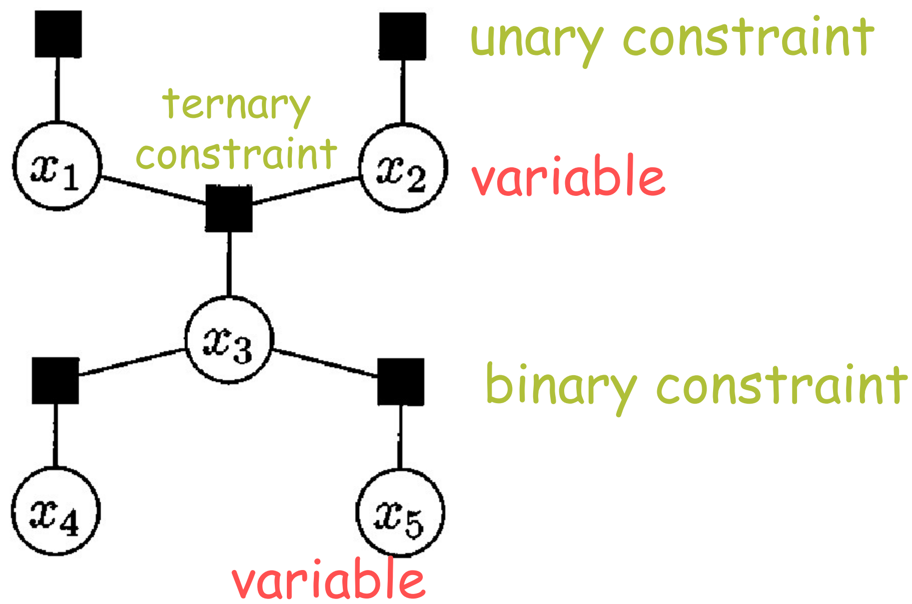
weight w ➔
if satisfied, factor=**1**
if violated, factor=**exp(-w)**

- Measure goodness of an assignment by the **product of all the factors** (>= 0).
  - How can we reduce previous slide to this?
    - There, each constraint was either satisfied or not (simple case).
    - There, good score meant **small** total weight for **violated** constraints.

figure thanks to Brian Potetz

Draw problem structure as a "factor graph"

Each constraint ("factor") is a function of the values of its variables.

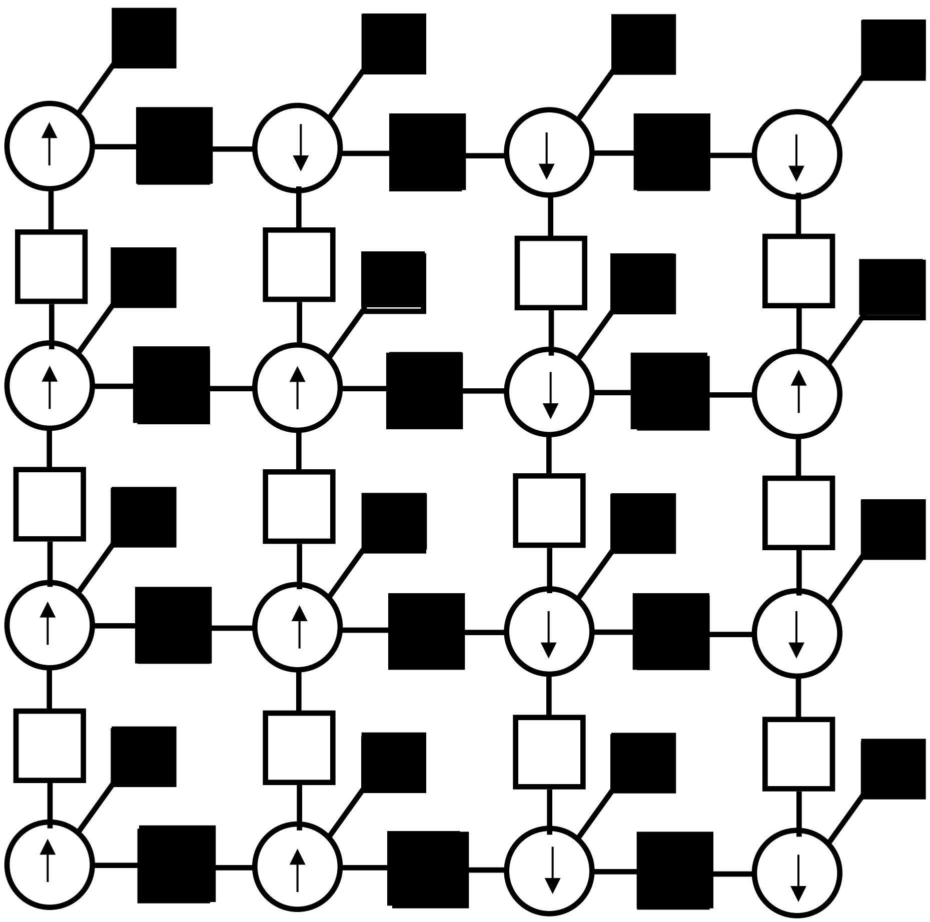


- Measure goodness of an assignment by the **product of all the factors** (>= 0).
- Models like this show up ***all the time***.

figure thanks to Brian Potetz

Example: Ising Model

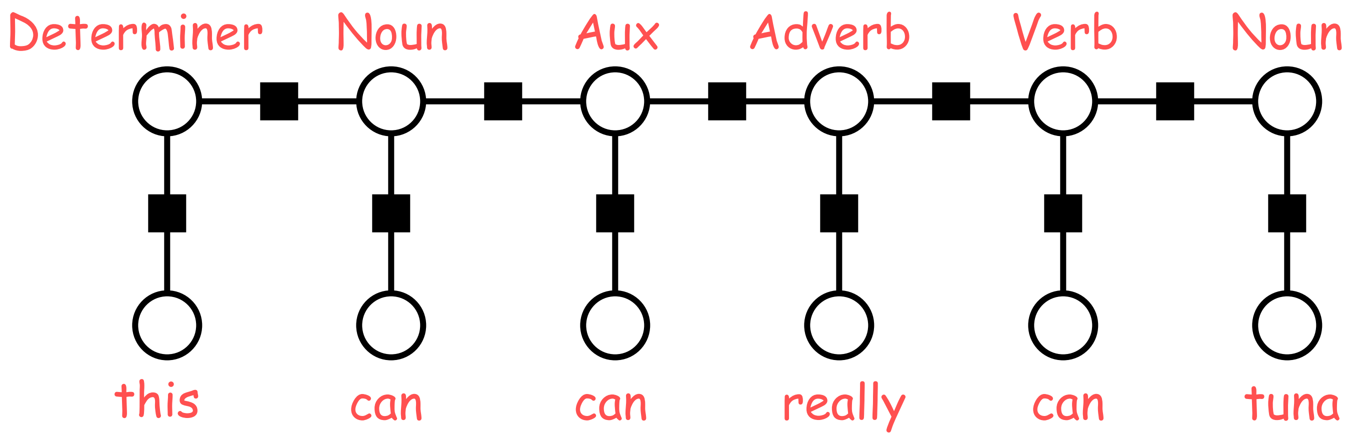
(soft version of graph coloring, on a grid graph)



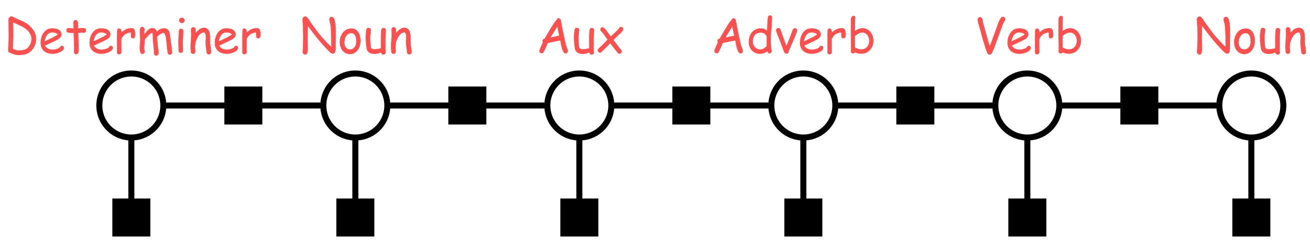
Model	Physics
Boolean vars	Magnetic polarity at points on the plane
Binary equality constraints	?
Unary constraints	?
MAX-SAT	?

figure thanks to ???

Example: Parts of speech
(or other sequence labeling problems)

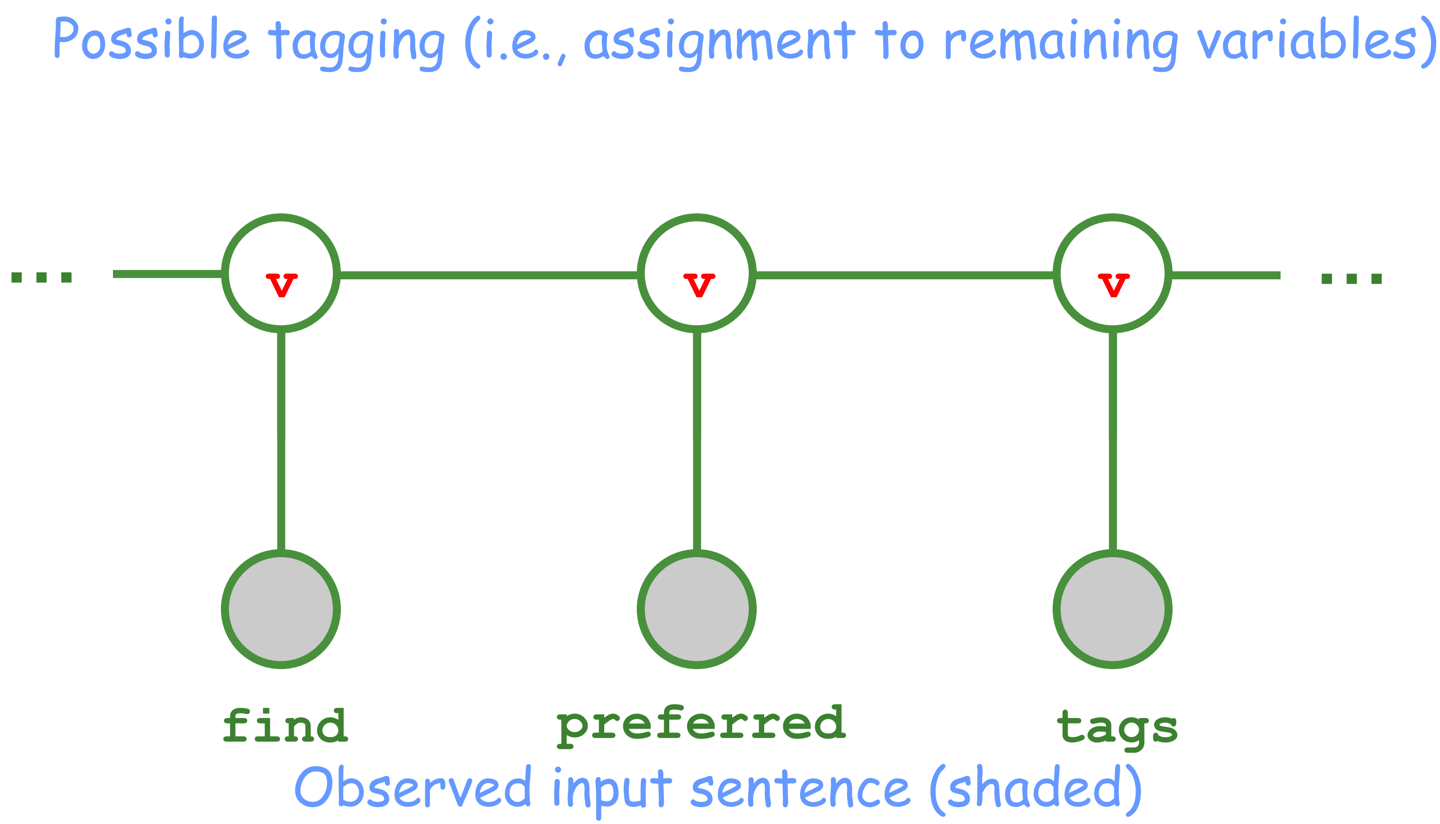


Or, if the input words are given, you can customize the factors to them:



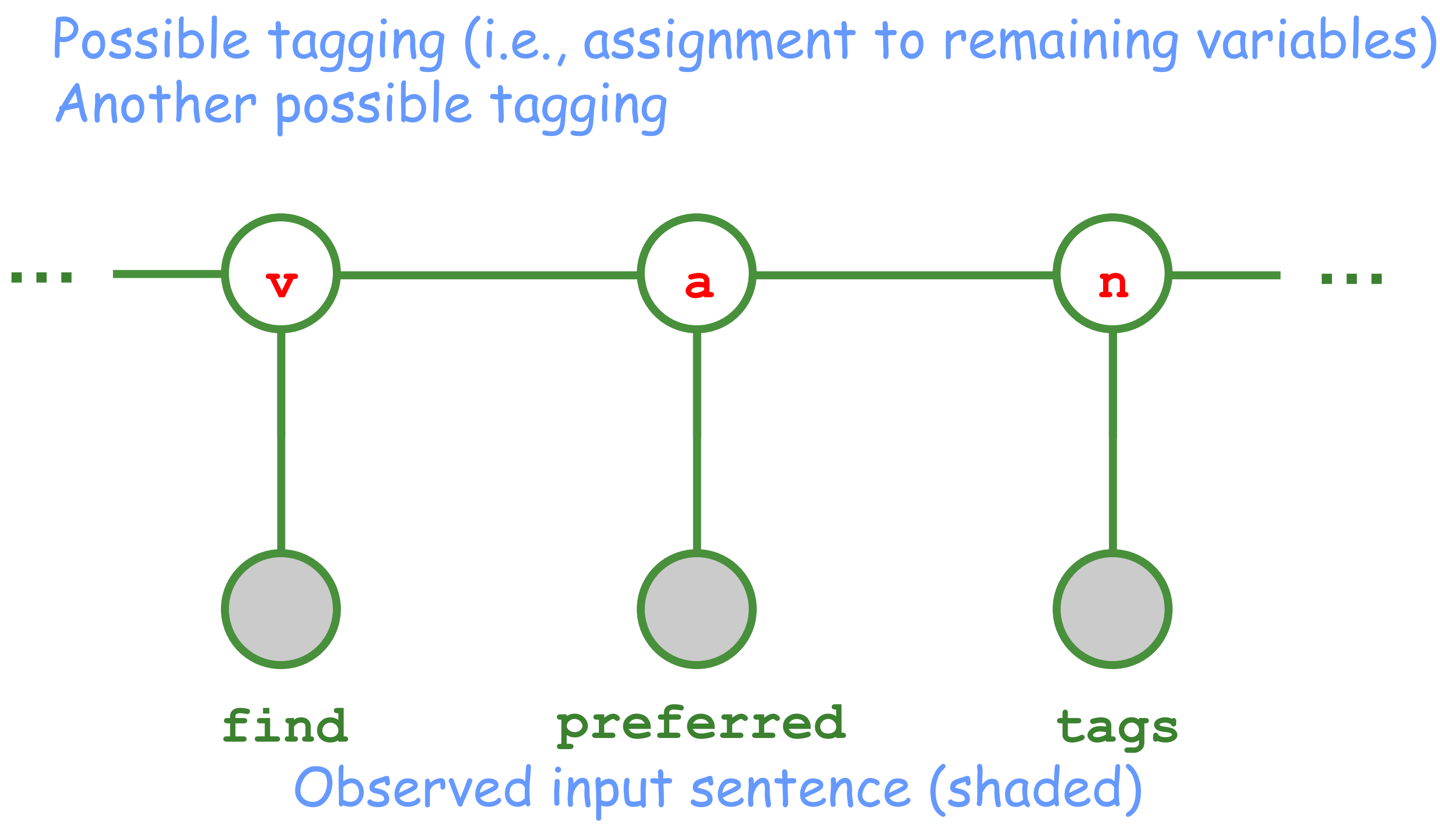
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging



