

# 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 ▶▶

1.1 変数消去とは ▶▶

1.2 条件付き確率テーブル ▶▶

## 2. 変数消去法 ▶▶

2.1 事前確率における変数消去 ▶▶

2.2 Elimination orderingの構成 ▶▶

2.3 事後確率における変数消去 ▶▶

## 3. クリークツリーを用いた推論 ▶▶

3.1 Junction Treeアルゴリズムとは ▶▶

3.2 Message Passingの基本 ▶▶

## 4. Junction Tree アルゴリズム ▶▶

4.1 Shafer-Shenoyアルゴリズム ▶▶

4.2 Lauritzen-Spiegelhalter  
アルゴリズム ▶▶

4.3 Huginアルゴリズム ▶▶

## 5. 厳密な推論から近似推論へ ▶▶

## 付録 ▶▶

# 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  $P(\mathbf{X}_A) = \sum_{\mathbf{X}_{V \setminus A}} P(\mathbf{X}_A, \mathbf{X}_{V \setminus A})$

## (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  $\mathbf{X}_B = \mathbf{e}$  が所与での確率分布は

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

… 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 $X_1, \dots, X_N$ 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  $p(X_2) = \sum_{\mathbf{X}_{\setminus 2}} p(\mathbf{X})$  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 $\mathbf{X}_{\setminus 2}$ 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

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

Chain ruleと条件付き独立性を使って計算を効率化する



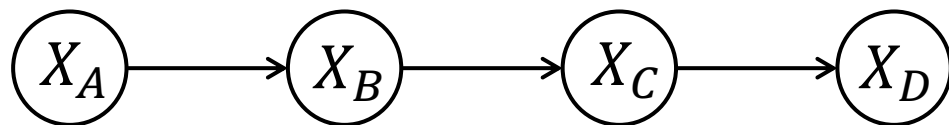
# 具体例

- 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

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



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), 次のページで見えます。

# BNによる計算量の削減

愚直に $p(d)$ を求めるには  
3×2<sup>4</sup>回の積と2×(2<sup>3</sup> - 1)回の和が必要

$$\begin{aligned} &P(a^1)P(b^1|a^1)P(c^1|b^1)P(d^1|c^1) \\ &+P(a^2)P(b^1|a^2)P(c^1|b^1)P(d^1|c^1) \\ &+P(a^1)P(b^2|a^1)P(c^1|b^2)P(d^1|c^1) \\ &+P(a^2)P(b^2|a^2)P(c^1|b^2)P(d^1|c^1) + \dots \end{aligned}$$

計算量の削減：62回→18回

$$\begin{aligned} &\{P(a^1)P(b^1|a^1) + P(a^2)P(b^1|a^2)\}P(c^1|b^1)P(d^1|c^1) \\ &+ \{P(a^1)P(b^2|a^1) + P(a^2)P(b^2|a^2)\}P(c^1|b^2)P(d^1|c^1) + \dots \end{aligned}$$

$a$ の和をとる：  
 $\tau_1(b) = \sum_a P(a)P(b|a)$   
(2回の和と4回の積)

$$\begin{aligned} &\tau_1(b^1)P(c^1|b^1)P(d^1|c^1) + \tau_1(b^2)P(c^1|b^2)P(d^1|c^1) \\ &+ \tau_1(b^1)P(c^2|b^1)P(d^1|c^2) + \tau_1(b^2)P(c^2|b^2)P(d^1|c^2) + \dots \end{aligned}$$

$$\begin{aligned} &\{\tau_1(b^1)P(c^1|b^1) + \tau_1(b^2)P(c^1|b^2)\}P(d^1|c^1) \\ &+ \{\tau_1(b^1)P(c^2|b^1) + \tau_1(b^2)P(c^2|b^2)\}P(d^1|c^2) + \dots \end{aligned}$$

$c$ の和をとり $p(d)$ を得る  
(2回の和と4回の積)

$$\begin{aligned} &\tau_2(c_1)P(d^1|c^1) + \tau_2(c_2)P(d^1|c^2) \\ &+ \tau_2(c_1)P(d^2|c^1) + \tau_2(c_2)P(d^2|c^2) \end{aligned}$$

$b$ の和をとる：  
 $\tau_2(c) = \sum_b P(b)P(c|b)$   
(2回の和と4回の積)

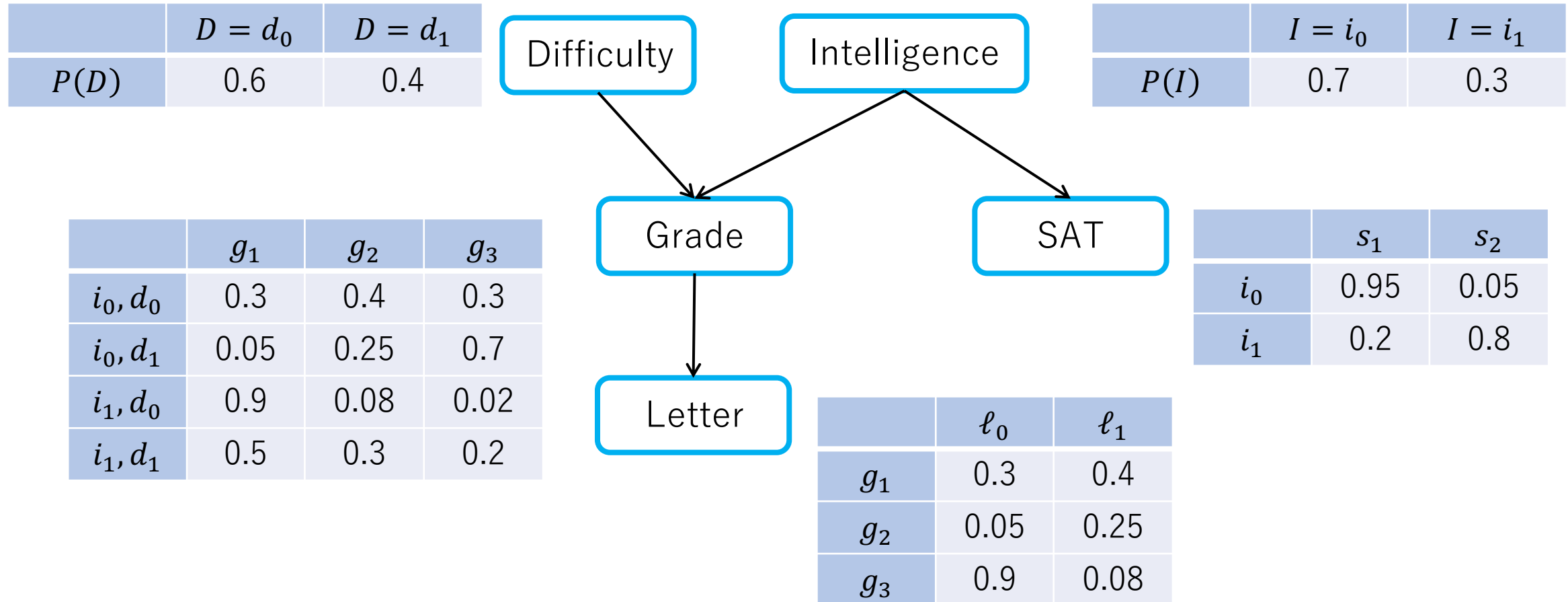
## 1.2 条件付き確率テーブル

# 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

# Conditional Probability Table (CPT)

## Student Bayesian networkの例



# 変数消去は行列演算とみなせる

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

	$g_1$	$g_2$	$g_3$	$\times$		$\ell_0$	$\ell_1$	$=$		$\ell_0$	$\ell_1$
$i_0, d_0$	0.3	0.4	0.3		$g_1$	0.3	0.4		$i_0, d_0$	0.38	0.24
$i_0, d_1$	0.05	0.25	0.7		$g_2$	0.05	0.25		$i_0, d_1$	0.66	0.14
$i_1, d_0$	0.9	0.08	0.02		$g_3$	0.9	0.08		$i_1, d_0$	0.29	0.38
$i_1, d_1$	0.5	0.3	0.2						$i_1, d_1$	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 $\mathbf{X}$ が取りうる値を $\text{val}[\mathbf{X}]$ とする.

- Factor $\phi$ は次のように定義される関数である

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

- $\phi$ に含まれる変数の集合をスコープとよび,  $\text{Scope}[\phi]$ 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  $\phi(X_i, \mathbf{X}_{\setminus i})$  を  $X_i$  について和をとることを  
周辺化 (marginalization) とよぶ。

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

- 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

$$\begin{aligned}P(X_A, X_B, X_C, X_D) &= P(X_A)P(X_B|X_A)P(X_C|X_B)P(X_D|X_C) \\ &= \phi_A(A)\phi_B(B, A)\phi_C(C, B)\phi_D(D, C)\end{aligned}$$

- ◆ 全てのFactor集合を $\Phi$ 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

- $\Phi$ : 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 ( $|v| = k$ )  $\triangleleft v$ の $i$ 番目の要素を $v_i$ とする

Output

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

For  $i = 1, \dots, k$

$\Phi_{\text{el}} \leftarrow \{\phi \in \Phi : X_{v_i} \in \text{Scope}[\phi]\}$

$\triangleleft$  消去したい変数が入っているファクター

$\Phi_{\text{re}} \leftarrow \Phi \setminus \Phi_{\text{el}}$

$\triangleleft$  消去したい変数が入っていないファクター

$\psi \leftarrow \prod_{\phi \in \Phi_{\text{el}}} \phi$

$\triangleleft$  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

$\tau \leftarrow \sum_{X_{v_i}} \psi$

$\triangleleft$  変数の和をとる

$\Phi \leftarrow \Phi_{\text{re}} \cup \tau$

$\triangleleft$  新しいファクターのセット

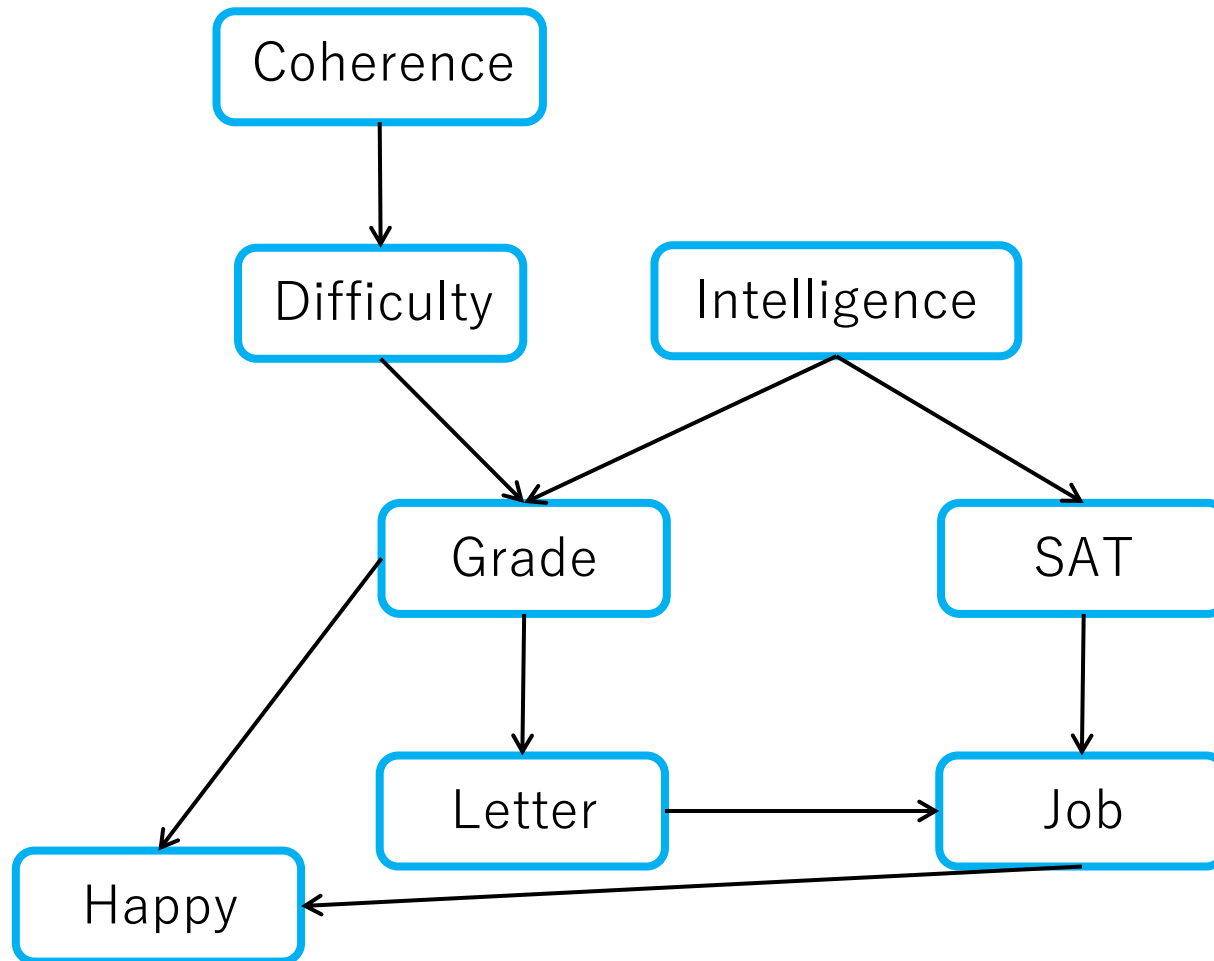
end For

$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$

$\triangleleft$  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



- $\Phi = \{p(X_i | X_{\pi(i)}) | i = 1, \dots, N\}$
- $\pi(i)$  is 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

