グラフィカルモデルにおける
厳密な推論

統計数理研究所　数理・推論研究系

坂田 綾香

ayaka@ism.ac.jp

2021年度「計算推論基礎」講義資料

参考文献

- 「確率的グラフィカルモデル」共立出版 １章 ⏩
  - ✦ 変数消去の基本的手続きについて説明されている
- Koller & Friedman, ”Probabilistic Graphical Models” 9章，10章
  - ✦ クリークツリーの作り方について詳しい説明がある
  - ✦ Lauritzen-Spiegelhalterアルゴリズムの説明が主
- Jordan, “An Introduction to Probabilistic Graphical Models” 3, 4, 17章 ⏩
  - ✦ HUGINアルゴリズムの説明が主
- Lepar & Shenoy, arXiv:1301.7394 ⏩
  - ✦ いろいろなアルゴリズムを比較している

目次

1. はじめに

グラフィカルモデルにおける推論

- 目標：周辺化分布，事後分布を評価する
  - 一般に変数消去と呼ばれる
  - グラフの構造を使って計算量を抑える方法を紹介
    - ✦グラフィカルモデルを使う利点のひとつ

1.1 変数消去とは

変数消去とは

注目する変数以外の変数の状態和をとり，

注目する変数に関する確率分布を得ることを**変数消去**とよぶ.

(1) **事前確率**における変数消去

- あるノード集合$A \subset V$についての確率分布は $P(\boldsymbol{X}_A) = \sum_{\boldsymbol{X}_{V \setminus A}} P\left(\boldsymbol{X}_A, \boldsymbol{X}_{V \setminus A}\right)$

(2) **事後確率**における変数消去

- あるノード集合$A, B, C \subset V$をdisjoint setとすると，
  evidence$\boldsymbol{X}_B = \boldsymbol{e}$が所与での確率分布は

$$P(\boldsymbol{X}_A | \boldsymbol{X}_B = \boldsymbol{e}) = \frac{\sum_{\boldsymbol{X}_C} P(\boldsymbol{X}_A, \boldsymbol{X}_B = \boldsymbol{e}, \boldsymbol{X}_C)}{\sum_{\boldsymbol{X}_A} \sum_{\boldsymbol{X}_C} P(\boldsymbol{X}_A, \boldsymbol{X}_B = \boldsymbol{e}, \boldsymbol{X}_C)}$$

… 分子だけでは規格化されていないので確率とみなせない

変数消去の計算量

- N個の変数$X_1, \ldots, X_N$があり，それぞれk通りの値を取りうるとする
- 例として，X_2の分布を得るには $p(X_2) = \sum_{\boldsymbol{X}_{\backslash 2}} p(\boldsymbol{X})$ を評価する必要がある.
  - ✦ X_2がとりうるk通りの値について$\boldsymbol{X}_{\backslash 2}$の和をとるので，$O(k \times k^{n-1})$の計算量
  - ✦ 愚直なアルゴリズムでは，指数関数オーダーの計算量が必要 (NP-hard)
- (一部)解決策：Chain ruleを使う

$$p(X_1, \ldots, X_N) = p(X_N | X_1, \ldots, X_{N-1}) \ldots p(X_3 | X_1, X_2) p(X_2 | X_1) p(X_1)$$

Chain ruleと条件付き独立性を使って計算を効率化する

具体例

- 4つの確率変数の同時確率分布$p(X_A, X_B, X_C, X_D)$からX_Dのみに関する確率分布を得たい

$$p(X_D) = \sum_{X_A} \sum_{X_B} \sum_{X_C} p(X_A, X_B, X_C, X_D)$$

- ベイジアンネットワークにより確率分布が次のように表現されている場合

$$p(X_A, X_B, X_C, X_D) = p(X_A)p(X_B|X_A)p(X_C|X_B)p(X_D|X_C)$$

$$X_A \rightarrow X_B \rightarrow X_C \rightarrow X_D$$

和の取り方を工夫することで計算量を抑えることができる

各変数が二値変数である場合について($X_A = a_1$ or a_2など)，次のページで見てみます．

BNによる計算量の削減

愚直に$p(d)$を求めるには

3×2^4回の積と$2\times(2^3-1)$回の和が必要

$$P(a^1)P(b^1|a^1)P(c^1|b^1)P(d^1|c^1)$$
$$+P(a^2)P(b^1|a^2)P(c^1|b^1)P(d^1|c^1)$$
$$+P(a^1)P(b^2|a^1)P(c^1|b^2)P(d^1|c^1)$$
$$+P(a^2)P(b^2|a^2)P(c^1|b^2)P(d^1|c^1)+\cdots$$

$$\{P(a^1)P(b^1|a^1)+P(a^2)P(b^1|a^2)\}P(c^1|b^1)P(d^1|c^1)$$
$$+\{P(a^1)P(b^2|a^1)+P(a^2)P(b^2|a^2)\}P(c^1|b^2)P(d^1|c^1)+\cdots$$

aの和をとる：

$\tau_1(b)=\sum_a P(a)P(b|a)$

(2回の和と4回の積)

$$\tau_1(b^1)P(c^1|b^1)P(d^1|c^1)+\tau_1(b^2)P(c^1|b^2)P(d^1|c^1)$$
$$+\tau_1(b^1)P(c^2|b^1)P(d^1|c^2)+\tau_1(b^2)P(c^2|b^2)P(d^1|c^2)+\cdots$$

計算量の削減：62回→18回

$$\{\tau_1(b^1)P(c^1|b^1)+\tau_1(b^2)P(c^1|b^2)\}P(d^1|c^1)$$
$$+\{\tau_1(b^1)P(c^2|b^1)+\tau_1(b^2)P(c^2|b^2)\}P(d^1|c^2)+\cdots$$

bの和をとる：

$\tau_2(c)=\sum_b P(b)P(c|b)$

(2回の和と4回の積)

$$\tau_2(c_1)P(d^1|c^1)+\tau_2(c_2)P(d^1|c^2)$$
$$+\tau_2(c_1)P(d^2|c^1)+\tau_2(c_2)P(d^2|c^2)$$

cの和をとり$p(d)$を得る

(2回の和と4回の積)

1.2 条件付き確率テーブル

Conditional Probability Table (CPT)

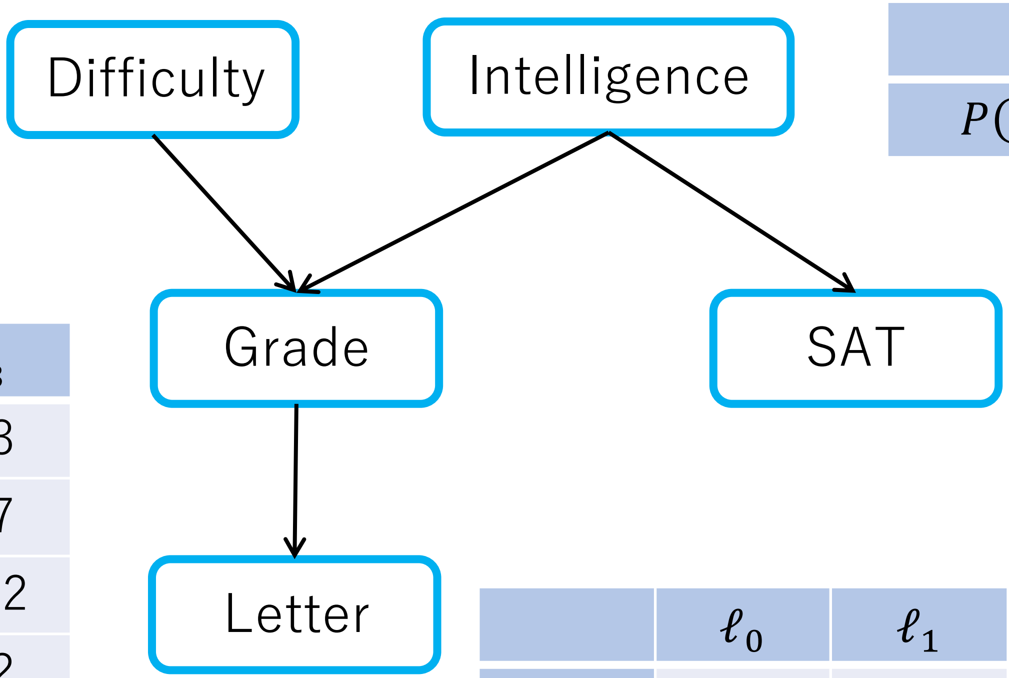
- BNを使って変数消去するには，条件付き確率が必要.
- ここでは条件付き確率が既知の場合を考える
- 条件付き確率一覧をConditional probability table (CPT)として表現することもある
- 条件付き確率を推定する場合もあるが，ここでは扱わない

「Probabilistic Graphical Models」16章，17章参照

Conditional Probability Table (CPT)

Student Bayesian networkの例

	$D = d_0$	$D = d_1$
$P(D)$	0.6	0.4



	$I = i_0$	$I = i_1$
$P(I)$	0.7	0.3

	g_1	g_2	g_3
i_0, d_0	0.3	0.4	0.3
i_0, d_1	0.05	0.25	0.7
i_1, d_0	0.9	0.08	0.02
i_1, d_1	0.5	0.3	0.2

	s_1	s_2
i_0	0.95	0.05
i_1	0.2	0.8

	ℓ_0	ℓ_1
g_1	0.3	0.4
g_2	0.05	0.25
g_3	0.9	0.08

変数消去は行列演算とみなせる

（例）：$P(I,D,L) = \sum_G P(I,D|G)P(G|L)$

	g_1	g_2	g_3
i_0, d_0	0.3	0.4	0.3
i_0, d_1	0.05	0.25	0.7
i_1, d_0	0.9	0.08	0.02
i_1, d_1	0.5	0.3	0.2

×

	ℓ_0	ℓ_1
g_1	0.3	0.4
g_2	0.05	0.25
g_3	0.9	0.08

=

	ℓ_0	ℓ_1
i_0, d_0	0.38	0.24
i_0, d_1	0.66	0.14
i_1, d_0	0.29	0.38
i_1, d_1	0.35	0.29

※ 小数点3桁で四捨五入

2. 事前確率における変数消去

定義：FactorとScope

- 確率変数$\boldsymbol{X}$が取りうる値を$\mathrm{val}[\boldsymbol{X}]$とする.
- Factorϕは次のように定義される関数である

$$\phi(\boldsymbol{X}): \mathrm{val}[\boldsymbol{X}] \rightarrow \mathbb{R}^{+}$$

- ϕに含まれる変数の集合をスコープとよび，$\mathrm{Scope}[\phi]$と表記する。

- Factorの例：✦ベイジアンネットワークにおける条件付き確率
  - 規格化されている

  ✦マルコフネットワークにおけるClique potential
  - 一般に規格化されていない

- 変数消去法はfactorに対する演算として表現できる

定義：周辺化 (Marginalization)

Factor $\phi(X_i, \boldsymbol{X}_{\setminus i})$を$X_i$について和をとることを

周辺化(marginalization) とよぶ.

$$\psi(\boldsymbol{X}_{\setminus i}) = \sum_{X_i} \phi(X_i, \boldsymbol{X}_{\setminus i})$$

- またはsumming out of X_iという。

Sum-Product

- 4変数の確率分布をファクターで表現する

$$P(X_A, X_B, X_C, X_D) = P(X_A)P(X_B|X_A)P(X_C|X_B)P(X_D|X_C)$$
$$= \phi_A(A)\phi_B(B,A)\phi_c(C,B)\phi_D(D,C)$$

✦ 全てのFactor集合をΦとすると,

$$P(X_A, X_B, X_C, X_D) = \prod_{\phi \in \Phi} \phi$$

- X_D以外の変数の周辺化は次のように表現される

$$P(D) = \sum_C \sum_B \sum_A P(A,B,C,D) = \sum_C \phi_D(D,C) \left(\sum_B \phi_C(C,B) \left(\sum_A \phi_B(B,A)\phi_A(A) \right) \right)$$

→ **Sum-Productアルゴリズム**として一般に表現できる

Sum-Product variable elimination algorithm

Input
- Φ: 全てのファクターのセット
- v: 消去したい変数の番号のセット$(|v| = k)$ ◁ vのi番目の要素をv_iとする

Output
- ϕ^*: 変数消去されたファクター $(X_v \not\subset \mathrm{Scope}[\phi^*])$

For $i = 1, \ldots, k$

$\Phi_{\mathrm{el}} \leftarrow \{\phi \in \Phi : X_{v_i} \in \mathrm{Scope}[\phi]\}$ ◁ 消去したい変数が入っているファクター

$\Phi_{\mathrm{re}} \leftarrow \Phi \setminus \Phi_{\mathrm{el}}$ ◁ 消去したい変数が入っていないファクター

$\psi \leftarrow \prod_{\phi \in \Phi_{\mathrm{el}}} \phi$ ◁ 消去したい変数が入っているファクターを掛け合わせる

$\tau \leftarrow \sum_{X_{v_i}} \psi$ ◁ 変数の和をとる

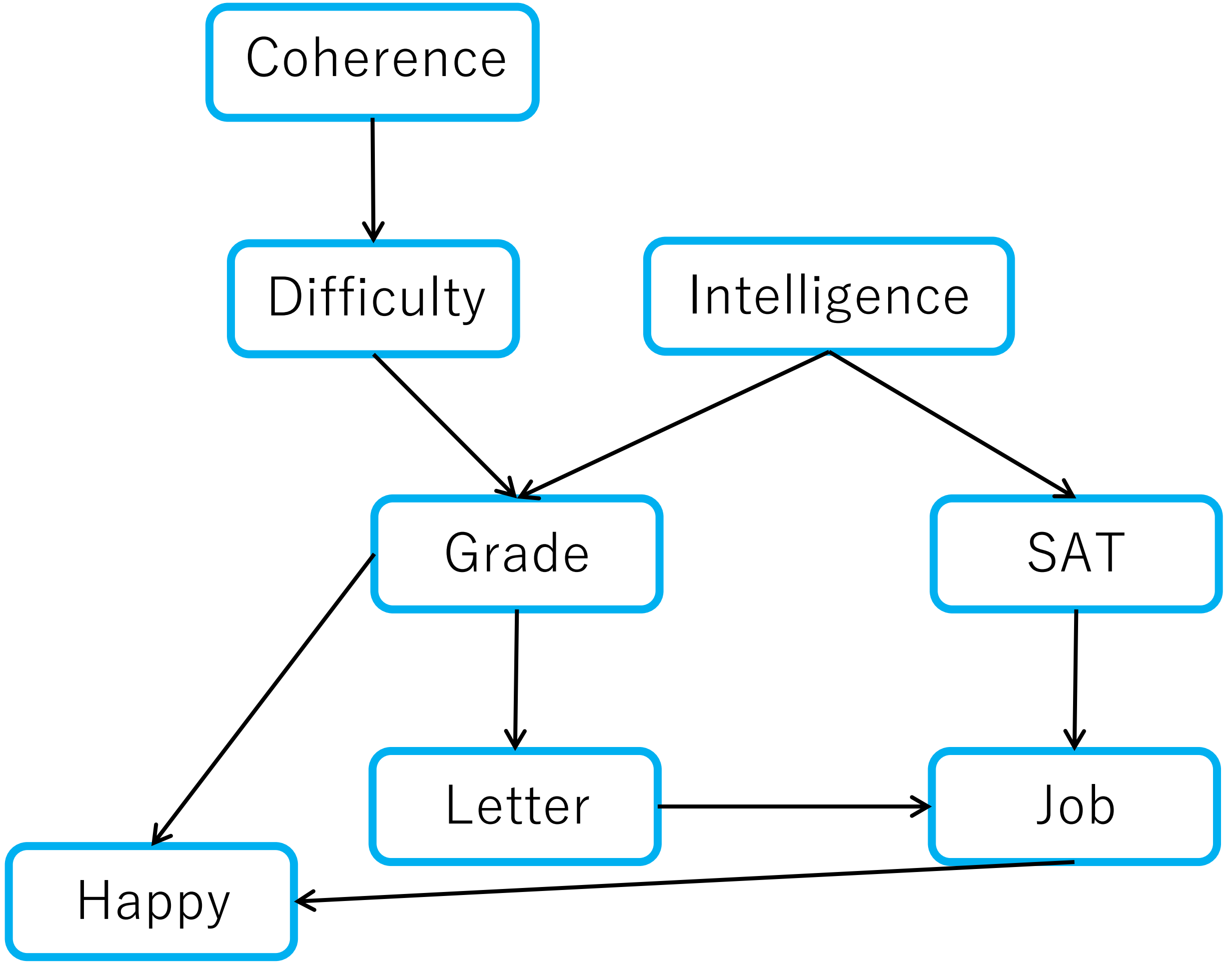
$\Phi \leftarrow \Phi_{\mathrm{re}} \cup \tau$ ◁ 新しいファクターのセット

end For

$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$ ◁ 変数消去されたファクターを掛け合わせる

VEアルゴリズムをBNに適用してみる

- Extended-Student Bayesian network



- $\Phi = \{p(X_i | X_{\pi(i)}) | i = 1, \ldots, N\}$
- $\pi(i)$はparent node

以下では，簡単のため
各ノードを頭文字で表す

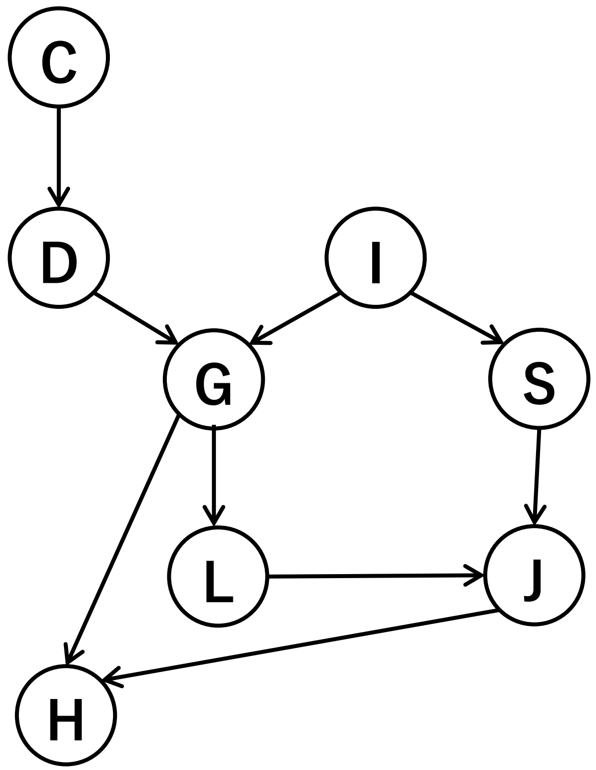
- BNの確率分布

$$P(C,D,I,G,S,L,J,H) = P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|S,L)P(H|G,J)$$
$$= \phi_C(C)\phi_D(D,C)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I)\phi_L(L,G)\phi_J(J,S,L)\phi_H(H,G,J)$$

- VEアルゴリズムにより$P(J)$を評価する

Eliminationの順番は**C → D → I → H → G → S → L**とする

1. **C**のElimination
   - ✦ $\psi_1(C,D) = \phi_C(C)\phi_D(D,C)$
   - ✦ $\tau_1(D) = \sum_C \psi_1(C,D)$
2. **D**のElimination
   - ✦ $\psi_2(G,I,D) = \phi_G(G,I,D)\tau_1(D)$
   - ✦ $\tau_2(G,I) = \sum_D \psi_2(G,I,D)$
3. **I**のElimination
   - ✦ $\psi_3(G,I,S) = \phi_I(I)\phi_S(S,I)\tau_2(G,I)$
   - ✦ $\tau_3(G,S) = \sum_I \psi_3(G,I,S)$



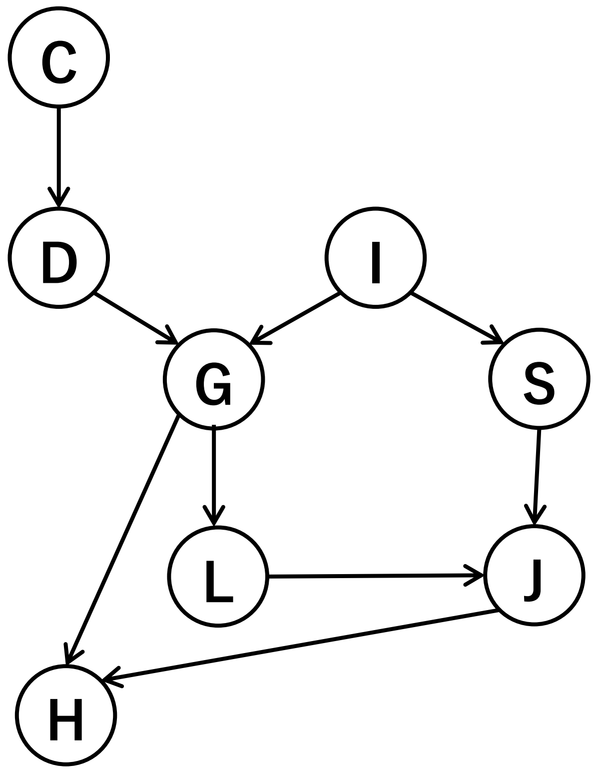
- BNの確率分布

$$P(C,D,I,G,S,L,J,H) = P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|S,L)P(H|G,J)$$
$$= \phi_C(C)\phi_D(D,C)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I)\phi_L(L,G)\phi_J(J,S,L)\phi_H(H,G,J)$$

- VEアルゴリズムにより$P(J)$を評価する

Eliminationの順番は**C → D → I → H → G → S → L**とする

4. **H**のElimination
   - ✦ $\psi_4(H,G,J) = \phi_H(H,G,J)$
   - ✦ $\tau_4(G,J) = \sum_H \psi_4(H,G,J) = 1$
5. **G**のElimination
   - ✦ $\psi_5(G,J,L,S) = \phi_L(L,G)\tau_3(G,S)\tau_4(G,J)$
   - ✦ $\tau_5(J,L,S) = \sum_G \psi_5(G,J,L,S)$
6. **S**のElimination
   - ✦ $\psi_6(J,L,S) = \phi_J(J,S,L)\tau_5(J,L,S)$
   - ✦ $\tau_6(J,L) = \sum_S \psi_6(J,L,S)$



- BNの確率分布

$$P(C,D,I,G,S,L,J,H) = P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|S,L)P(H|G,J)$$
$$= \phi_C(C)\phi_D(D,C)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I)\phi_L(L,G)\phi_J(J,S,L)\phi_H(H,G,J)$$