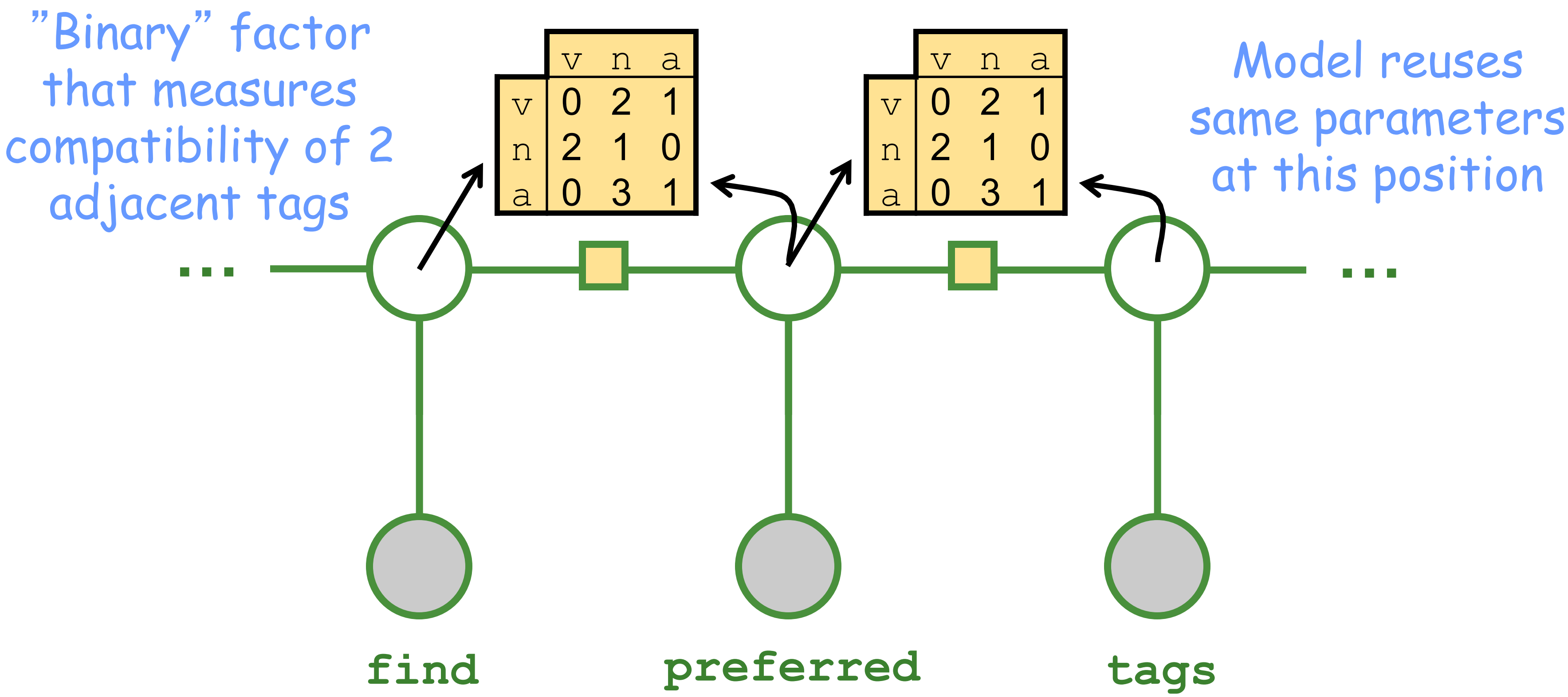
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging



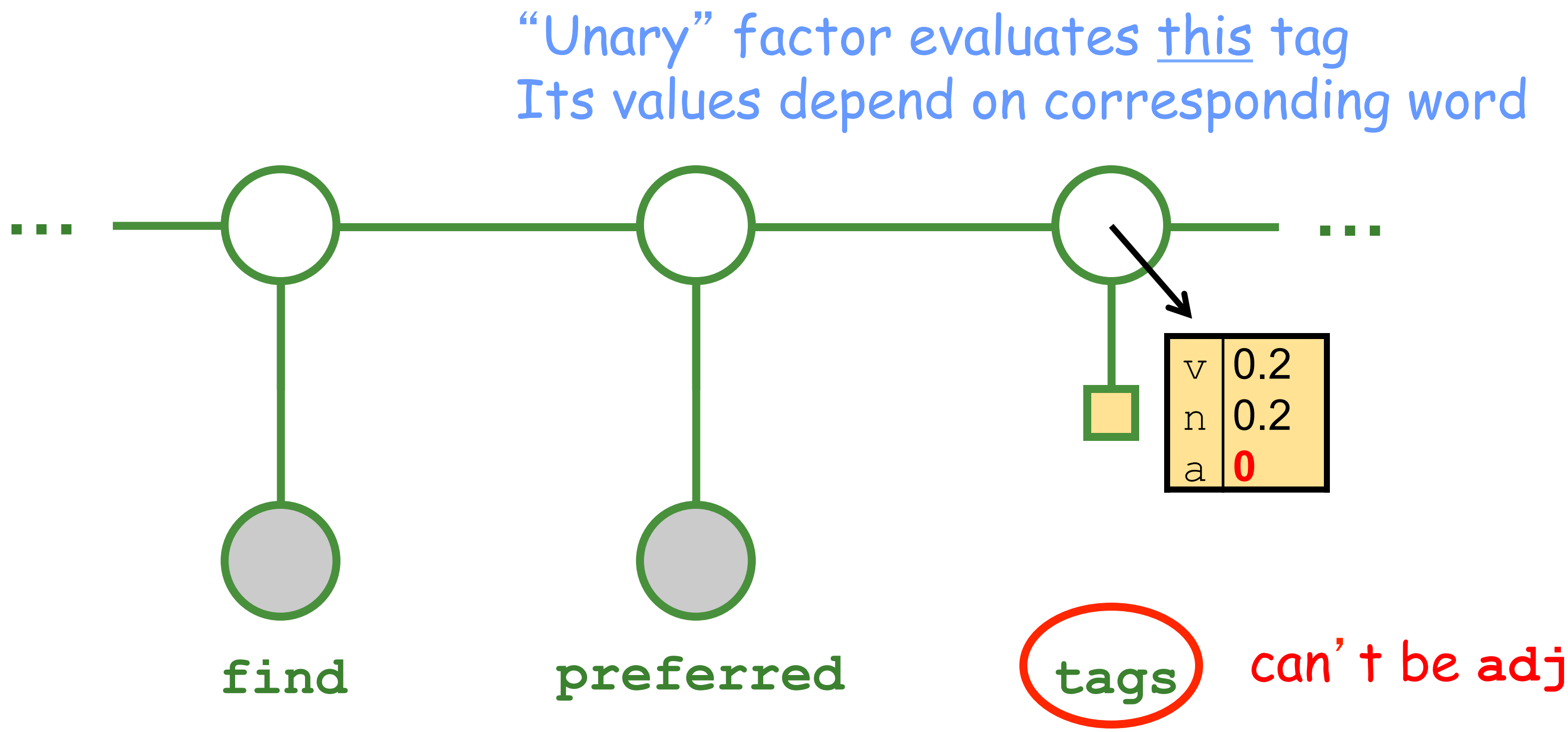
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging



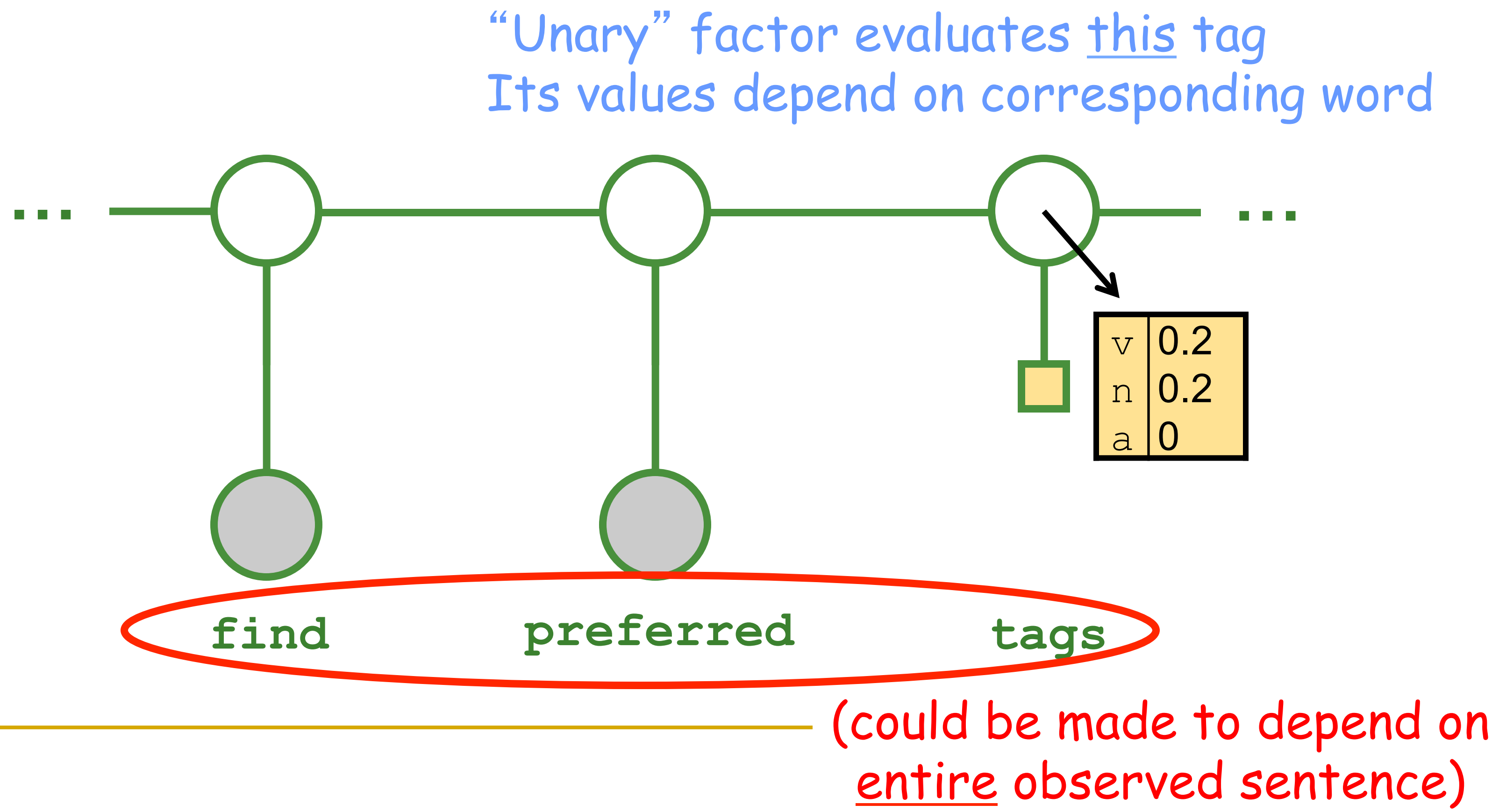
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging



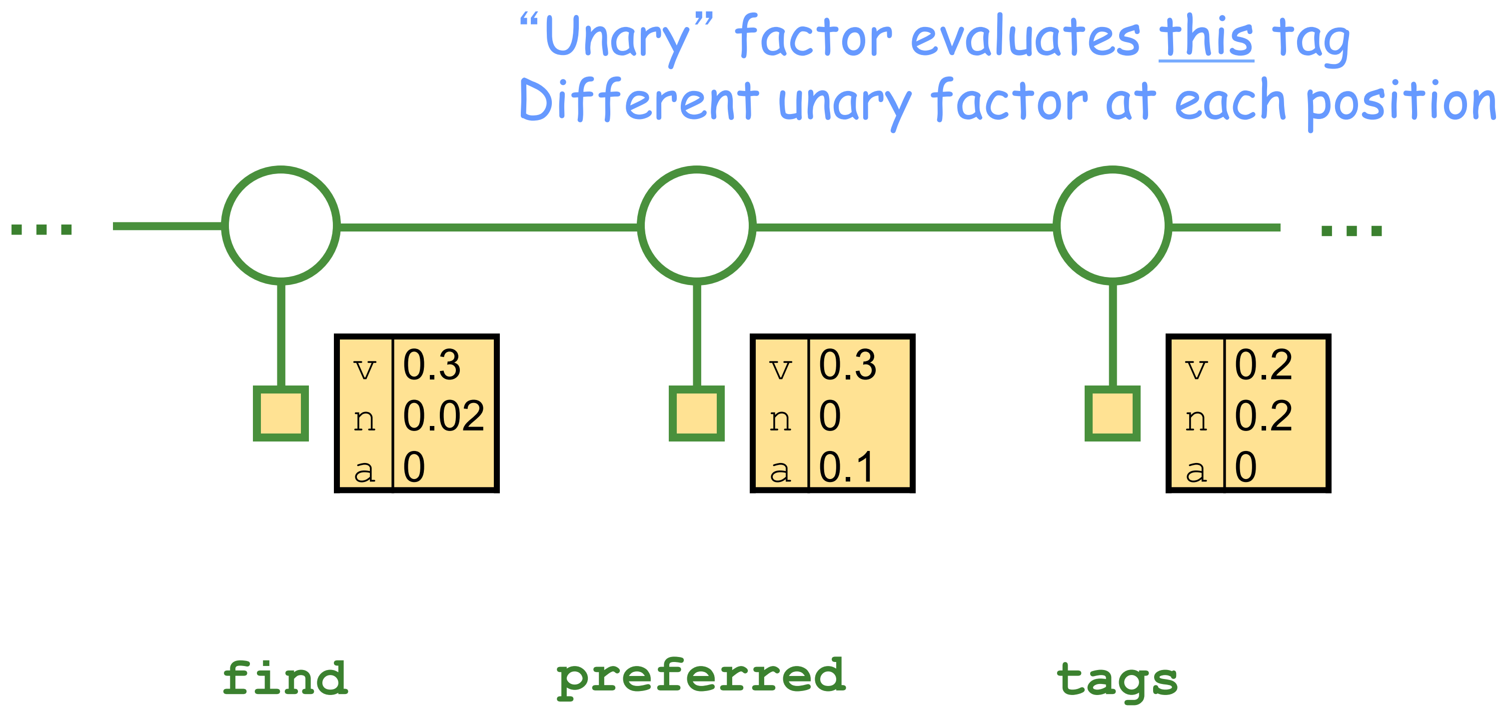
Local factors in a graphical model

- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging



Local factors in a graphical model

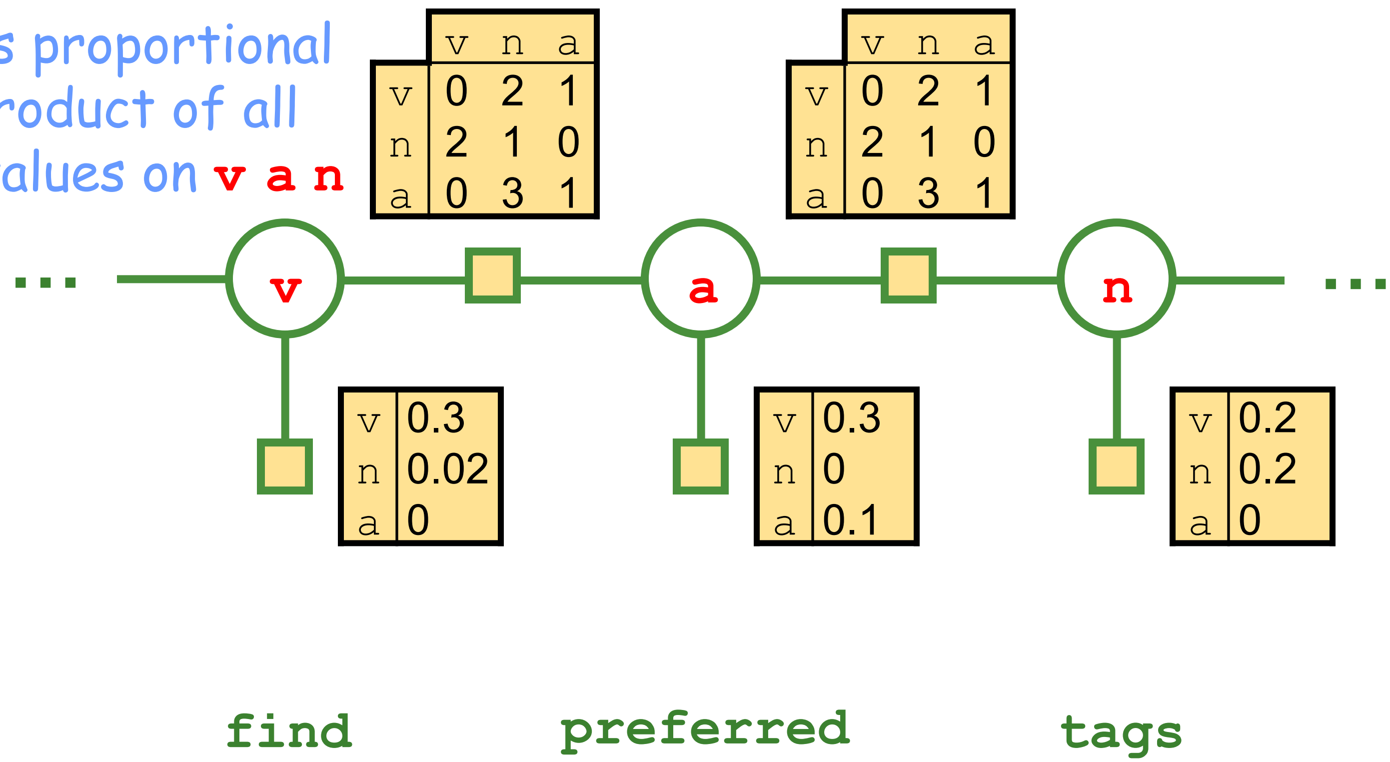
- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging



Local factors in a graphical model

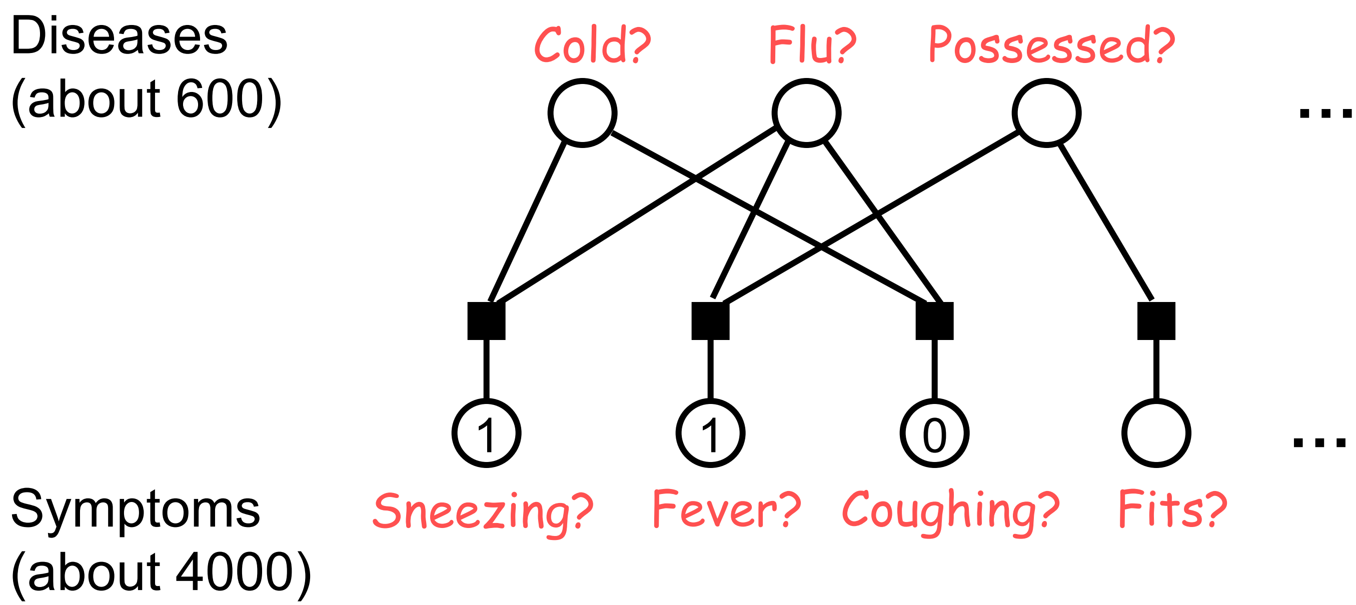
- First, a familiar example
  - Conditional Random Field (CRF) for POS tagging

p(**v a n**) is proportional to the product of all factors' values on **v a n**



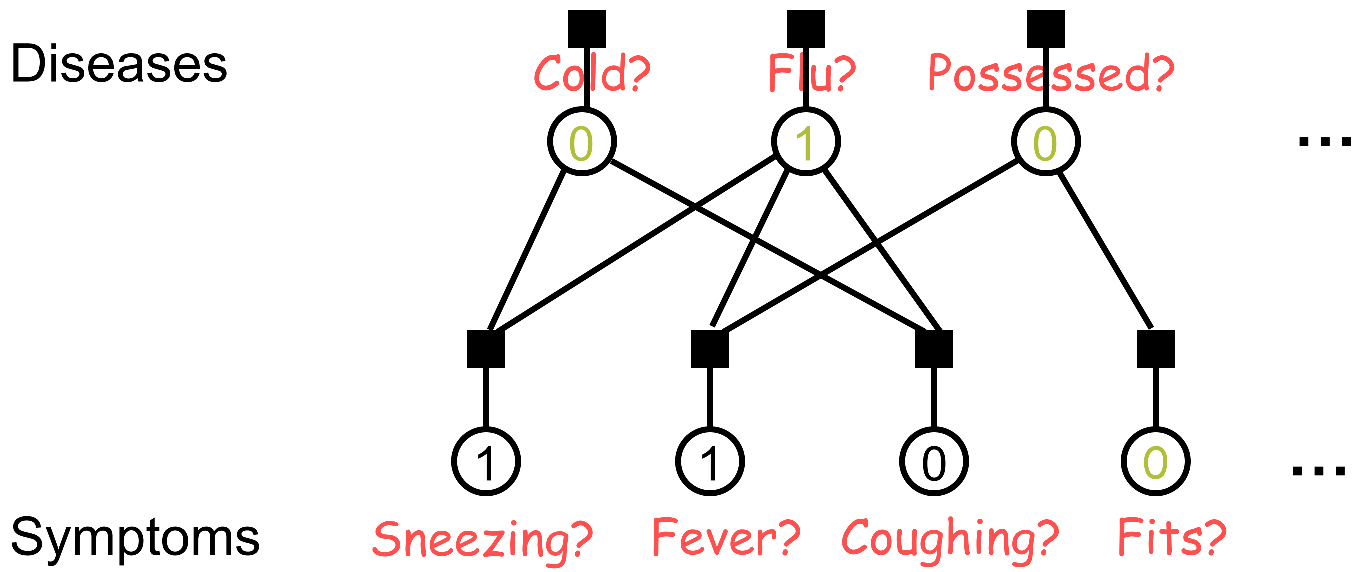
Example: Medical diagnosis (QMR-DT)

- Patient is sneezing with a fever; no coughing



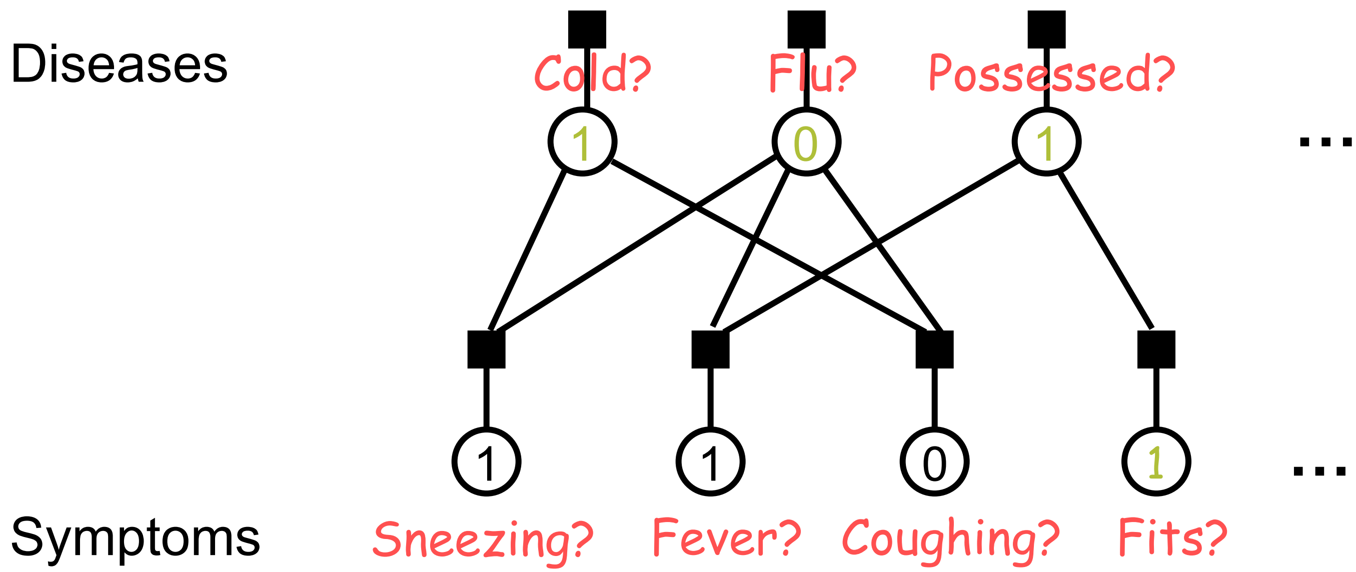
Example: Medical diagnosis

- Patient is sneezing with a fever; no coughing
  - Possible diagnosis: Flu (without coughing)
    - But maybe it's not flu season …



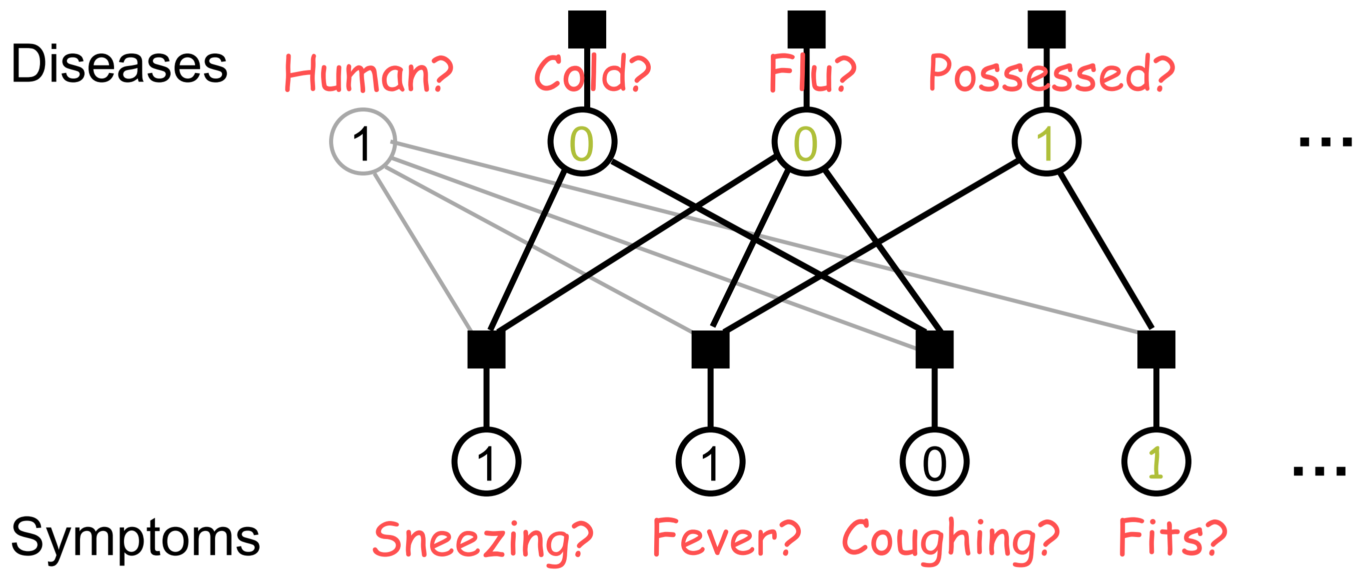
Example: Medical diagnosis

- Patient is sneezing with a fever; no coughing
  - Possible diagnosis: Cold (without coughing), and possessed (better ask about fits …)



Example: Medical diagnosis

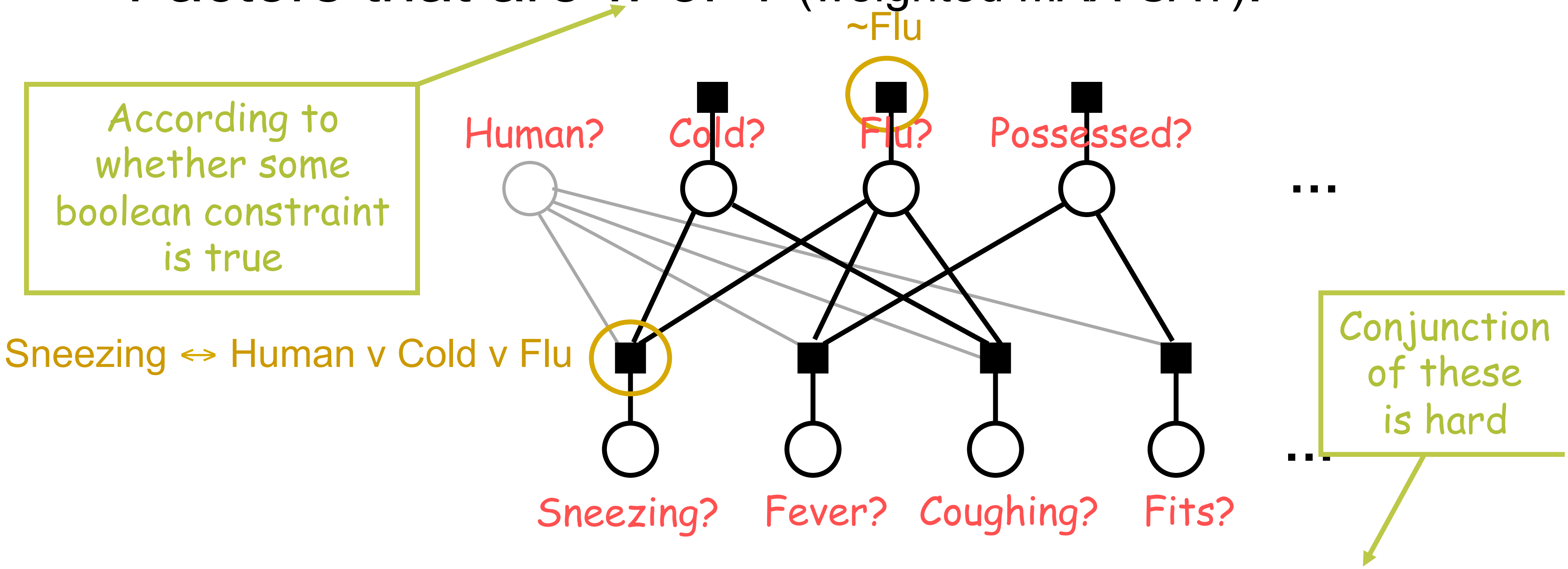
- Patient is sneezing with a fever; no coughing
  - Possible diagnosis: Spontaneous sneezing, and possessed (better ask about fits …)



Note: Here symptoms & diseases are boolean.
We could use real #s to denote degree.

Example: Medical diagnosis

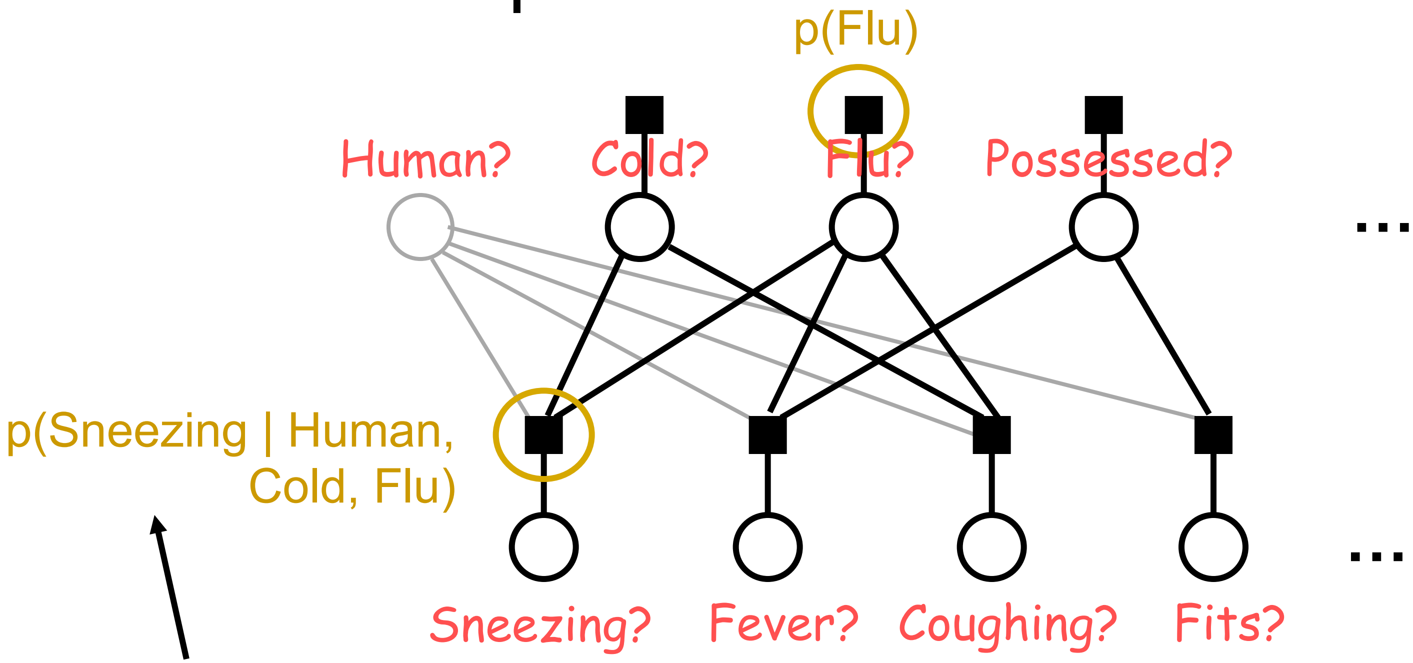
- What are the factors, exactly?
- Factors that are w or 1 (weighted MAX-SAT):



- If observe sneezing, get a disjunctive clause (Human v Cold v Flu)
- If observe non-sneezing, get unit clauses (~Human) ^ (~Cold) ^ (~Flu)

Example: Medical diagnosis

- What are the factors, exactly?
- Factors that are probabilities:



Use a little "noisy OR" model here:

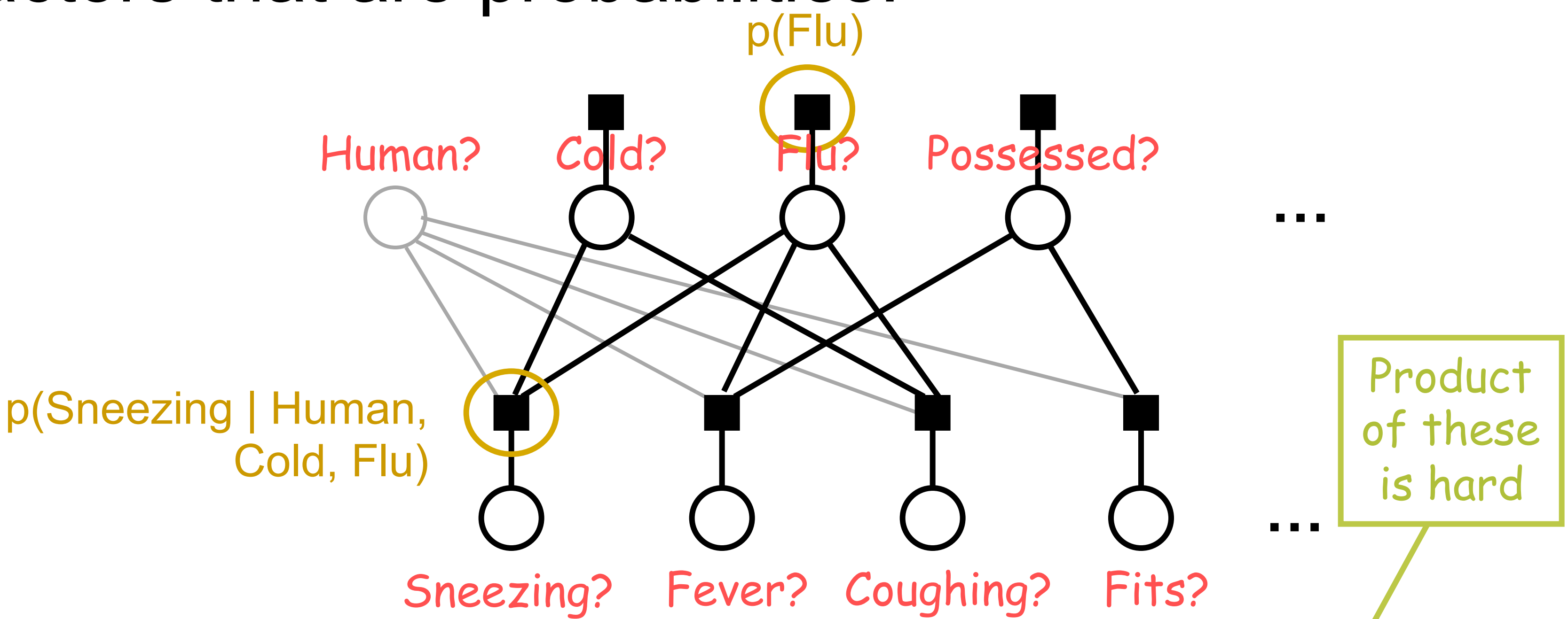
x = (Human,Cold,Flu), e.g., (1,1,0). More 1's should increase p(sneezing).

p(~sneezing | x) = exp(- w · x) e.g., w = (0.05, 2, 5)

Would get logistic regression model if we replaced exp by sigmoid, i.e., exp/(1+exp)

Example: Medical diagnosis

- What are the factors, exactly?
- Factors that are probabilities:

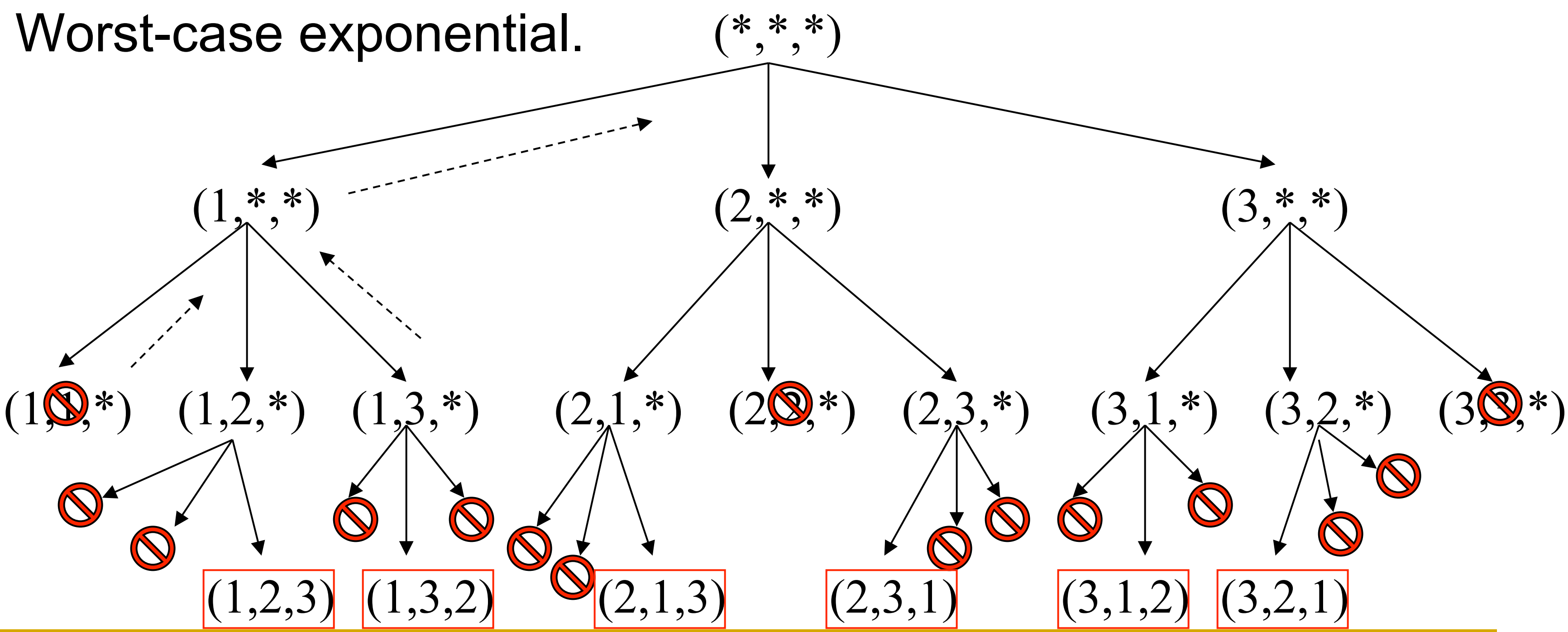


- If observe sneezing, get a factor (1 – exp(- w · x)) $(1 - 0.95^{Human}\, 0.14^{Cold}\, 0.007^{Flu})$
- If observe non-sneezing, get a factor exp(- w · x) $0.95^{Human}\, 0.14^{Cold}\, 0.007^{Flu}$

As w → ∞, approach Boolean case (product of all factors → 1 if SAT, 0 if UNSAT)

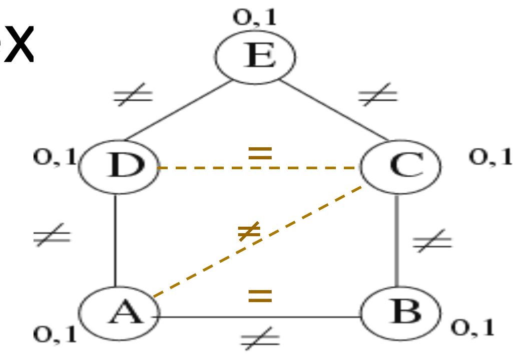
Technique #1: Branch and bound

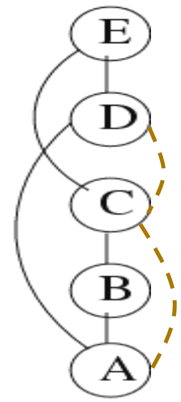
- Exact backtracking technique we've already studied.
  - And used via ECLiPSe's "minimize" routine.
- Propagation can help prune branches of the search tree (add a hard constraint that we must do better than best solution so far).
- Worst-case exponential.



Technique #2: Variable Elimination

- Exact technique we've studied; worst-case ex





Bucket E: E ≠ D, E ≠ C → D = C
Bucket D: D ≠ A → A ≠ C
Bucket C: C ≠ B → B = A
Bucket B: B ≠ A → contradiction
Bucket A:

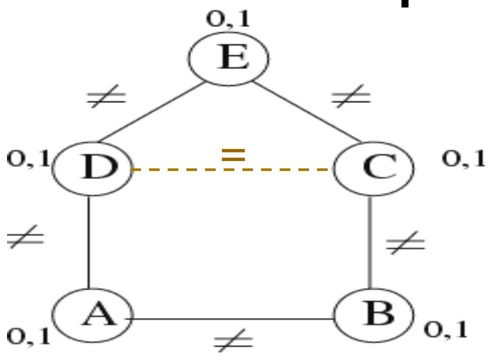
join all constraints in E's bucket
yielding a new constraint on D (and C)
now join all constraints in D's bucket …

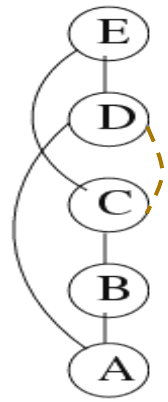
- But how do we do it for soft constraints?
  - How do we join soft constraints?

figure thanks to Rina Dechter

Technique #2: Variable Elimination

- Easiest to explain via Dyna.





- goal max= f1(A,B)*f2(A,C)*f3(A,D)*~~f4(C,E)*f5(D,E)~~.

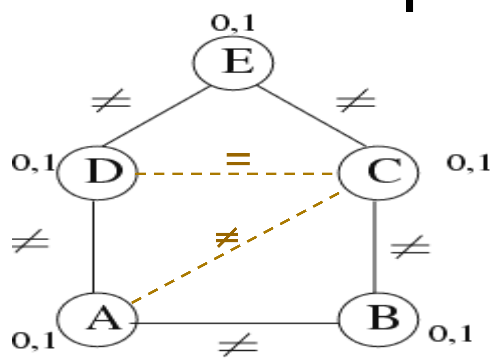
tempE(C,D)

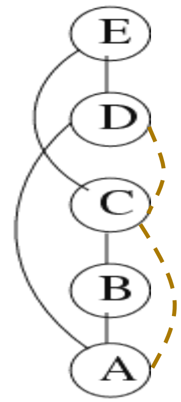
- tempE(C,D) max= f4(C,E)*f5(D,E).

to eliminate E,
join constraints mentioning E,
and project E out

Technique #2: Variable Elimination

- Easiest to explain via Dyna.





- goal max= f1(A,B)*f2(A,C)*~~f3(A,D)*tempE(C,D)~~.

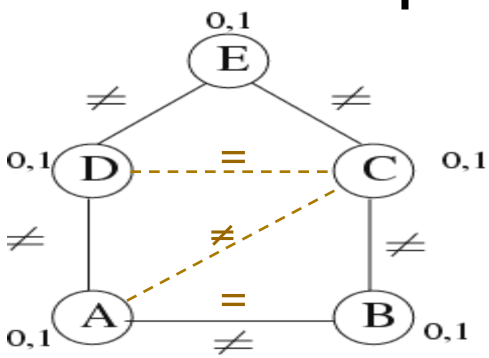
  tempD(A,C)

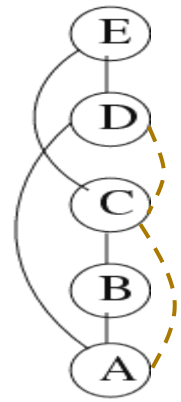
- tempD(A,C) max= f3(A,D)*tempE(C,D).
- tempE(C,D) max= f4(C,E)*f5(D,E).

to eliminate D,
join constraints mentioning D,
and project D out

Technique #2: Variable Elimination

- Easiest to explain via Dyna.





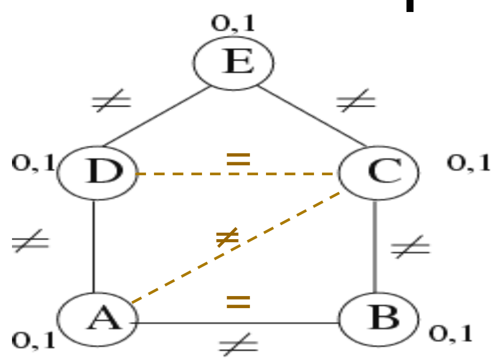
- goal max= f1(A,B)*~~f2(A,C)*tempD(A,C)~~.

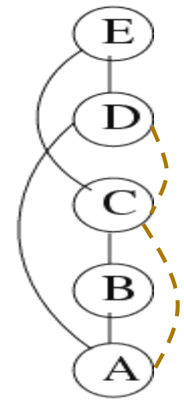
  tempC(A)

- tempC(A) max= f2(A,C)*tempD(A,C).
- tempD(A,C) max= f3(A,D)*tempE(C,D).
- tempE(C,D) max= f4(C,E)*f5(D,E).

Technique #2: Variable Elimination

- Easiest to explain via Dyna.

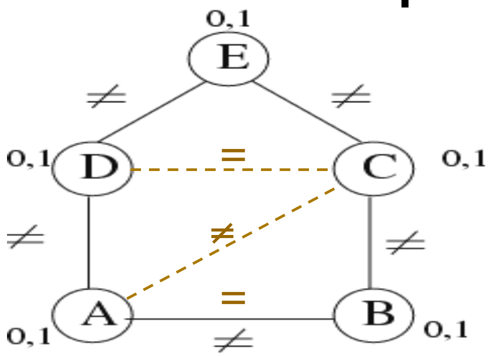


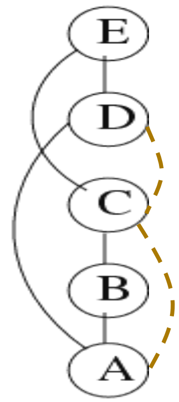


- goal max= tempC(A)*~~f1(A,B)~~. tempB(A)
- tempB(A) max= f1(A,B).
- tempC(A) max= f2(A,C)*tempD(A,C).
- tempD(A,C) max= f3(A,D)*tempE(C,D).
- tempE(C,D) max= f4(C,E)*f5(D,E).

Technique #2: Variable Elimination

- Easiest to explain via Dyna.



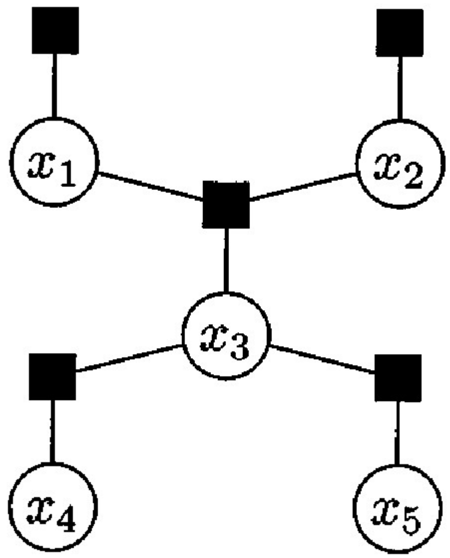


- goal max= tempC(A)*tempB(A).
- tempB(A) max= f1(A,B).
- tempC(A) max= f2(A,C)*tempD(A,C).
- tempD(A,C) max= f3(A,D)*tempE(C,D).
- tempE(C,D) max= f4(C,E)*f5(D,E).

Probabilistic interpretation of factor graph

("undirected graphical model")

Each factor is a function >= 0 of the values of its variables.



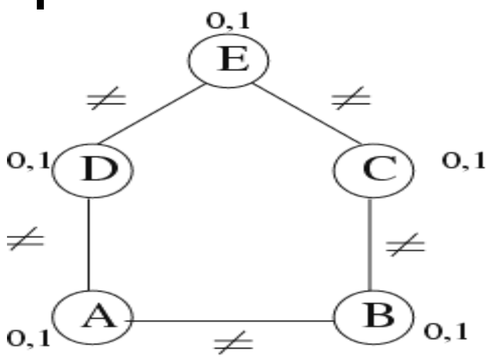
Measure goodness of an assignment by the product of all the factors.

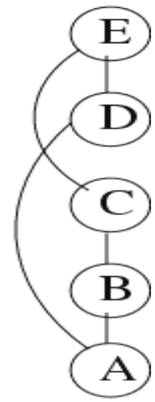
- For any assignment x = (x1,…,x5), define u(x) = product of all factors, e.g., u(x) = f1(x)*f2(x)*f3(x)*f4(x)*f5(x).
- We'd like to interpret u(x) as a **probability distribution** over all 2^5 assignments.

- Do we have u(x) >= 0? Yes. ☺
- Do we have ∑ u(x) = 1? No. ∑ u(x) = Z for some Z. ☹
- So u(x) is *not* a probability distribution.
- But p(x) = u(x)/Z is!

Z is hard to find … (the "partition function")

- Exponential time with this Dyna program.