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

- 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

$$\blacklozenge \psi_1(C, D) = \phi_C(C)\phi_D(D, C)$$

$$\blacklozenge \tau_1(D) = \sum_C \psi_1(C, D)$$

2. **D**のElimination

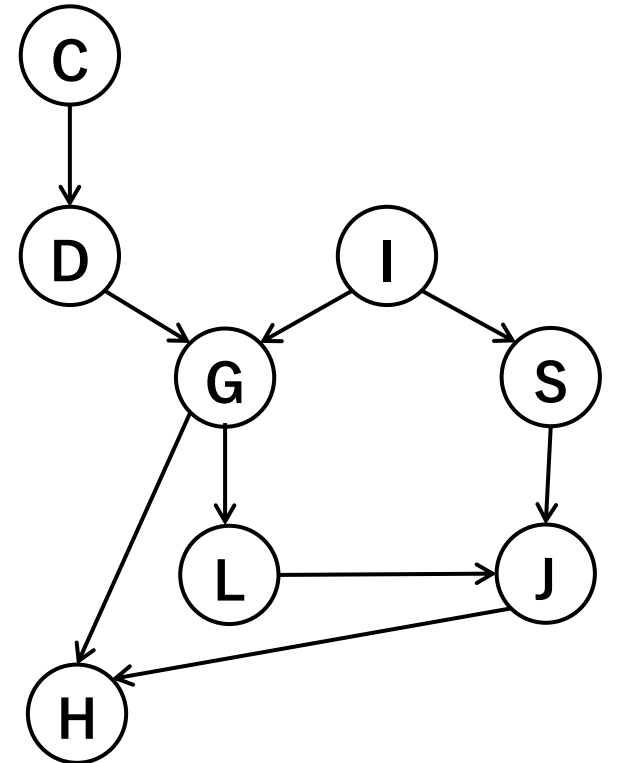
$$\blacklozenge \psi_2(G, I, D) = \phi_G(G, I, D)\tau_1(D)$$

$$\blacklozenge \tau_2(G, I) = \sum_D \psi_2(G, I, D)$$

3. **I**のElimination

$$\blacklozenge \psi_3(G, I, S) = \phi_I(I)\phi_S(S, I)\tau_2(G, I)$$

$$\blacklozenge \tau_3(G, S) = \sum_I \psi_3(G, I, S)$$



- BNの確率分布

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

- 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

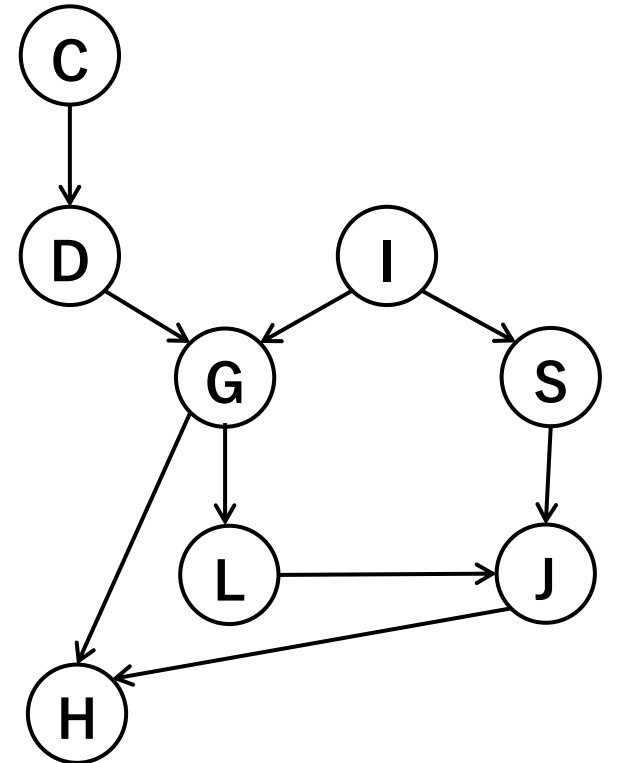
- ◆  $\psi_4(H, G, J) = \phi_H(H, G, J)$
- ◆  $\tau_4(G, J) = \sum_H \psi_4(H, G, J) = 1$

#### 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

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

- VEアルゴリズムにより $P(J)$ を評価する

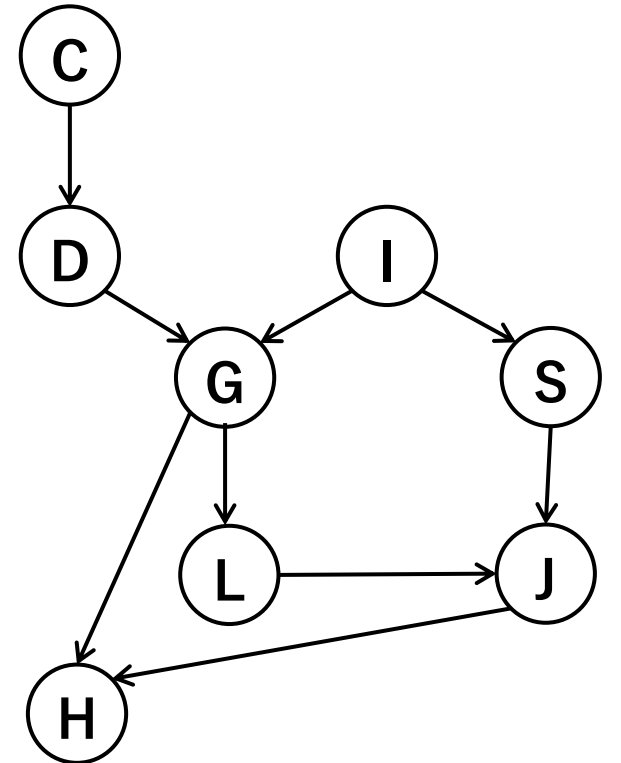
Eliminationの順番は**C** → **D** → **I** → **H** → **G** → **S** → **L**とする

7. **L**のElimination

$$\blacklozenge \psi_7(J, L) = \tau_6(J, L)$$

$$\blacklozenge \tau_7(J) = \sum_L \psi_7(H, G, J)$$

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キープの大きさ

Sum-Product variable elimination algorithm ◀ より抜粋

$$\Phi_{el} \leftarrow \{\phi \in \Phi: X_{v_i} \in \text{Scope}[\phi]\}$$

◁ 消去したい変数が入っているファクター

$$\Phi_{re} \leftarrow \Phi \setminus \Phi_{el}$$

◁ 消去したい変数が入っていないファクター

$$\psi \leftarrow \prod_{\phi \in \Phi_{el}} \phi$$

◁ 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[ $\psi$ ]の全状態に対応する $\psi$ の値を評価するためには

$O(\exp|\text{Scope}[\Phi_{el}]|)$ 回の掛け算が必要

◆ 例えば全変数が二値をとる場合,  $2^{|\text{Scope}[\Phi_{el}]|} \times |\Phi_{el}|$ 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.



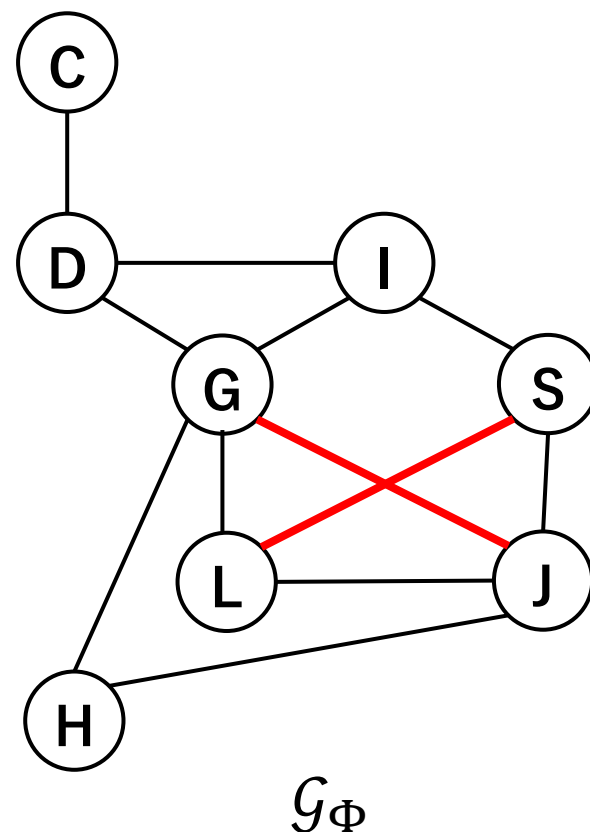
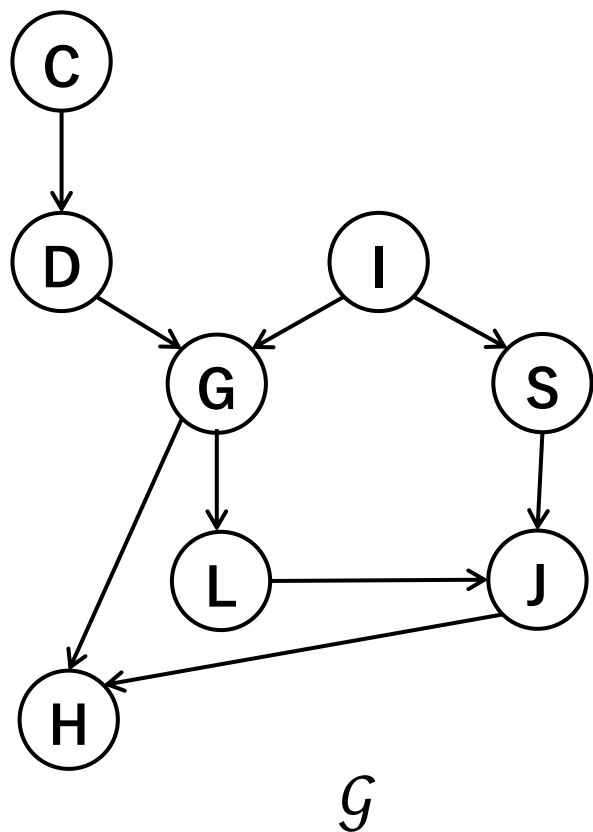
## 2.1 Elimination orderingの構成

# 準備

- ファクター集合 $\Phi$ のscopeを $\text{Scope}[\Phi] = \bigcup_{\phi \in \Phi} \text{Scope}[\phi]$ 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.
  - $\mathcal{V} = \text{Scope}[\Phi]$
  - $\{X_i, X_j\} \in \text{Scope}[\phi]$ となるような $\phi \in \Phi$ が1つでもあったら $E_{ij} \in E$

# $\mathcal{G}_\Phi$ の例

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



# 定理：実効グラフはMinimal I-map

$\mathbf{X} = \text{Scope}[\Phi]$ とする.