- VEアルゴリズムにより$P(J)$を評価する

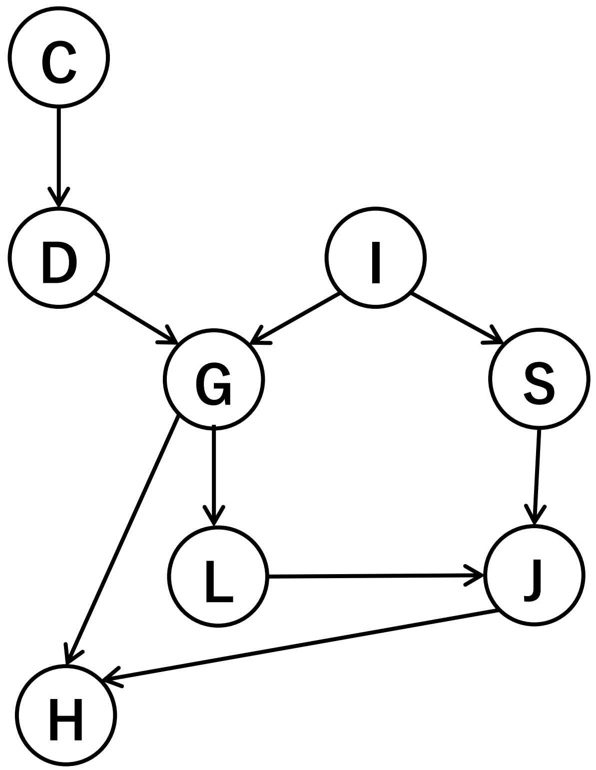
Eliminationの順番は**C → D → I → H → G → S → L**とする

7. **L**のElimination

✦ $\psi_7(J,L) = \tau_6(J,L)$

✦ $\tau_7(J) = \sum_L \psi_7(H,G,J)$

計算の手続きはEliminationの順番(**Elimination ordering**)に依存する



変数消去の順番とファクターのスコープの大きさ

Sum-Product variable elimination algorithm ⏪ より抜粋

$\Phi_{\mathrm{el}} \leftarrow \{\phi \in \Phi : X_{v_i} \in \mathrm{Scope}[\phi]\}$ ◁ 消去したい変数が入っているファクター

$\Phi_{\mathrm{re}} \leftarrow \Phi \setminus \Phi_{\mathrm{el}}$ ◁ 消去したい変数が入っていないファクター

$\psi \leftarrow \prod_{\phi \in \Phi_{\mathrm{el}}} \phi$ ◁ 消去したい変数が入っているファクターを掛け合わせる

Scope[ψ]の全状態に対応するψの値を評価するためには
$O(\exp|\mathrm{Scope}[\Phi_{\mathrm{el}}]|)$回の掛け算が必要

- 例えば全変数が二値をとる場合，$2^{|\mathrm{Scope}[\Phi_{\mathrm{el}}]|} \times |\Phi_{\mathrm{el}}|$回の掛け算が必要

Scopeの大きさを抑えるElimination orderingを採用して計算量を抑える

→ 次ページ以降で，グラフの構造からelimination orderingを決める方法を紹介します．

2.1 Elimination orderingの構成

準備

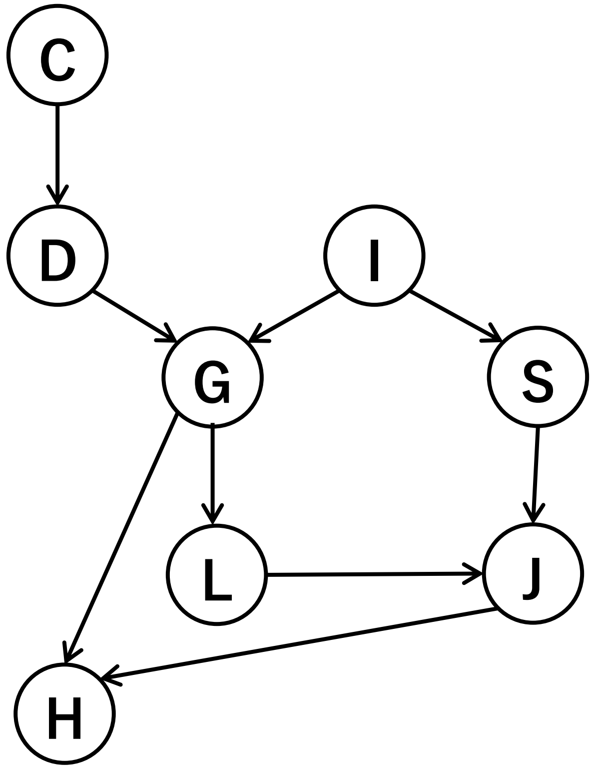
- ファクター集合Φのscopeを$\mathrm{Scope}[\Phi] = \cup_{\phi\in\Phi} \mathrm{Scope}[\phi]$とする.

- ファクター集合の**実効グラフ**とよぶ無向グラフ$\mathcal{G}_{\Phi} = \{\mathcal{V}, E\}$を定義する.
  - $\mathcal{V} = \mathrm{Scope}[\Phi]$
  - $\{X_i, X_j\} \in \mathrm{Scope}[\phi]$となるような$\phi \in \Phi$が1つでもあったら$E_{ij} \in \mathcal{E}$

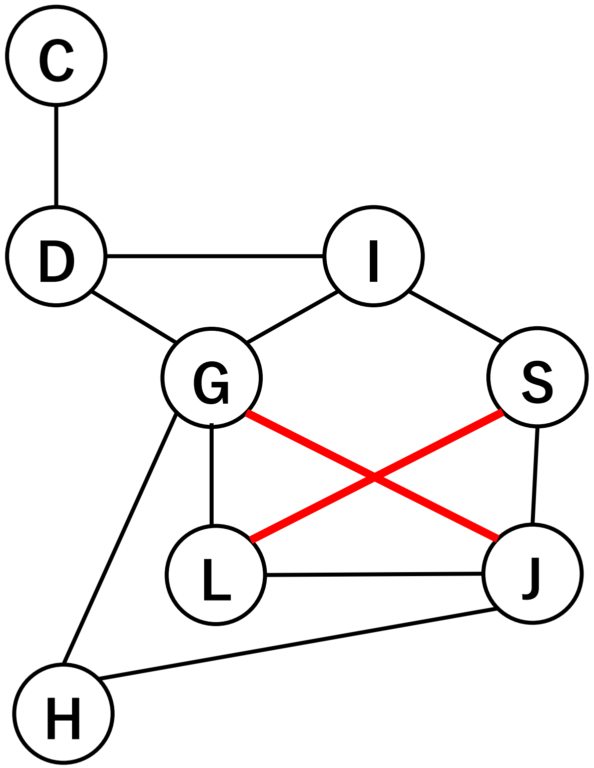
$\mathcal{G}_\Phi$の例

$$P(C,D,I,G,S,L,J,H) = P(C)P(D|C)P(I)P(G|I,D)P(S|I)P(L|G)P(J|S,L)P(H|G,J)$$

$$= \phi_C(C)\phi_D(D,C)\phi_I(I)\phi_G(G,I,D)\phi_S(S,I)\phi_L(L,G)\phi_J(J,S,L)\phi_H(H,G,J)$$



$\mathcal{G}$



$\mathcal{G}_\Phi$

定理：実効グラフはMinimal I-map

$\boldsymbol{X} = \mathrm{Scope}[\Phi]$とする．

Factorから定義される確率分布$P(X) = \frac{1}{Z}\prod_{\phi\in\Phi}\phi \ \left(Z = \sum_X \prod_{\phi\in\Phi}\phi\right)$について，

Markov Network $\{\mathcal{G}_\Phi, p_{\mathcal{G}_\Phi}\}$は$P$のminimal I-mapである．

- 特にBayesian network $\mathcal{G}$から定義されるファクター集合Φについて，
  無向グラフ$\mathcal{G}_\Phi$は$\mathcal{G}$のモラルグラフである．

- 実効グラフをもとに，Elimination Orderingを考える．

定義：Fill edge

ファクター集合ΦからXを消去をして得られたファクター集合を$\Phi_{\backslash X}$とする.

また対応する実効的グラフ$\mathcal{G}_{\phi}$と$\mathcal{G}_{\Phi_{\backslash X}}$とする.

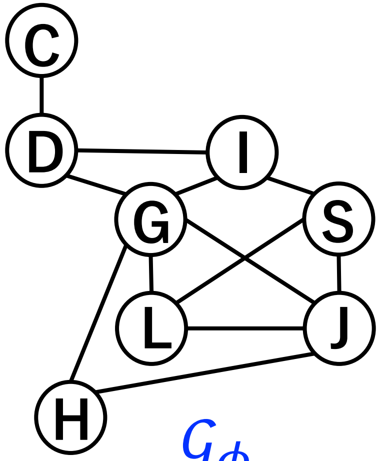
$\mathcal{G}_{\Phi_{\backslash X}}$が$\mathcal{G}_{\phi}$に含まれないedgeをもつとき，これをfill edgeとよぶ.

(例) extended student networkで**C** → **D** → **I** → と消去する場合,

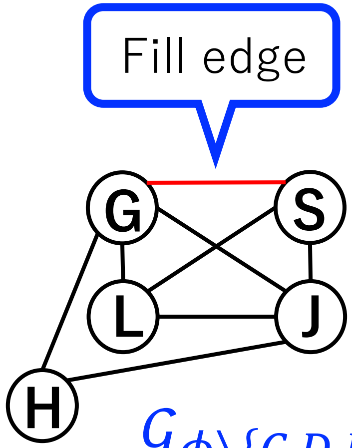
IのEliminationによりファクター τ_3が生じる

- ✦ $\psi_3(G,I,S) = \phi_I(I)\phi_S(S,I)\tau_2(G,I)$
- ✦ $\tau_3(G,S) = \sum_I \psi_3(G,I,S)$

よって$\mathcal{G}_{\Phi_{\backslash\{C,D,I\}}}$はfill edge $G-S$を持つ.

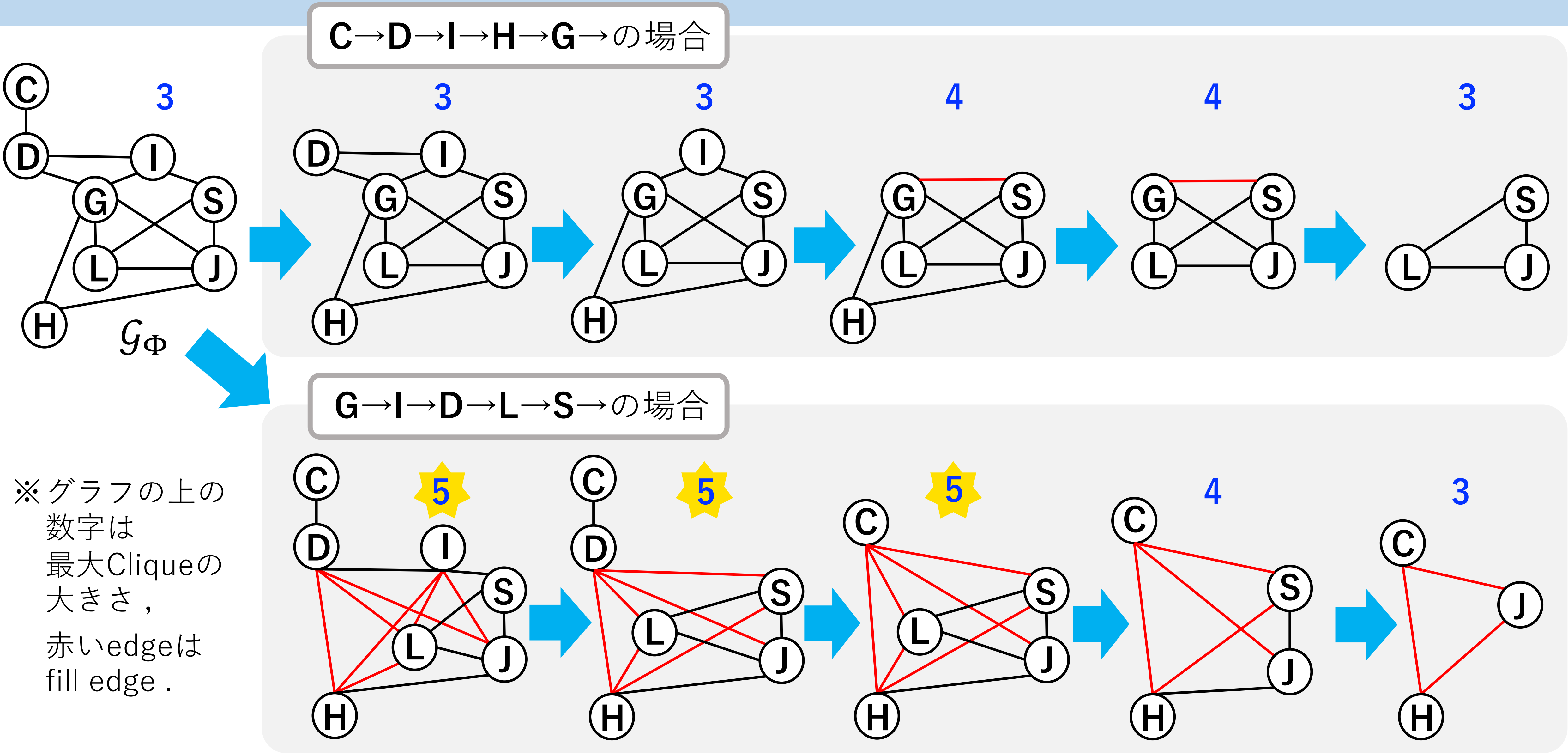


$\mathcal{G}_{\phi}$



$\mathcal{G}_{\phi\backslash\{C,D,I\}}$

Elimination Ordering依存性



※グラフの上の数字は最大Cliqueの大きさ，赤いedgeはfill edge .

定義：Induced graph

ファクター集合をΦ，elimination orderingを$\prec$とする．

VE algorithmにより，$\prec$の通りに変数消去を行う際に生じた

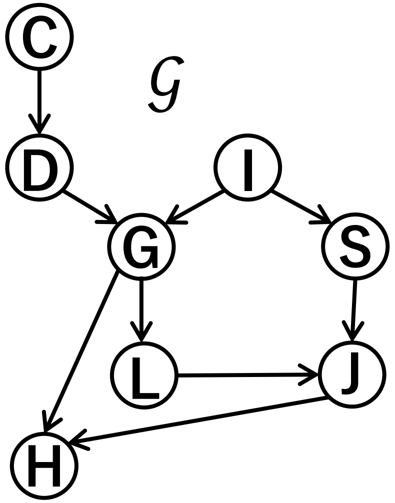
全てのFill edgeを$\mathcal{G}_{\Phi}$に加えたグラフをInduced graphとよび$\mathcal{I}_{\Phi,\prec}$と表す．

- $\prec$にしたがって，$1\sim i$番目に消去される変数を$\prec(i)$と表記する．
- グラフ$\mathcal{G}$のedge集合を$\mathcal{E}_{\mathcal{G}}$と表すと，

$$\mathcal{E}_{\mathcal{I}_{\Phi,\prec}} = \mathcal{E}_{\mathcal{G}_{\Phi}} \cup \mathcal{E}_{\mathcal{G}_{\Phi\backslash\prec(1)}} \cup \mathcal{E}_{\mathcal{G}_{\Phi\backslash\prec(2)}} \cup \cdots \cup \mathcal{E}_{\mathcal{G}_{\Phi\backslash\prec(N)}}$$

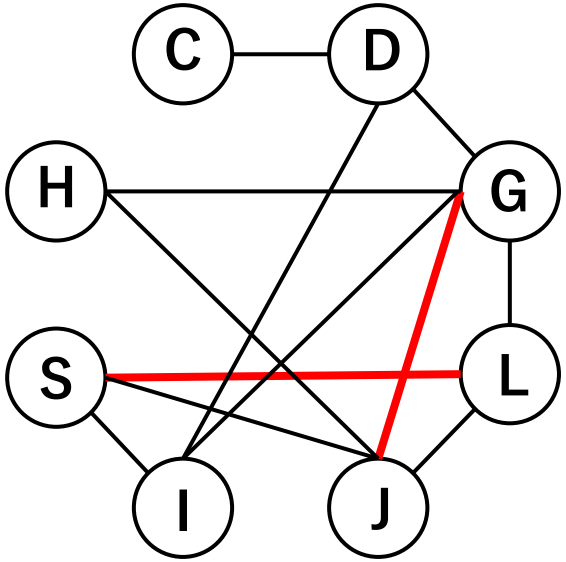
Induced graphのOrderingへの依存性

- 赤：モラル化により追加されたedge
- 青：fill edges



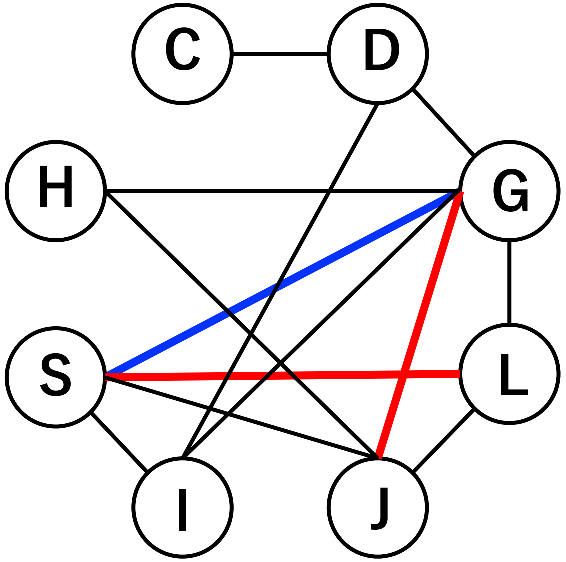
$\mathcal{G}$

$\mathcal{G}_\Phi$



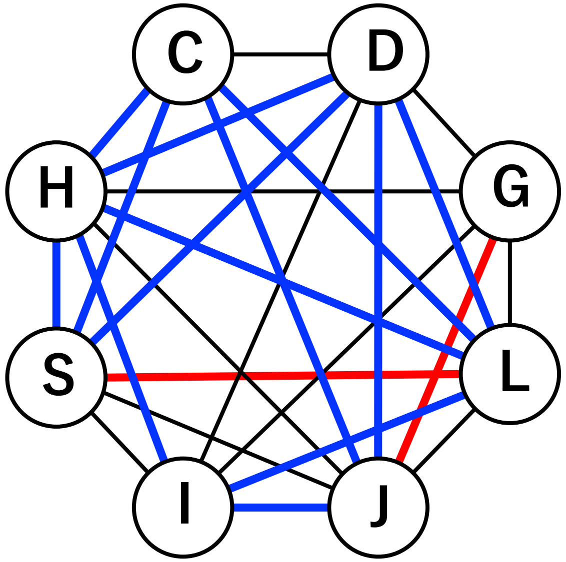
$\mathcal{J}_{\Phi,\prec}$

$\prec$ ：C → D → I → H → G → S → L



$\mathcal{J}_{\Phi,\prec}$

$\prec$ ：G→I→D→ L→S→C→H



※ 比較のため八角形状で表す

定理：Induced graphにおけるClique

Factor set ΦとElimination ordering $\prec$ のもとでの

Induced graphを$\mathcal{I}_{\Phi,\prec}$とする.

1. 変数消去過程において現れるすべてのfactorのscopeは,

   $\mathcal{I}_{\Phi,\prec}$ においてクリークを構成する.

2. $\mathcal{I}_{\Phi,\prec}$における全てのクリークは,

   変数消去過程のある中途ステップで現れるfactorのscopeである.

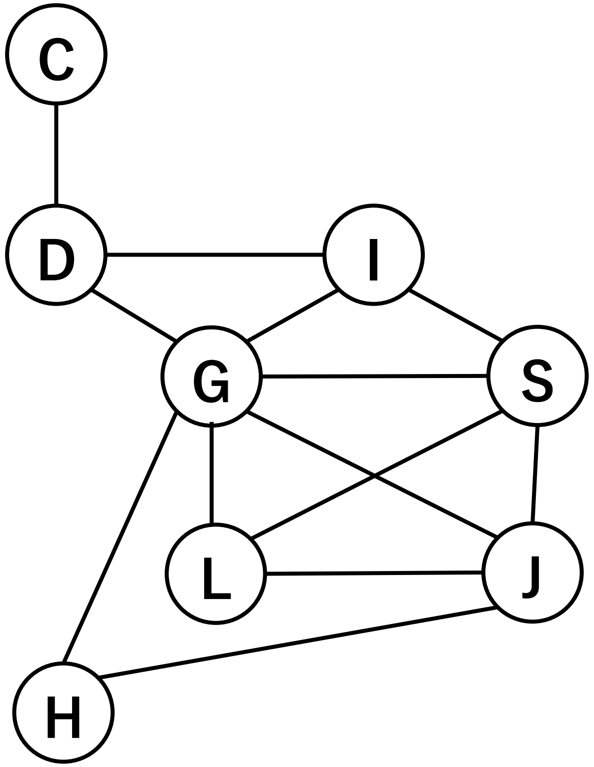
◆ 証明 ◆

1. Induced graphの定義から明らか.

2. 変数消去過程から明らか.（変数消去過程が進めば，factorの最大クリークは小さくなる）

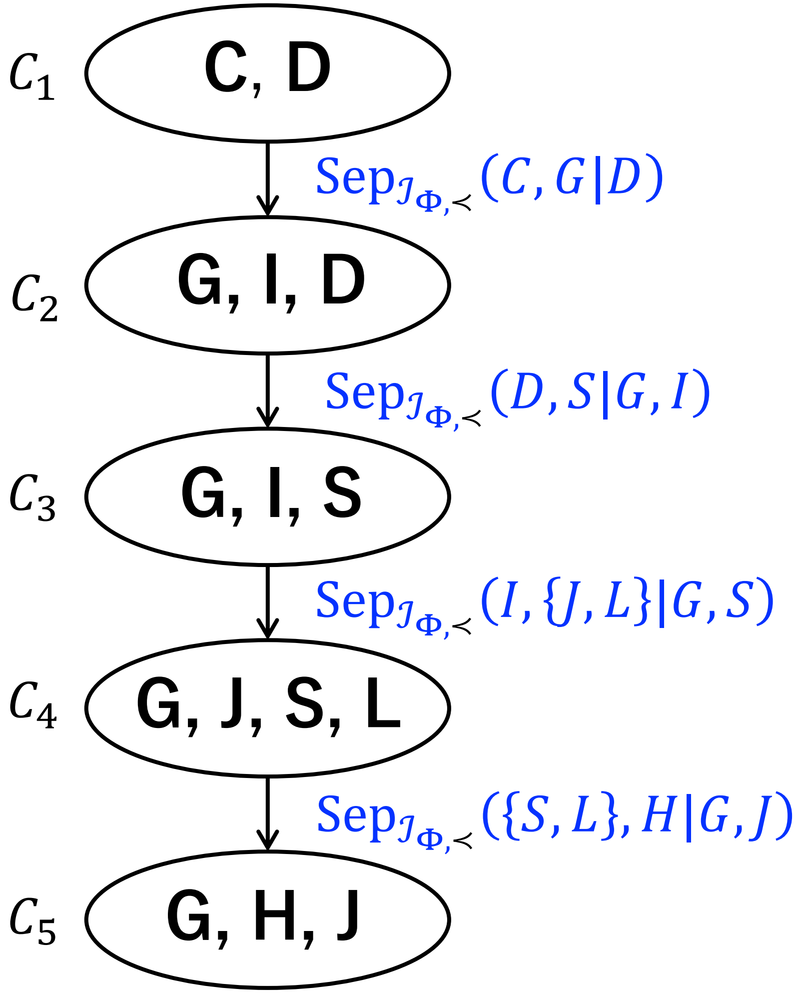
Clique tree for Induced Graph

$\prec$: $C \rightarrow D \rightarrow I \rightarrow H \rightarrow G \rightarrow S \rightarrow L$