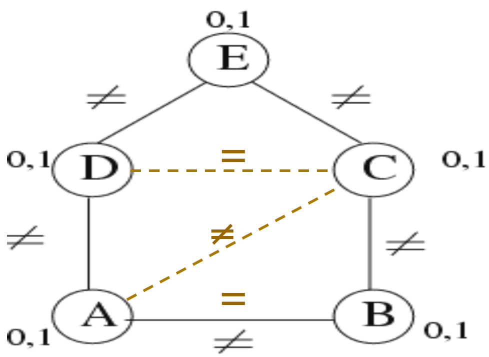
- goal ~~max~~= f1(A,B)*f2(A,C)*f3(A,D)*f4(C,E)*f5(D,E).

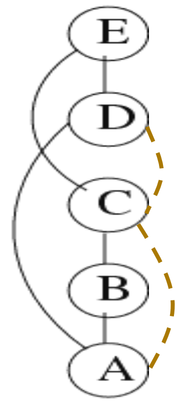
  +=

This explicitly sums over all 2^5 assignments.
We can do better by variable elimination …
(although still exponential time in worst case).
Same algorithm as before: just replace max= with +=.

Z is hard to find … (the "partition function")

- Faster version of Dyna program, after var elim.





- goal += tempC(A)*tempB(A).
- tempB(A) += f1(A,B).
- tempC(A) += f2(A,C)*tempD(A,C).
- tempD(A,C) += f3(A,D)*tempE(C,D).
- tempE(C,D) += f4(C,E)*f5(D,E).

Why a probabilistic interpretation?

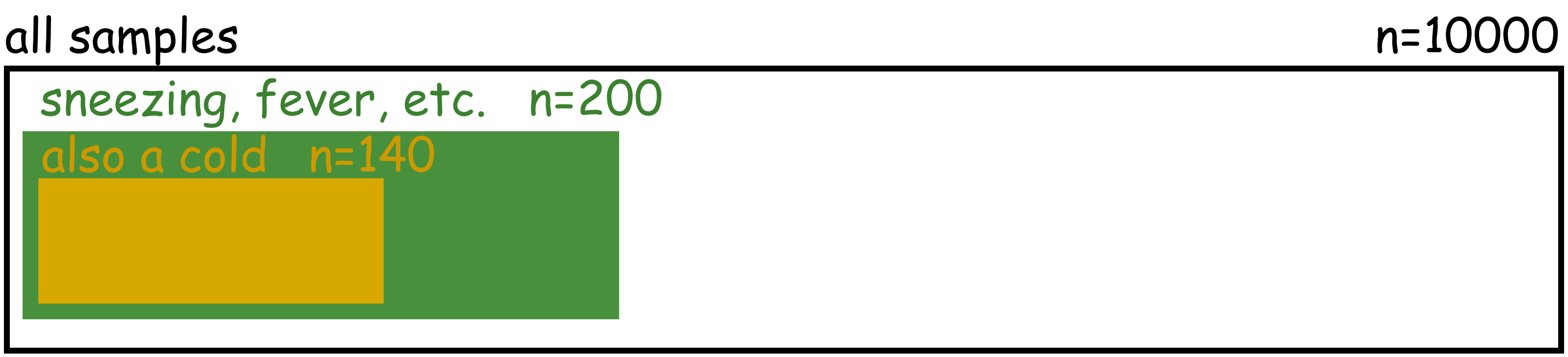
1. Allows us to make **predictions**.
   - You're sneezing with a fever & no cough.
   - Then what is the *probability* that you have a cold?
2. Important in **learning** the factor functions.
   - Maximize the probability of training data.
3. Central to deriving fast **approximation** algorithms.
   - "Message passing" algorithms where nodes in the factor graph are repeatedly updated based on adjacent nodes.
   - Many such algorithms. E.g., survey propagation is the current best method for random 3-SAT problems. Hot area of research!

Probabilistic interpretation ➔ Predictions

You're sneezing with a fever & no cough.

Then what is the *probability* that you have a cold?

- Randomly sample 10000 assignments from p(x).
- In 200 of them (2%), patient is sneezing with a fever and no cough.
- In 140 (1.4%) of those, the patient also has a cold.



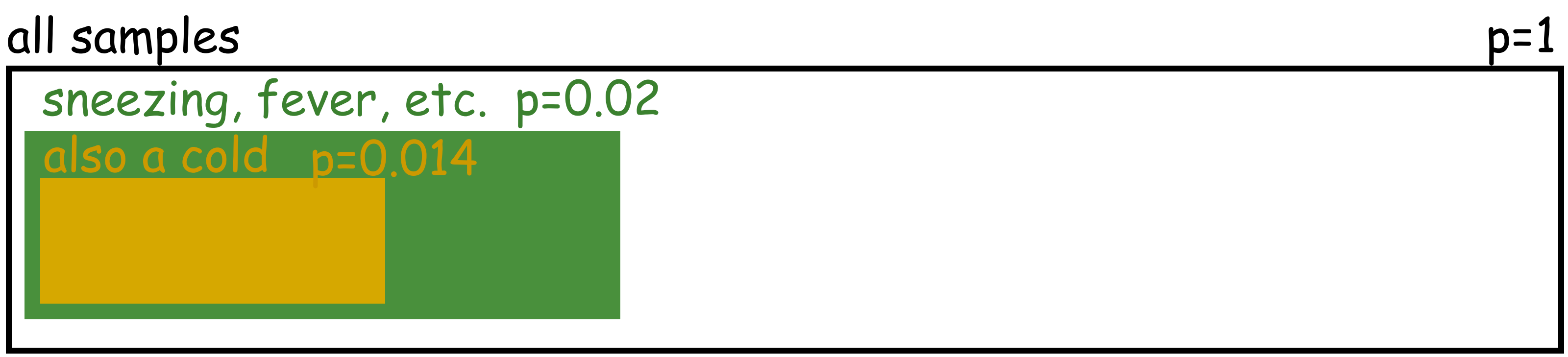
answer: 70% (140/200)

Probabilistic interpretation → Predictions

You're sneezing with a fever & no cough.

Then what is the *probability* that you have a cold?

- Randomly sample 10000 assignments from p(x).
- In 200 of them (2%), patient is sneezing with a fever and no cough.
- In 140 (1.4%) of those, the patient <u>also</u> has a cold.



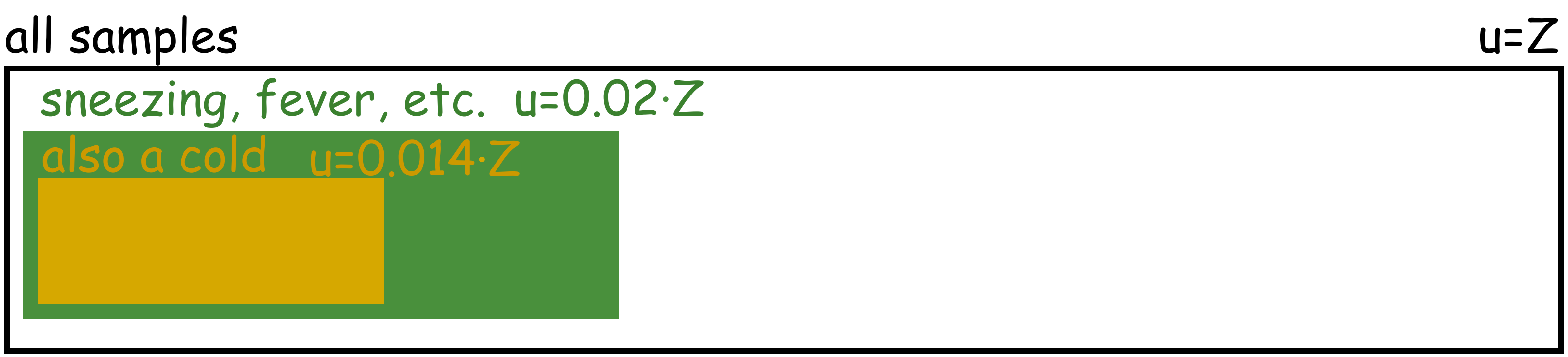
answer: 70% (0.014/0.02)

Probabilistic interpretation → Predictions

You're sneezing with a fever & no cough.

Then what is the *probability* that you have a cold?

- Randomly sample 10000 assignments from p(x).
- In 200 of them (2%), patient is sneezing with a fever and no cough.
- In 140 (1.4%) of those, the patient also has a cold.



answer: 70% (0.014·Z / 0.02·Z)

Probabilistic interpretation → Predictions

You're sneezing with a fever & no cough.

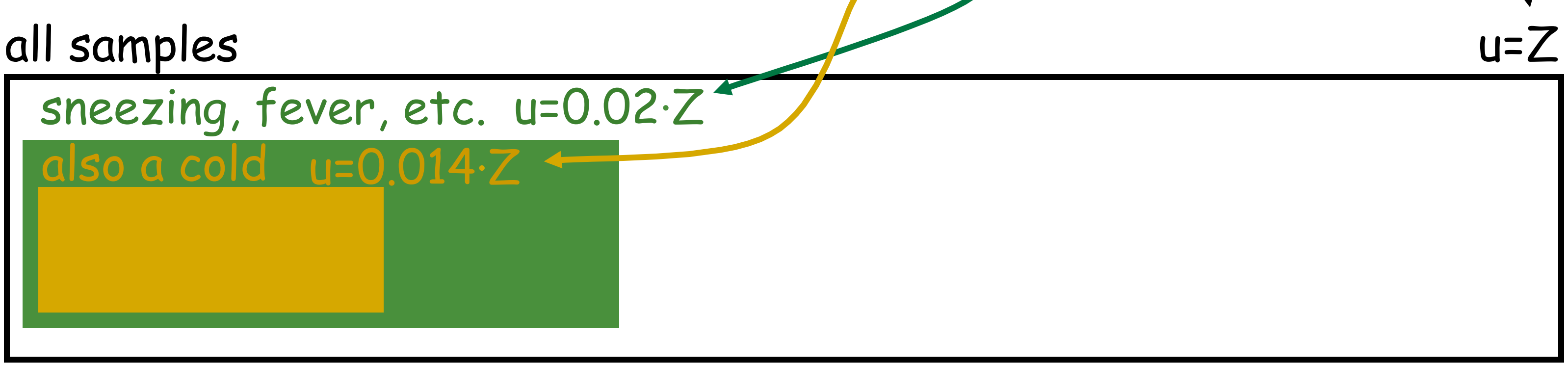
Then what is the *probability* that you have a cold?

- ~~Randomly sample 10000 assignments from p(x).~~ Could we compute exactly instead?

~~Remember, we can find this by variable elimination~~ unnecessary

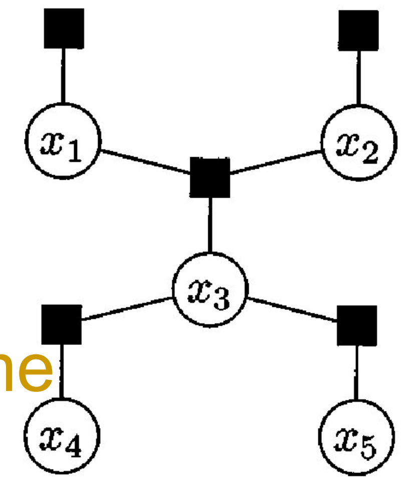
This too: just add unary constraints Sneezing=1,Fever=1,Cough=0

This too: one more unary constraint Cold=1



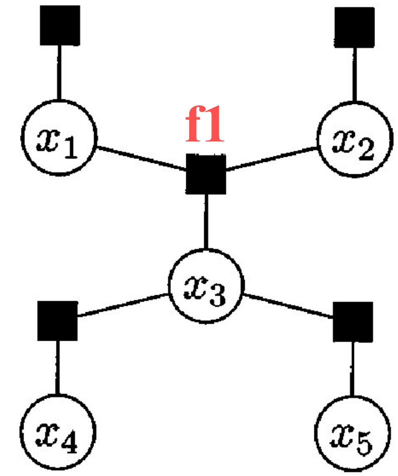
answer: 70% (0.014·Z / 0.02·Z)

Probabilistic interpretation → Learning



- How likely is it for (X1,X2,X3) = (1,0,1) (according to **real data**)? 90% of the time
- How likely is it for (X1,X2,X3) = (1,0,1) (according to **the full model**)? 55% of the time
  - I.e., if you randomly sample many assignments from p(x), 55% of assignments have (1,0,1).
  - E.g., 55% have (Cold, ~Cough, Sneeze): too few.
- To learn a better p(x), we adjust the factor functions to bring the second ratio from 55% up to 90%.

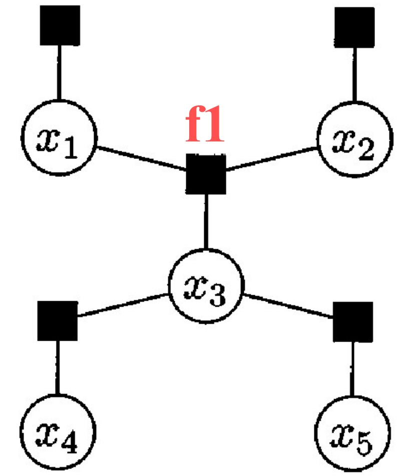
Probabilistic interpretation → Learning



- How likely is it for (X1,X2,X3) = (1,0,1) (according to **real data**)? 90% of the time
- How likely is it for (X1,X2,X3) = (1,0,1) (according to **the full model**)? 55% of the time
- To learn a better p(x), we adjust the factor functions to bring the second ratio from 55% up to 90%.

- By increasing f1(1,0,1), we can increase the model's probability that (X1,X2,X3) = (1,0,1).
- Unwanted ripple effect: This will also increase the model's probability that X3=1, and hence will change the probability that X5=1, and …
- So we have to change all the factor functions at once to make all of them match real data.
- Theorem: This is always possible. (gradient descent or other algorithms)
  - Theorem: The resulting learned function p(x) maximizes p(**real data**).

Probabilistic interpretation → Learning



- How likely is it for (X1,X2,X3) = (1,0,1) (according to **real data**)? 90% of the time
- How likely is it for (X1,X2,X3) = (1,0,1) (according to **the full model**)? 55% of the time
- To learn a better p(x), we adjust the factor functions to bring the second ratio from 55% up to 90%.

- By increasing f1(1,0,1), we can increase the model's probability that (X1,X2,X3) = (1,0,1).
- Unwanted ripple effect: This will also increase the model's probability that X3=1, and hence will change the probability that X5=1, and …
- So we have to change all the factor functions at once to make all of them match real data.
- Theorem: This is always possible. (gradient descent or other algorithms)
  - Theorem: The resulting learned function p(x) maximizes p(**real data**).

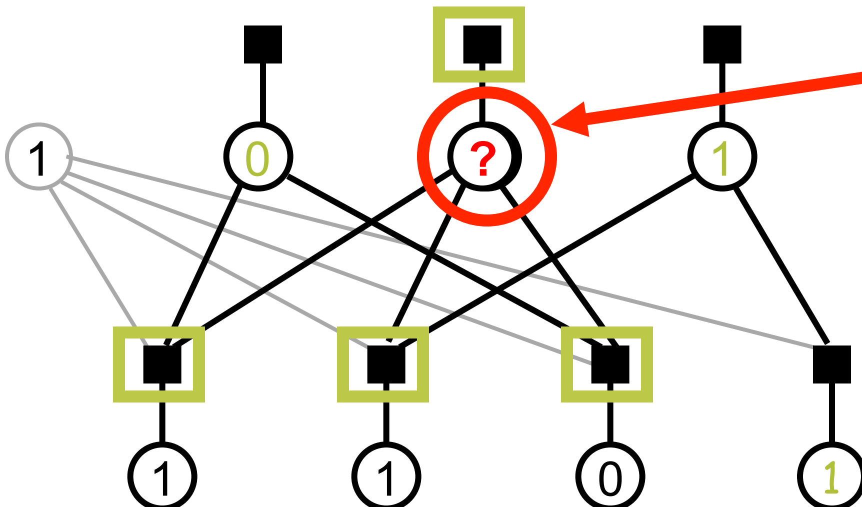
Probabilistic interpretation → Approximate constraint satisfaction

3. Central to deriving fast **approximation** algorithms.
   - "Message passing" algorithms where nodes in the factor graph are repeatedly updated based on adjacent nodes.

- Gibbs sampling / simulated annealing
- Mean-field approximation and other variational methods
- Belief propagation
- Survey propagation

How do we sample from p(x)?