Factorから定義される確率分布 $P(X) = \frac{1}{Z} \prod_{\phi \in \Phi} \phi$  ( $Z = \sum_X \prod_{\phi \in \Phi} \phi$ )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 $\Phi$ 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 $\Phi$ から $X$ を消去をして得られたファクター集合を $\Phi_{\setminus X}$ とする。

また対応する実効的グラフ $\mathcal{G}_\Phi$ と $\mathcal{G}_{\Phi_{\setminus X}}$ とする。

$\mathcal{G}_{\Phi_{\setminus X}}$ が $\mathcal{G}_\Phi$ に含まれないedgeをもつとき、これをfill edgeとよぶ。

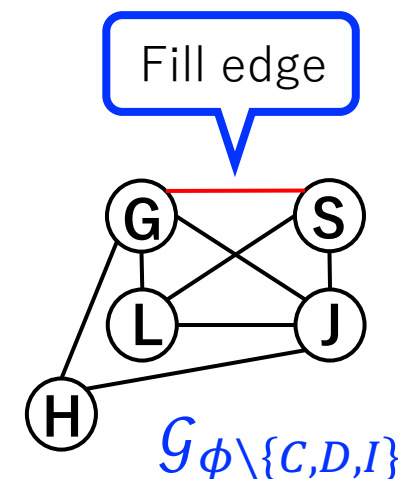
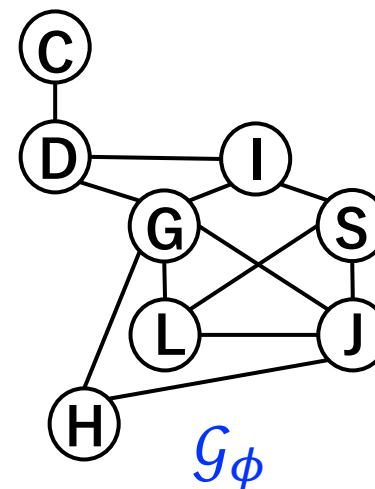
(例) extended student networkで $\mathbf{C} \rightarrow \mathbf{D} \rightarrow \mathbf{I} \rightarrow$ と消去する場合、

$\mathbf{I}$ のEliminationによりファクター  $\tau_3$ が生じる

$$\blacklozenge \psi_3(G, I, S) = \phi_I(I) \phi_S(S, I) \tau_2(G, I)$$

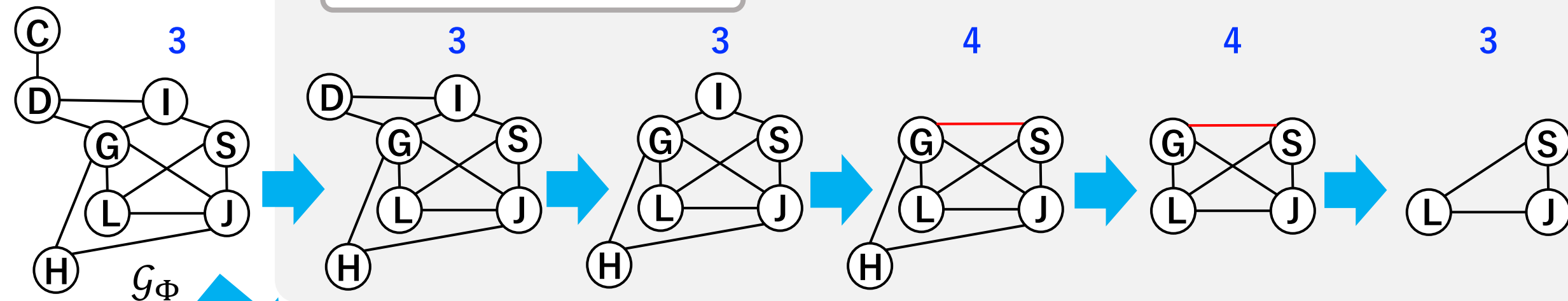
$$\blacklozenge \tau_3(\textcolor{red}{G}, \textcolor{red}{S}) = \sum_I \psi_3(G, I, S)$$

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

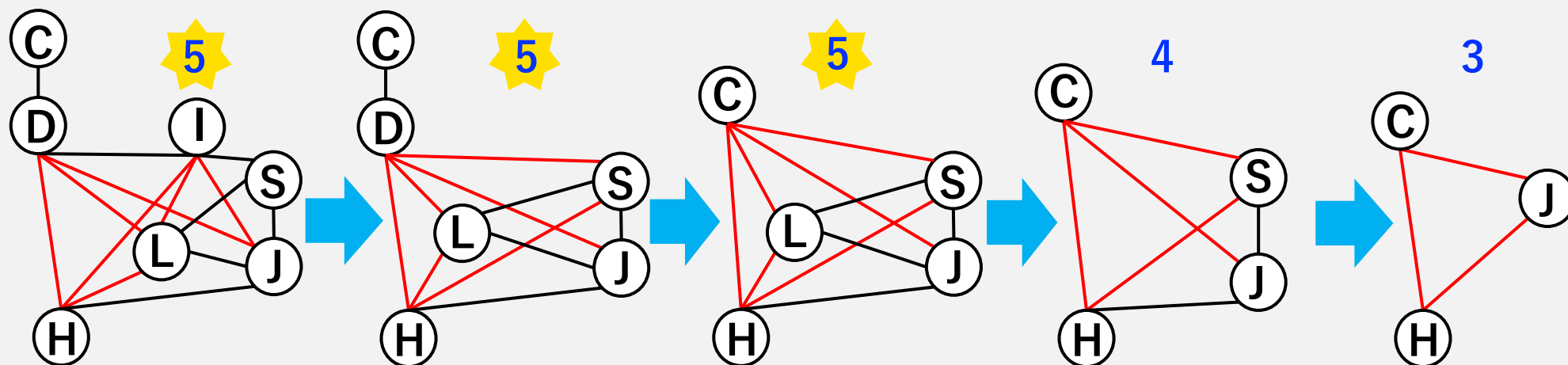


# Elimination Ordering依存性

$C \rightarrow D \rightarrow I \rightarrow H \rightarrow G \rightarrow$  の場合



$G \rightarrow I \rightarrow D \rightarrow L \rightarrow S \rightarrow$  の場合



※ 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 $\Phi$ , elimination orderingを $<$ とする.

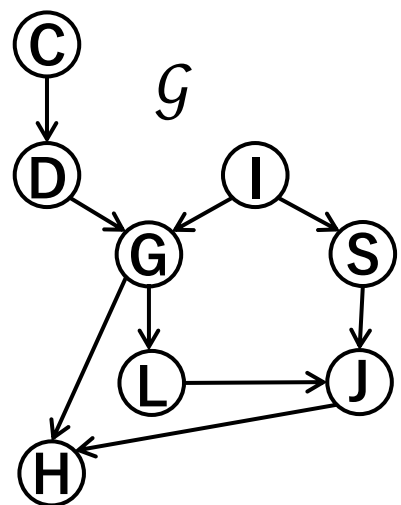
VE algorithmにより,  $<$ の通りに変数消去を行う際に生じた

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

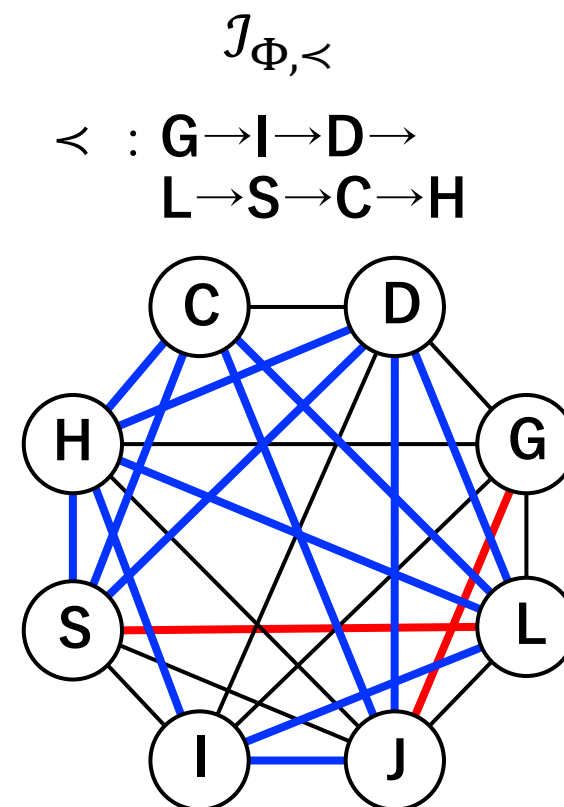
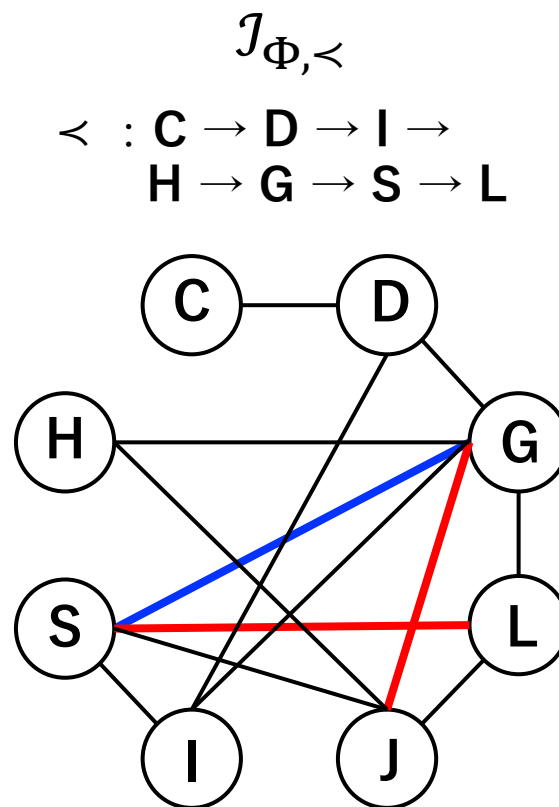
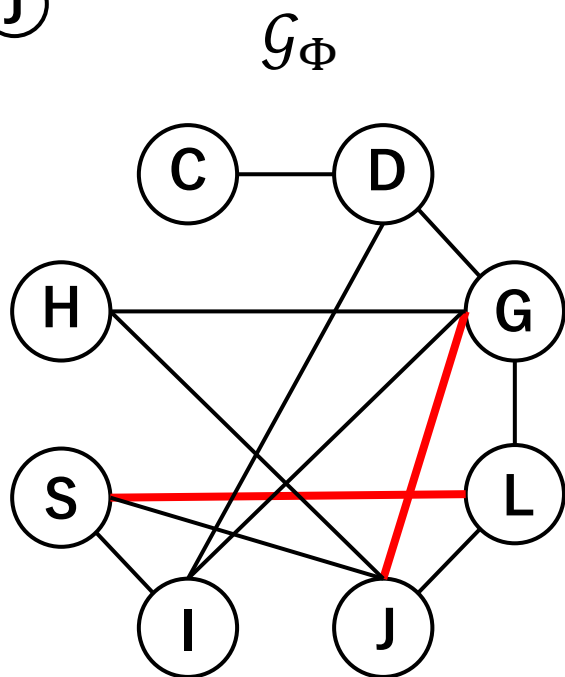
- $<$ にしたがって,  $1 \sim i$ 番目に消去される変数を $<(i)$ と表記する.
- グラフ $\mathcal{G}$ のedge集合を $\mathcal{E}_{\mathcal{G}}$ と表すと,

$$\mathcal{E}_{\mathcal{I}_{\Phi, <}} = \mathcal{E}_{\mathcal{G}_\Phi} \cup \mathcal{E}_{\mathcal{G}_{\Phi \setminus <(1)}} \cup \mathcal{E}_{\mathcal{G}_{\Phi \setminus <(2)}} \cup \dots \cup \mathcal{E}_{\mathcal{G}_{\Phi \setminus <(N)}}$$

# 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



※ 比較のため八角形状で表す



# 定理：Induced graphにおけるClique

Factor set  $\Phi$  と Elimination ordering  $\prec$  のもとでの Induced graph を  $\mathcal{I}_{\Phi, \prec}$  とする.

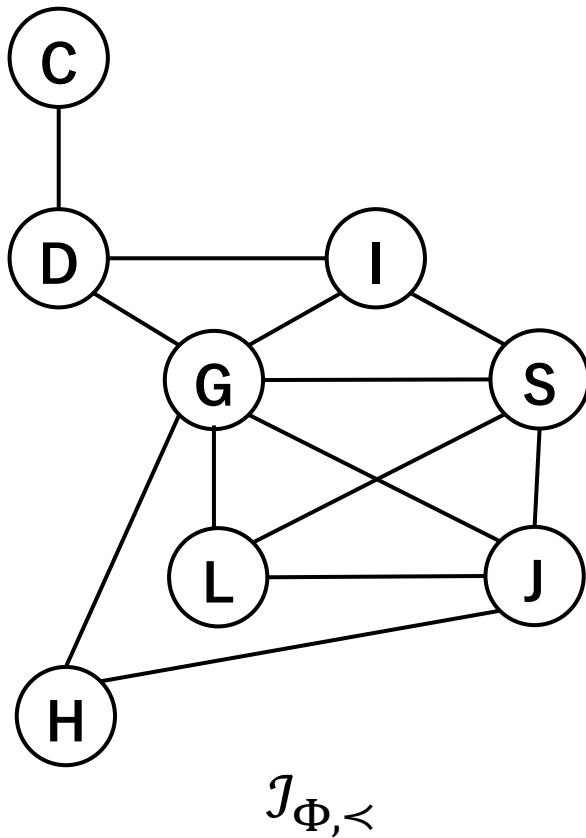
1. 変数消去過程において現れるすべてのfactorのscopeは,  
 $\mathcal{I}_{\Phi, \prec}$  においてクリークを構成する.
2.  $\mathcal{I}_{\Phi, \prec}$  における全てのクリークは,  
変数消去過程のある中途ステップで現れるfactorのscopeである.