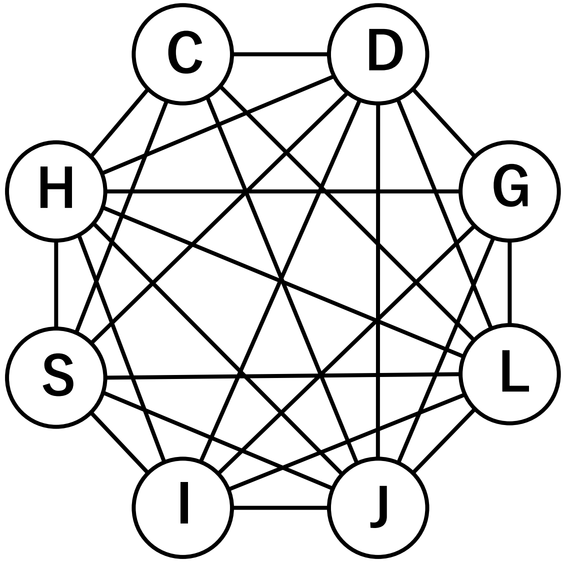
$\mathcal{J}_{\Phi,\prec}$

Clique Tree T



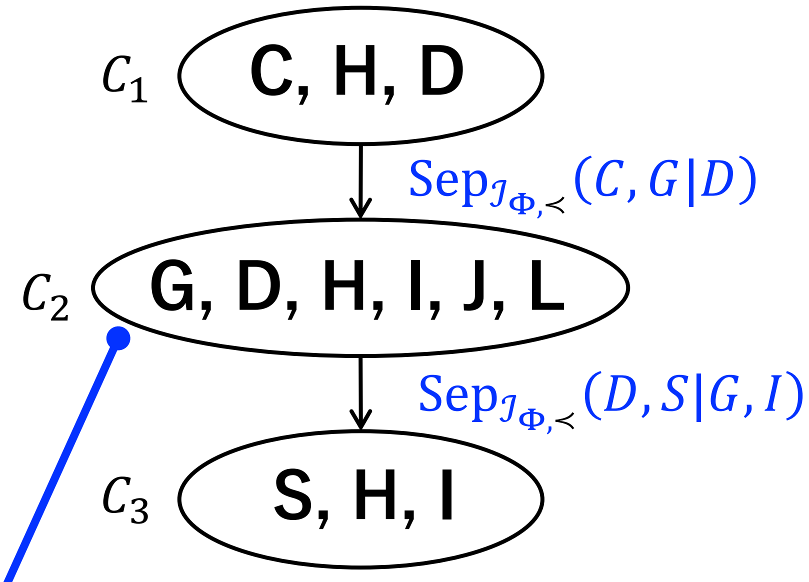
Clique tree for Induced Graph

$\prec$: G → I → D → L → S → C → H→



$\mathcal{J}_{\Phi,\prec}$

Clique Tree T



クリークが大きいので計算量が多い！

クリークサイズを小さくする

- Induced graphの最大クリークのノード数−１を**Induced width**とよぶ.
- 次の問題はNP-hardであることが知られる.

「グラフGと，あるKが与えられたとき，
induced widthがK以下になるelimination orderingがあるかどうか決定せよ」

- すなわち，optimalなelimination orderingを与えることはNP-hardである.
- Induced graphの性質や発見的手法を使えば
（最適とまではいかないかもしれないが）良いorderingを見つけることはできる.

VEの計算量を少なくするelimination orderingの構成方法

ここでは以下の方法を扱う

1. ファクター集合Φのもとで定義される実効的無向グラフ$\mathcal{G}_\Phi$を含むコーダルグラフ$\mathcal{H}_\Phi$を作る.
   - その際，largest cliqueが最も小さくなるようにtriangulation※する.
2. $\mathcal{H}_\Phi$を使ってelimination orderingを作る.

※ クリークサイズを小さくするtriangulationを見つける方法はNP-hardであるが,
いくつかの方法が知られている

- レビュー論文：Heggernes, Discrete Mathematics (2006) ⏩

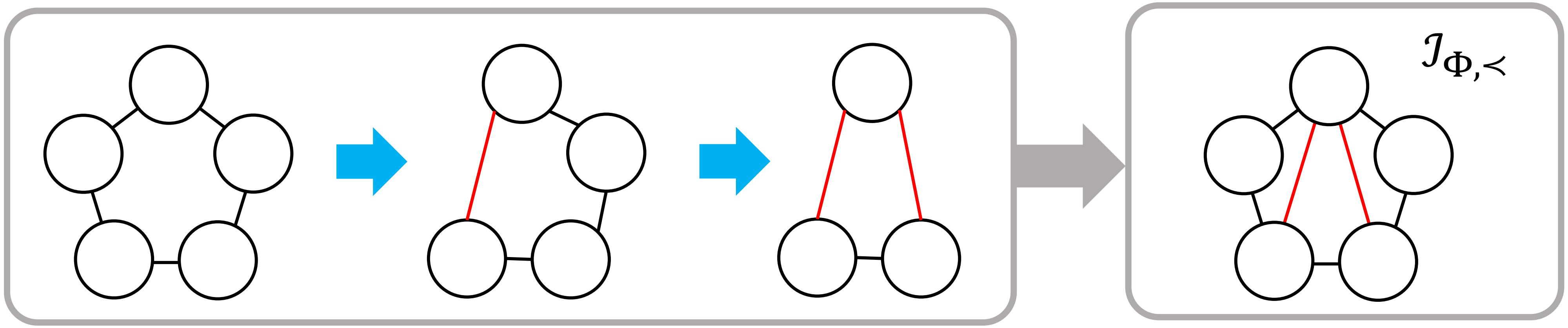
定理：$\prec \Rightarrow$コーダルグラフ

あらゆるElimination ordering $\prec$とFactor集合$\boldsymbol{\Phi}$について，

Induced graph $\mathcal{I}_{\boldsymbol{\Phi},\prec}$はコーダルグラフである．

◆ 証明 ◆

- $\mathcal{G}_{\boldsymbol{\Phi}}$に長さ$k$のループがあるとき，ループに属す変数のeliminationにより加わるfill edgeはInduced graphにおいてサイズ3のクリークと長さ$k-1$のループをつくる
- ループが長さ3になるまで，eliminationによりfill edgeが加わる
- よってInduced graphには，長さ3以下のループしか存在しない



定理：コーダルグラフ⇒ $\prec$

あらゆるコーダルグラフ$\mathcal{H}$は，変数消去によりグラフにfill edgeを加えない

Elimination ordering $\prec$をもつ.

つまり，コーダルグラフ$\mathcal{H}$は,

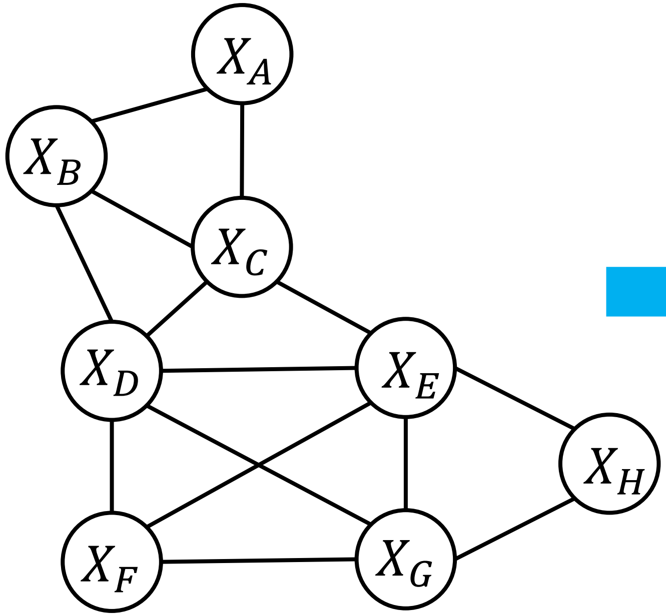
あるElimination ordering $\prec$のInduced graphに対応する.

◆ 証明 ◆

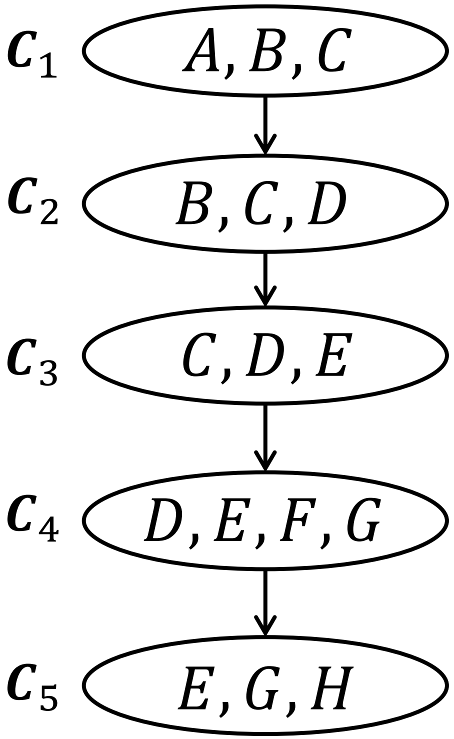
- コーダルグラフ$\mathcal{H}$はクリークツリー$\mathcal{T}$で表現される.
- Leafに対応するクリーク$\boldsymbol{C}_k$は，隣接クリークに含まれない変数X_iを持つ.
- クリークの定義よりX_iは$\boldsymbol{C}_k \setminus X_i$と全結合なので，$X_i$の消去によりfill edgeは生じない.
- X_iを消去したグラフを$\mathcal{H}'$もコーダルグラフであるので,
  同様にLeafに対応するクリークの中で隣接クリークに含まれない変数を消去すれば良い.

証明を絵で描くと

Chordal Graph $\mathcal{H}$



Clique Tree $\mathcal{T}$



Elimination Ordering

X_Aの和をとると$\boldsymbol{C}_1$が消える

↓

X_Bの和をとると$\boldsymbol{C}_2$が消える

↓

X_Cの和をとると$\boldsymbol{C}_3$が消える

↓

X_D, X_Fの和をとると$\boldsymbol{C}_4$が消える

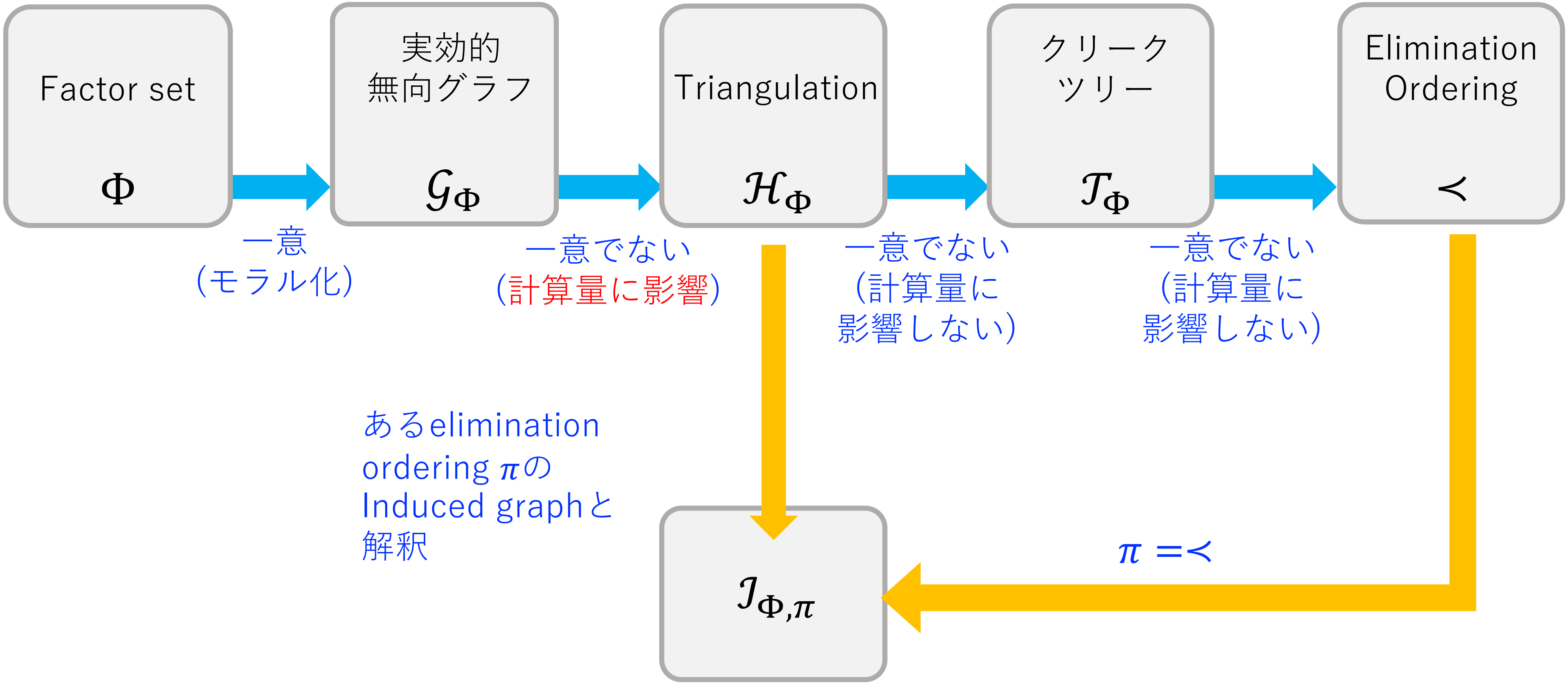
↓

X_E, X_G, X_Hの和をとると$\boldsymbol{C}_5$が消える

$\mathcal{H}$は$\prec = A \to B \to C \to \boxed{D \to F} \to \boxed{E \to G \to H}$のInduced Graphに対応

※ $\boxed{}$ の中で入れ替え可能

まとめ：elimination orderingの構成方法



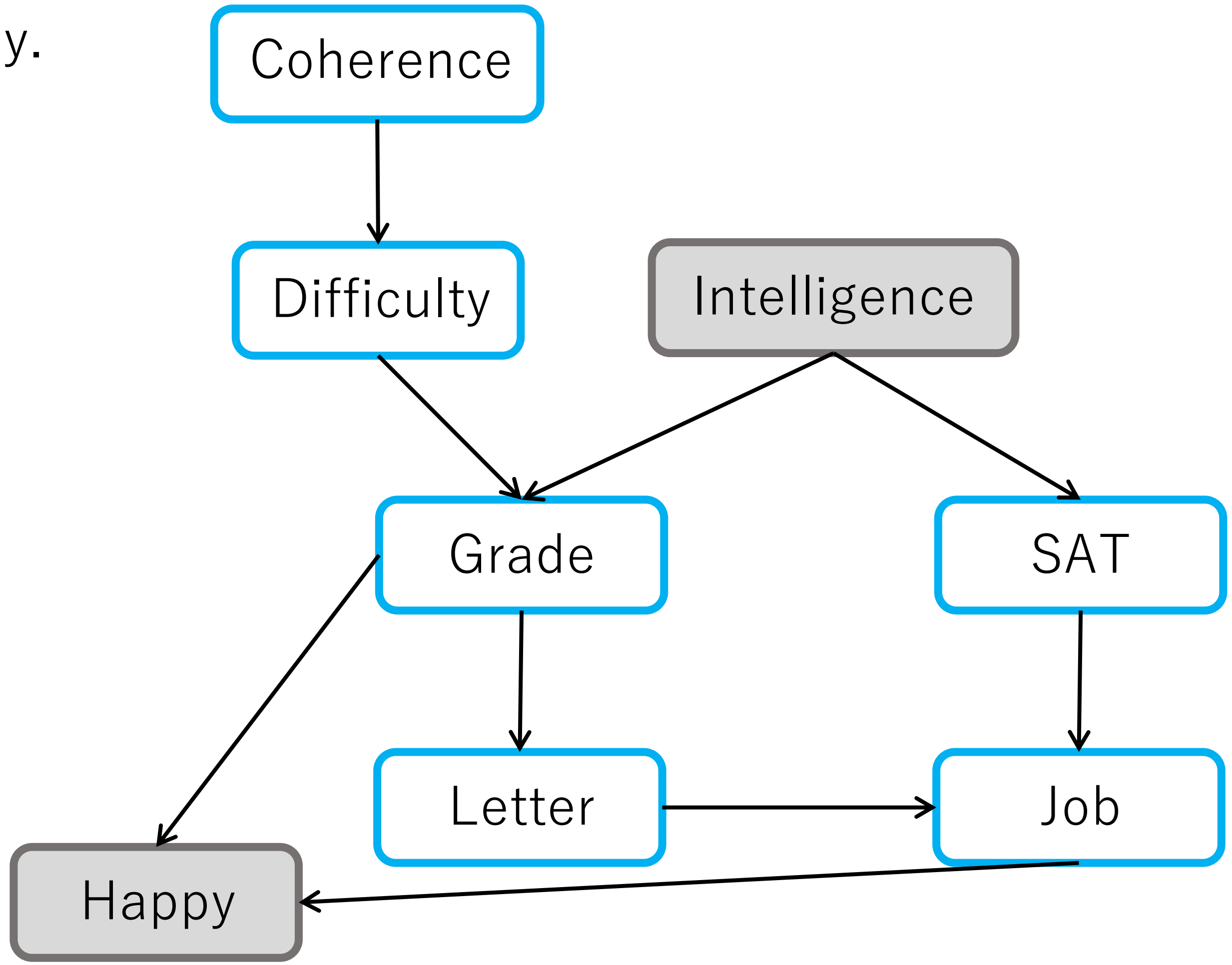
2.3. 事後確率における変数消去

事後確率とは

- エビデンスを得た後の確率分布を事後確率と呼ぶ
- 例：$\boldsymbol{X}_B = \boldsymbol{e}$というエビデンスを得たとする $(B \subset \mathcal{V})$
  - ✦ $P\left(\boldsymbol{X}_{\mathcal{V}\backslash B}, \boldsymbol{X}_B = \boldsymbol{e}\right)$は確率ではない（規格化されていない）
  - ✦ $P\left(\boldsymbol{X}_{\mathcal{V}\backslash B} \middle| \boldsymbol{X}_B = \boldsymbol{e}\right) = \dfrac{P(\boldsymbol{X}_{\mathcal{V}\backslash B}, \boldsymbol{X}_B = \boldsymbol{e})}{P(\boldsymbol{X}_B = \boldsymbol{e})}$ とすれば確率になる
    - $P(\boldsymbol{X}_B = \boldsymbol{e}) = \sum_{\boldsymbol{X}_{\mathcal{V}\backslash B}} P\left(\boldsymbol{X}_{\mathcal{V}\backslash B}, \boldsymbol{X}_B = \boldsymbol{e}\right)$ を評価する必要がある

Evidenceを得たあとの確率を評価する

- $I = i_1, H = h_0$を観測したとする
  - The student is intelligent and unhappy.
- このとき$P(J|I = i_1, H = h_0)$を評価したい
- Evidence によりconditionされたBNの確率分布はGibbs分布により表現できるため，事後確率の評価にマルコフネットワークが使える(次ページの補題参照)



補題：Conditioned BN and MN

Bayesian network $\{\mathcal{D}, p_{\mathcal{B}}(\boldsymbol{X})\}$において，観測$\boldsymbol{X}_A = \boldsymbol{e}$が得られたとする．
このとき$\boldsymbol{X}_{\mathcal{V}\setminus A}$の確率分布$p_{\mathcal{B}}\left(\boldsymbol{X}_{\mathcal{V}\setminus A} \middle| \boldsymbol{X}_A = \boldsymbol{e}\right)$は，
次の特徴を持つGibbs分布により与えられる．

- ファクター$\phi_i = p_{\mathcal{B}}\left(X_i \middle| \boldsymbol{X}_{\pi(i)\setminus A}, \{X_j = e_j\colon j \in \pi(i) \text{ and } j \in A\}\right)$からなる
- 分配関数は$p_{\mathcal{B}}(\boldsymbol{X}_A = \boldsymbol{e})$と一致

◆ 証明 ◆

$$p_{\mathcal{B}}\left(\boldsymbol{X}_{V\setminus A}, \boldsymbol{X}_A = \boldsymbol{e}\right) = \prod_{i \in A} p_{\mathcal{B}}\left(X_i = e_i \middle| \boldsymbol{X}_{\pi(i)\setminus A}, \{X_j = e_j\colon j \in \pi(i) \text{ and } j \in A\}\right)$$
$$\times \prod_{i \in V\setminus A} p_{\mathcal{B}}\left(X_i \middle| \boldsymbol{X}_{\pi(i)\setminus A}, \{X_j = e_j\colon j \in \pi(i) \text{ and } j \in A\}\right)$$

$\boldsymbol{X}_{V\setminus A}$の分布として規格化されていないので，分配関数$\sum_{X_{V\setminus A}} p_{\mathcal{B}}\left(\boldsymbol{X}_{V\setminus A}, \boldsymbol{X}_A = \boldsymbol{e}\right) = p_{\mathcal{B}}(\boldsymbol{X}_A = \boldsymbol{e})$が必要

Sum-Product VE for conditional probabilities

Input
- Φ : Evidence所与のもとでの全てのファクターのセット
- ε: Evidenceを与えられた変数のラベルセット
- v : 消去したい変数の番号のセット($|v| = k$)

Output
- ϕ^*: 変数消去されたファクター ($X_v \not\subset \mathrm{Scope}[\phi^*]$)
- Z : 分配関数

For $i = 1, \ldots, k$

$\Phi_{\mathrm{el}} \leftarrow \{\phi \in \Phi : X_{v_i} \in \mathrm{Scope}[\phi]\}$ ◁ 消去したい変数が入っているファクター

$\Phi_{\mathrm{re}} \leftarrow \Phi \setminus \Phi_{\mathrm{el}}$ ◁ 消去したい変数が入っていないファクター

$\psi \leftarrow \prod_{\phi \in \Phi_{\mathrm{el}}} \phi$ ◁ 消去したい変数が入っているファクターを掛け合わせる