- Gibbs sampler: (should remind you of stochastic SAT solvers)
  - Pick a random starting assignment.
  - Repeat n times: Pick a variable and possibly flip it, at random
  - Theorem: Our new assignment is a random sample from a distribution close to p(x) (converges to p(x) as n → ∞)

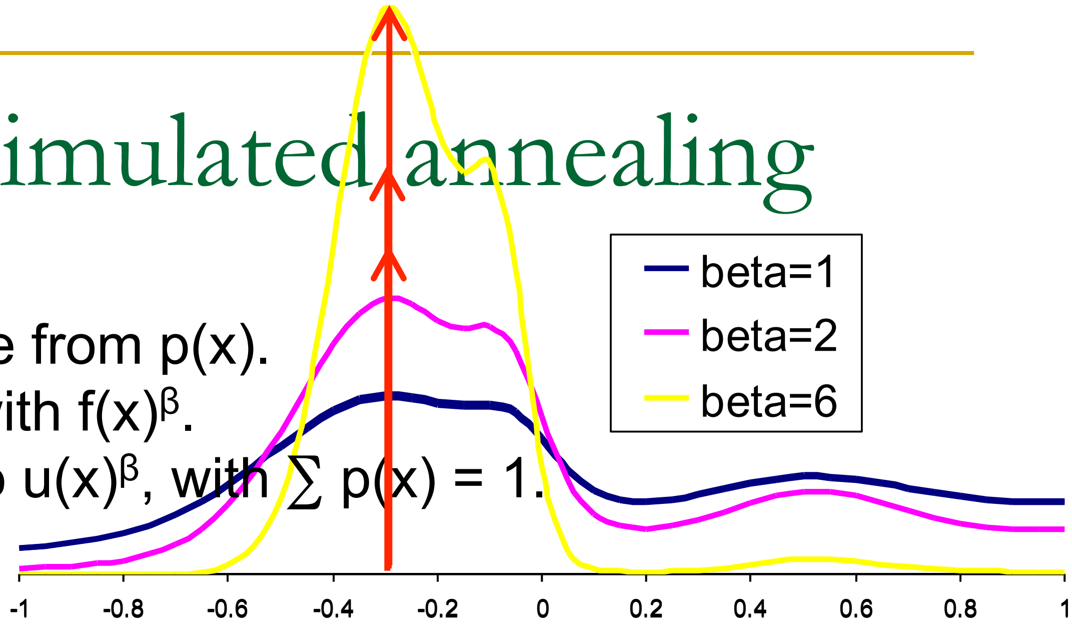


How do we decide whether new value should be 0 or 1?

If u(x) is twice as big when set at 0 than at 1, then pick 1 with prob 2/3, pick 0 with prob 1/3.

It's a local computation to determine that flipping the variable doubles u(x), since only these factors of u(x) change.

Technique #3: Simulated annealing



- Gibbs sampler can sample from p(x).
- Replace each factor f(x) with $f(x)^\beta$.
- Now p(x) is proportional to $u(x)^\beta$, with $\sum p(x) = 1$.
- What happens as $\beta \rightarrow \infty$?

- Sampler turns into a maximizer!
  - Let x* be the value of x that maximizes p(x).
  - For very large β, a single sample is almost always equal to x*.

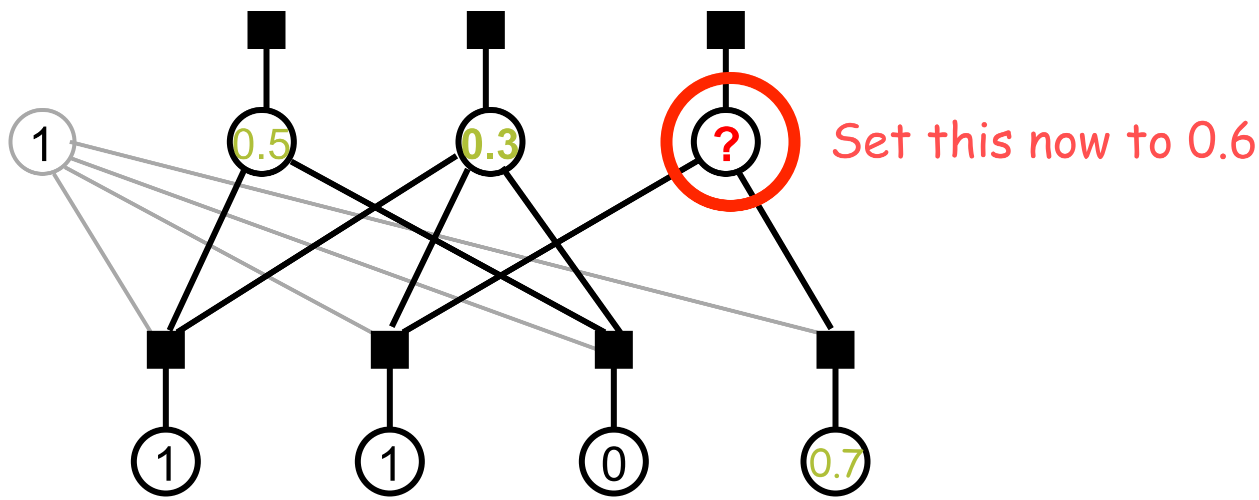
- Why doesn't this mean P=NP?
  - As $\beta \rightarrow \infty$, need to let $n \rightarrow \infty$ too to preserve quality of approx.
    - Sampler rarely goes down *steep* hills, so stays in local maxima for ages.
  - Hence, simulated annealing: <u>gradually</u> increase β as we flip variables.
  - Early on, we're flipping quite freely

Technique #4: Variational methods

- To work exactly with p(x), we'd need to compute quantities like Z, which is NP-hard.
  - (e.g., to predict whether you have a cold, or to learn the factor functions)
- We saw that Gibbs sampling was a good (but slow) approximation that didn't require Z.
- The mean-field approximation is sort of like a deterministic "averaged" version of Gibbs sampling.
  - In Gibbs sampling, nodes flutter on and off – you can ask how often x3 was 1.
  - In mean-field approximation, every node maintains a belief about how often it's 1. This belief is updated based on the beliefs at adjacent nodes. No randomness.
  - [details beyond the scope of this course, but within reach]

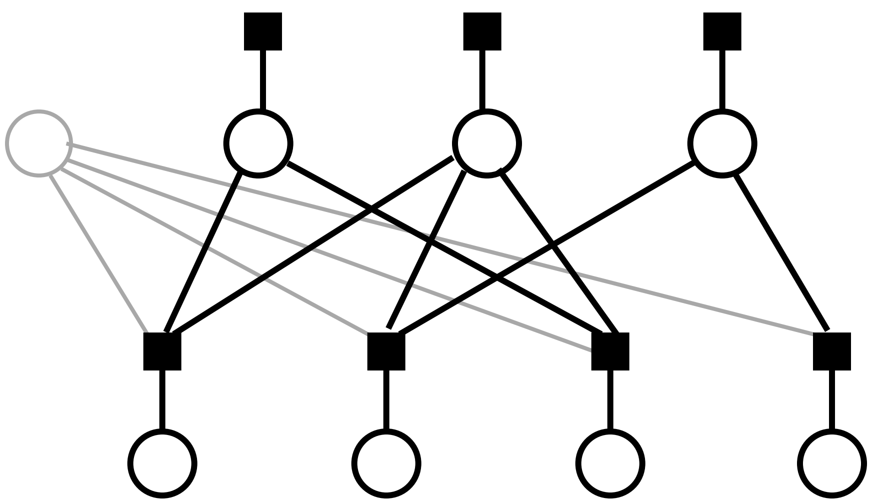
Technique #4: Variational methods

- The mean-field approximation is sort of like a deterministic "averaged" version of Gibbs sampling.
  - In Gibbs sampling, nodes flutter on and off – you can ask how often x3 was 1.
  - In mean-field approximation, every node maintains a belief about how often it's 1. This belief is repeatedly updated based on the beliefs at adjacent nodes. No randomness.

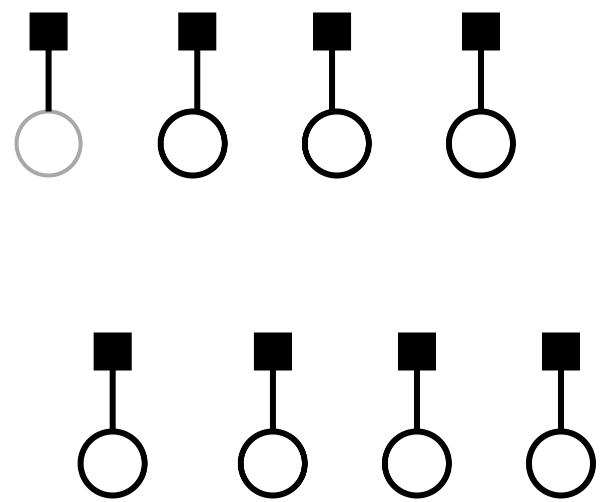


Technique #4: Variational methods

- The mean-field approximation is sort of like a deterministic "averaged" version of Gibbs sampling.
  - Can frame this as seeking an optimal approximation of this p(x) …



… by a p(x) defined as a product of simpler factors (easy to work with):



Technique #4: Variational methods

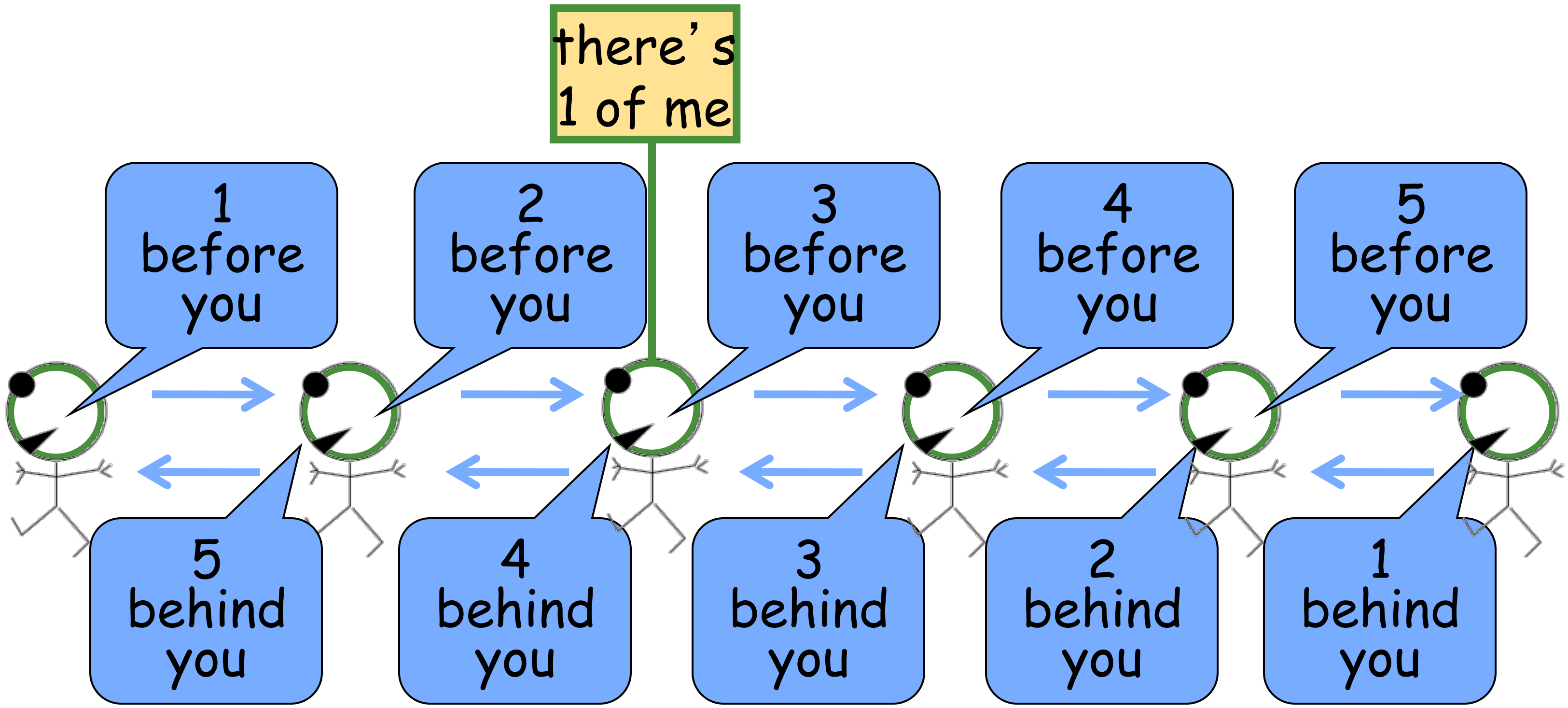
- More sophisticated version: Belief Propagation
  - The soft version of arc consistency
    - Arc consistency: some of my values become impossible ➔ so do some of yours
    - Belief propagation: some of my values become unlikely ➔ so do some of yours
      - Therefore, your other values become more likely
    - Note: Belief propagation has to be more careful than arc consistency about not having X's influence on Y feed back and influence X as if it were separate evidence. Consider constraint X=Y.
      - But there will be feedback when there are cycles in the factor graph – which hopefully are long enough that the influence is not great. If no cycles (a tree), then the beliefs are exactly correct. In this case, BP boils down to a dynamic programming algorithm on the tree.
  - Can also regard it as Gibbs sampling without the randomness
    - That's what we said about mean-field, too, but this is an even better approx.
    - Gibbs sampling lets you see:
      - how often x1 takes each of its 2 values, 0 and 1.
      - how often (x1,x2,x3) takes each of its 8 values such as (1,0,1). (This is needed in learning if (x1,x2,x3) is a factor.)
    - Belief propagation estimates these probabilities by "message passing."
    - Let's see how it works!

Technique #4: Variational methods

- Mean-field approximation
- Belief propagation
- Survey propagation:
  - Like belief propagation, but also assess the belief that the value of this variable doesn't matter! Useful for solving hard random 3-SAT problems.
- Generalized belief propagation: Joins constraints, roughly speaking.
- Expectation propagation: More approximation when belief propagation runs too slowly.
- Tree-reweighted belief propagation: …