## ◆ 証明 ◆

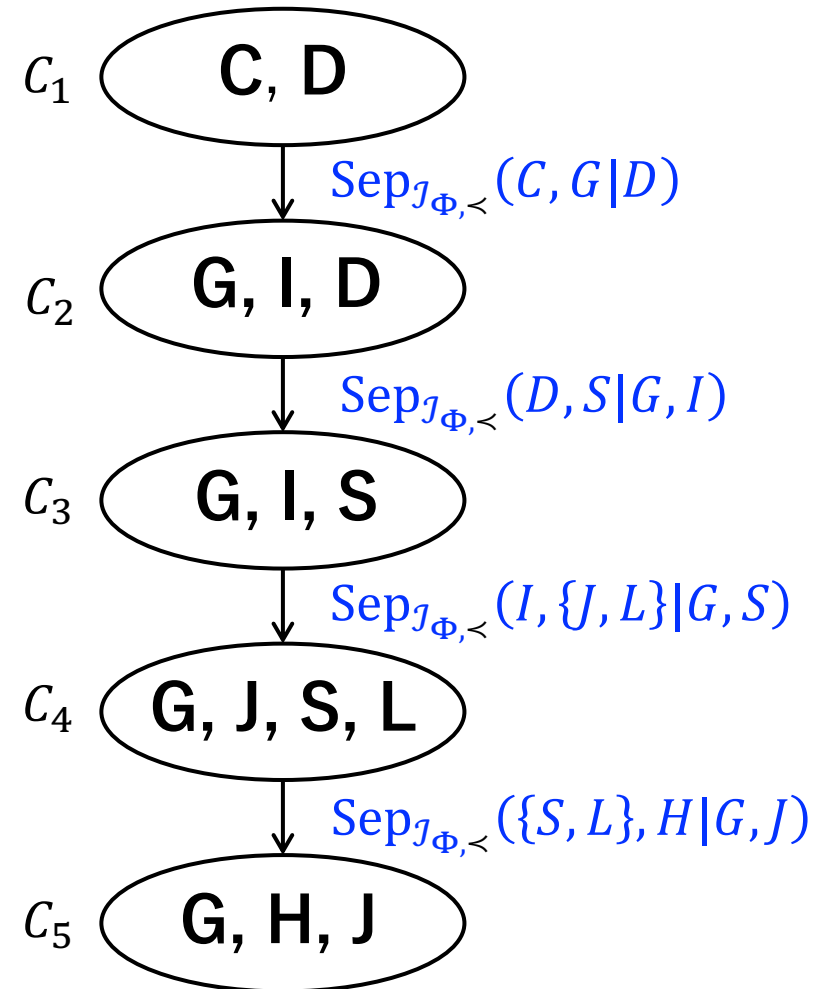
1. Induced graphの定義から明らか.
2. 変数消去過程から明らか. (変数消去過程が進めば, factorの最大クリークは小さくなる)

# Clique tree for Induced Graph

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

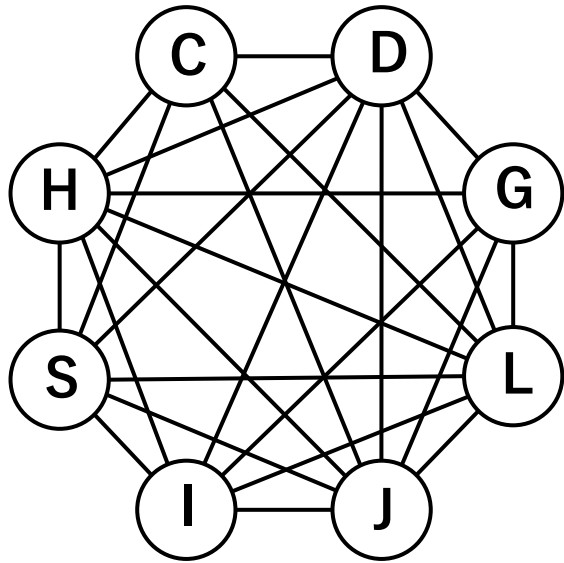


Clique Tree  $T$



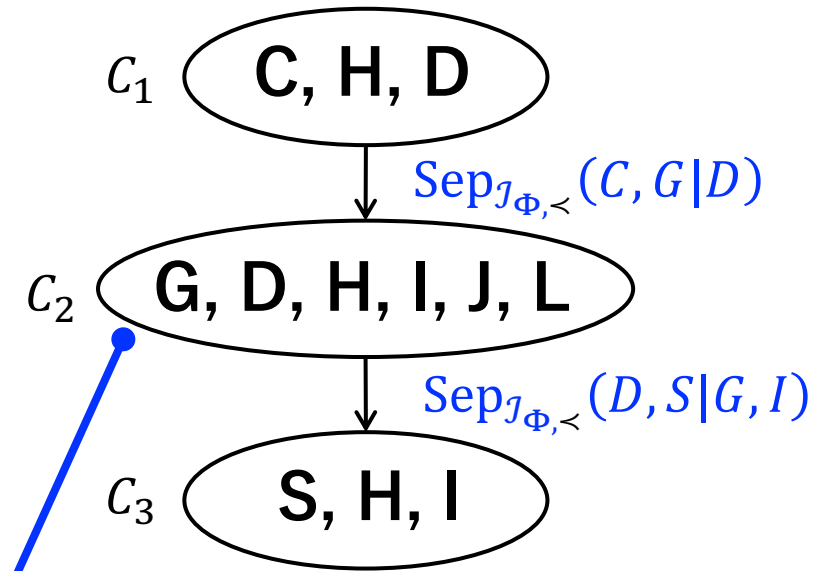
# Clique tree for Induced Graph

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



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

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 $\Phi$ のもとで定義される実効的無向グラフ $G_\Phi$ 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

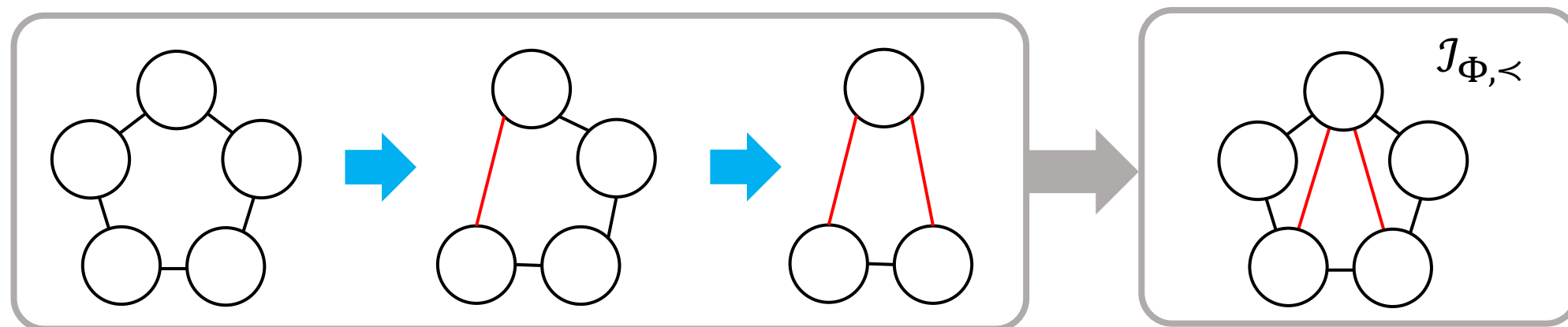
- レビュー論文: Heggernes, Discrete Mathematics (2006) 

# 定理： $\prec \Rightarrow$ コーダルグラフ

あらゆる Elimination ordering  $\prec$  と Factor 集合  $\Phi$  について、  
Induced graph  $\mathcal{I}_{\Phi, \prec}$  は コーダルグラフ である。

## ◆ 証明 ◆

- $\mathcal{G}_{\Phi}$  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 $\Rightarrow$ $<$

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  $<$  をもつ。

つまり、コーダルグラフ  $\mathcal{H}$  は、

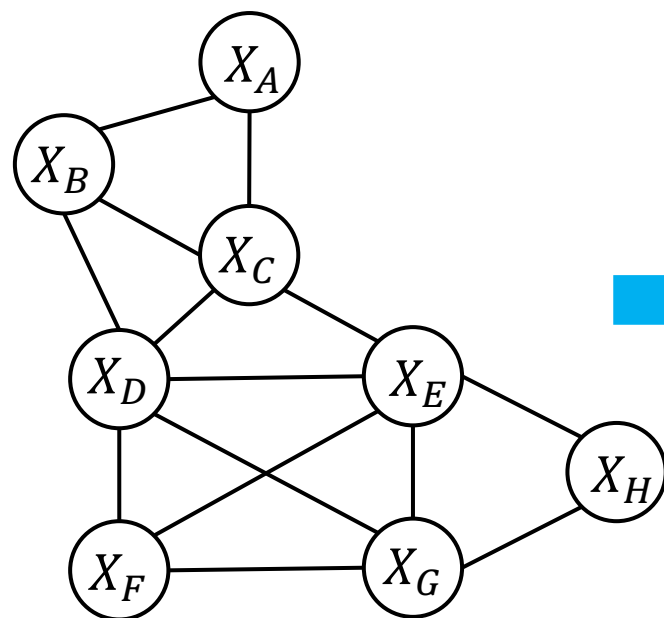
ある Elimination ordering  $<$  の Induced graph に対応する。

## ◆ 証明 ◆

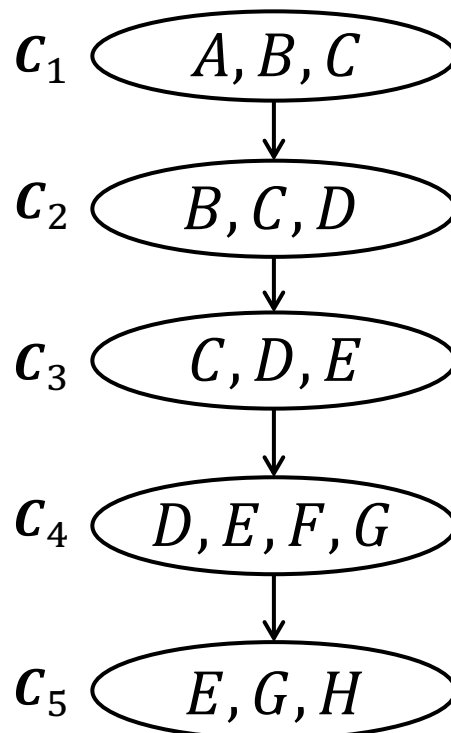
- コーダルグラフ  $\mathcal{H}$  はクリークツリー  $\mathcal{T}$  で表現される。
- Leaf に対応するクリーク  $\mathbf{C}_k$  は、隣接クリークに含まれない変数  $X_i$  を持つ。
- クリークの定義より  $X_i$  は  $\mathbf{C}_k \setminus X_i$  と全結合なので、 $X_i$  の消去により fill edge は生じない。
- $X_i$  を消去したグラフを  $\mathcal{H}'$  もコーダルグラフであるので、  
同様に Leaf に対応するクリークの中で隣接クリークに含まれ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

