

Searching Possible Worlds

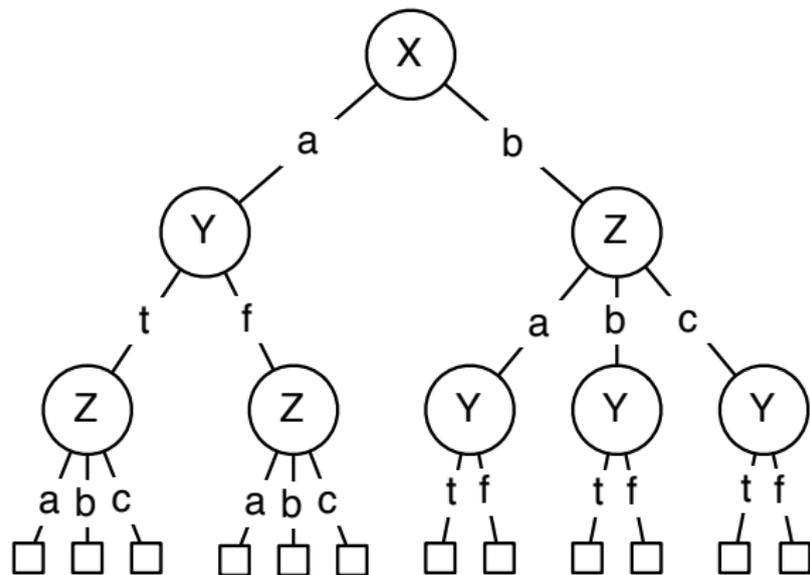
- Can we estimate the probabilities by only enumerating a few of the possible worlds?
- How can we enumerate just a few of the most probable possible worlds?
- Can we estimate the error in our estimates?
- Can we exploit the structure that variable elimination does?
- Can we exploit more structure?

The **search tree** has nodes labeled with variables, and is defined as follows:

- Each non-leaf node is labelled with a variable
- The arcs are labelled with values. There is a child for a node X for every value in the domain of X .
- A node cannot be labelled with the same label as an ancestor node.
- A path from the root corresponds to an assignment to a set of variables.
- In a full tree, every path from the root to a leaf contains all variables. The leaves correspond to possible worlds.

Example search tree

Suppose we have 3 variables, X with domain $\{a, b\}$, Y with domain $\{t, f\}$, and Z with domain $\{a, b, c\}$:



Basic Search Algorithm

$Q := \{\langle \rangle\};$

$W := \{\};$

While $Q \neq \{\}$ do

 choose and remove $\langle Y_1=v_1, \dots, Y_j=v_j \rangle$ from Q ;

 if $j = n$

$W \leftarrow W \cup \{\langle Y_1=v_1, \dots, Y_j=v_j \rangle\}$

 else

 Select a variable $Y_{j+1} \notin \{Y_1, \dots, Y_n\}$

$Q \leftarrow Q \cup \{\langle Y_1=v_1, \dots, Y_j=v_j, Y_{j+1}=v \rangle : v \in \text{dom}(Y_{j+1})\}$

Q is a set of paths from root to a leaf.

W is a set of generated possible worlds.

Properties of the Algorithm

- Each partial description can only be generated once. There is no need to check for multiple paths or loops in the search.
- The probability of a world W is

$$\prod_i P(X_i | \text{parents}(X_i))_W$$

- The cost of a partial assignment is the probability of the factors that are fully assigned by the partial assignment.

Estimating the Probabilities

Use W , at the start of an iteration of the while loop, as an approximation to the set of all possible worlds.

Let

$$P_W^g = \sum_{w \in W \wedge w \models g} P(w)$$

$$P_Q = 1 - P_W^{true}$$

Then

$$\leq P(g) \leq$$

Estimating the Probabilities

Use W , at the start of an iteration of the while loop, as an approximation to the set of all possible worlds.

Let

$$P_W^g = \sum_{w \in W \wedge w \models g} P(w)$$

$$P_Q = 1 - P_W^{true}$$

Then

$$P_W^g \leq P(g) \leq P_W^g + P_Q$$

Posterior Probabilities

Given the definition of conditional probability:

$$P(g|obs) = \frac{P(g \wedge obs)}{P(obs)}$$

We estimate the probability of a conditional probability:

$$\leq P(g|obs) \leq$$

Posterior Probabilities

Given the definition of conditional probability:

$$P(g|obs) = \frac{P(g \wedge obs)}{P(obs)}$$

We estimate the probability of a conditional probability:

$$\frac{P_W^{g \wedge obs}}{P_W^{obs} + P_Q} \leq P(g|obs) \leq \frac{P_W^{g \wedge obs} + P_Q}{P_W^{obs} + P_Q}$$

If we choose the midpoint as an estimate:

$$\text{Error} \leq$$

Posterior Probabilities

Given the definition of conditional probability:

$$P(g|obs) = \frac{P(g \wedge obs)}{P(obs)}$$

We estimate the probability of a conditional probability:

$$\leq P(g|obs) \leq$$

If we choose the midpoint as an estimate:

$$\text{Error} \leq \frac{P_Q}{2(P_W^{obs} + P_Q)}$$

As the computation progresses, the probability mass in the queue P_Q approaches zero.

Refinements

- We only need to consider the ancestors of the variables we are interested in. We can prune the rest before the search.
- When computing $P(\alpha)$, we prune partial descriptions if it can be determined whether α is true or false in that partial description.
- When computing $P(\bullet|OBS)$, we prune partial descriptions in which OBS is false.
- We want to generate the most likely possible worlds to minimize the error. One good search strategy is a depth-first search, pruning unlikely worlds.

Recursive Conditioning

- If an assignment disconnects the graph, each component can be evaluated separately.
- Computed values can be cached. We check the cache before evaluating any query.

Recursive Conditioning

```
procedure  $rc(Fs : \text{set of factors})$ :  
  if  $Fs = \{\}$  return 1  
  else if  $\exists v$  such that  $\langle Fs, v \rangle \in \text{cache}$   
    return  $v$   
  else if  $\exists F \in Fs$  such that  $\text{vars}(F) = \{\}$   
    return  $F \times rc(Fs \setminus F)$   
  else if  $Fs = Fs_1 \uplus Fs_2$  such that  $\text{vars}(Fs_1) \cap \text{vars}(Fs_2) = \{\}$   
    return  $rc(Fs_1) \times rc(Fs_2)$   
  else select variable  $X \in \text{vars}(Fs)$   
     $sum \leftarrow 0$   
    for each  $v \in \text{dom}(X)$   
       $sum \leftarrow sum + rc(\{F_{X=v} : F \in Fs\})$   
     $cache \leftarrow cache \cup \{\langle Fs, sum \rangle\}$   
  return  $sum$ 
```

Notes on the $rc(Fs)$ algorithm

- *cache* is a global variable that contains sets of pairs. It is initially empty.
- $vars(F)$ returns the unassigned variables in F
- $Fs = Fs_1 \uplus Fs_2$ is the disjoint union, meaning $Fs_1 \neq \{\}$, $Fs_2 \neq \{\}$, $Fs_1 \cap Fs_2 = \{\}$, $Fs = Fs_1 \cup Fs_2$