$\tau \leftarrow \sum_{X_{v_i}} \psi$ ◁ 変数の和をとる

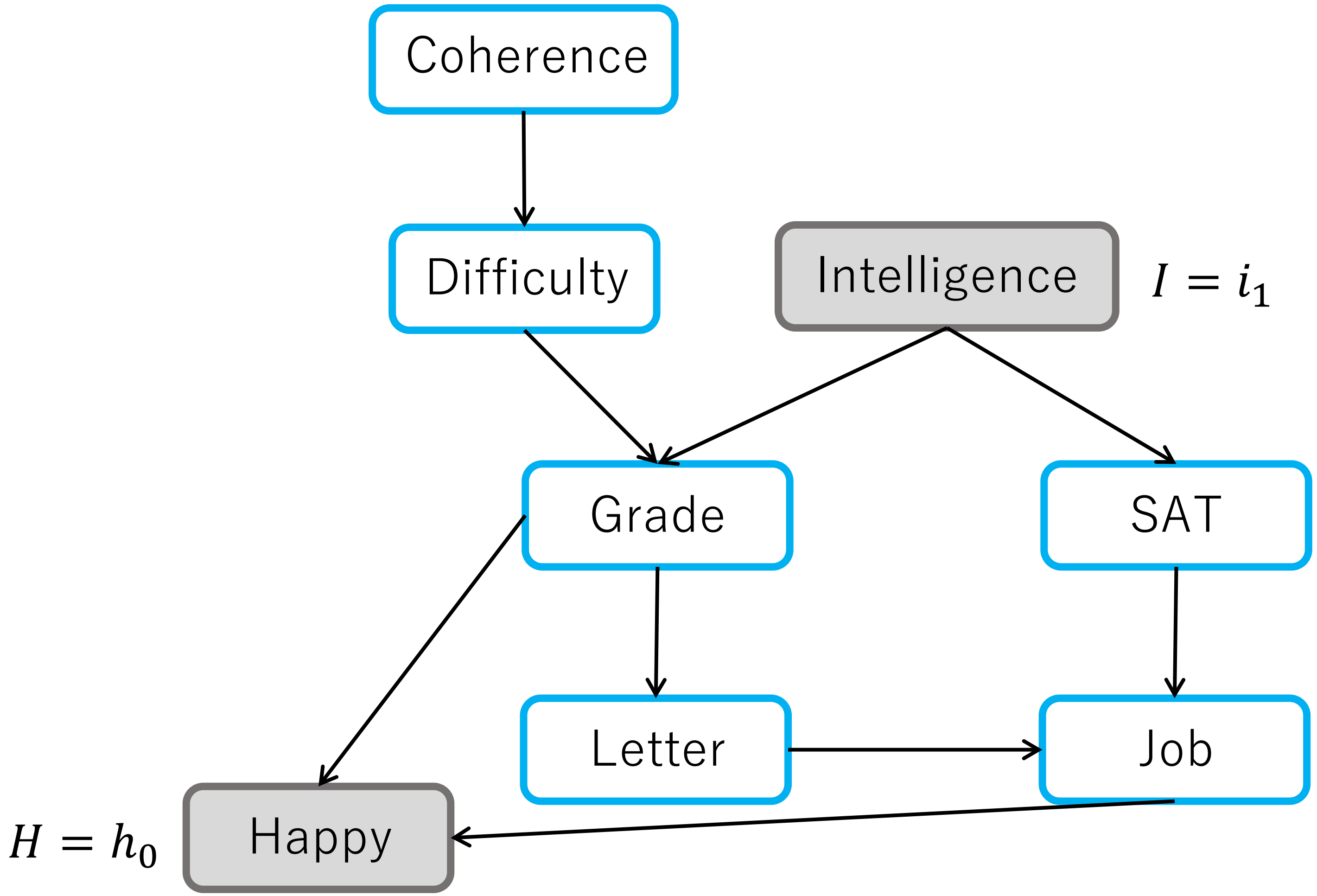
$\Phi \leftarrow \Phi_{\mathrm{re}} \cup \tau$ ◁ 新しいファクターのセット

end For

$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$ ◁ 変数消去されたファクターを掛け合わせる

$Z \leftarrow \sum_{\boldsymbol{X}_{V \setminus (\varepsilon \cup v)}} \phi^*\left(\boldsymbol{X}_{V \setminus (\varepsilon \cup v)}\right)$ ◁ 分配関数の評価

条件付き分布に対するSum-Product VEの例

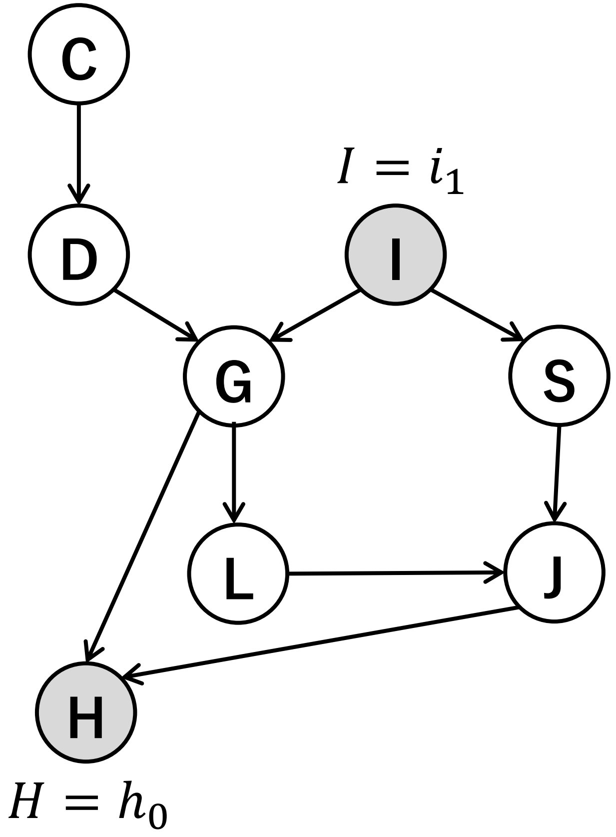


- グラフ上で定義される確率分布

$$P(C,D,I=i_0,G,S,L,J,H=h_0)$$
$$=P(C)P(D|C)P(I=i_0)P(G|I=i_0,D)P(S|I=i_0)P(L|G)P(J|S,L)P(H=h_0|G,J)$$
$$=\phi_C(C)\phi_D(D,C)\phi_I(I=i_0)\phi_G(G,I=i_0,D)\phi_S(S,I=i_0)\phi_L(L,G)\phi_J(J,S,L)\phi_H(H=h_0,G,J)$$

- VEアルゴリズムにより$P(J)$を評価する
  Eliminationの順番は**C** → **D** → **G** → **S** → **L**とする

1. **C**のElimination
   - ✦ $\psi_1(C,D)=\phi_C(C)\phi_D(D,C)$
   - ✦ $\tau_1(D)=\sum_C \psi_1(C,D)$
2. **D**のElimination
   - ✦ $\psi_2(G,D)=\phi_G(G,I=i_0,D)\tau_1(D)$
   - ✦ $\tau_2(G)=\sum_D \psi_2(G,D)$
3. **G**のElimination
   - ✦ $\psi_3(G,L,J)=\phi_L(L,G)\phi_H(H=h_0,G,J)\tau_2(G)$
   - ✦ $\tau_3(L,J)=\sum_G \psi_3(G,L,J)$



- グラフ上で定義される確率分布

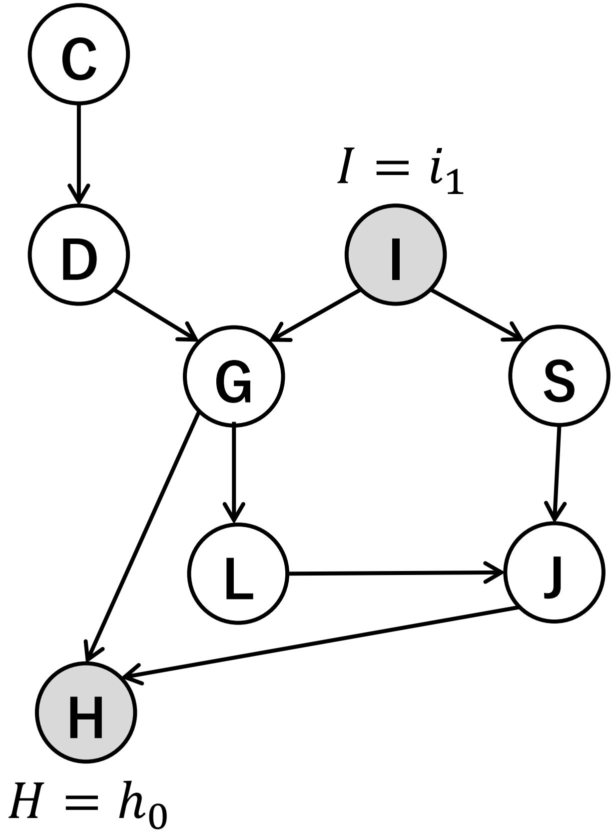
$$P(C, D, I = i_0, G, S, L, J, H = h_0)$$
$$= P(C)P(D|C)P(I = i_0)P(G|I = i_0, D)P(S|I = i_0)P(L|G)P(J|S, L)P(H = h_0|G, J)$$
$$= \phi_C(C)\phi_D(D, C)\phi_I(I = i_0)\phi_G(G, I = i_0, D)\phi_S(S, I = i_0)\phi_L(L, G)\phi_J(J, S, L)\phi_H(H = h_0, G, J)$$

- VEアルゴリズムにより$P(J)$を評価する

  Eliminationの順番は**C** → **D** → **G** → **S** → **L**とする

4. **S**のElimination
   - ✦ $\psi_4(S, L, J) = \phi_S(S, I = i_0)\phi_J(J, S, L)$
   - ✦ $\tau_4(L, J) = \sum_S \psi_4(S, L, J)$
5. **L**のElimination
   - ✦ $\psi_5(L, J) = \tau_3(L, J)\tau_4(L, J)$
   - ✦ $\tau_5(J) = \sum_L \psi_5(L, J)$

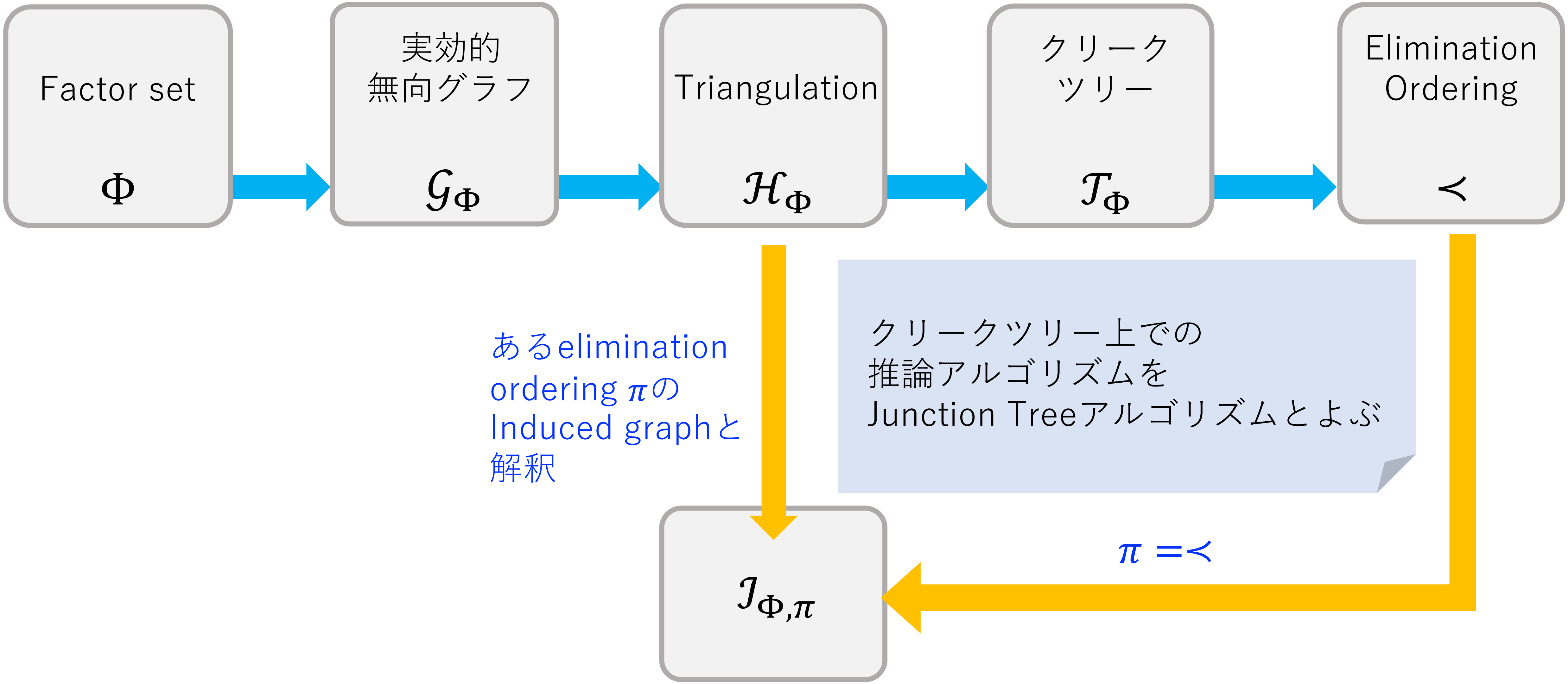
分配関数評価：$Z_J = \sum_J \tau_5(J)$ ➡ $P(J|I = i_1, H = h_0) = \frac{1}{Z_J}\tau_5(J)$



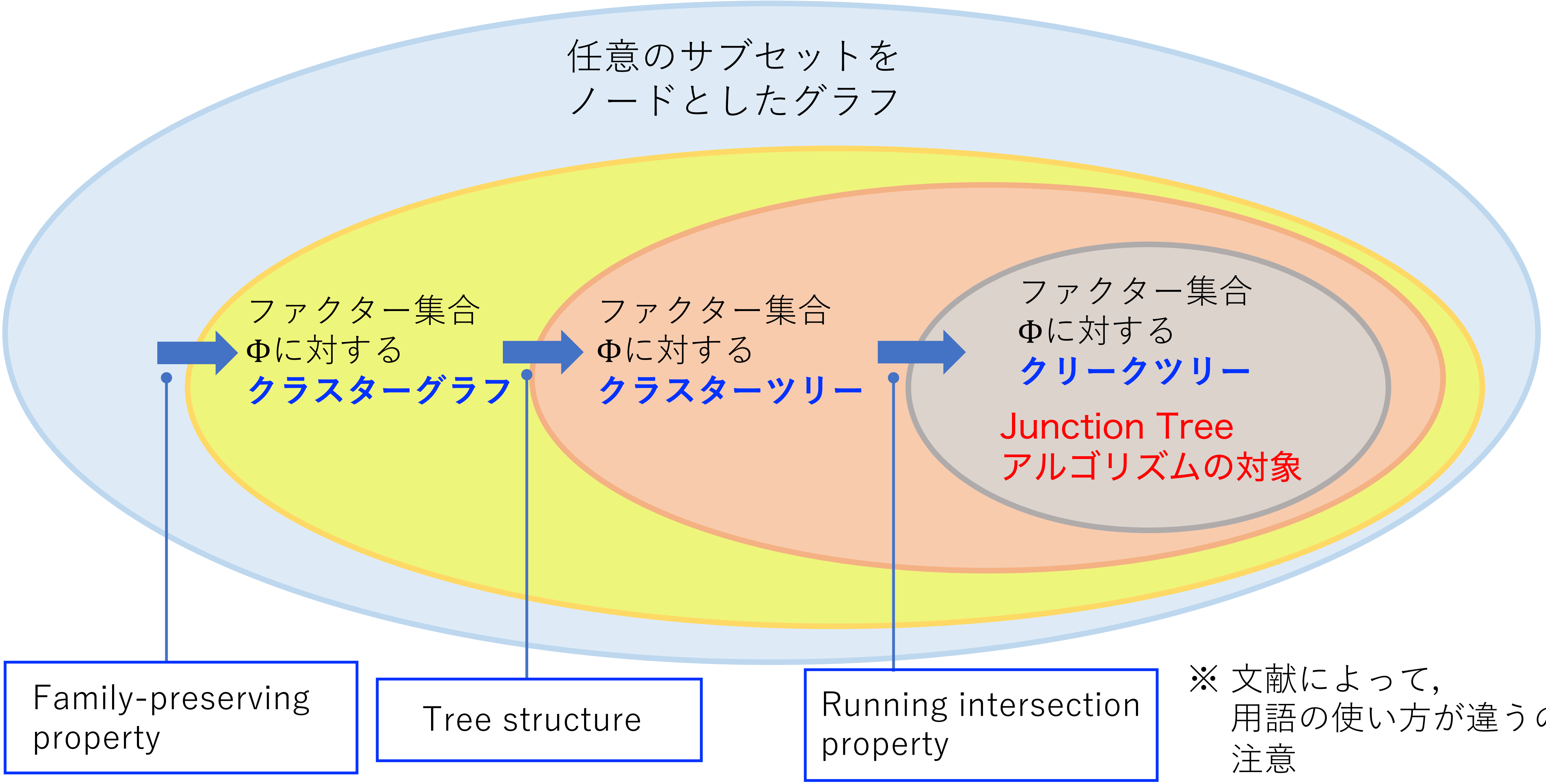
3. クリークツリーを用いた推論

Junction Tree algorithm

elimination orderingの構成方法(再掲)



準備：クリークツリー



※ 文献によって，
用語の使い方が違うので
注意

定義：Cluster graph

各ノードiが変数集合のサブセット$\boldsymbol{C}_i$に対応し，ファクター集合$\boldsymbol{\Phi}$に対して

Family-preserving※である無向グラフを**クラスターグラフ**とよぶ.

クラスターグラフにおけるノードはクラスターとよばれる.

またクラスター$\boldsymbol{C}_i$と$\boldsymbol{C}_j$をつなぐedgeはsepset $\boldsymbol{C}_i \cap \boldsymbol{C}_j$を表す.

※ ファクター$\forall\phi \in \Phi$が$\mathrm{Scope}[\phi] \subseteq \boldsymbol{C}_i\ \exists i$を満たすことを

family-preservingとよぶ.

- VEの手続きはCluster graphによって表現することができる.
  この場合，クラスターグラフはツリーであるので，**クラスターツリー**と呼ばれる.

定義：Running Intersection Property

ファクターΦのもとでのクラスターツリーを$\mathcal{T}$とする．

$X \in \boldsymbol{C}_i$，$X \in \boldsymbol{C}_j$を満たす変数Xがあるとき，クラスターツリー$\mathcal{T}$において

$\boldsymbol{C}_i$と$\boldsymbol{C}_j$間のユニークな経路上の全てのクラスターにXが含まれる．

これを**Running Intersection Property**という．

- クリーク$\boldsymbol{C}_i$から$\boldsymbol{C}_j$にメッセージが送られる際に変数$\boldsymbol{X}$が消去されたとする．

  → $\boldsymbol{C}_j$には$\boldsymbol{X}$が含まれない

- Running intersection propertyより，$\boldsymbol{C}_j$より上流で$\boldsymbol{X}$は現れない．

  ※ Junction tree propertyということもある

定義：Clique Tree

Running Intersection Propertyを満たすクラスターツリーは
クリークツリーとよばれる

- Junction Tree, Join Treeともよばれる.
- 前回の講義で導入した次の定義と等価である.

ノード$\{\boldsymbol{C}_i\}$とエッジ$\{\mathrm{Sep}_{ij}\}$を持つツリー$\mathcal{T}$が次の性質を満たすとき,
無向グラフ$\mathcal{G}$に対するクリークツリーとよぶ.

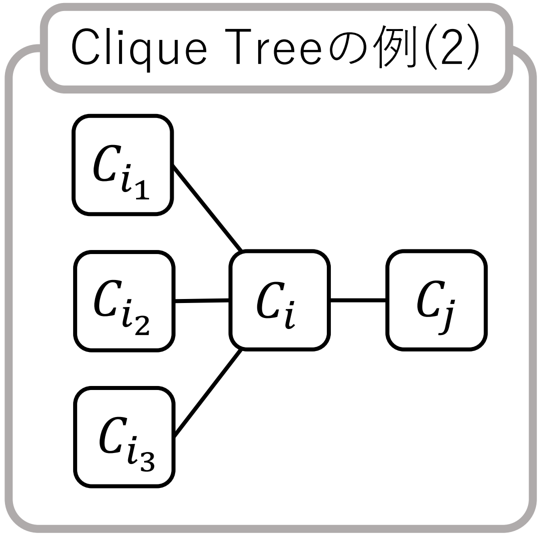
- $\mathcal{G}$の極大クリークは$\mathcal{T}$のノードである.
- $\mathcal{G}$においてSep_{ij}が$\boldsymbol{C}_i$より手前のクリークと$\boldsymbol{C}_j$より後のクリークを分ける.