Great Ideas in ML: Message Passing

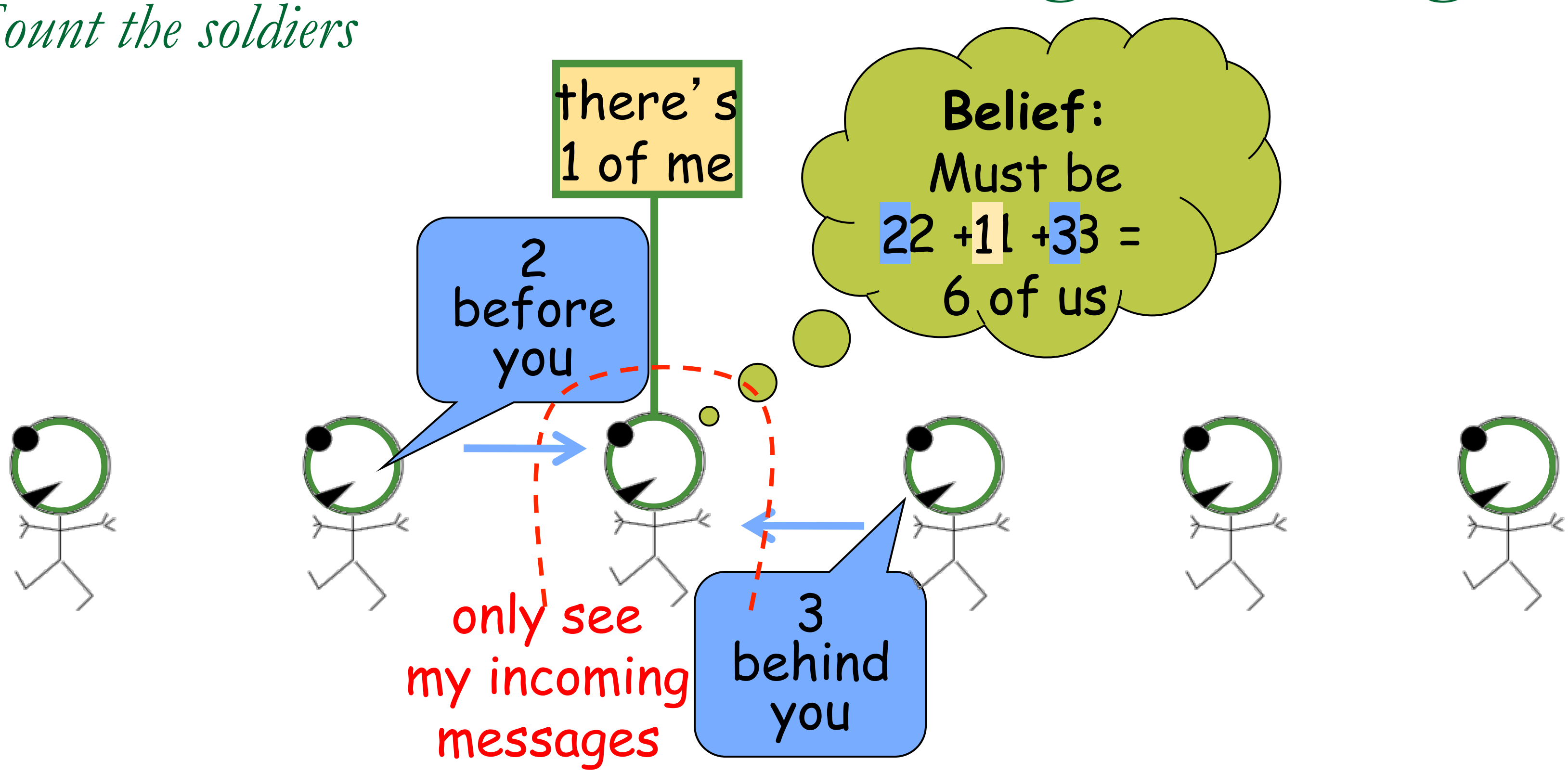
Count the soldiers



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

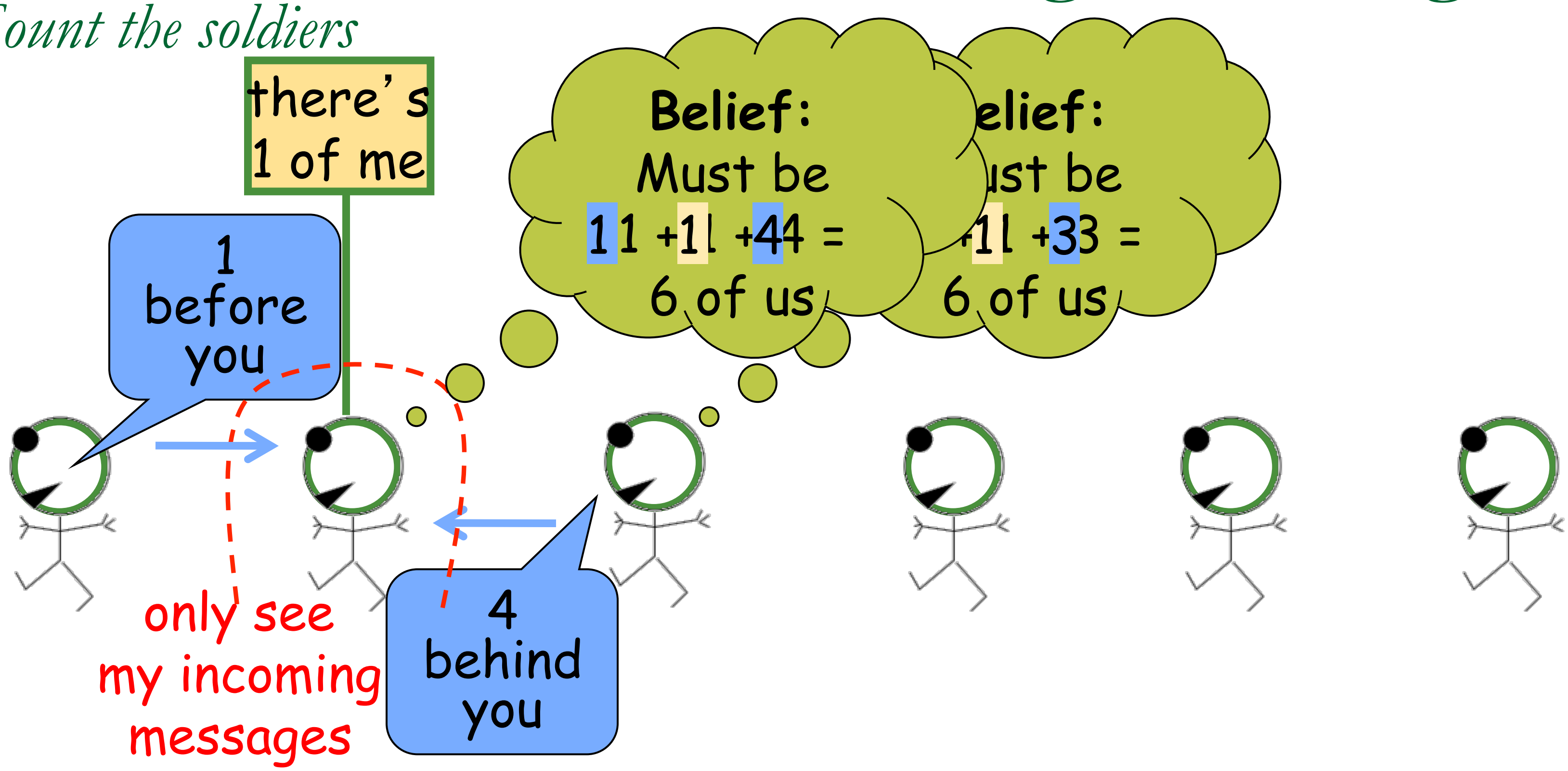
Count the soldiers



adapted from MacKay (2003) textbook

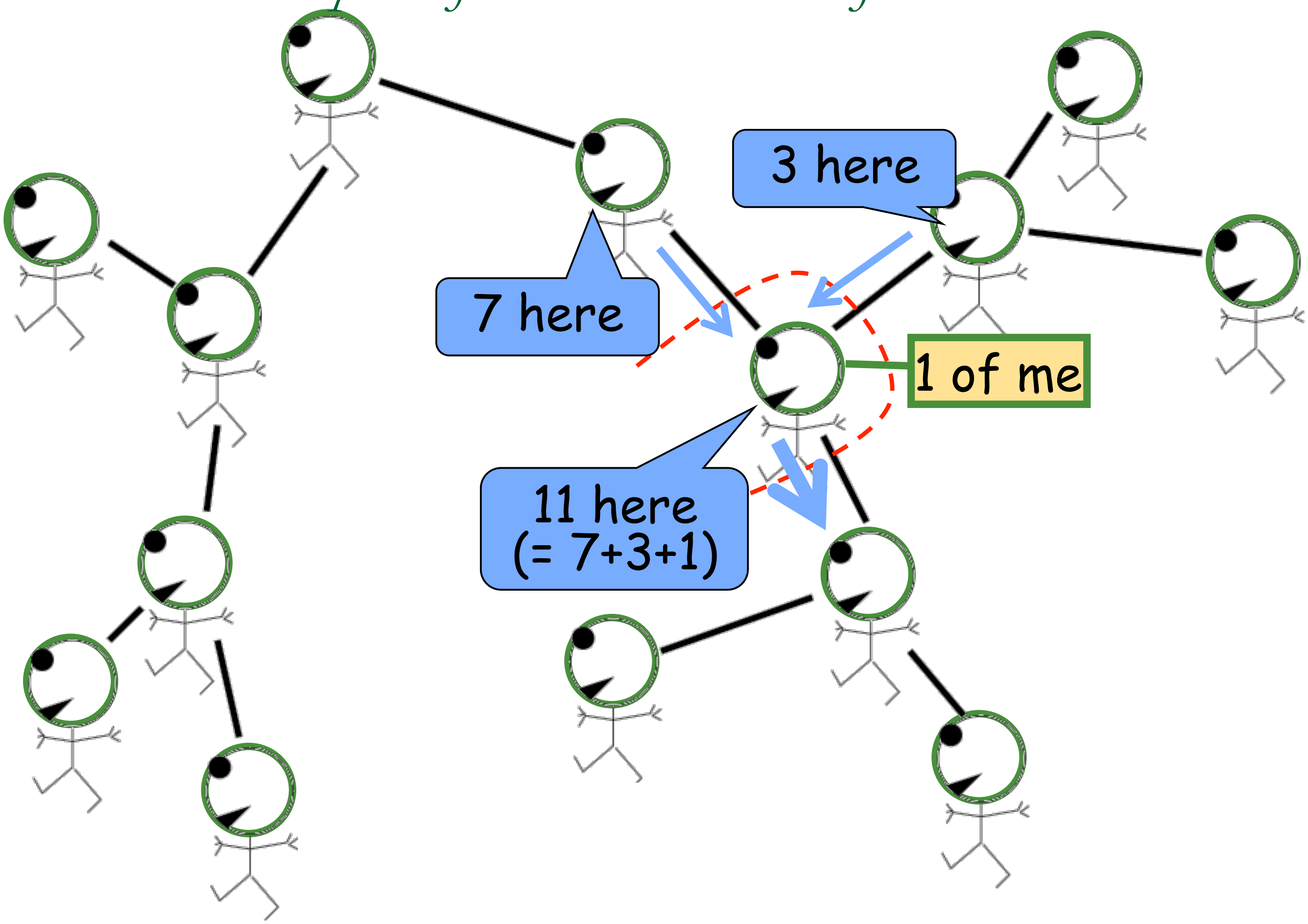
Great Ideas in ML: Message Passing

Count the soldiers



Great Ideas in ML: Message Passing

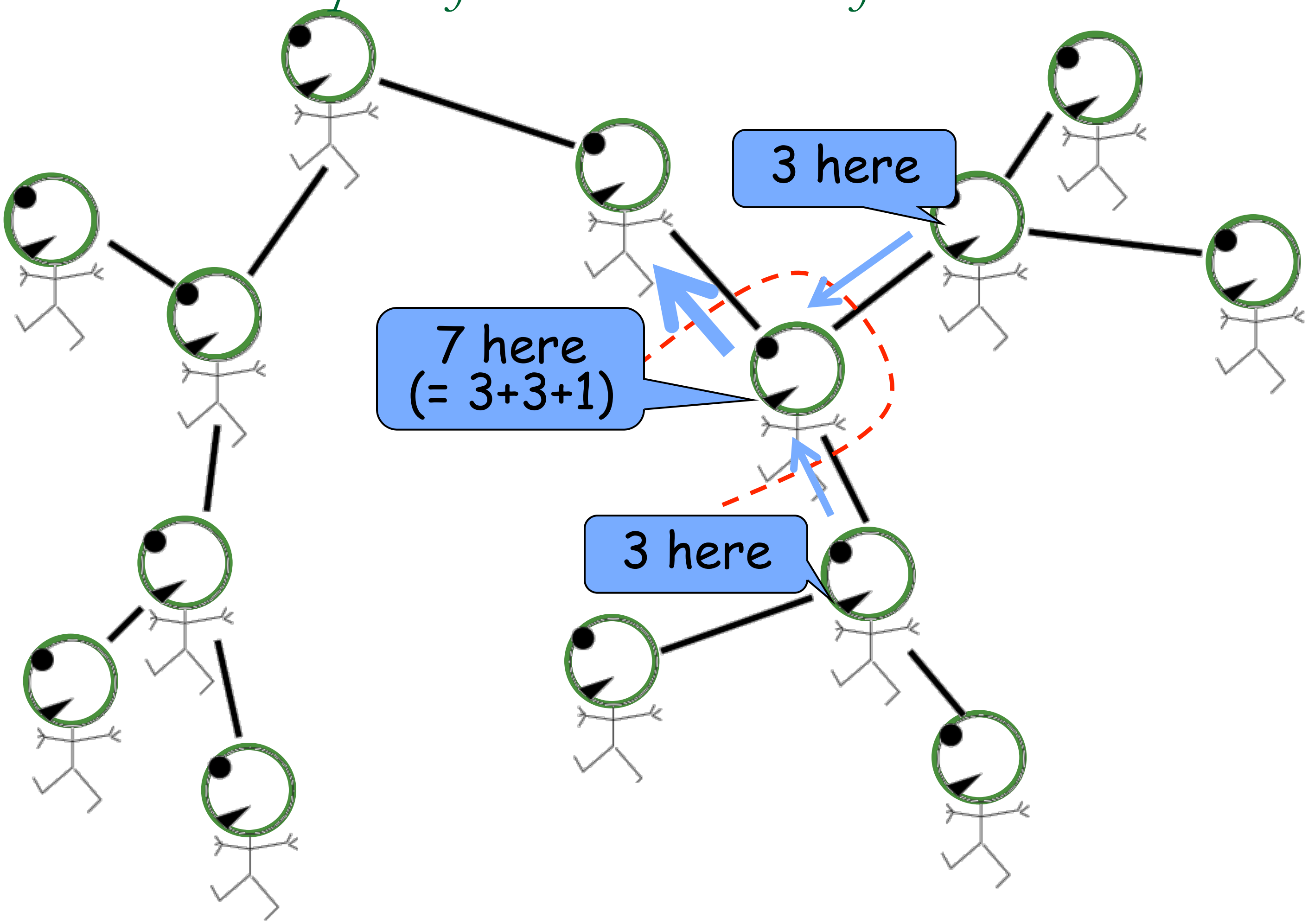
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

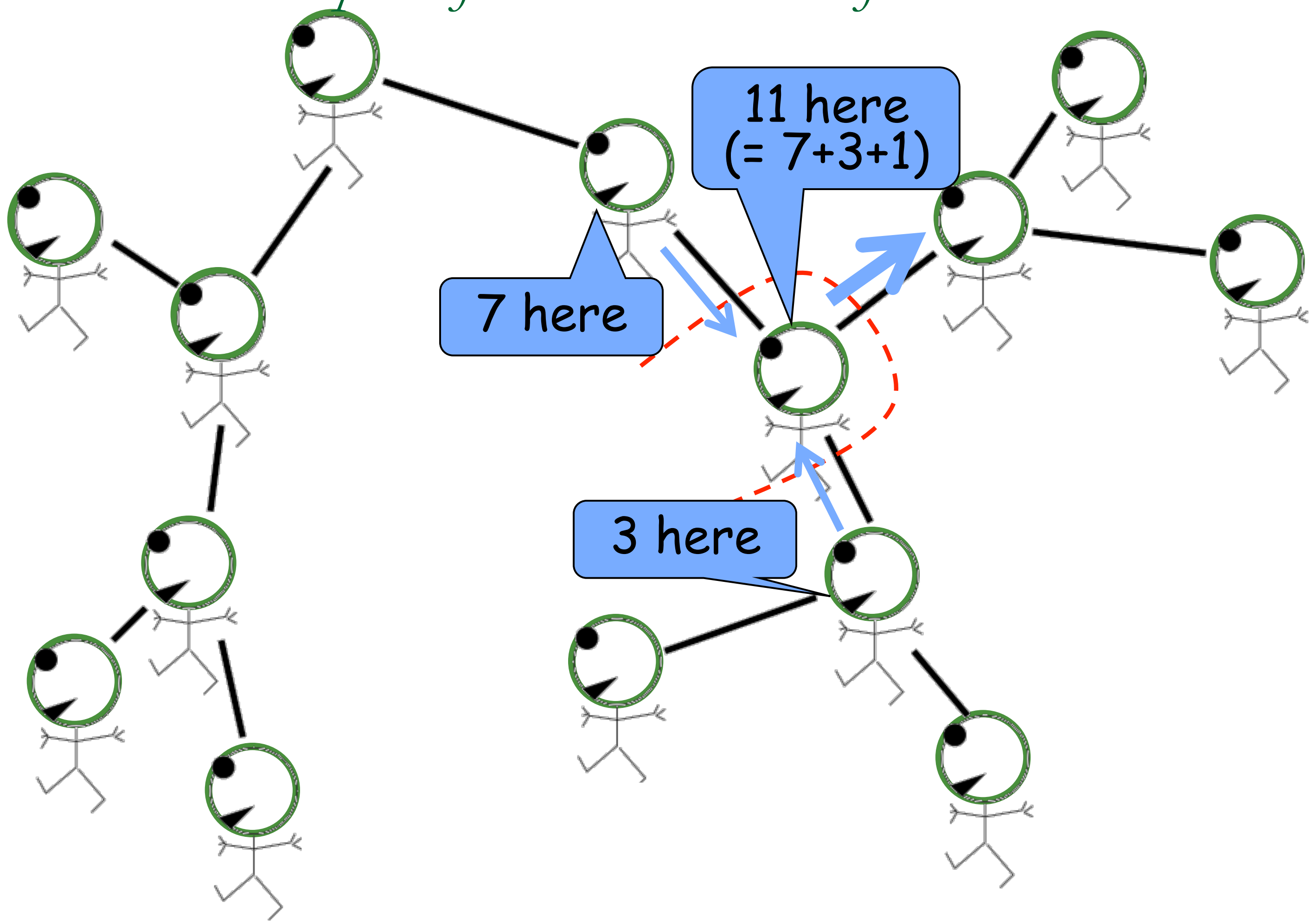
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