# 証明を絵で描くと

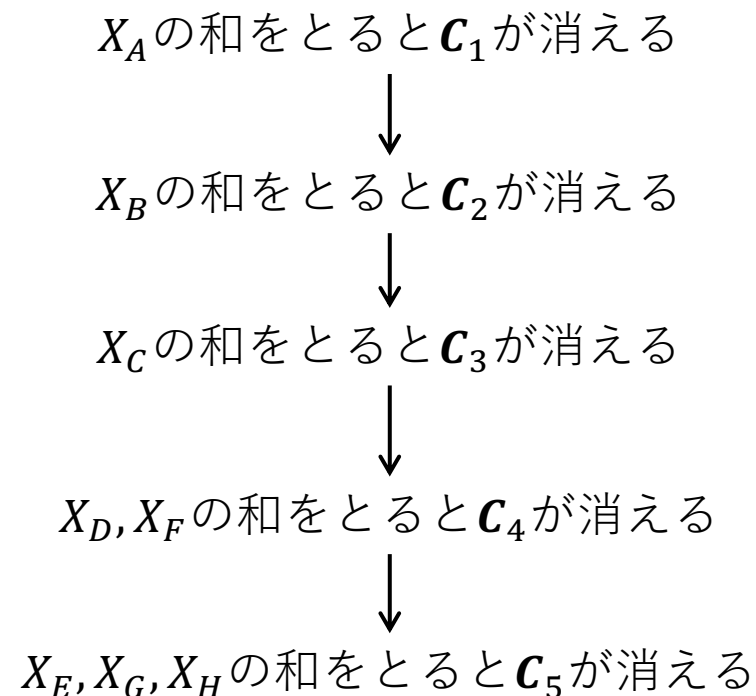
Chordal Graph  $\mathcal{H}$



Clique Tree  $\mathcal{T}$



Elimination Ordering

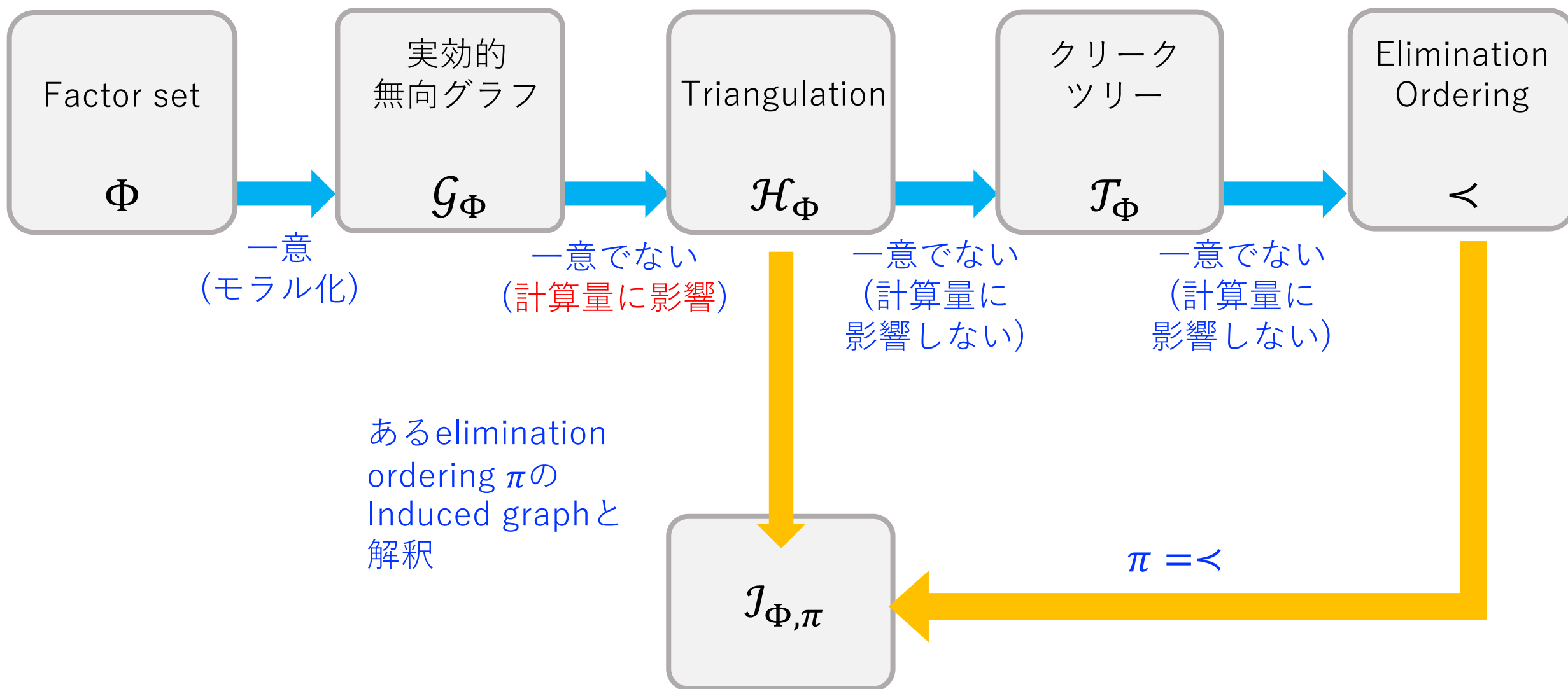


$\mathcal{H}$  は  $\leq A \rightarrow B \rightarrow C \rightarrow \boxed{D \rightarrow F} \rightarrow \boxed{E \rightarrow G \rightarrow H}$  の Induced Graph に対応

※  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： $\mathbf{X}_B = \mathbf{e}$ というエビデンスを得たとする ( $B \subset \mathcal{V}$ )

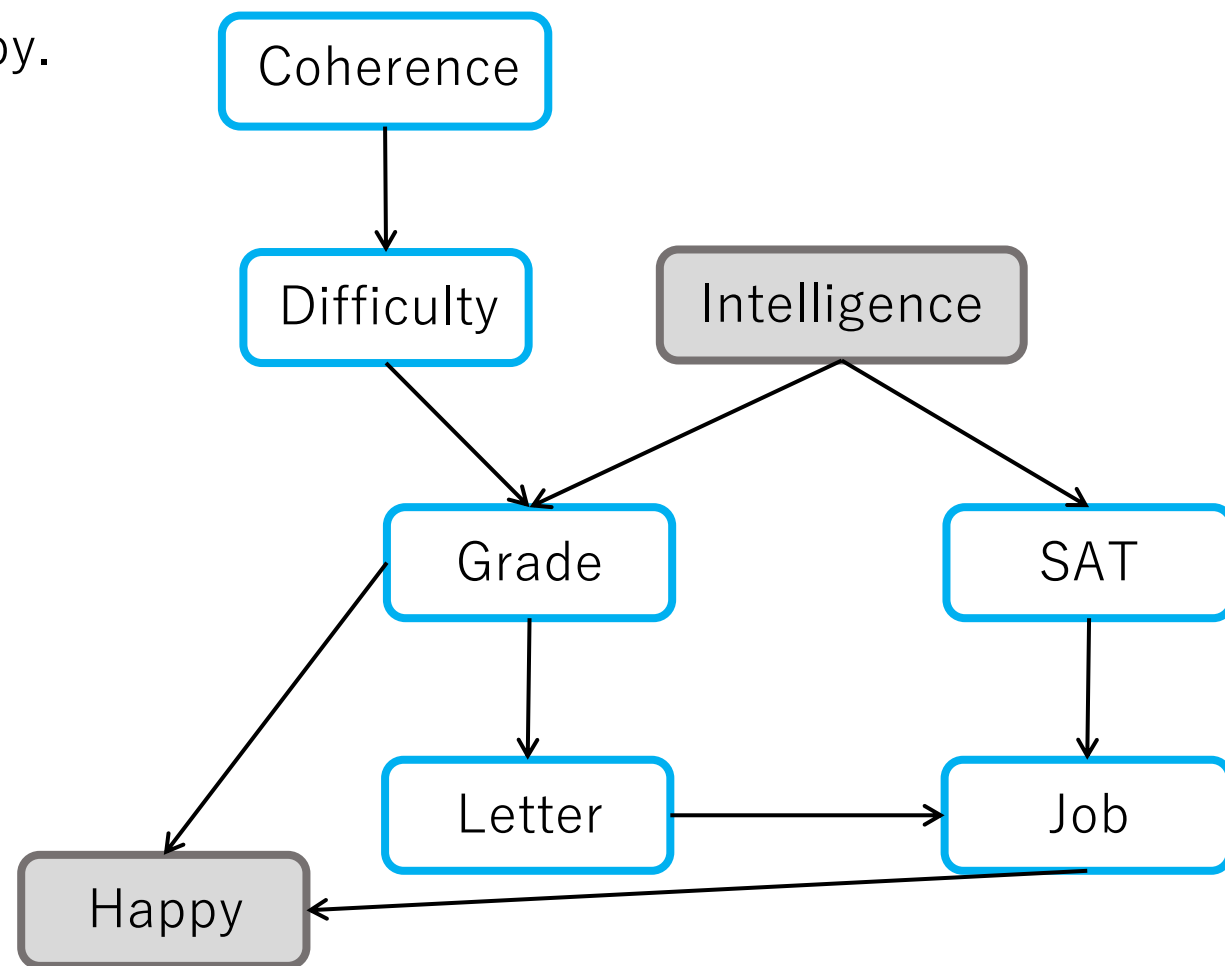
◆  $P(\mathbf{X}_{\mathcal{V} \setminus B}, \mathbf{X}_B = \mathbf{e})$ は確率ではない (規格化されていない)

◆  $P(\mathbf{X}_{\mathcal{V} \setminus B} | \mathbf{X}_B = \mathbf{e}) = \frac{P(\mathbf{X}_{\mathcal{V} \setminus B}, \mathbf{X}_B = \mathbf{e})}{P(\mathbf{X}_B = \mathbf{e})}$  とすれば確率になる

- $P(\mathbf{X}_B = \mathbf{e}) = \sum_{\mathbf{X}_{\mathcal{V} \setminus B}} P(\mathbf{X}_{\mathcal{V} \setminus B}, \mathbf{X}_B = \mathbf{e})$  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  $\{\mathcal{D}, p_B(\mathbf{X})\}$ において、観測 $\mathbf{X}_A = \mathbf{e}$ が得られたとする。  
このとき $\mathbf{X}_{V \setminus A}$ の確率分布 $p_B(\mathbf{X}_{V \setminus A} | \mathbf{X}_A = \mathbf{e})$ は、  
次の特徴を持つGibbs分布により与えられる。

- ファクター $\phi_i = p_B(X_i | \mathbf{X}_{\pi(i) \setminus A}, \{X_j = e_j : j \in \pi(i) \text{ and } j \in A\})$ からなる
- 分配関数は $p_B(\mathbf{X}_A = \mathbf{e})$ と一致

◆ 証明 ◆

$$p_B(\mathbf{X}_{V \setminus A}, \mathbf{X}_A = \mathbf{e}) = \prod_{i \in A} p_B(X_i = e_i | \mathbf{X}_{\pi(i) \setminus A}, \{X_j = e_j : j \in \pi(i) \text{ and } j \in A\}) \\ \times \prod_{i \in V \setminus A} p_B(X_i | \mathbf{X}_{\pi(i) \setminus A}, \{X_j = e_j : j \in \pi(i) \text{ and } j \in A\})$$

$\mathbf{X}_{V \setminus A}$ の分布として規格化されていないので、分配関数 $\sum_{\mathbf{X}_{V \setminus A}} p_B(\mathbf{X}_{V \setminus A}, \mathbf{X}_A = \mathbf{e}) = p_B(\mathbf{X}_A = \mathbf{e})$ が必要

# Sum-Product VE for conditional probabilities

Input

- $\Phi$  : Evidence所与のもとでの全てのファクターのセット
- $\varepsilon$  : 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

- $\phi^*$  : 変数消去されたファクター ( $X_v \notin \text{Scope}[\phi^*]$ )
- $Z$  : 分配関数

For  $i = 1, \dots, k$

$$\Phi_{\text{el}} \leftarrow \{\phi \in \Phi : X_{v_i} \in \text{Scope}[\phi]\}$$