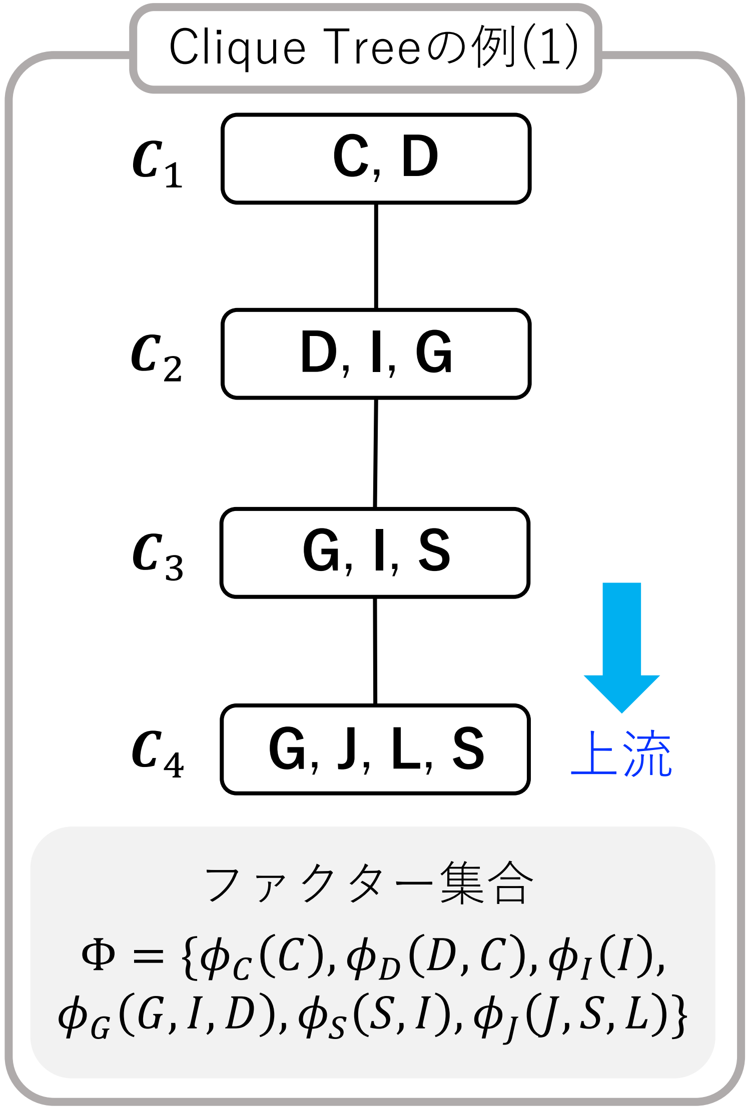
3.1 Junction Treeアルゴリズムとは

準備と基本的考え方

- $\mathcal{V}_{\prec(i\to j)}$：$\boldsymbol{C}_i$含む下流のCliqueから$\mathbf{Sep}_{ij}$を除いた変数集合
- $\mathcal{F}_{\prec(i\to j)}$：$\boldsymbol{C}_i$含む下流のCliqueに対応するファクター集合
  - ✦ 右のClique Treeの例(1)の場合,
    $\mathcal{V}_{\prec(3\to 4)} = \{C, D, I\}$, $\mathcal{F}_{\prec(3\to 4)} = \{\phi(C), \phi(C, D), \phi(D, I, G), \phi(I), \phi(S, I)\}$
- Running intersection propertyより，Clique Treeの例(2)の場合
  - $\mathcal{V}_{\prec(i_k\to i)}$ $(k = 1,2,3)$はdisjoint
  - $\mathcal{F}_{\prec(i_k\to j)}$ $(k = 1,2,3)$はdisjoint
  - $\mathcal{V}_{\prec(i\to j)} = \mathcal{V}_{\prec(i_1\to i)} \cup \cdots \cup \mathcal{V}_{\prec(i_k\to i)}$
  - $\mathcal{F}_{\prec(i\to j)} = \mathcal{F}_{\prec(i_1\to i)} \cup \cdots \cup \mathcal{F}_{\prec(i_k\to i)} \cup \{\phi \mid \alpha(\phi) = i\}$

→ $\mathcal{V}_{\prec(i_k\to i)}$ $(k = 1,2,3)$について独立に和を取ることができる

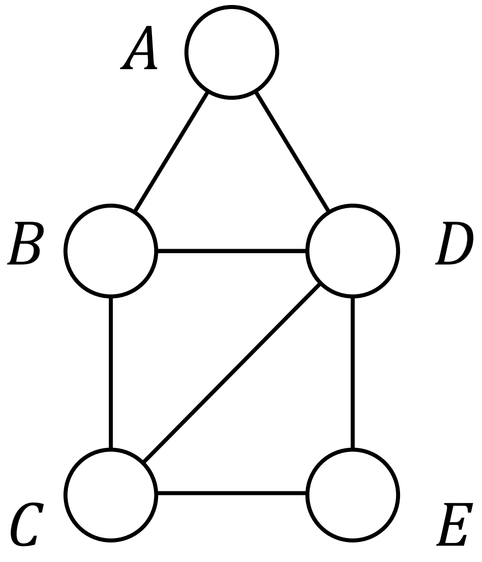




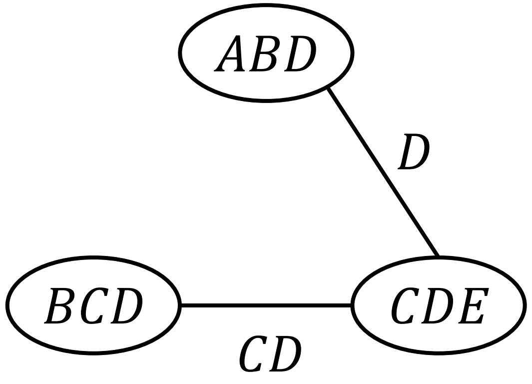
Junction Treeアルゴリズムの手続き

1. Moralization：実効グラフを作る
2. Evidenceがあれば入れる
3. Triangulationによりコーダルグラフを作る
4. Clique Treeを作る
5. Message Passing Protocolによりクリークについての周辺化分布を得る
   - ✦ 3つの代表的アルゴリズムを紹介

Clique Treeの作り方について

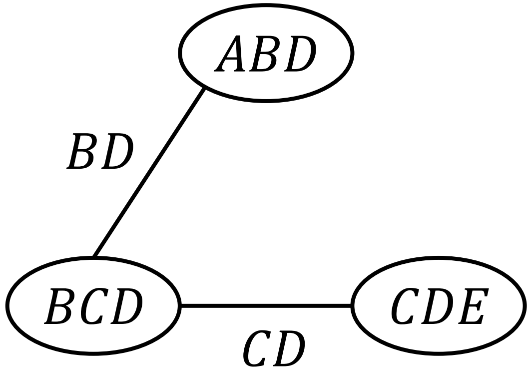


Clique Treeではない



※ BについてRunning intersection propertyが満たされていない

Clique Tree



詳しくは

- Maximum spanning tree problemの一種ととらえたClique Treeの構成方法
  - ✦ アルゴリズムイントロダクション　近代科学社⏩
  - ✦ Prim’s algorithm

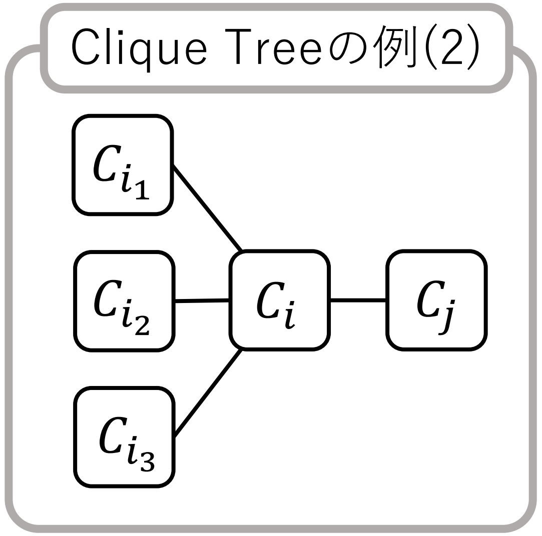
3.2 Message Passingの基本

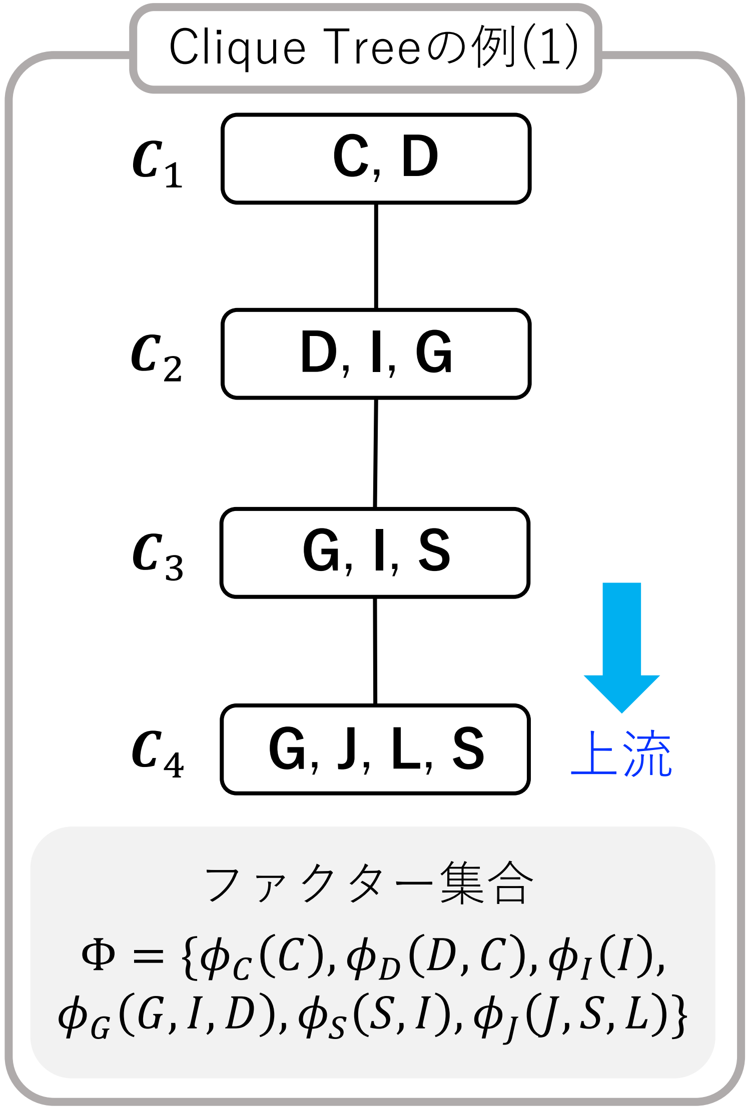
ルート以外のクラスターを周辺化する

準備と基本的考え方(再掲)

- $\mathcal{V}_{\prec(i\to j)}$：$\boldsymbol{C}_i$含む下流のCliqueから$\mathbf{Sep}_{ij}$を除いた変数集合
- $\mathcal{F}_{\prec(i\to j)}$：$\boldsymbol{C}_i$含む下流のCliqueに対応するファクター集合
  - ✦ 右のClique Treeの例(1)の場合,
    $\mathcal{V}_{\prec(3\to 4)} = \{C, D, I\}$, $\mathcal{F}_{\prec(3\to 4)} = \{\phi(C), \phi(C,D), \phi(D,I,G), \phi(I), \phi(S,I)\}$
- Running intersection propertyより，Clique Treeの例(2)の場合
  - $\mathcal{V}_{\prec(i_k\to i)}\ (k=1,2,3)$はdisjoint
  - $\mathcal{F}_{\prec(i_k\to j)}\ (k=1,2,3)$はdisjoint
  - $\mathcal{V}_{\prec(i\to j)} = \mathcal{V}_{\prec(i_1\to i)} \cup \cdots \cup \mathcal{V}_{\prec(i_k\to i)}$
  - $\mathcal{F}_{\prec(i\to j)} = \mathcal{F}_{\prec(i_1\to i)} \cup \cdots \cup \mathcal{F}_{\prec(i_k\to i)} \cup \{\phi \mid \alpha(\phi) = i\}$

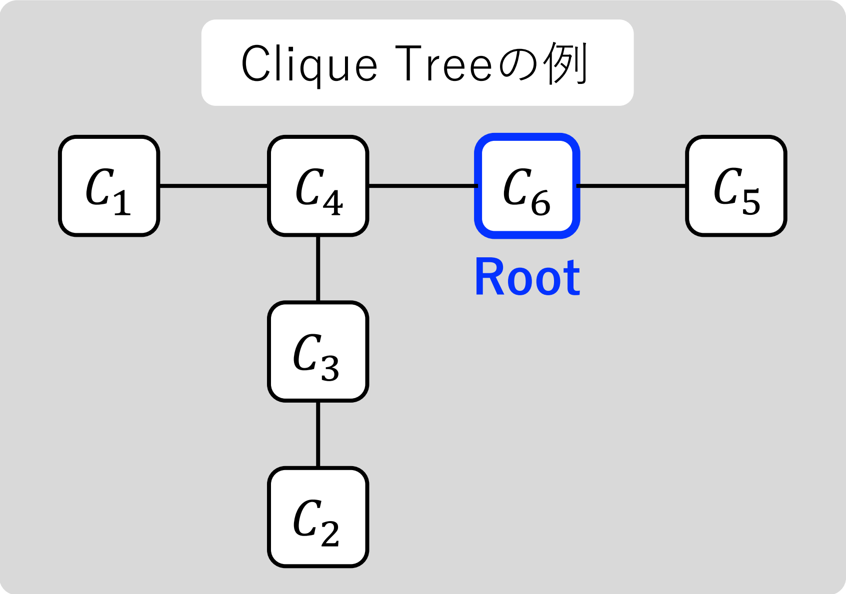
→ $\mathcal{V}_{\prec(i_k\to i)}\ (k=1,2,3)$について独立に和を取ることができる





目標

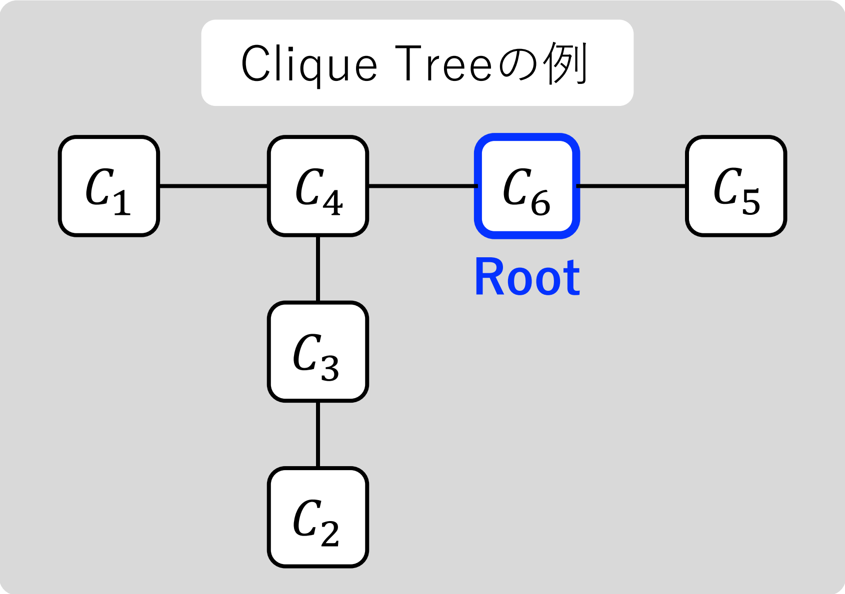
- $\boldsymbol{C}_r$をルートクリークとし，$\tilde{P}_\Phi(\boldsymbol{C}_r) = \sum_{\boldsymbol{X}\backslash\boldsymbol{C}_r} \prod_{\phi\in\Phi} \phi$を求める
  - ルート：
    メッセージが最後に到達するクリーク
  - リーフ：
    下流にクリークを持たないクリーク
    (右図の場合は$\boldsymbol{C}_1, \boldsymbol{C}_2, \boldsymbol{C}_5$)
  - 上流：ルートに向かう方向
    ここではツリーを考えるので，上流クリークは1つだけ
  - 下流：ルートから遠ざかる方向



Clique-Tree Message Passingの手続き

- ファクターϕがassignされているクリークを$\alpha(\phi)$とする.
  - ✦あるリーフクリーク$\boldsymbol{C}_i$から始め，ルートに向かって和をとっていく
- $\boldsymbol{C}_i$の近接クリークのラベル集合を∂_iとし，そのうち上流クリークを∂_i^+とする
- クリークに対してポテンシャル$\psi_j(\boldsymbol{C}_j)$を定義する.
  - ✦初期条件は$\psi_j = \varphi_j \equiv \prod_{\phi:\alpha(\phi)=j} \phi$とする.
  - ✦各ファクターは1つのクリークにassignされているので$\prod_{\phi\in\Phi} \phi = \prod_j \varphi_j(\boldsymbol{C}_j)$
- 各クリーク$\boldsymbol{C}_i$で**Message**$\delta_{i\to j}$ $(j = \partial_i^+)$を評価して上流に渡し，
  ルートでMessageをまとめて**Belief**を評価する

Clique-Tree Message Passingの例(1)



- 和を実行する順番：$\prec_C = 2, 3, 1, 4, 5, 6$とする
  - ✦ $\prec_C = 1,2,3,4,5,6$, $\prec_C = 5,1,2,3,4,6$などでもよい

    (2は3より先，3は4より先，1は4より先，
    が満たされていれば良い)
- 近接クリーク：$\boldsymbol{\partial} = \{4, 3, (2,4), (1,3,6), 6, (4,5)\}$
- 上流の近接クリーク：$\boldsymbol{\partial}^+ = \{4, 3, 4, 6, 6, \emptyset\}$

$\psi_2(\boldsymbol{C}_2) \leftarrow \varphi_2$

$\delta_{2\to3}(\mathrm{Sep}_{2,3}) \leftarrow \sum_{\boldsymbol{C}_2\setminus\mathrm{Sep}_{2,3}} \psi_2(\boldsymbol{C}_2)$

$\psi_3(\boldsymbol{C}_3) \leftarrow \varphi_3\delta_{2\to3}$

$\delta_{3\to4}(\mathrm{Sep}_{3,4}) \leftarrow \sum_{\boldsymbol{C}_3\setminus\mathrm{Sep}_{3,4}} \psi_3(\boldsymbol{C}_3)$

$\psi_1(\boldsymbol{C}_1) \leftarrow \varphi_1$

$\delta_{1\to4}(\mathrm{Sep}_{1,4}) \leftarrow \sum_{\boldsymbol{C}_1\setminus\mathrm{Sep}_{1,4}} \psi_1(\boldsymbol{C}_1)$

$\psi_4(\boldsymbol{C}_4) \leftarrow \varphi_4\delta_{1\to4}\delta_{3\to4}$

$\delta_{4\to6}(\mathrm{Sep}_{4,6}) \leftarrow \sum_{\boldsymbol{C}_4\setminus\mathrm{Sep}_{4,6}} \psi_4(\boldsymbol{C}_4)$

$\psi_5(\boldsymbol{C}_5) \leftarrow \varphi_5$

$\delta_{5\to6}(\mathrm{Sep}_{5,6}) \leftarrow \sum_{\boldsymbol{C}_5\setminus\mathrm{Sep}_{5,6}} \psi_5(\boldsymbol{C}_5)$

Belief:

$\beta_6(\boldsymbol{C}_6) \leftarrow \varphi_6\delta_{4\to6}\delta_{5\to6}$

Clique-Tree Message Passingの例(1)

$\tilde{P}_\Phi(\boldsymbol{C}_6) = \sum_{\boldsymbol{X}\backslash\boldsymbol{C}_6} \prod_{\phi\in\Phi} \phi$と$\beta_6(\boldsymbol{C}_6) = \varphi_6\delta_{4\to6}\delta_{5\to6}$が一致しているか確認

$$\beta_6(\boldsymbol{C}_6) = \varphi_6\delta_{4\to6}\delta_{5\to6}$$

$$= \varphi_6 \sum_{\boldsymbol{C}_4\backslash\mathrm{Sep}_{4,6}} \varphi_4\delta_{1\to4}\delta_{3\to4} \sum_{\boldsymbol{C}_5\backslash\mathrm{Sep}_{5,6}} \varphi_5$$

$$= \varphi_6 \sum_{\boldsymbol{C}_4\backslash\mathrm{Sep}_{4,6}} \varphi_4 \sum_{\boldsymbol{C}_1\backslash\mathrm{Sep}_{1,4}} \varphi_1 \sum_{\boldsymbol{C}_3\backslash\mathrm{Sep}_{3,4}} \varphi_3\delta_{2\to3} \sum_{\boldsymbol{C}_5\backslash\mathrm{Sep}_{5,6}} \varphi_5$$

$$= \varphi_6 \sum_{\boldsymbol{C}_4\backslash\mathrm{Sep}_{4,6}} \varphi_4 \sum_{\boldsymbol{C}_1\backslash\mathrm{Sep}_{1,4}} \varphi_1 \sum_{\boldsymbol{C}_3\backslash\mathrm{Sep}_{3,4}} \varphi_3 \sum_{\boldsymbol{C}_2\backslash\mathrm{Sep}_{2,3}} \varphi_2 \sum_{\boldsymbol{C}_5\backslash\mathrm{Sep}_{5,6}} \varphi_5$$

$$= \sum_{X\backslash\boldsymbol{C}_6} \prod_{\phi\in\Phi} \phi$$

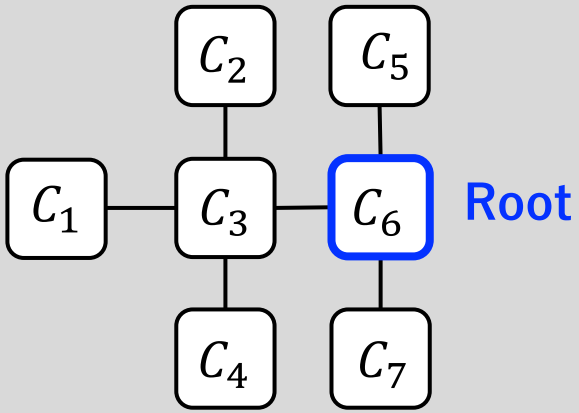
$$\psi_i = \prod_{\phi:\alpha(\phi)=i} \phi$$

$$\prod_i \psi_i = \prod_{\phi\in\Phi} \phi$$

$\mathrm{Sep}_{ij} \in \boldsymbol{C}_j$ なので

Clique-Tree Message Passingの例(2)

Clique Treeの例



- 和を実行する順番：$\prec_C = 1, 2, 4, 3, 5, 7, 6$とする
- 近接クリーク：$\boldsymbol{\partial} = \{3, 3, (1,2,4,6), 3,6, (3,5,7), 6\}$
- 上流の近接クリーク：$\boldsymbol{\partial}^+ = \{3, 3, 6, 3, 6, \emptyset, 6\}$

$\psi_1(\boldsymbol{C}_1) \leftarrow \varphi_1$

$\delta_{1\to3}(\mathrm{Sep}_{1,3}) \leftarrow \sum_{\boldsymbol{C}_1\backslash\mathrm{Sep}_{1,3}} \psi_1(\boldsymbol{C}_1)$

$\psi_2(\boldsymbol{C}_2) \leftarrow \varphi_2$

$\delta_{2\to3}(\mathrm{Sep}_{2,3}) \leftarrow \sum_{\boldsymbol{C}_2\backslash\mathrm{Sep}_{2,3}} \psi_2(\boldsymbol{C}_2)$

$\psi_4(\boldsymbol{C}_4) \leftarrow \varphi_4$

$\delta_{4\to3}(\mathrm{Sep}_{4,3}) \leftarrow \sum_{\boldsymbol{C}_4\backslash\mathrm{Sep}_{4,3}} \psi_2(\boldsymbol{C}_4)$

$\psi_3(\boldsymbol{C}_3) \leftarrow \varphi_3\delta_{4\to3}\delta_{1\to3}\delta_{2\to3}$

$\delta_{3\to6}(\mathrm{Sep}_{3,6}) \leftarrow \sum_{\boldsymbol{C}_3\backslash\mathrm{Sep}_{3,6}} \psi_3(\boldsymbol{C}_3)$

$\psi_5(\boldsymbol{C}_5) \leftarrow \varphi_5$

$\delta_{5\to6}(\mathrm{Sep}_{5,6}) \leftarrow \sum_{\boldsymbol{C}_5\backslash\mathrm{Sep}_{5,6}} \psi_5(\boldsymbol{C}_5)$