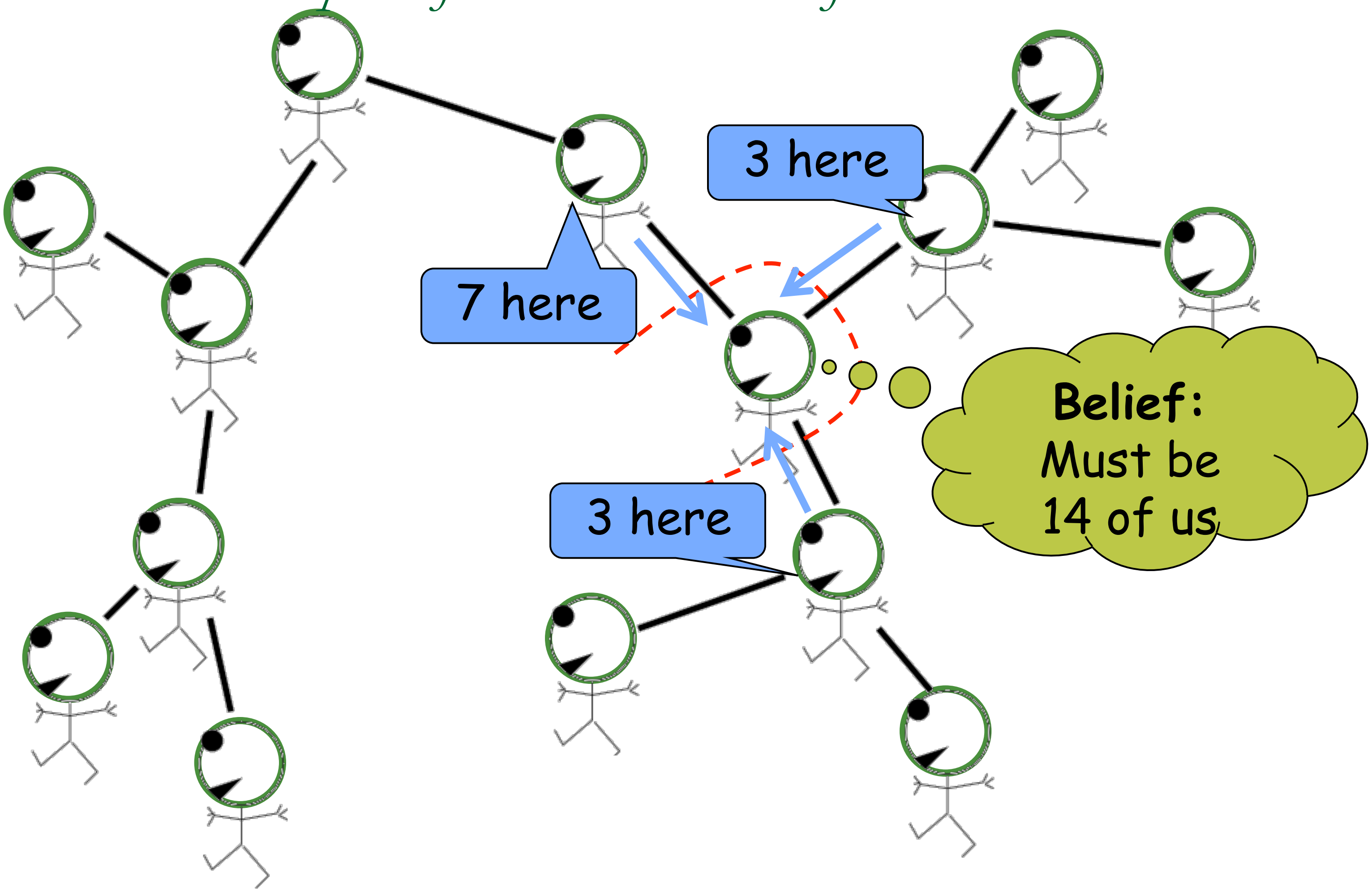
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

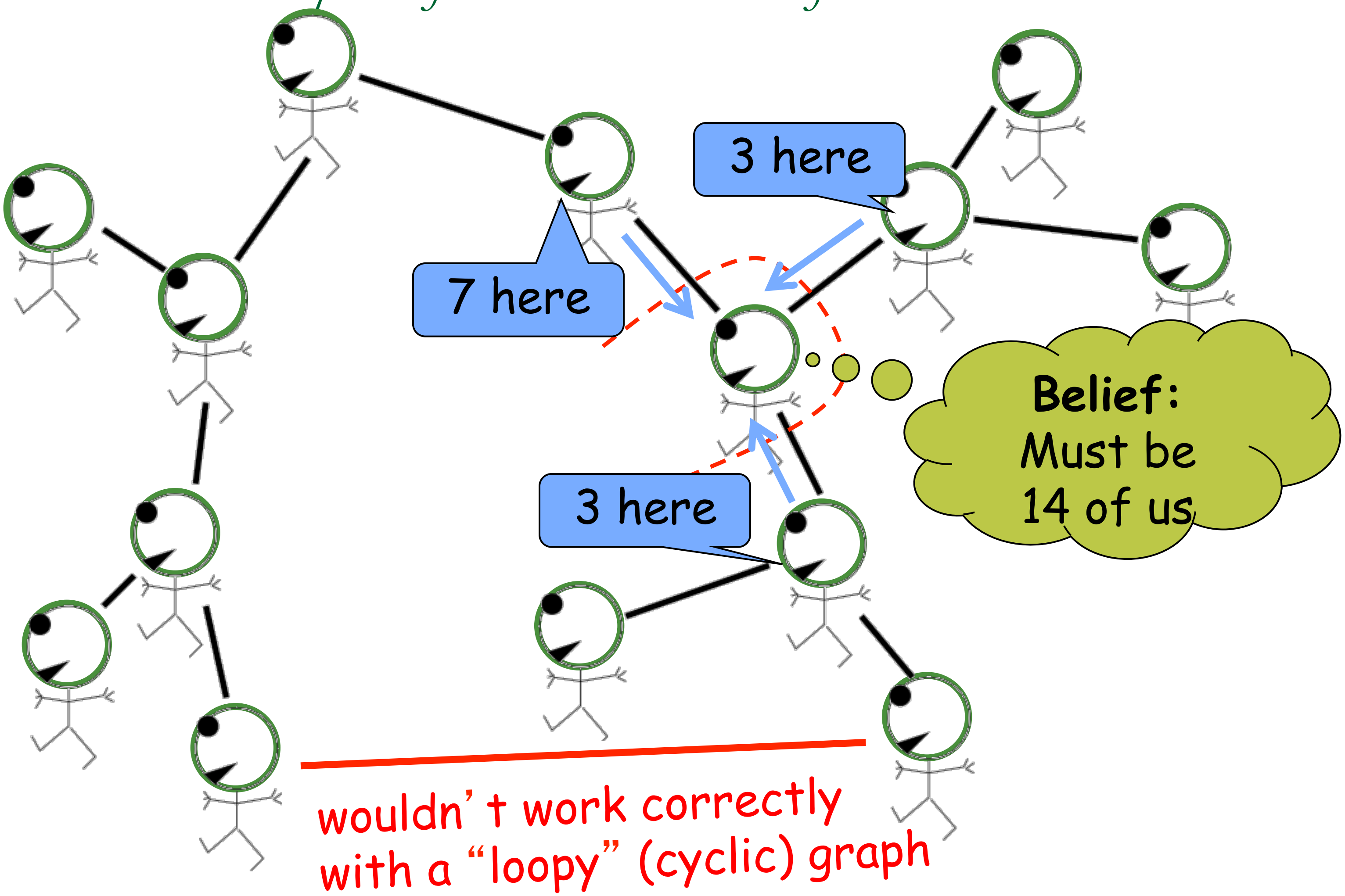
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

Great Ideas in ML: Message Passing

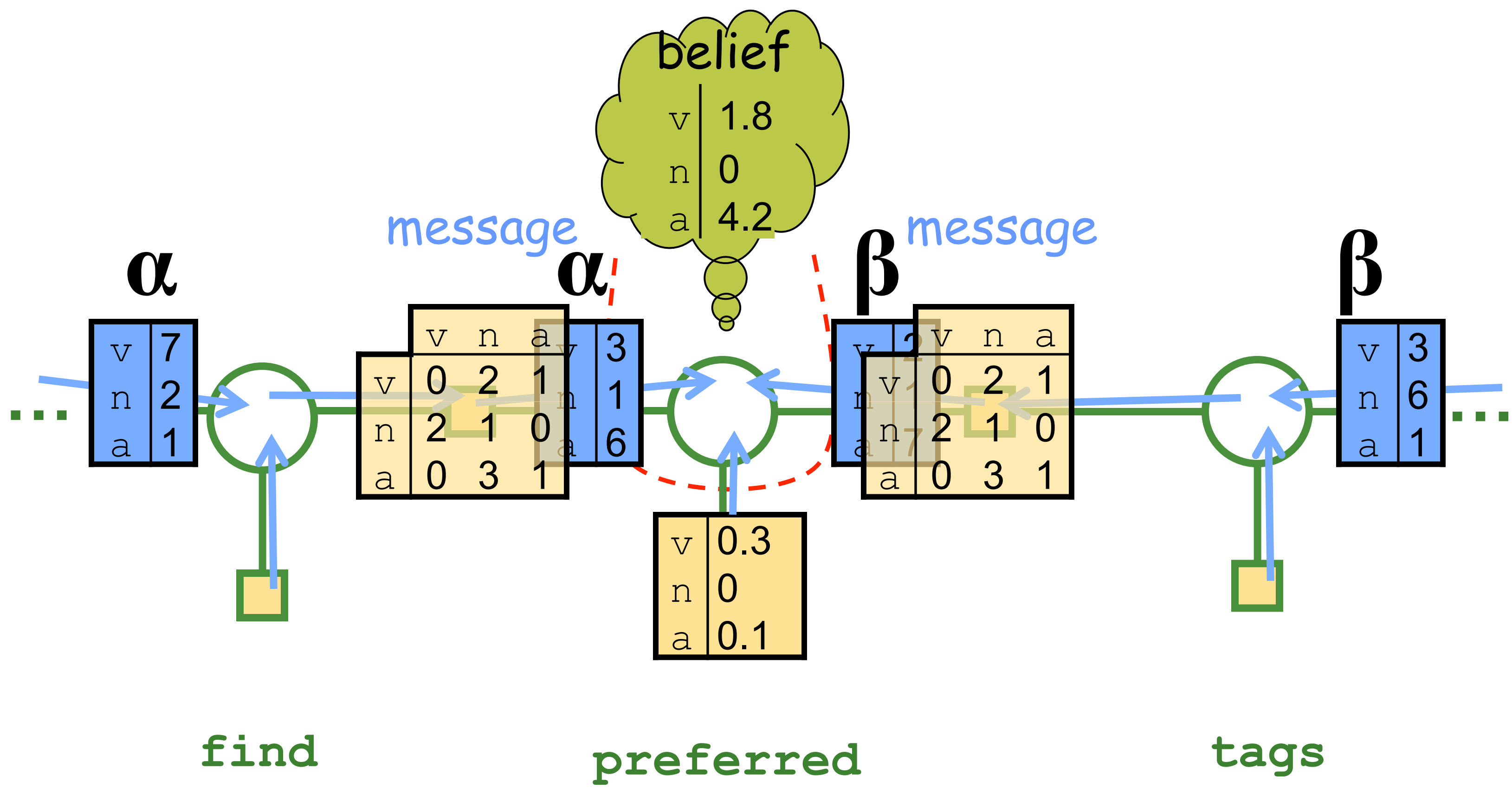
Each soldier receives reports from all branches of tree



adapted from MacKay (2003) textbook

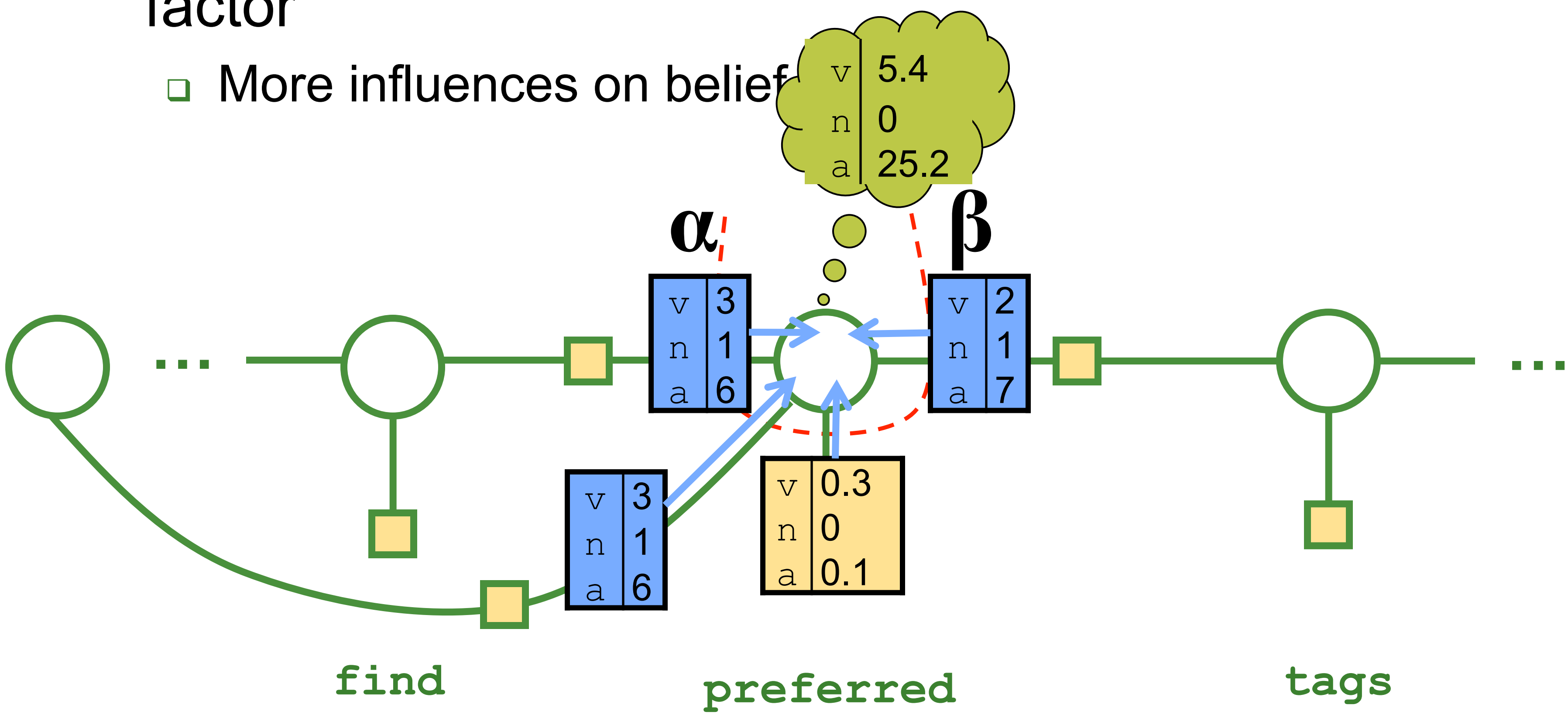
Great ideas in ML: Belief Propagation

- In the CRF, message passing = forward-backward



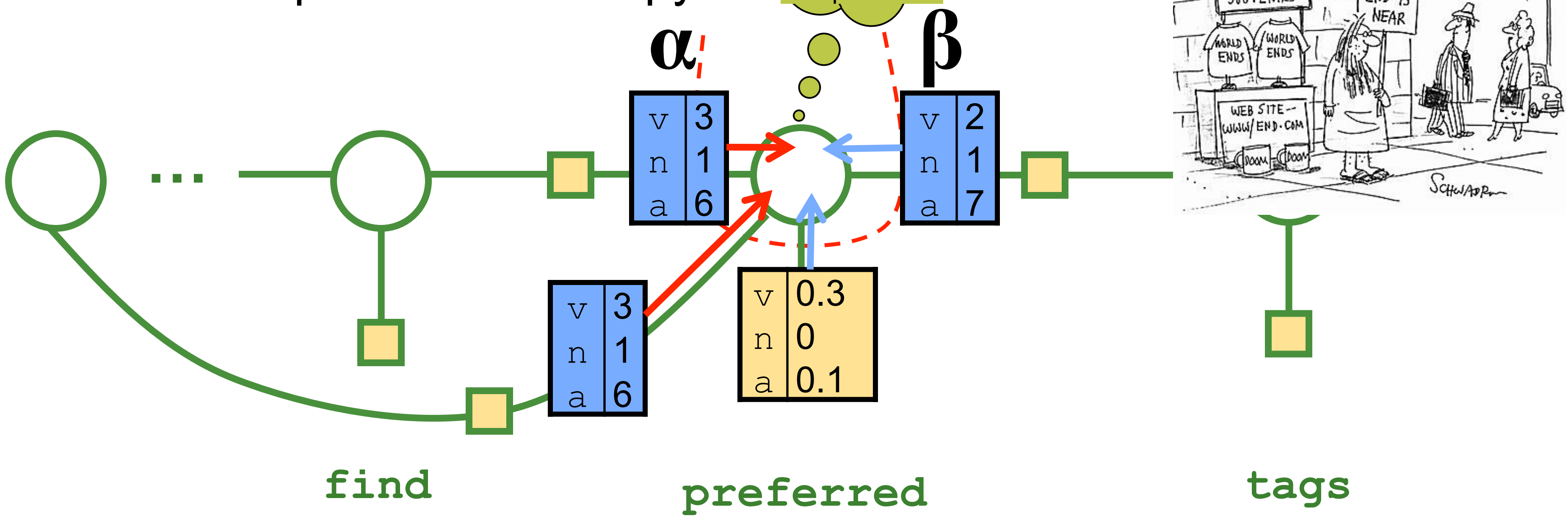
Great ideas in ML: Loopy Belief Propagation

- Extend CRF to “skip chain” to capture non-local factor
  - More influences on belief



Great ideas in ML: Loopy Belief Propagation

- Extend CRF to "skip chain" to capture non-local factor
  - More influences on belief
  - Graph becomes loopy ☹



Technique #4: Variational methods

- Mean-field approximation
- Belief propagation
- Survey propagation:
  - Like belief propagation, but also assess the belief that the value of this variable doesn't matter! Useful for solving hard random 3-SAT problems.
- Generalized belief propagation: Joins constraints, roughly speaking.
- Expectation propagation: More approximation when belief propagation runs too slowly.
- Tree-reweighted belief propagation: …