◁ 消去したい変数が入っているファクター

$$\Phi_{\text{re}} \leftarrow \Phi \setminus \Phi_{\text{el}}$$

◁ 消去したい変数が入っていないファクター

$$\psi \leftarrow \prod_{\phi \in \Phi_{\text{el}}} \phi$$

◁ 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

$$\Phi \leftarrow \Phi_{\text{re}} \cup \tau$$

◁ 新しいファクターのセット

end For

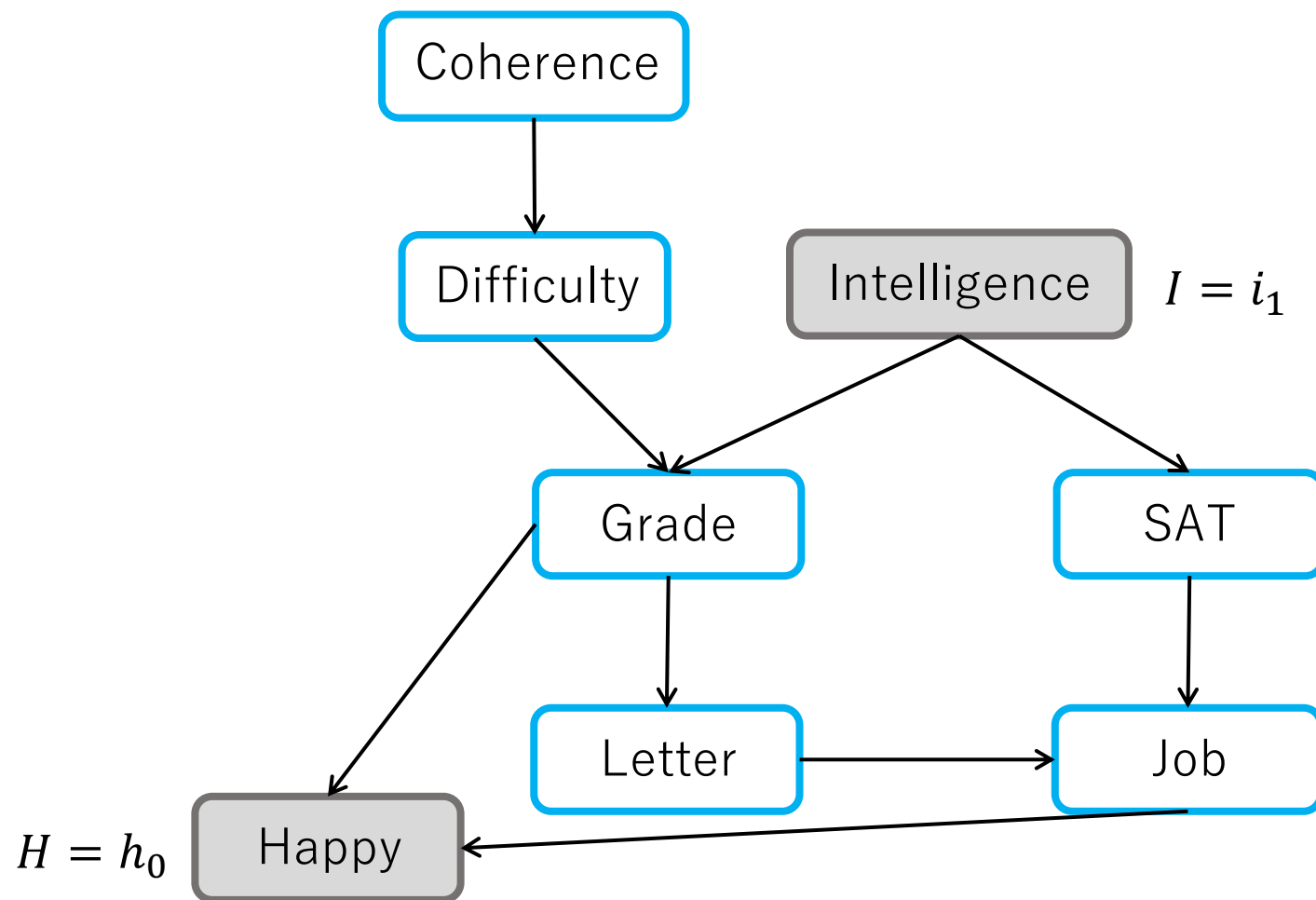
$$\phi^* \leftarrow \prod_{\phi \in \Phi} \phi$$

◁ 変数消去されたファクターを掛け合わせる

$$Z \leftarrow \sum_{X_{V \setminus (\varepsilon \cup v)}} \phi^*(X_{V \setminus (\varepsilon \cup v)})$$

◁ 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

$$\blacklozenge \psi_1(C, D) = \phi_C(C)\phi_D(D, C)$$

$$\blacklozenge \tau_1(D) = \sum_C \psi_1(C, D)$$

### 2. **D**のElimination

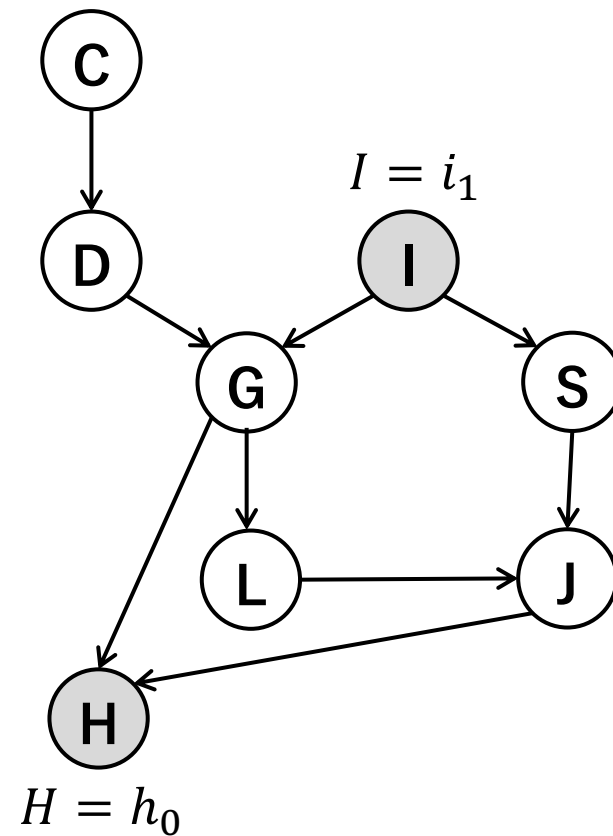
$$\blacklozenge \psi_2(G, D) = \phi_G(G, I = i_0, D)\tau_1(D)$$

$$\blacklozenge \tau_2(G) = \sum_D \psi_2(G, D)$$

### 3. **G**のElimination

$$\blacklozenge \psi_3(G, L, J) = \phi_L(L, G)\phi_H(H = h_0, G, J)\tau_2(G)$$

$$\blacklozenge \tau_3(L, J) = \sum_G \psi_3(G, L, J)$$





- 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

$$\blacklozenge \psi_4(S, L, J) = \phi_S(S, I = i_0)\phi_J(J, S, L)$$

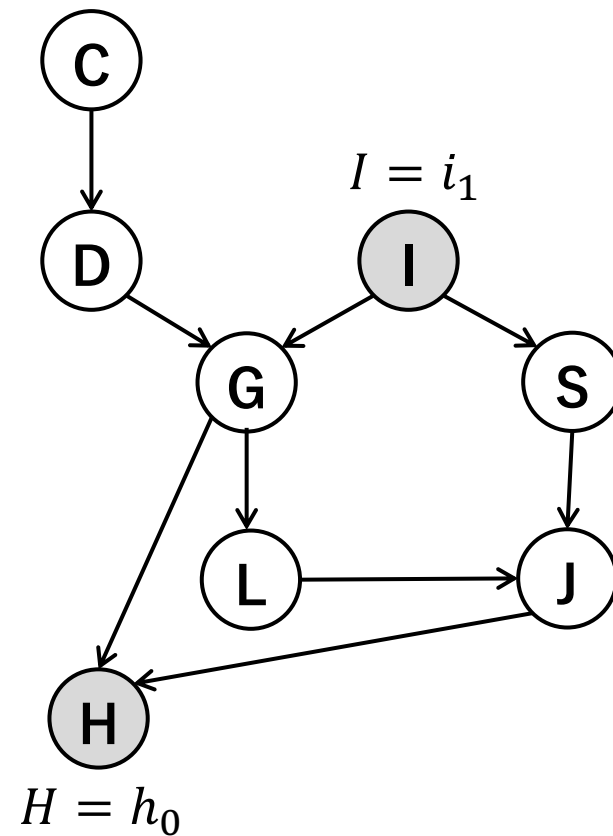
$$\blacklozenge \tau_4(L, J) = \sum_S \psi_4(S, L, J)$$