$\psi_7(\boldsymbol{C}_7) \leftarrow \varphi_7$

$\delta_{7\to6}(\mathrm{Sep}_{7,6}) \leftarrow \sum_{\boldsymbol{C}_7\backslash\mathrm{Sep}_{7,6}} \psi_7(\boldsymbol{C}_7)$

Belief:

$$\beta_6(\boldsymbol{C}_6) \leftarrow \varphi_6\delta_{3\to6}\delta_{5\to6}\delta_{7\to6}$$

Sum-Product Message Passing for Clique Trees

Input
- Φ: 全てのファクターのセット
- $\mathcal{T}$：Clique tree with k cliques
- α: ファクターのクリークへのassignment ◁ $\alpha(\phi)$: ファクターϕが対応するクリーク
- $\boldsymbol{C}_r$：クリークのルート

Output
- $\beta_r(\boldsymbol{C}_r)$：Belief ◁ ルート以外のクリークを消去したファクター

Initialize： For each clique $\boldsymbol{C}_i$, set $\varphi_i(\boldsymbol{C}_i) \leftarrow \prod_{\phi_j:\alpha(\phi_j)=i} \phi_j$

Get appropriate ordering $\prec_C$ to be $\prec_C(1)$ is a leaf and $\prec_C(k) = r$

For $j = 1, \dots, k-1$

$\quad i \leftarrow \prec_C(j)$

$\quad \psi_i(\boldsymbol{C}_i) \leftarrow \varphi_i \prod_{\gamma \in \partial i \setminus \partial i^+} \delta_{\gamma \to i}$

$\quad \delta_{i \to \partial_i^+}\left(\mathrm{Sep}_{i,\partial i^+}\right) \leftarrow \sum_{\boldsymbol{C}_i \setminus \mathrm{Sep}_{i,\partial i^+}} \psi_i(\boldsymbol{C}_i)$

end For

$\beta_r(\boldsymbol{C}_r) \leftarrow \psi_r \prod_{k \in \partial r} \delta_{k \to r}$

4. Junction Treeアルゴリズム

代表的なアルゴリズム

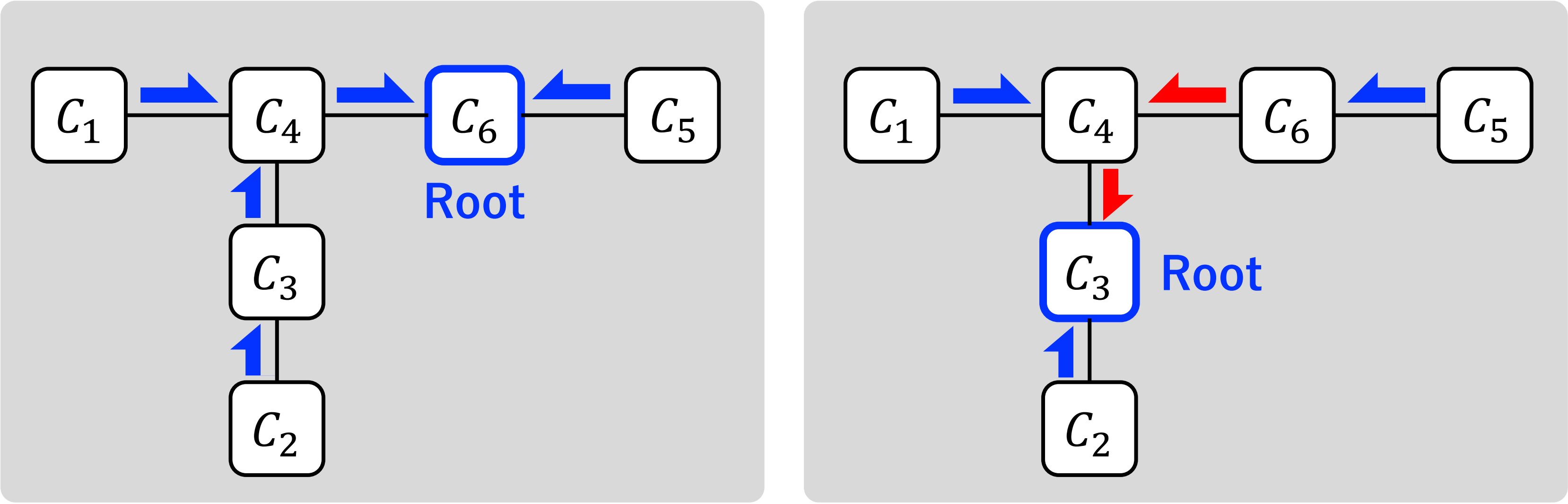
- Shafer-Shenoyアルゴリズム
  - ✦Sum-product message passing
- Lauritzen-Spiegelhalterアルゴリズム
  - ✦ Belief update型
- Huginアルゴリズム
  - ✦ Separator potentialの導入

4.1 Shafer-Shenoy algorithm

Sum-product message passingにより
各クリークの周辺化分布を得る

二方向のmessage

- クリークツリーの各edgeには，二方向のmessageが定義可能



青：上流へ向かう流れ

赤：下流へ向かう流れ

とする

(逆でも問題ない)

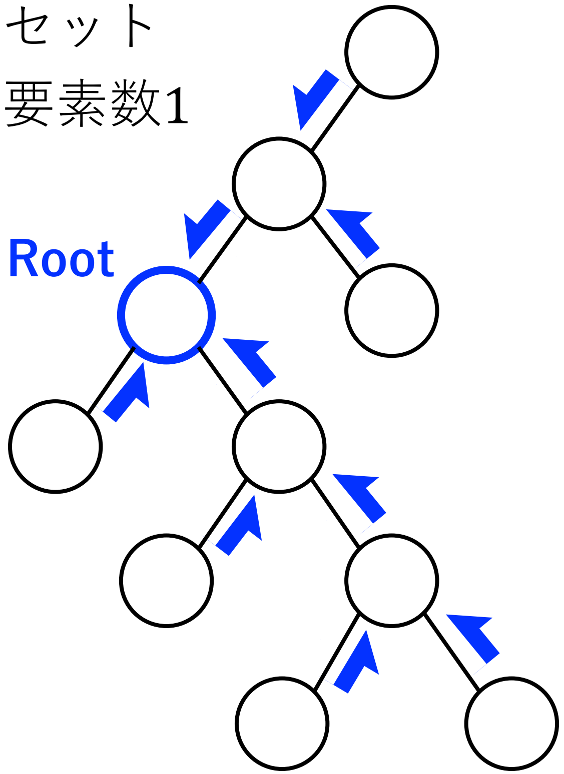
- どこがrootになっていても，messageは同じ

→ あらかじめ全てのedgeで二方向のmessageを評価しておくと便利

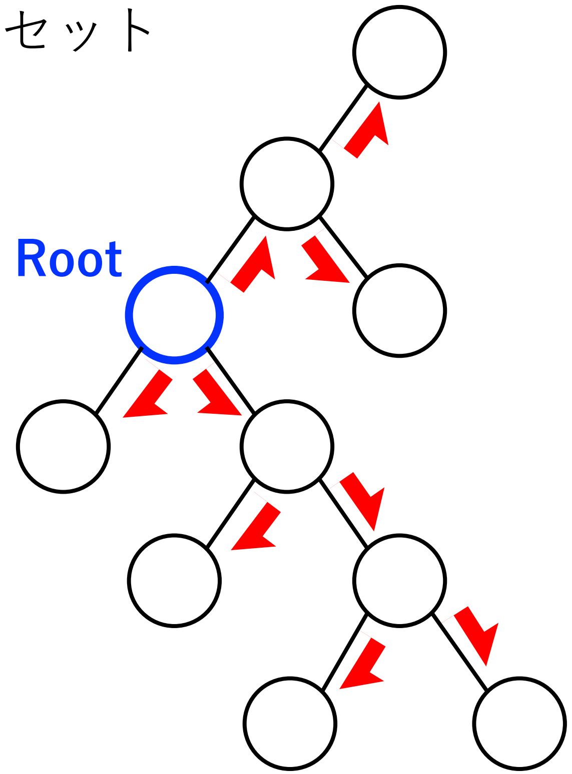
二つのプロトコルを定義

- COLLECT_MESSAGE(i)
  - 下流からノードiに入るメッセージを"集めて"送る
  - 出力はiから上流ノードへのメッセージのセット

  ※ ツリーなので要素数1



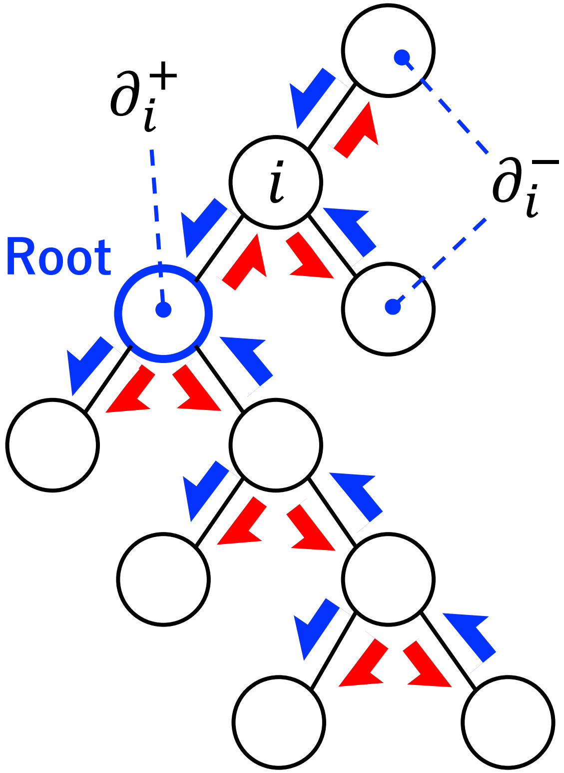
- DISTRIBUTE_MESSAGE(i)
  - 上流からノードiに入るメッセージを"集めて"送る
  - 出力はiから下流ノードへのメッセージのセット



COLLECT/DISTRIBUTE_MESSAGEの中身

※ ここではルートへ向かう向きを上流，逆の向きを下流とよぶ．

- $\boldsymbol{C}_i$の近接クリークのうち上流クリークを∂_i^+, 下流クリークを∂_i^-とする



✦ $\delta_{i\to\partial_i^+} = \mathrm{COLLECT\_MESSAGE\_SS}(i, \varphi_i, \{\delta_{\gamma\to i} | \gamma \in \partial_i^-\})$

$$\psi_{i\to\partial_i^+}(\boldsymbol{C}_i) \leftarrow \varphi_i \prod_{\gamma\in\partial i^-} \delta_{\gamma\to i}$$

$$\delta_{i\to\partial_i^+}(\mathrm{Sep}_{i,\partial i^+}) \leftarrow \sum_{\boldsymbol{C}_i \backslash \mathrm{Sep}_{i,\partial i^+}} \psi_{i\to\partial_i^+}(\boldsymbol{C}_i)$$

✦ $\{\delta_{i\to j} | j \in \partial_i^-\} = \mathrm{DISTRIBUTE\_MESSAGE\_SS}(i, \varphi_i, \delta_{\partial_i^+\to i})$

$$\psi_{i\to j}(\boldsymbol{C}_i) \leftarrow \varphi_i \prod_{\gamma\in\partial i\backslash j} \delta_{\gamma\to i}$$

$$\delta_{i\to j}(\mathrm{Sep}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i \backslash \mathrm{Sep}_{i,j}} \psi_{i\to j}(\boldsymbol{C}_i)$$

- クリーク数をkとすると，合計$2(k-1)$回呼び出す．

Calibration using sum-product message passing in a clique tree

Input
- Φ: 全てのファクターのセット
- $\mathcal{T}$ ：Clique tree with k cliques
- α: ファクターのクリークへのassignment ◁ $\alpha(\phi)$: ファクターϕが対応するクリーク

Output
- $\beta_i(\boldsymbol{C}_i), i = 1, \ldots, k$ ：Belief ◁ 各クリーク以外の全てのクリークを消去したファクター

Initialize：
- Topological sorting of cliques
- For each clique $\boldsymbol{C}_i$, set $\varphi_i(\boldsymbol{C}_i) \leftarrow \prod_{\phi_j:\alpha(\phi_j)=i} \phi_j$

For $i = 1, \ldots, k-1$ ◁ Rootに向かってメッセージを更新

  COLLECT_MESSAGE_SS(i)

end For

For $i = k, \ldots, 2$ ◁ Rootから逆方向にメッセージを更新

  DISTRIBUTE_MESSAGE_SS(i)

end For

For each clique $\boldsymbol{C}_i$, set $\beta_i(\boldsymbol{C}_i) \leftarrow \varphi_i \prod_{k\in\partial i} \delta_{k\to i}$ ◁ 各クリークについてBeliefの評価

定義：Calibration

- 二つの近接クリーク$\boldsymbol{C}_i$と$\boldsymbol{C}_j$が $\sum_{\boldsymbol{C}_i \backslash \mathrm{Sep}_{ij}} \beta_i(\boldsymbol{C}_i) = \sum_{\boldsymbol{C}_j \backslash \mathrm{Sep}_{ij}} \beta_j(\boldsymbol{C}_j)$ を満たすとき

  $\boldsymbol{C}_i$と$\boldsymbol{C}_j$ は**calibrate**されたという.
- 全ての近接クリークペアがcalibrateされたとき,

  クリークツリー$\mathcal{T}$はcalibratedであるという.

Calibrateされていることを確認

- $\beta_i(\boldsymbol{C}_i) = \varphi_i \prod_{k\in\partial i} \delta_{k\to i} = \delta_{j\to i}\varphi_i \prod_{k\in\partial i\backslash j} \delta_{k\to i}$

$$\to \sum_{\boldsymbol{C}_i\backslash \mathrm{Sep}_{ij}} \beta_i(\boldsymbol{C}_i) = \delta_{j\to i} \sum_{\boldsymbol{C}_i\backslash \mathrm{Sep}_{ij}} \varphi_i \prod_{k\in\partial i\backslash j} \delta_{k\to i} = \delta_{j\to i}\delta_{i\to j}$$

- $\beta_j(\boldsymbol{C}_j) = \varphi_j \prod_{k\in\partial j} \delta_{k\to j} = \delta_{i\to j}\varphi_j \prod_{k\in\partial j\backslash i} \delta_{k\to j}$

$$\to \sum_{\boldsymbol{C}_j\backslash \mathrm{Sep}_{ij}} \beta_j(\boldsymbol{C}_j) = \delta_{i\to j} \sum_{\boldsymbol{C}_j\backslash \mathrm{Sep}_{ij}} \varphi_j \prod_{k\in\partial j\backslash i} \delta_{k\to j} = \delta_{i\to j}\delta_{j\to i}$$

定義：sepset belief

Calibratedクリークツリーにおける$\beta_i(\boldsymbol{C}_i)$をclique beliefとよび,

また**sepset belief** $\mu_{ij}(\mathrm{Sep}_{ij})$を次のように定義する.

$$\mu_{ij}(\mathrm{Sep}_{ij}) = \sum_{\boldsymbol{C}_i \backslash \mathrm{Sep}_{ij}} \beta_i(\boldsymbol{C}_i) = \sum_{\boldsymbol{C}_j \backslash \mathrm{Sep}_{ij}} \beta_j(\boldsymbol{C}_j)$$

- 前ページより$\mu_{ij} = \delta_{i\to j}\delta_{j\to i}$

Reparametrization

Clique belief $\beta_i(\boldsymbol{C}_i)$とSepset belief $\mu_{ij}(\mathrm{Sep}_{ij})$により

$\tilde{P}_\Phi(\boldsymbol{X}) = \prod_{\phi\in\Phi}\phi$は次のように与えられる

$$\tilde{P}_\Phi(\boldsymbol{X}) = \frac{\prod_{i\in\mathcal{V}_\mathcal{T}}\beta_i(\boldsymbol{C}_i)}{\prod_{(ij)\in\mathcal{E}_\mathcal{T}}\mu_{ij}(\mathrm{Sep}_{ij})}$$

- $\mathcal{V}_\mathcal{T}$：クリークツリー$\mathcal{T}$のノード集合
- $\mathcal{E}_\mathcal{T}$：クリークツリー$\mathcal{T}$のエッジ集合

◆ 証明 ◆

- $\beta_i = \varphi_i\prod_{j\in\partial_i}\delta_{j\to i}$より，　$\prod_{i\in\mathcal{V}_\mathcal{T}}\beta_i = \left(\prod_{i\in\mathcal{V}_\mathcal{T}}\varphi_i\right)\left(\prod_{i\in\mathcal{V}_\mathcal{T}}\prod_{j\in\partial_i}\delta_{j\to i}\right)$
- $\mu_{ij} = \delta_{i\to j}\delta_{j\to i}$より，　$\prod_{(ij)\in\mathcal{E}_\mathcal{T}}\mu_{ij} = \prod_{(ij)\in\mathcal{E}_\mathcal{T}}\delta_{i\to j}\delta_{j\to i}$

※ $\varphi_i = \prod_{\phi:\alpha(\phi)=i}\phi$

打ち消す

Clique beliefとSepset beliefは，
規格化されていない確率分布$\tilde{P}_\Phi$をReparametrizeしたものといえる．

Reparametrizationの一般化

- $\mathcal{T}$をファクターΦのもとでのクリークツリーとする
- β_iをCalibrated Belief, μ_{ij}をSepset beliefとする
- $Q_{\mathcal{T}}(\boldsymbol{X}) = \frac{\prod_{i\in\mathcal{V}_{\mathcal{T}}}\beta_i(\boldsymbol{C}_i)}{\prod_{(ij)\in\mathcal{E}_{\mathcal{T}}}\mu_{ij}(\mathrm{Sep}_{ij})}$とする

このとき次の関係が成立する

$$\tilde{P}_{\Phi}(\boldsymbol{X}) \propto Q_{\mathcal{T}}(\boldsymbol{X}) \Leftrightarrow \beta_i(\boldsymbol{C}_i) \propto \tilde{P}_{\Phi}(\boldsymbol{C}_i)$$

▶ 証明の概要

- Running intersection propertyが成立するとき， $\mathbf{Sep}_{i,\partial_i^+}$所与のもとで $\boldsymbol{C}_i$と$\boldsymbol{C}_i$の非下流クリークに含まれる変数は独立であるという事実を用いる

定理：クリークツリーにおける条件付き独立性

ノード集合$\mathcal{V}$のもと，クリークツリー$\mathcal{T}$のリーフC_iを考える．

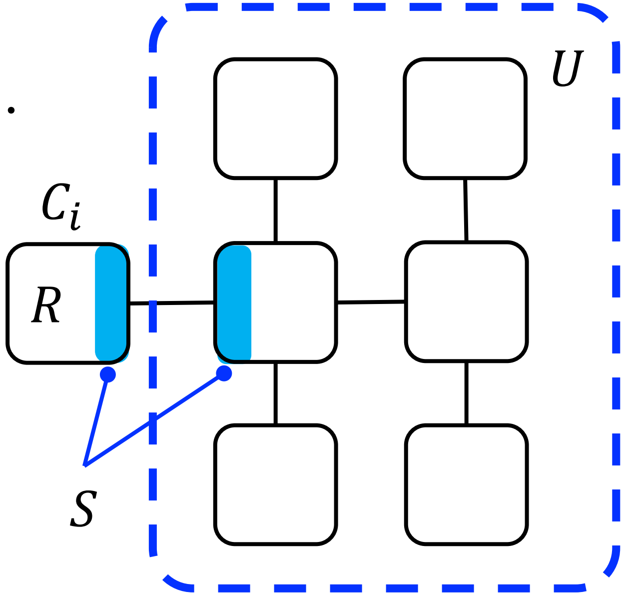
C_iと，ある隣接クリークC_j間のsepsetを$\mathbf{Sep}_{ij}$とする．

このとき，$R_i = C_i \setminus \mathbf{Sep}_{ij}$，$U_i = \mathcal{V} \setminus C_i$とすると，$R_i \perp\!\!\!\perp U_i | \mathbf{Sep}_{ij}$が成立する．

▶ 証明の概要： Running intersection propertyから，
R_iとU_iが$\mathbf{Sep}_{ij}$により分離されることを背理法で証明．

◆ 証明 ◆

- あるノード$A \in R_i$と近接ノード$n \in U_i$を含む最大complete subsetがあったとする．
    - このcomplete subset はC_iではない($n \notin C_i$なので)
- Running intersection propertyより$R_i \cap U_i = \emptyset$なので
$A_i \in R_i$はC以外のクリークに入らないため，
このようなcomplete setは存在しない．



4.2 Lauritzen-Spiegelhalterアルゴリズム

Belief updateによるMessage Passing

Factor Division

- $\boldsymbol{X}, \boldsymbol{Y}$を互いにdisjointな変数集合とする.
- 二つのファクター $\phi_1(\boldsymbol{X}, \boldsymbol{Y})$と$\phi_2(\boldsymbol{Y})$について,
  $\boldsymbol{X}, \boldsymbol{Y}$の値ごとに除算を次のように定義する

$$\psi(\boldsymbol{X}, \boldsymbol{Y}) = \frac{\phi_1(\boldsymbol{X}, \boldsymbol{Y})}{\phi_2(\boldsymbol{Y})}$$

※ $0/0$の場合は0とする.

- Factor Divisionにより，Shafer-Shenoyアルゴリズムのメッセージは次のように表される

$$\delta_{i\to j} = \frac{\sum_{\boldsymbol{C}_i \backslash \mathrm{Sep}_{ij}} \beta_i}{\delta_{j\to i}}$$

- Lauritzen-Spiegelhalterアルゴリズムでは，この関係を用いる

Calibration using belief propagation in clique tree
(Lauritzen-Spiegelhalterアルゴリズム)

`Input` • Φ: 全てのファクターのセット
• $\mathcal{T}$：Clique tree with k cliques
• α: ファクターのクリークへのassignment ◁ $\alpha(\phi)$: ファクターϕが対応するクリーク

`Output` • $\beta_i(\boldsymbol{C}_i), i = 1, \dots, k$：Belief

```
Initialize
```
• `For each clique` $\boldsymbol{C}_i$`, set` $\beta_i \leftarrow \prod_{\phi_j:\alpha(\phi_j)=i} \phi_j$
• `For each edge` $(i,j) \in \mathcal{E}_\mathcal{T}$`, set` $\mu_{ij} \leftarrow 1$

`While exists an uninformed clique in` $\mathcal{T}$ ◁ 収束していないクリークがある場合

`Select` $(i,j) \in \mathcal{E}_\mathcal{T}$

$\sigma_{i\to j} \leftarrow \sum_{\boldsymbol{C}_i \backslash \mathrm{Sep}_{ij}} \beta_i$

$\beta_j \leftarrow \beta_j \cdot \frac{\sigma_{i\to j}}{\mu_{ij}}$

$\mu_{ij} \leftarrow \sigma_{i\to j}$

`end For`

※ $(i,j) \in \mathcal{E}_\mathcal{T}$の更新の順番に影響を受けない.
- すでに正しく評価されているβ_iについて, (i, ∂_i)が選ばれてもβ_iに変化はない.