#### 5. **L**のElimination

$$\blacklozenge \psi_5(L, J) = \tau_3(L, J)\tau_4(L, J)$$

$$\blacklozenge \tau_5(J) = \sum_L \psi_5(L, J)$$

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



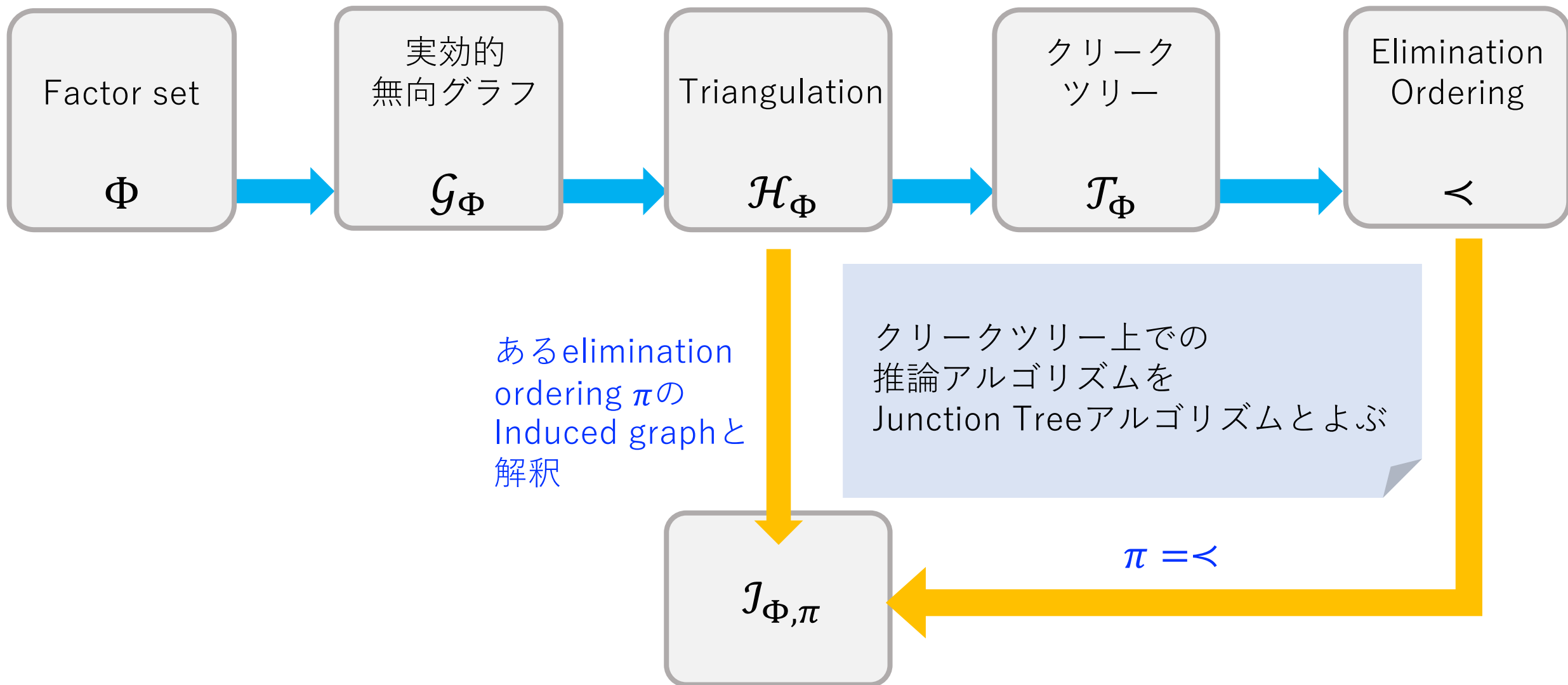
# まとめ

- 変数消去について，基本的な考え方を学んだ

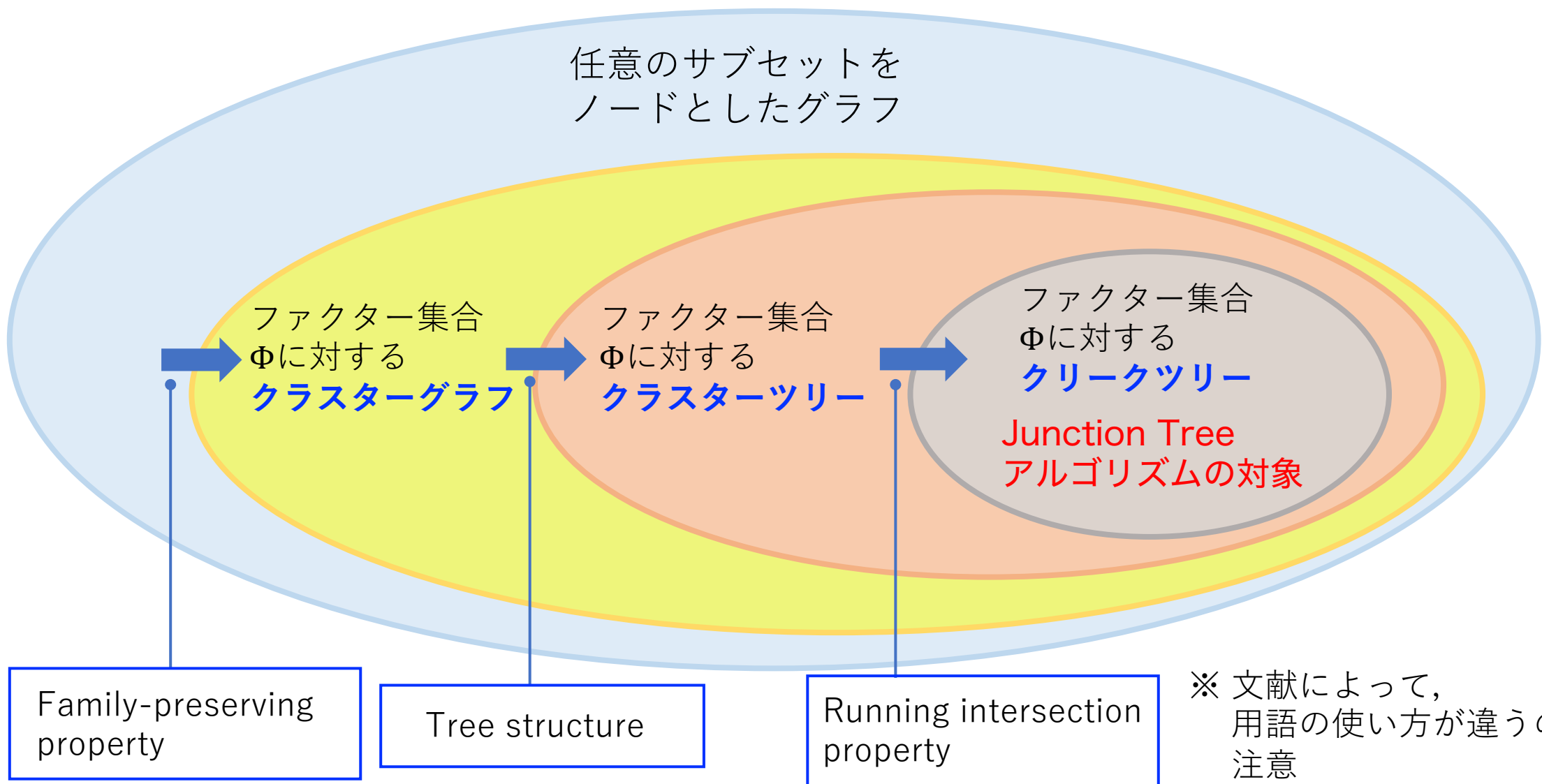
### 3. クリークツリーを用いた推論

Junction Tree algorithm

# elimination orderingの構成方法(再掲)



# 準備：クリークツリー



# 定義：Cluster graph

各ノード $i$ が変数集合のサブセット $\mathbf{C}_i$ に対応し，ファクター集合 $\Phi$ 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 $\mathbf{C}_i$ と $\mathbf{C}_j$ をつなぐedgeはsepset  $\mathbf{C}_i \cap \mathbf{C}_j$ を表す.

※ ファクター $\forall \phi \in \Phi$ が $\text{Scope}[\phi] \subseteq \mathbf{C}_i \exists i$ 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 $\Phi$ のもとでのクラスターツリーを $\mathcal{T}$ とする.

$X \in \mathbf{C}_i$ ,  $X \in \mathbf{C}_j$ 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  
 $\mathbf{C}_i$ と $\mathbf{C}_j$ 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 $\mathbf{C}_i$ から $\mathbf{C}_j$ にメッセージが送られる際に変数 $\mathbf{X}$ が消去されたとする.  
→  $\mathbf{C}_j$ には $\mathbf{X}$ が含まれない
- Running intersection propertyより,  $\mathbf{C}_j$ より上流で $\mathbf{X}$ 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 $\{\mathbf{C}_i\}$ とエッジ $\{\text{Sep}_{ij}\}$ 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 $\text{Sep}_{ij}$ が $\mathbf{C}_i$ より手前のクリークと $\mathbf{C}_j$ より後のクリークを分ける.



## 3.1 Junction Tree アルゴリズムとは

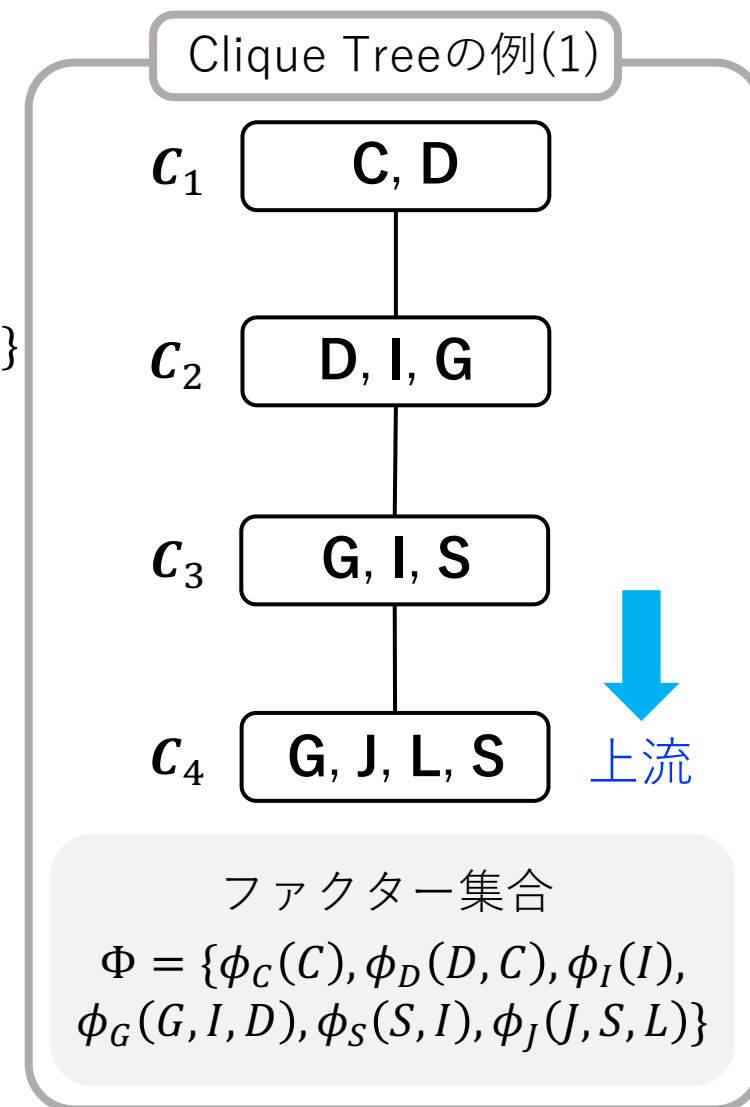
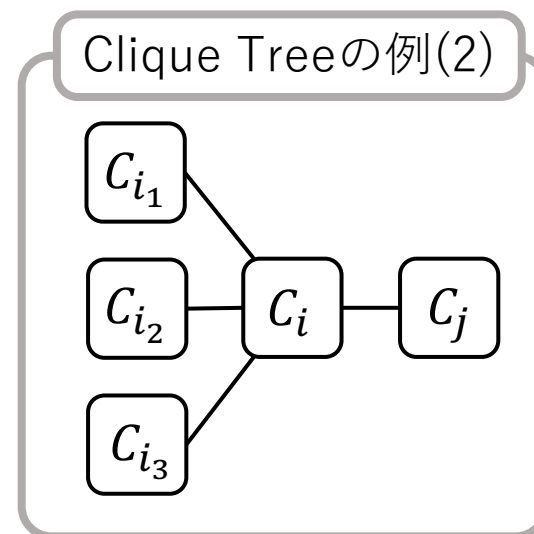
# 準備と基本的考え方

- $\mathcal{V}_{<(i \rightarrow j)}$  :  $\mathbf{C}_i$ 含む下流のCliqueから $\text{Sep}_{ij}$ を除いた変数集合
- $\mathcal{F}_{<(i \rightarrow j)}$  :  $\mathbf{C}_i$ 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,  
 $\mathcal{V}_{<(3 \rightarrow 4)} = \{C, D, I\}$ ,  $\mathcal{F}_{<(3 \rightarrow 4)} = \{\phi(C), \phi(C, D), \phi(D, I, G), \phi(I), \phi(S, I)\}$

- Running intersection propertyより, Clique Treeの例(2)の場合

- $\mathcal{V}_{<(i_k \rightarrow i)}$  ( $k = 1, 2, 3$ )はdisjoint
- $\mathcal{F}_{<(i_k \rightarrow j)}$  ( $k = 1, 2, 3$ )はdisjoint
- $\mathcal{V}_{<(i \rightarrow j)} = \mathcal{V}_{<(i_1 \rightarrow i)} \cup \dots \cup \mathcal{V}_{<(i_k \rightarrow i)}$
- $\mathcal{F}_{<(i \rightarrow j)} = \mathcal{F}_{<(i_1 \rightarrow i)} \cup \dots \cup \mathcal{F}_{<(i_k \rightarrow i)} \cup \{\phi \mid \alpha(\phi) = i\}$

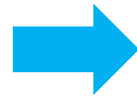
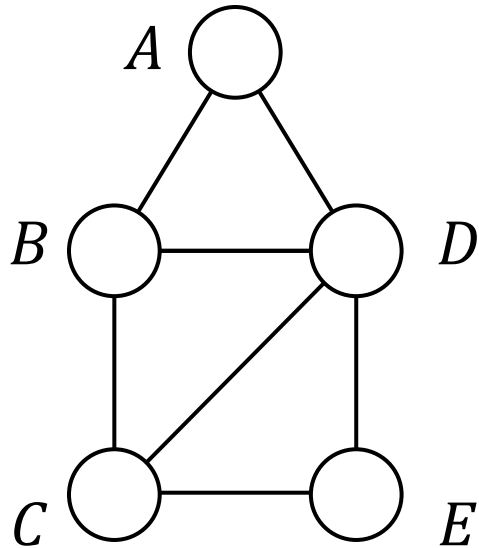
→  $\mathcal{V}_{<(i_k \rightarrow i)}$  ( $k = 1, 2, 3$ )について独立に和を取ることができる