Clique Tree Invariant

- Lauritzen-Spiegelhalter algorithmで,

  t回更新したあとのBeliefを$\beta_i^{(\mathrm{LS},t)}, \mu_{ij}^{(\mathrm{LS},t)}$とする.

LSアルゴリズムでは，ステップtで次の関係が満たされるとき,

$$\tilde{P}_\Phi(\boldsymbol{X}) = \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i^{(\mathrm{LS},t)}}{\prod_{(ij) \in \mathcal{E}_\mathcal{T}} \mu_{ij}^{(\mathrm{LS},t)}}$$

$t' > t$においても同様の関係が成立する.

$$\tilde{P}_\Phi(\boldsymbol{X}) = \frac{\prod_{i \in \mathcal{V}_\mathcal{T}} \beta_i^{(\mathrm{LS},t')}}{\prod_{(ij) \in \mathcal{E}_\mathcal{T}} \mu_{ij}^{(\mathrm{LS},t')}}$$

この性質をClique Tree Invariantとよぶ.

Clique Tree Invariant

◆ 証明 ◆

- LSアルゴリズムでは，各ステップで1つのエッジが更新される
  - ここでは(k, ℓ)を更新してみる

$$\sigma_{k\to\ell}^{(t+1)} = \sum_{\boldsymbol{C}_k \backslash \mathrm{Sep}_{k\ell}} \beta_k^{(t)}$$

$$\beta_\ell^{(t+1)} \leftarrow \beta_\ell^{(t)} \cdot \frac{\sigma_{k\to\ell}^{(t+1)}}{\mu_{k\ell}^{(t)}}$$

$$\mu_{k\ell}^{(t+1)} \leftarrow \sigma_{k\to\ell}^{(t+1)}$$

初期条件は $\beta_i^{(\mathrm{LS},0)} \leftarrow \prod_{\phi_j : \alpha(\phi_j)=i} \phi_j$, $\mu_{ij}^{(\mathrm{LS},0)} \leftarrow 1$なので

$\prod_{i\in\mathcal{V}_\mathcal{T}} \beta_i^{(\mathrm{LS},0)}(\boldsymbol{C}_i) = \prod_{\phi\in\Phi} \phi$となり$\tilde{P}_\Phi(\boldsymbol{X})$と一致

- 代入すると

$$\frac{\prod_{i\in\mathcal{V}_\mathcal{T}} \beta_i^{(\mathrm{LS},t+1)}}{\prod_{(ij)\in\mathcal{E}_\mathcal{T}} \mu_{ij}^{(\mathrm{LS},t+1)}} = \frac{\beta_\ell^{(t)} \cdot \frac{\sigma_{k\to\ell}^{(t+1)}}{\mu_{k\ell}^{(t)}}}{\mu_{k\ell}^{(t+1)} \prod_{i\in\partial_\ell \backslash k} \mu_{i\ell}^{(t)}} \prod_{j\neq\ell} \frac{\beta_j^{(t)}}{\prod_{i\in\partial_j} \mu_{ij}^{(t)}} = \frac{\prod_{i\in\mathcal{V}_\mathcal{T}} \beta_i^{(\mathrm{LS},t)}}{\prod_{(ij)\in\mathcal{E}_\mathcal{T}} \mu_{ij}^{(\mathrm{LS},t)}}$$

よってステップtで$\frac{\prod_{i\in\mathcal{V}_\mathcal{T}} \beta_i^{(\mathrm{LS},t)}}{\prod_{(ij)\in\mathcal{E}_\mathcal{T}} \mu_{ij}^{(\mathrm{LS},t)}} = \tilde{P}_\Phi(\boldsymbol{X})$が成立するとき， $t' > t$でも成立する

Clique Tree Invariant

†補足†

- 初期条件を $\beta_i^{(\mathrm{LS},0)} \leftarrow \prod_{\phi_j:\alpha(\phi_j)=i} \phi_j$，また $\mu_{ij}^{(\mathrm{LS},0)} \leftarrow 1$とすると，

$$\frac{\prod_{i\in\mathcal{V}_\mathcal{T}} \beta_i^{(\mathrm{LS},0)}}{\prod_{(ij)\in\mathcal{E}_\mathcal{T}} \mu_{ij}^{(\mathrm{LS},0)}} = \prod_i \prod_{\phi:\alpha(\phi)=i} \phi = \tilde{P}_{\boldsymbol{\Phi}}(\boldsymbol{X})$$

となるので，$t > 1$で常にClique Tree Invariantである．

Shafer-ShenoyとLauritzen-Spiegelhalterの等価性

- SSでの初期条件を$\varphi_i \ (i \in \mathcal{V}_{\mathcal{T}}), \delta_{i\to j}^{(\mathrm{SS},0)}, \delta_{j\to i}^{(\mathrm{SS},0)} \ \big((i,j) \in \mathcal{E}_{\mathcal{T}}\big)$とする
- この初期条件から次のように決まるbeliefを，LSの初期条件とする

$$\beta_i^{(\mathrm{LS},0)} = \varphi_i \prod_{k\in\partial_i} \delta_{k\to i}^{(\mathrm{SS},0)}, \qquad \mu_{ij}^{(\mathrm{LS},0)} = \delta_{i\to j}^{(\mathrm{SS},0)} \delta_{j\to i}^{(\mathrm{SS},0)}$$

※ Calibrateな表式

- このとき，SS，LSについてそれぞれ１ステップ更新しても同じ式が成り立つ

$$\beta_i^{(\mathrm{LS},1)} = \varphi_i \prod_{k\in\partial_i} \delta_{k\to i}^{(\mathrm{SS},1)}, \qquad \mu_{ij}^{(\mathrm{LS},1)} = \delta_{i\to j}^{(\mathrm{SS},1)} \delta_{j\to i}^{(\mathrm{SS},1)}$$

Shafer-ShenoyとLauritzen-Spiegelhalterの等価性

◆ 証明 ◆

- 同じ(i, j)を更新する
（(i, j)以外については$t = 1$と$t = 0$での値が同じとする）

SP: $\delta_{i \to j}^{(\mathrm{SS},1)} = \sum_{\boldsymbol{c}_i \setminus \mathrm{Sep}_{i,j}} \varphi_i \prod_{\gamma \in \partial_i \setminus j} \delta_{\gamma \to i}^{(\mathrm{SS},0)} \quad \cdots (1)$

LS: $\beta_j^{(\mathrm{LS},0)} = \varphi_j \prod_{k \in \partial_j} \delta_{k \to j}^{(\mathrm{SP},0)} \quad \cdots (2)$

$\mu_{ij}^{(\mathrm{LS},0)} = \delta_{i \to j}^{(\mathrm{SP},0)} \delta_{j \to i}^{(\mathrm{SP},0)} \quad \cdots (3)$

$\sigma_{i \to j}^{(\mathrm{LS},1)} = \sum_{\boldsymbol{c}_i \setminus \mathrm{Sep}_{ij}} \beta_i^{(\mathrm{LS},0)} \quad \cdots (4)$

$\beta_j^{(\mathrm{LS},1)} = \beta_j^{(\mathrm{LS},0)} \cdot \frac{\sigma_{i \to j}^{(\mathrm{LS},1)}}{\mu_{ij}^{(\mathrm{LS},0)}} \quad \cdots (5)$

$\mu_{ij}^{(\mathrm{LS},1)} = \sigma_{i \to j}^{(\mathrm{LS},1)} \quad \cdots (6)$

- (4)に(2)を代入すると

$$\sigma_{i \to j}^{(\mathrm{LS},1)} = \sum_{\boldsymbol{c}_i \setminus \mathrm{Sep}_{ij}} \varphi_i \prod_{k \in \partial_i} \delta_{k \to i}^{(\mathrm{SS},0)}$$

$$= \delta_{j \to i}^{(\mathrm{SS},0)} \sum_{\boldsymbol{c}_i \setminus \mathrm{Sep}_{ij}} \varphi_i \prod_{k \in \partial_i \setminus j} \delta_{k \to i}^{(\mathrm{SS},0)}$$

$$= \delta_{j \to i}^{(\mathrm{SS},0)} \delta_{i \to j}^{(\mathrm{SS},1)} \quad \cdots (4')$$

- (5)に(2), (3), (4')を代入すると

$$\beta_j^{(\mathrm{LS},1)} = \varphi_j \prod_{k \in \partial_j} \delta_{k \to j}^{(\mathrm{SS},0)} \cdot \frac{\delta_{j \to i}^{(SS,0)} \delta_{i \to j}^{(SS,1)}}{\delta_{i \to j}^{(SS,0)} \delta_{j \to i}^{(SS,0)}}$$

$$= \varphi_j \delta_{i \to j}^{(\mathrm{SP},1)} \prod_{k \in \partial_j \setminus i} \delta_{k \to j}^{(\mathrm{SP},0)}$$

- $\delta_{a \to b}^{(\mathrm{SS},1)} = \delta_{a \to b}^{(\mathrm{SS},0)}\ (a \neq i, b \neq j)$とすると

$$\beta_j^{(\mathrm{LS},1)} = \varphi_j \prod_{k \in \partial_j} \delta_{k \to j}^{(\mathrm{SS},1)}$$

$$\mu_{ij}^{(\mathrm{LS},1)} = \delta_{j \to i}^{(\mathrm{SS},1)} \delta_{i \to j}^{(\mathrm{SS},1)}$$

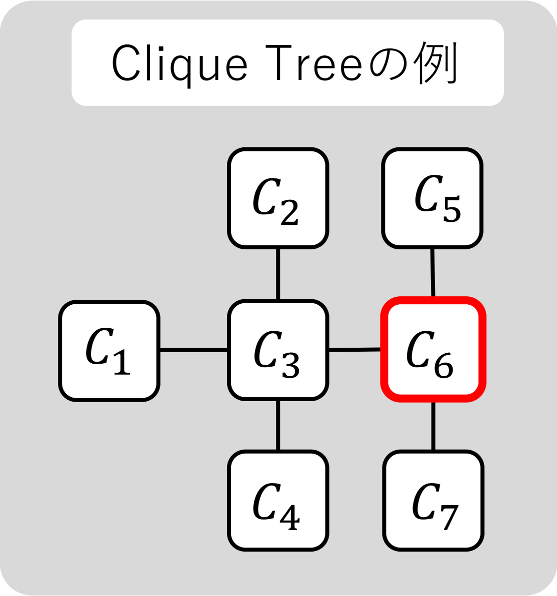
Incremental Update

- ファクター集合ΦのもとでのCalibrated clique treeに新しいファクターϕ'を加えることを考える.
  - ここではScope$[\phi'] \subseteq \boldsymbol{C}_i$とする
- 新しいファクター集合$\Phi' = \Phi \cup \phi'$のもとで$\tilde{P}_{\Phi'}(\boldsymbol{X})$を得たい.
  - ここではShafer-Shenoyアルゴリズムを使う

(例) 右図でScope$[\phi'] \subseteq \boldsymbol{C}_6$となる$\phi'$が加わったとする

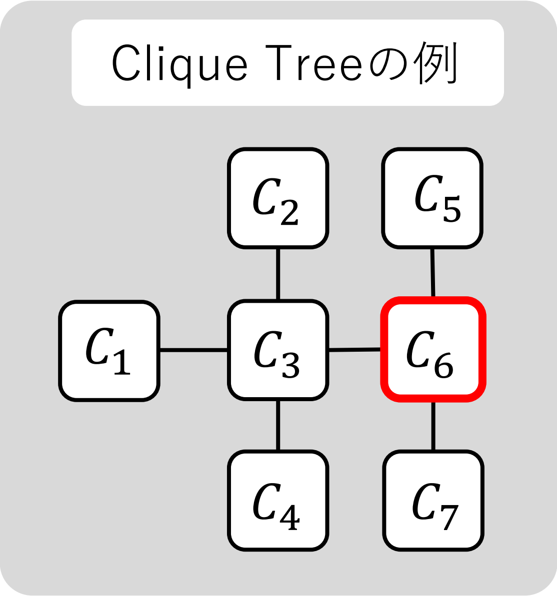
- 初期条件を$\varphi'_i \leftarrow \varphi_i \phi'$ $(i = 6)$と変える
- Upward pathについて： $\boldsymbol{C}_6$より上流が変化
- Downward pathについて：$\boldsymbol{C}_6$より下流が変化

→一般の形で更新式を書いてみる(次のページ) ：これを**Incremental Update**とよぶ



Incremental Update

- 以下では，クリークの番号が大きい方を上流とする
- ファクター集合Φにおいてすでに得られているmessageをδとする．



1. Upward pathの修正

   - iから親クリークについて

     $$\psi'_{i\to\partial_i^+}(\boldsymbol{C}_i) \leftarrow \varphi'_i \prod_{\gamma\in\partial_i\backslash\partial_i^+} \delta_{\gamma\to i}$$

     iに入ってくるmessageに変更はない．

     $$\delta'_{i\to\partial_i^+}\left(\mathrm{Sep}_{i,\partial_i^+}\right) \leftarrow \sum_{\boldsymbol{C}_i\backslash\mathrm{Sep}_{i,\partial_i^+}} \psi'_{i\to\partial_i^+}(\boldsymbol{C}_i)$$

   - $j=\partial_i^+$から上流について

     $$\psi'_{j\to\partial_j^+}(\boldsymbol{C}_i) \leftarrow \varphi_j \delta'_{i\to j} \prod_{\gamma\in\partial_j\backslash\partial_j^+,i} \delta_{\gamma\to j}$$

     i以外から入ってくるmessageに変更はない．

     $$\delta'_{j\to\partial_j^+}\left(\mathrm{Sep}_{j,\partial_j^+}\right) \leftarrow \sum_{\boldsymbol{C}_j\backslash\mathrm{Sep}_{j,\partial_j^+}} \psi'_{j\to\partial_j^+}(\boldsymbol{C}_j)$$

Incremental Update

- 以下では，クリークの番号が大きい方を上流とする
- ファクター集合Φにおいてすでに得られているmessageをδとする.

2. Downward pathの修正

  - iから近接下流クリークについて

    $$\psi'_{i\to\partial_i^-}(\boldsymbol{C}_i) \leftarrow \varphi'_i \prod_{\gamma\in\partial_i\setminus\partial_i^-} \delta_{\gamma\to i}$$

    iに入ってくるmessageに変更はない.

    $$\delta'_{i\to\partial_i^-}(\mathrm{Sep}_{i,\partial_i^-}) \leftarrow \sum_{\boldsymbol{C}_i\setminus\mathrm{Sep}_{i,\partial_i^-}} \psi'_{i\to\partial_i^-}(\boldsymbol{C}_i)$$

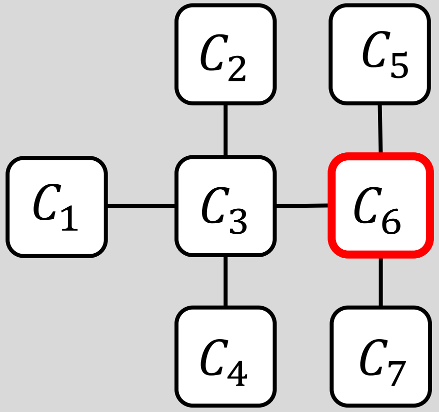
  - $j\in\partial_i^-$から下流について

    $$\psi'_{j\to\partial_j^-}(\boldsymbol{C}_i) \leftarrow \varphi_j \delta'_{i\to j} \prod_{\gamma\in\partial_j\setminus\partial_j^-,i} \delta_{\gamma\to j}$$

    i以外から入ってくるmessageに変更はない.

    $$\delta'_{j\to\partial_j^-}\left(\mathrm{Sep}_{j,\partial_j^-}\right) \leftarrow \sum_{\boldsymbol{C}_j\setminus\mathrm{Sep}_{j,\partial_j^-}} \psi'_{j\to\partial_j^-}(\boldsymbol{C}_j)$$

Clique Treeの例



Incremental Update: Evidenceの印加

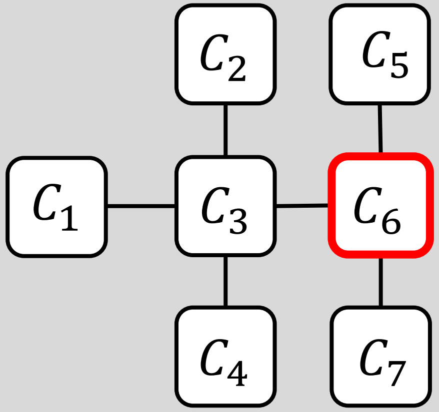
- ファクター集合Φのもとでの Calibrated clique treeに新しいファクターϕ'を加えることを考える.
  - ここでは$\mathrm{Scope}[\phi'] \subseteq \boldsymbol{C}_i$とする
- 新しいファクター集合$\Phi' = \Phi \cup \phi'$のもとで$\tilde{P}_{\Phi'}(\boldsymbol{X})$を得たい.
- Evidenceを与えることも，新しいファクターの追加と見なすことができる

$$\tilde{P}_{\Phi}(\boldsymbol{X}_{\backslash A}, \boldsymbol{X}_A = \boldsymbol{e}) = \mathbb{I}(\boldsymbol{X}_A = \boldsymbol{e}) \prod_{\phi \in \Phi} \phi = \prod_{\phi \in \Phi'} \phi$$

ϕ'とみなす

- Shafer-Shenoyアルゴリズムを使ったメッセージの修正は一般のファクターの場合と同じ

Clique Treeの例



4.3 Huginアルゴリズム

Separator potentialの利用

基本：クリークツリーにおける確率分布の表現

- Running intersection propertyのもとでの条件付き独立性より

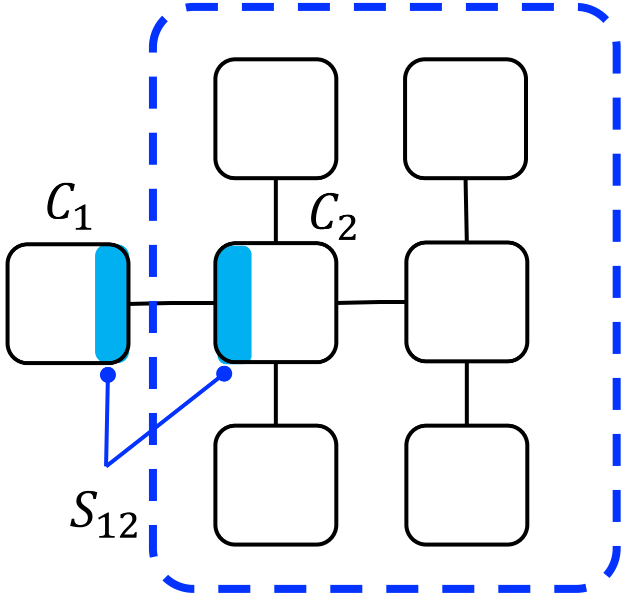
$$p(C_1 \setminus \mathrm{Sep}_{12}, \mathrm{Sep}_{12}, \mathcal{V} \setminus C_1) = p(C_1 \setminus \mathrm{Sep}_{12} | \mathrm{Sep}_{12}, \mathcal{V} \setminus C_1) p(\mathrm{Sep}_{12} | \mathcal{V} \setminus C_1) p(\mathcal{V} \setminus C_1)$$

$$\to p(C_1 \setminus \mathrm{Sep}_{12} | \mathrm{Sep}_{12}) p(\mathrm{Sep}_{12} | \mathcal{V} \setminus C_1) p(\mathcal{V} \setminus C_1)$$

$$= \frac{p(C_1 \setminus \mathrm{Sep}_{12}, \mathrm{Sep}_{12}) p(\mathrm{Sep}_{12}, \mathcal{V} \setminus C_1)}{p(\mathrm{Sep}_{12})} = \frac{p(C_1) p\big((\mathcal{V} \setminus C_1) \cup \mathrm{Sep}_{12}\big)}{p(\mathrm{Sep}_{12})}$$

$$p\big((\mathcal{V} \setminus C_1) \cup \mathrm{Sep}_{12}\big) = p\big(C_2 \setminus \mathrm{Sep}_{23}, \mathrm{Sep}_{23}, \mathcal{V} \setminus (C_1 \cup C_2)\big)$$

$$\to \frac{p(C_2) p\big((\mathcal{V} \setminus (C_1 \cup C_2)) \cup \mathrm{Sep}_{23}\big)}{p(\mathrm{Sep}_{23})}$$



- これを続けると，次の表現を得る．

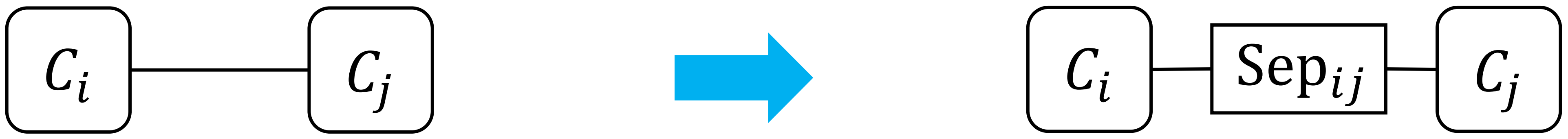
$$p(\boldsymbol{X}) = \frac{\prod_i p(\boldsymbol{C}_i)}{\prod_{(i,j) \in \mathcal{E}_{\mathcal{T}}} p\big(\mathrm{Sep}_{ij}\big)}$$

Hugin architecture

- Separator nodeとSeparator potential ϕを導入

Clique tree

Hugin architecture



- $\mathrm{Sep}_{ij} = \boldsymbol{C}_i \cap \boldsymbol{C}_j$

- $p(\boldsymbol{X}) = \dfrac{\prod_{i \in \mathcal{V}_{\mathcal{T}}} \psi_i\left(\boldsymbol{X}_{C_i}\right)}{\prod_{(i,j) \in \mathcal{E}_{\mathcal{T}}} \phi_{ij}\left(\boldsymbol{X}_{\mathrm{Sep}_{ij}}\right)}$という形を満たす$\psi_i\left(\boldsymbol{X}_{C_i}\right)$と$\phi_{ij}\left(\boldsymbol{X}_{\mathrm{Sep}_{ij}}\right)$を構成する
  - 分配関数は$\phi_{\emptyset}(\emptyset)$として表現されているとする

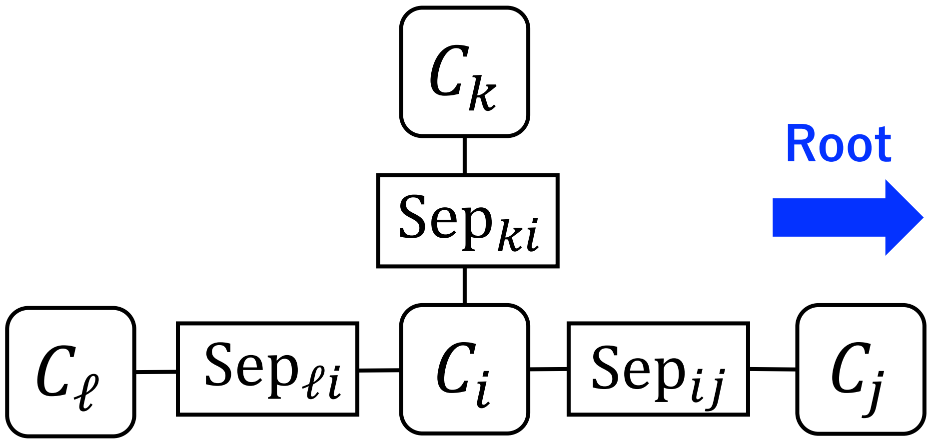
Huginアルゴリズムにおけるメッセージの更新

- $\{\phi^*_{ij}, \psi^*_j\} \leftarrow$COLLECT_MESSAGE_HUGIN$(i, \psi_i, \phi_{ij}, \psi_j)$

$$\phi^*_{ij}(\mathrm{Sep}_{ij}) \leftarrow \sum_{\boldsymbol{C}_i \setminus \mathrm{Sep}_{ij}} \psi_i(\boldsymbol{C}_i)$$

$$\psi^*_j(\boldsymbol{C}_j) \leftarrow \frac{\phi^*_{ij}(\mathrm{Sep}_{ij})}{\phi_{ij}(\mathrm{Sep}_{ij})} \psi_j(\boldsymbol{C}_j)$$

$$\psi_j(\boldsymbol{C}_j) \leftarrow \psi^*_j(\boldsymbol{C}_j)$$



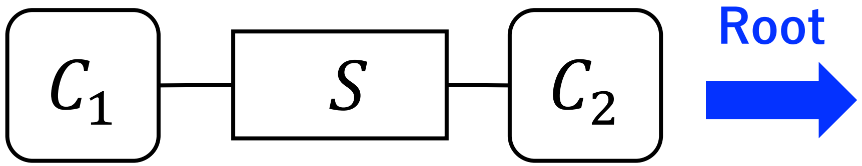
- $\{\phi^{**}_{ik}, \psi^{**}_k | k \in \partial^-_i\} \leftarrow$DISTRIBUTE_MESSAGE_HUGIN$(i, \psi^*_i, \{\phi^*_{ki}, \psi^*_k | k \in \partial^-_i\})$

$$\phi^{**}_{ki}(\mathrm{Sep}_{ki}) \leftarrow \sum_{\boldsymbol{C}_k \setminus \mathrm{Sep}_{ki}} \psi^*_k(\boldsymbol{C}_k)$$

$$\psi^{**}_k(\boldsymbol{C}_k) \leftarrow \frac{\phi^{**}_{ki}(\mathrm{Sep}_{ki})}{\phi^*_{ki}(\mathrm{Sep}_{ki})} \psi^*_k(\boldsymbol{C}_k)$$

$$\psi^*_k(\boldsymbol{C}_k) \leftarrow \psi^{**}_k(\boldsymbol{C}_k)$$

HuginアルゴリズムにおけるClique Tree Invariant



- あるステップ$t+1$でのCOLLECT_MESSAGE_HUGIN

$$\phi_S^{(t+1)}(S) \leftarrow \sum_{\boldsymbol{C}_i \setminus S} \psi_1^{(t)}(\boldsymbol{C}_1)$$

$$\psi_2^{(t+1)}(\boldsymbol{C}_2) \leftarrow \frac{\phi_S^{(t+1)}(S)}{\phi_S^{(t)}(S)} \psi_2^{(t)}(\boldsymbol{C}_2)$$

$$\frac{\psi_1^{(t+1)}\psi_2^{(t+1)}}{\phi_S^{(t+1)}} = \frac{\psi_1^{(t)}\psi_2^{(t)}\phi_S^{(t+1)}}{\phi_S^{(t)}\phi_S^{(t+1)}} = \frac{\psi_1^{(t)}\psi_2^{(t)}}{\phi_S^{(t)}}$$

※ $\psi_1^{(t+1)} = \psi_1^{(t)}$とした.

- あるステップ$t+2$でのDISTRIBUTE_MESSAGE_HUGIN

$$\phi_S^{(t+2)}(S) \leftarrow \sum_{\boldsymbol{C}_2 \setminus S} \psi_2^{(t+1)}(\boldsymbol{C}_2)$$

$$\psi_1^{(t+2)}(\boldsymbol{C}_1) \leftarrow \frac{\phi_S^{(t+2)}(S)}{\phi_S^{(t+1)}(S)} \psi_1^{(t+1)}(\boldsymbol{C}_1)$$

$$\frac{\psi_1^{(t+2)}\psi_2^{(t+2)}}{\phi_S^{(t+2)}} = \frac{\psi_1^{(t+1)}\psi_2^{(t+1)}\phi_S^{(t+2)}}{\phi_S^{(t+1)}\phi_S^{(t+2)}} = \frac{\psi_1^{(t+1)}\psi_2^{(t+1)}}{\phi_S^{(t+1)}}$$

※ $\psi_2^{(t+2)} = \psi_2^{(t+1)}$とした.

Huginアルゴリズムのメッセージと周辺分布

Huginアルゴリズムにより更新したclique potentialとseparator potentialはlocal marginal probabilityに比例する.

- 証明の概要 (詳細は次ページ)
  - 帰納法により$\psi_i^{**}(\boldsymbol{C}_i) = \sum_{\boldsymbol{X} \backslash \boldsymbol{C}_i} p(\boldsymbol{X})$, $\phi_{ij}^{**}(\mathrm{Sep}_{ij}) = \sum_{\boldsymbol{X} \backslash \mathrm{Sep}_{ij}} p(\boldsymbol{X})$を証明する
    - $N = 1$だと自明に成立
    - $N = k$で成立しているとき$N = k + 1$でも成立することをみる

Huginアルゴリズムのメッセージと周辺分布

Huginアルゴリズムにより更新したclique potentialとseparator potentialは

local marginal probabilityに比例する．

◆ 証明 ◆

- $\boldsymbol{C}_i$をleaf cliqueとし，近接クリークを$\boldsymbol{C}_j$とする．
- クリークツリーにおける条件付き独立性より$p(\boldsymbol{X}) = p\left(C_i, \mathrm{Sep}_{ij}, \boldsymbol{X}_{\setminus C_i}\right) = p\left(C_i \middle| \mathrm{Sep}_{ij}\right) p\left(\mathrm{Sep}_{ij}, \boldsymbol{X}_{\setminus C_i}\right)$
- $\boldsymbol{C}_i$のないクリークツリーで$p\left(\mathrm{Sep}_{ij}, \boldsymbol{X}_{\setminus C_i}\right) = \frac{\prod_{k \neq i} \psi_k^{**}(C_k)}{\prod_{(k,\ell) \in \mathcal{E}_{\mathcal{T}} \setminus C_i} \phi_{S_{k\ell}}^{**}(\mathrm{Sep}_{k\ell})}$, また

  $\psi_k^{**}(C_k) = p(C_k)$, $\phi_{k\ell}^{**}(\mathrm{Sep}_{k\ell}) = p(\mathrm{Sep}_{k\ell})$が成立しているとする．

- $\boldsymbol{C}_i$を含むクリークツリーでのHuginアルゴリズムにより$p\left(\boldsymbol{C}_i \middle| \mathrm{Sep}_{ij}\right) = \frac{\psi_i^{**}(C_i)}{\phi_{ij}^{**}(\mathrm{Sep}_{ij})}$を得る．

  ここで$\sum_{C_j \setminus \mathrm{Sep}_{ij}} \psi_j^{**}\left(\boldsymbol{C}_j\right) = \phi_{ij}^{**}\left(\mathrm{Sep}_{ij}\right) = p\left(\mathrm{Sep}_{ij}\right)$なので$\psi_i^{**}(C_i) = p(C_i)$

HuginとShafer-Shenoyの対応関係

HuginアルゴリズムとShafer-Shenoyアルゴリズムにおけるメッセージは次のように対応する

$$\delta_{j\to i} \leftrightarrow \frac{\phi_{ij}^{**}(\mathrm{Sep}_{ij})}{\phi_{ij}^{*}(\mathrm{Sep}_{ij})}, \qquad \forall i \in \partial_j^-$$

$$\delta_{i\to j} \leftrightarrow \frac{\phi_{ij}^{*}(\mathrm{Sep}_{ij})}{\phi_{ij}(\mathrm{Sep}_{ij})}, \qquad \forall j \in \partial_i^+$$

▶ 詳細は⏩

● Shafer-Shenoyアルゴリズム

✦ COLLECT_MESSAGE_SS

$\psi_{i\to\partial_i^+}(\boldsymbol{C}_i) \leftarrow \varphi_i \prod_{\gamma\in\partial i^-} \delta_{\gamma\to i}$ ◁ Computational cost ⇧

$\delta_{i\to\partial_i^+}(\mathrm{Sep}_{i,\partial i^+}) \leftarrow \sum_{\boldsymbol{C}_i\backslash\mathrm{Sep}_{i,\partial i^+}} \psi_{i\to\partial_i^+}(\boldsymbol{C}_i)$

✦ DISTRIBUTE_MESSAGE_SS

$\psi_{i\to j}(\boldsymbol{C}_i) \leftarrow \varphi_i \prod_{\gamma\in\partial i\backslash j} \delta_{\gamma\to i}$ ◁ Computational cost ⇧

$\delta_{i\to j}(\mathrm{Sep}_{i,j}) \leftarrow \sum_{\boldsymbol{C}_i\backslash\mathrm{Sep}_{i,j}} \psi_{i\to j}(\boldsymbol{C}_i)$

✦ 周辺化分布の評価 ◁ Computational cost ⇧

$$\beta_i(\boldsymbol{C}_i) \leftarrow \varphi_i \prod_{k\in\partial i} \delta_{k\to i}$$

$$p(\boldsymbol{C}_i) \leftarrow \frac{\beta_i(\boldsymbol{C}_i)}{\sum_{C_i}\beta_i(\boldsymbol{C}_i)}$$

● Huginアルゴリズム

✦ COLLECT_MESSAGE_HUGIN

$$\phi_{ij}^*(\mathrm{Sep}_{ij}) \leftarrow \sum_{\boldsymbol{C}_i\backslash\mathrm{Sep}_{ij}} \psi_i(\boldsymbol{C}_i)$$

$$\psi_j^*(\boldsymbol{C}_j) \leftarrow \frac{\phi_{ij}^*(\mathrm{Sep}_{ij})}{\phi_{ij}(\mathrm{Sep}_{ij})}\psi_j(\boldsymbol{C}_j)$$

$$\psi_j(\boldsymbol{C}_j) \leftarrow \psi_j^*(\boldsymbol{C}_j)$$

✦ DISTRIBUTE_MESSAGE_HUGIN

$$\phi_{ki}^{**}(\mathrm{Sep}_{ki}) \leftarrow \sum_{\boldsymbol{C}_k\backslash\mathrm{Sep}_{ki}} \psi_k^*(\boldsymbol{C}_k)$$

$\psi_k^{**}(\boldsymbol{C}_k) \leftarrow \frac{\phi_{ki}^{**}(\mathrm{Sep}_{ki})}{\phi_{ki}^*(\mathrm{Sep}_{ki})}\psi_k^*(\boldsymbol{C}_k)$ ◁ Memory cost ⇧

$$\psi_k^*(\boldsymbol{C}_k) \leftarrow \psi_k^{**}(\boldsymbol{C}_k)$$

- どちらも$O(\exp(\mathrm{width}))$の計算量だが，係数はHuginの方が小さい.
- Sepset potentialを保存する分，Huginの方がメモリーを要求する

5. 厳密な推論から近似推論へ

近似推論の方が現実的

- 厳密推論の場合，クリークの大きさの指数関数回の計算が必要
  …グラフの構造によっては計算量が大きくなってしまう

- 近似推論により計算量の削減が試みられている
  … 計算しやすいグラフ構造を仮定

- “Probabilistic Graphical Models” 11章参照

付録 (1)

Reparametrizationの一般化の証明

$\tilde{P}_\Phi(\boldsymbol{X}) \propto Q_\mathcal{T}(\boldsymbol{X}) \Leftarrow \beta_i(\boldsymbol{C}_i) \propto \tilde{P}_\Phi(\boldsymbol{C}_i)$の証明

- 適当なルートクリークC_rについて考える．

✦Chain-ruleとRunning Intersection Propertyから

$$\tilde{P}_\Phi(\boldsymbol{X}) = \tilde{P}_\Phi(\boldsymbol{C}_r)\prod_{i\neq r}\tilde{P}_\Phi(\boldsymbol{C}_i|C_{i+1},\dots,C_r) \to \tilde{P}_\Phi(\boldsymbol{C}_r)\prod_{i\neq r}\tilde{P}_\Phi\left(\boldsymbol{C}_i\middle|\mathrm{Sep}_{i,\partial_i^+}\right)$$

✦$\mathcal{Q}_\mathcal{T}(\boldsymbol{X})$を変形すると

$$\mathcal{Q}_\mathcal{T}(\boldsymbol{X}) = \beta_r(C_r)\prod_{i\neq r}\frac{\beta_i(\boldsymbol{C}_i)}{\mu_{i\partial_i^+}\left(\mathrm{Sep}_{i,\partial_i^+}\right)} = \beta_r(C_r)\prod_{i\neq r}\frac{\beta_i(\boldsymbol{C}_i)}{\sum_{C_i\backslash\mathrm{Sep}_{i,\partial_i^+}}\beta_i(C_i)} = \beta_r(C_r)\prod_{i\neq r}\beta_i\left(\boldsymbol{C}_i|\mathrm{Sep}_{i,\partial_i^+}\right)$$

- よって，　$\beta_i(\boldsymbol{C}_i) \propto \tilde{P}_\Phi(\boldsymbol{C}_i)\ \forall i$であれば$\tilde{P}_\Phi(\boldsymbol{X}) \propto Q_\mathcal{T}(\boldsymbol{X})$である．

$\tilde{P}_\Phi(\boldsymbol{X}) \propto Q_\mathcal{T}(\boldsymbol{X}) \Rightarrow \beta_i(\boldsymbol{C}_i) \propto \tilde{P}_\Phi(\boldsymbol{C}_i)$の証明

- 適当なルートクリークC_rについて考える．

✦Chain-ruleとRunning Intersection Propertyから

$$P_\Phi(\boldsymbol{X}) = \frac{1}{Z}\tilde{P}_\Phi(\boldsymbol{C}_r)\prod_{i\neq r}\tilde{P}_\Phi(\boldsymbol{C}_i|C_{i+1},\dots,C_r) \to \frac{1}{Z}\tilde{P}_\Phi(\boldsymbol{C}_r)\prod_{i\neq r}\tilde{P}_\Phi\left(\boldsymbol{C}_i\middle|\mathrm{Sep}_{i,\partial_i^+}\right)$$

$$P_\Phi(\boldsymbol{C}_r) = \frac{1}{Z}\tilde{P}_\Phi(\boldsymbol{C}_r)\sum_{\boldsymbol{X}\setminus\boldsymbol{C}_r}\prod_{i\neq r}\tilde{P}_\Phi(\boldsymbol{C}_i|C_{i+1},\dots,C_r)$$

✦ $\mathcal{Q}_\mathcal{T}$ について$p_\beta\left(\boldsymbol{C}_i\setminus\mathrm{Sep}_{ij}\middle|\mathrm{Sep}_{ij}\right) = \frac{\beta_i(\boldsymbol{C}_i)}{\sum_{\boldsymbol{C}_i\setminus\mathrm{Sep}_{ij}}\beta_i(\boldsymbol{C}_i)}$と定義するとこれは確率分布である．

$$\mathcal{Q}_\mathcal{T}(\boldsymbol{C}_r) = \beta_r(\boldsymbol{C}_r)\sum_{\boldsymbol{X}\setminus\boldsymbol{C}_r}\prod_{i\neq r}p_\beta\left(\boldsymbol{C}_i\setminus\mathrm{Sep}_{ij}\middle|\mathrm{Sep}_{ij}\right) = \beta_r(\boldsymbol{C}_r)$$

- よって$\tilde{P}_\Phi(\boldsymbol{X}) \propto Q_\mathcal{T}(\boldsymbol{X})$ のとき$\beta_r(\boldsymbol{C}_r) \propto \tilde{P}_\Phi(\boldsymbol{C}_r)$.

✦ あらゆるrについて同じ議論が成立する．

付録 (2)

Hugin とShafer-Shenoyの対応

- HUGINでの上流へのメッセージの伝搬

$$\phi_{ki}^{*}(\mathrm{Sep}_{ki}) = \sum_{\boldsymbol{C}_k \backslash \boldsymbol{S}_{ki}} \psi_k(\boldsymbol{C}_k), \ \forall k \in \partial_i^-$$

$$\psi_i^{*}(\boldsymbol{C}_i) = \psi_i(\boldsymbol{C}_j) \prod_{k \in \partial_i^-} \frac{\phi_{ki}^{*}(\mathrm{Sep}_{ki})}{\phi_{ki}(\mathrm{Sep}_{ki})}$$

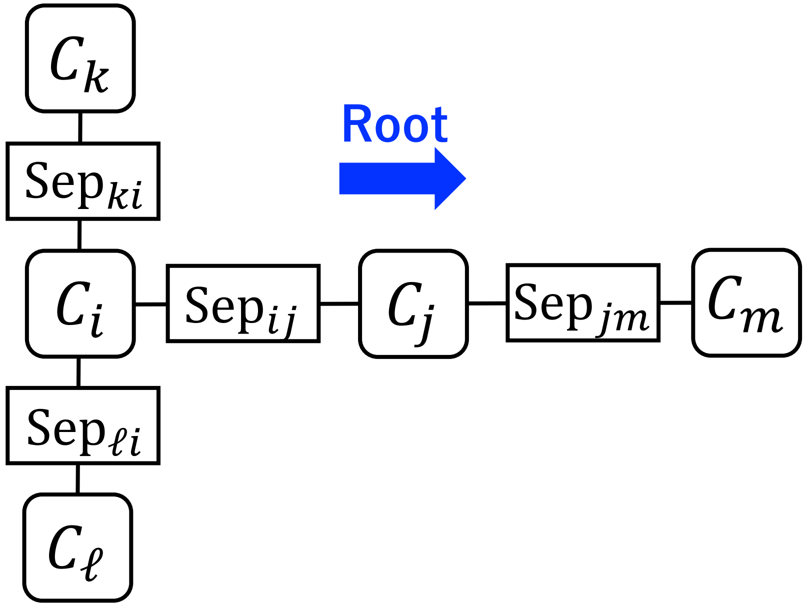
- HUGINでの下流へのメッセージの伝搬

$$\phi_{jm}^{**}(\mathrm{Sep}_{jm}) = \sum_{\boldsymbol{C}_m \backslash \boldsymbol{S}_{jm}} \psi_m^{*}(\boldsymbol{C}_m) \ \ \forall m \in \partial_j^+,$$

$$\psi_j^{**}(\boldsymbol{C}_i) = \psi_j^{*}(\boldsymbol{C}_j) \prod_{m \in \partial_j^+} \frac{\phi_{jm}^{**}(\mathrm{Sep}_{jm})}{\phi_{jm}^{*}(\mathrm{Sep}_{jm})}$$

$$= \psi_j(\boldsymbol{C}_j) \frac{\phi_{ij}^{*}(\mathrm{Sep}_{kj})}{\phi_{ij}(\mathrm{Sep}_{kj})} \prod_{k \in \partial_j^- \backslash i} \frac{\phi_{kj}^{*}(\mathrm{Sep}_{kj})}{\phi_{kj}(\mathrm{Sep}_{kj})} \prod_{m \in \partial_j^+} \frac{\phi_{jm}^{**}(\mathrm{Sep}_{jm})}{\phi_{jm}^{*}(\mathrm{Sep}_{jm})}$$

よって $$\sum_{C_j \backslash \mathrm{Sep}_{ij}} \psi_j^{**}(\boldsymbol{C}_j) = \phi_{ij}^{*}(\mathrm{Sep}_{kj}) \sum_{C_j \backslash \mathrm{Sep}_{ij}} \psi_j(\boldsymbol{C}_j) \prod_{k \in \partial_j^- \backslash i} \frac{\phi_{kj}^{*}(\mathrm{Sep}_{kj})}{\phi_{kj}(\mathrm{Sep}_{kj})} \prod_{m \in \partial_j^+} \frac{\phi_{jm}^{**}(\mathrm{Sep}_{jm})}{\phi_{jm}^{*}(\mathrm{Sep}_{jm})}$$



$\sum_{\boldsymbol{C}_j \backslash \mathrm{Sep}_{ij}} \psi_j^{**}(\boldsymbol{C}_j) = \phi_{ij}^{**}(\mathrm{Sep}_{ij})$なので

$$\phi_{ij}^{**}(\mathrm{Sep}_{ij}) = \phi_{ij}^{*}(\mathrm{Sep}_{ij}) \sum_{C_j \backslash \mathrm{Sep}_{ij}} \psi_j(\boldsymbol{C}_j) \prod_{k \in \partial_j^- \backslash i} \frac{\phi_{kj}^{*}(\mathrm{Sep}_{kj})}{\phi_{kj}(\mathrm{Sep}_{kj})} \prod_{m \in \partial_j^+} \frac{\phi_{jm}^{**}(\mathrm{Sep}_{jm})}{\phi_{jm}^{*}(\mathrm{Sep}_{jm})} \quad \cdots \square$$

- Shafer-Shenoyより

$$\mu_{j \to i}(\mathrm{Sep}_{ij}) = \sum_{\boldsymbol{C}_j \backslash \mathrm{Sep}_{ij}} \psi_j(C_j) \prod_{k \neq i} \mu_{k \to j} \qquad \cdots \blacksquare$$

□と■の比較から,

$$\mu_{j \to i} \leftrightarrow \frac{\phi_{ij}^{**}(\mathrm{Sep}_{ij})}{\phi_{ij}^{*}(\mathrm{Sep}_{ij})}, \qquad \forall i \in \partial_j^-$$

$$\mu_{i \to j} \leftrightarrow \frac{\phi_{ij}^{*}(\mathrm{Sep}_{ij})}{\phi_{ij}(\mathrm{Sep}_{ij})}, \qquad \forall j \in \partial_i^+$$

※ $\phi_{ki}(\mathrm{Sep}_{ki}) = 1 \ \forall k \in \partial_i^-$を利用した