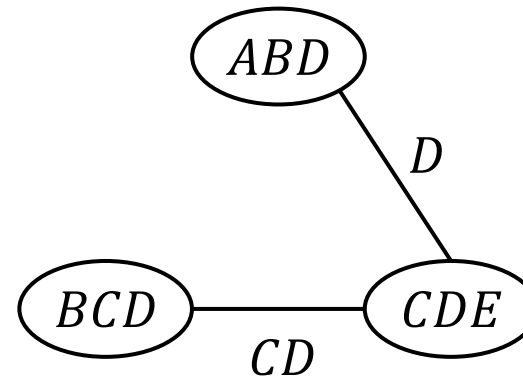
# Junction Tree アルゴリズムの手続き

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

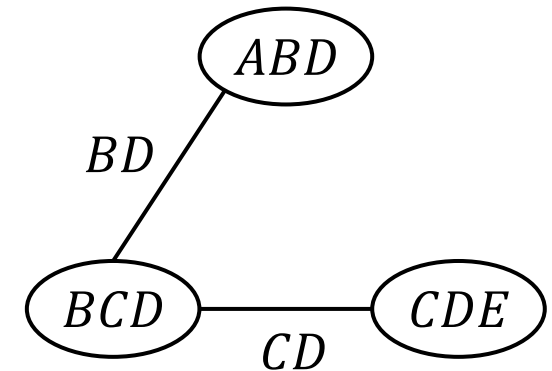


Clique Treeではない




※  $B$ 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

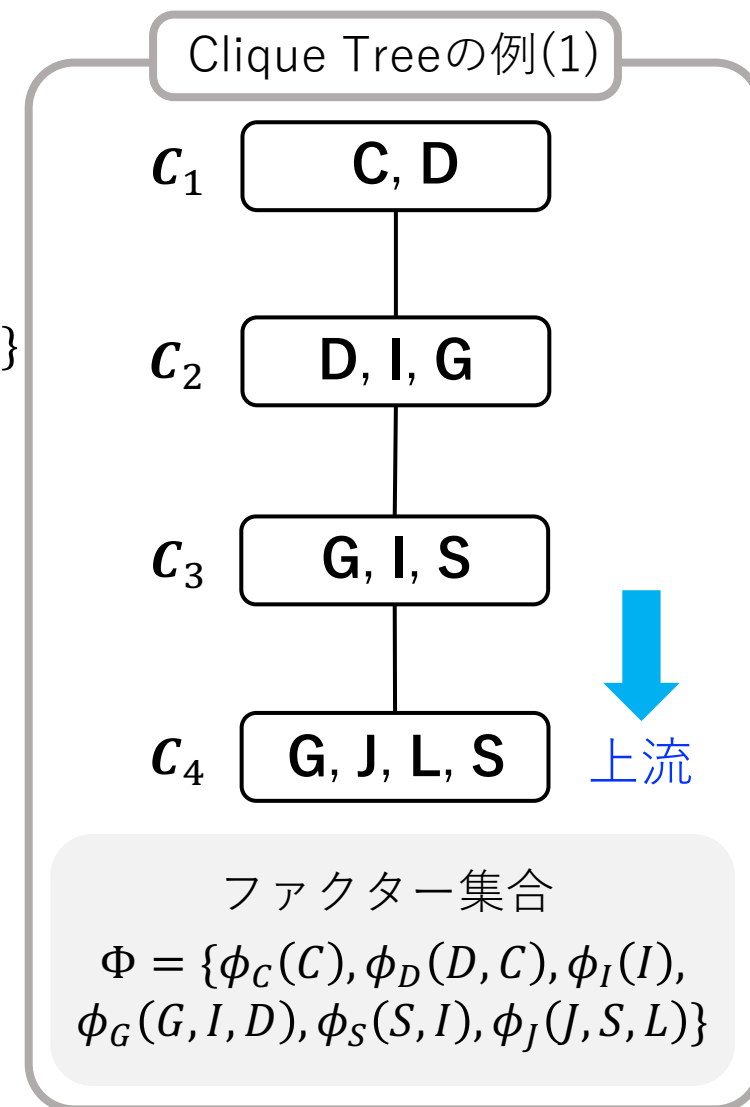
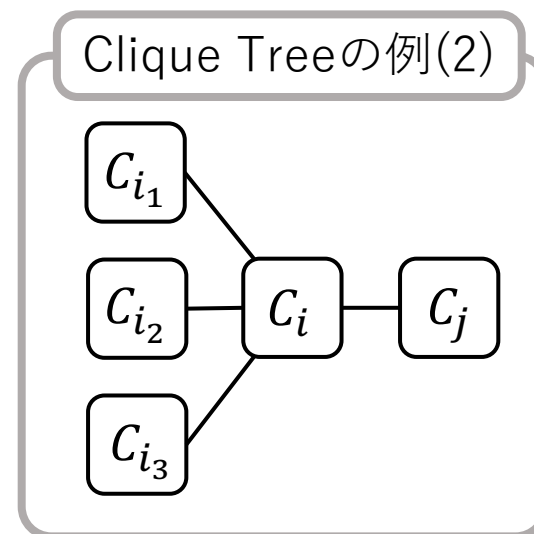
# 準備と基本的考え方(再掲)

- $\mathcal{V}_{<(i \rightarrow j)}$  :  $\mathbf{C}_i$ 含む下流のCliqueから $\text{Sep}_{ij}$ を除いた変数集合
- $\mathcal{F}_{<(i \rightarrow j)}$  :  $\mathbf{C}_i$ 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,  
 $\mathcal{V}_{<(3 \rightarrow 4)} = \{C, D, I\}$ ,  $\mathcal{F}_{<(3 \rightarrow 4)} = \{\phi(C), \phi(C, D), \phi(D, I, G), \phi(I), \phi(S, I)\}$

- Running intersection propertyより, Clique Treeの例(2)の場合

- $\mathcal{V}_{<(i_k \rightarrow i)}$  ( $k = 1, 2, 3$ )はdisjoint
- $\mathcal{F}_{<(i_k \rightarrow j)}$  ( $k = 1, 2, 3$ )はdisjoint
- $\mathcal{V}_{<(i \rightarrow j)} = \mathcal{V}_{<(i_1 \rightarrow i)} \cup \dots \cup \mathcal{V}_{<(i_k \rightarrow i)}$
- $\mathcal{F}_{<(i \rightarrow j)} = \mathcal{F}_{<(i_1 \rightarrow i)} \cup \dots \cup \mathcal{F}_{<(i_k \rightarrow i)} \cup \{\phi \mid \alpha(\phi) = i\}$

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



# 目標

- $\mathbf{c}_r$  をルートクリークとし,  $\tilde{P}_{\Phi}(\mathbf{c}_r) = \sum_{\mathbf{x} \setminus \mathbf{c}_r} \prod_{\phi \in \Phi} \phi$  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 :

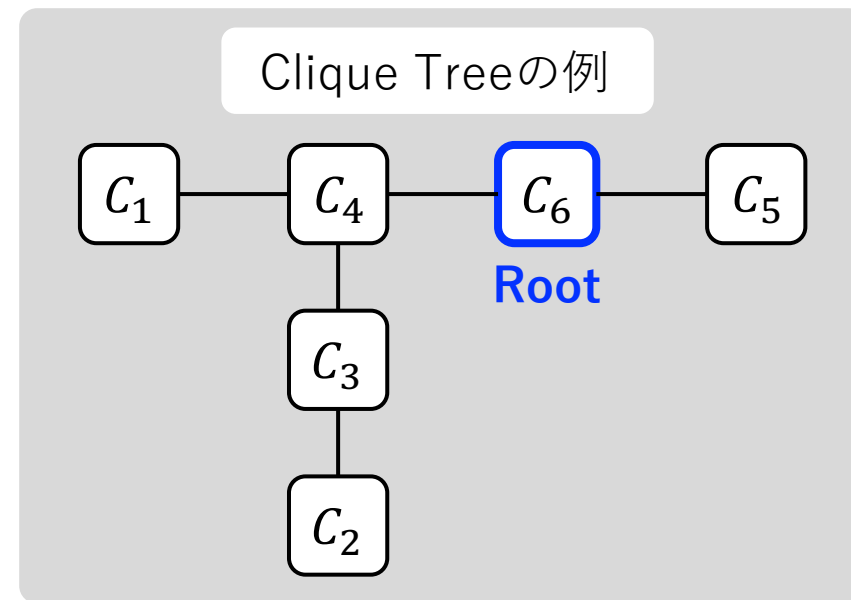
下流にクリークを持たないクリーク

(右図の場合は  $\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_5$ )

- 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



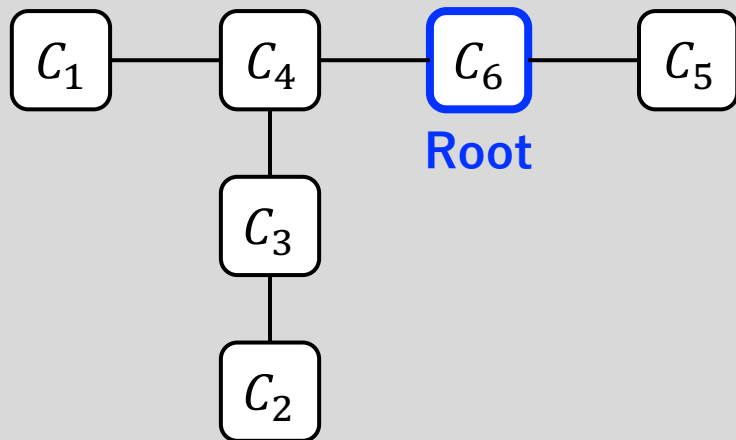
# Clique-Tree Message Passingの手続き

- ファクター $\phi$ 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 $\mathbf{C}_i$ から始め, ルートに向かって和をとっていく
- $\mathbf{C}_i$ の近接クリークのラベル集合を $\partial_i$ とし, そのうち上流クリークを $\partial_i^+$ とする
- クリークに対してポテンシャル $\psi_j(\mathbf{C}_j)$ 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 $\prod_{\phi \in \Phi} \phi = \prod_j \varphi_j(\mathbf{C}_j)$
- 各クリーク $\mathbf{C}_i$ で**Message** $\delta_{i \rightarrow j}$  ( $j = \partial_i^+$ )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)

Clique Treeの例



- 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： $\boldsymbol{\theta} = \{4, 3, (2, 4), (1, 3, 6), 6, (4, 5)\}$
- 上流の近接クリーク： $\boldsymbol{\theta}^+ = \{4, 3, 4, 6, 6, \emptyset\}$

$$\begin{aligned} \psi_2(\mathbf{C}_2) &\leftarrow \varphi_2 \\ \delta_{2 \rightarrow 3}(\text{Sep}_{2,3}) &\leftarrow \sum_{\mathbf{C}_2 \setminus \text{Sep}_{2,3}} \psi_2(\mathbf{C}_2) \end{aligned}$$

$$\begin{aligned} \psi_3(\mathbf{C}_3) &\leftarrow \varphi_3 \delta_{2 \rightarrow 3} \\ \delta_{3 \rightarrow 4}(\text{Sep}_{3,4}) &\leftarrow \sum_{\mathbf{C}_3 \setminus \text{Sep}_{3,4}} \psi_3(\mathbf{C}_3) \end{aligned}$$

$$\begin{aligned} \psi_1(\mathbf{C}_1) &\leftarrow \varphi_1 \\ \delta_{1 \rightarrow 4}(\text{Sep}_{1,4}) &\leftarrow \sum_{\mathbf{C}_1 \setminus \text{Sep}_{1,4}} \psi_1(\mathbf{C}_1) \end{aligned}$$

Belief:

$$\beta_6(\mathbf{C}_6) \leftarrow \varphi_6 \delta_{4 \rightarrow 6} \delta_{5 \rightarrow 6}$$

$$\begin{aligned} \psi_5(\mathbf{C}_5) &\leftarrow \varphi_5 \\ \delta_{5 \rightarrow 6}(\text{Sep}_{5,6}) &\leftarrow \sum_{\mathbf{C}_5 \setminus \text{Sep}_{5,6}} \psi_5(\mathbf{C}_5) \end{aligned}$$

$$\begin{aligned} \psi_4(\mathbf{C}_4) &\leftarrow \varphi_4 \delta_{1 \rightarrow 4} \delta_{3 \rightarrow 4} \\ \delta_{4 \rightarrow 6}(\text{Sep}_{4,6}) &\leftarrow \sum_{\mathbf{C}_4 \setminus \text{Sep}_{4,6}} \psi_4(\mathbf{C}_4) \end{aligned}$$

# Clique-Tree Message Passingの例(1)

$\tilde{P}_\Phi(\mathbf{c}_6) = \sum_{X \setminus \mathbf{c}_6} \prod_{\phi \in \Phi} \phi$  と  $\beta_6(\mathbf{c}_6) = \varphi_6 \delta_{4 \rightarrow 6} \delta_{5 \rightarrow 6}$  が一致しているか確認

$$\begin{aligned}
 \beta_6(\mathbf{c}_6) &= \varphi_6 \delta_{4 \rightarrow 6} \delta_{5 \rightarrow 6} \\
 &= \varphi_6 \sum_{\mathbf{c}_4 \setminus \text{Sep}_{4,6}} \varphi_4 \delta_{1 \rightarrow 4} \delta_{3 \rightarrow 4} \sum_{\mathbf{c}_5 \setminus \text{Sep}_{5,6}} \varphi_5 \\
 &= \varphi_6 \sum_{\mathbf{c}_4 \setminus \text{Sep}_{4,6}} \varphi_4 \sum_{\mathbf{c}_1 \setminus \text{Sep}_{1,4}} \varphi_1 \sum_{\mathbf{c}_3 \setminus \text{Sep}_{3,4}} \varphi_3 \delta_{2 \rightarrow 3} \sum_{\mathbf{c}_5 \setminus \text{Sep}_{5,6}} \varphi_5 \\
 &= \varphi_6 \sum_{\mathbf{c}_4 \setminus \text{Sep}_{4,6}} \varphi_4 \sum_{\mathbf{c}_1 \setminus \text{Sep}_{1,4}} \varphi_1 \sum_{\mathbf{c}_3 \setminus \text{Sep}_{3,4}} \varphi_3 \sum_{\mathbf{c}_2 \setminus \text{Sep}_{2,3}} \varphi_2 \sum_{\mathbf{c}_5 \setminus \text{Sep}_{5,6}} \varphi_5 \\
 &= \sum_{X \setminus \mathbf{c}_6} \prod_{\phi \in \Phi} \phi
 \end{aligned}$$

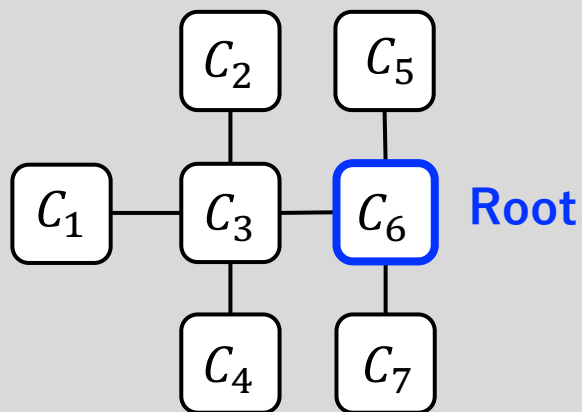
$$\psi_i = \prod_{\phi: \alpha(\phi)=i} \phi$$

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

$\text{Sep}_{ij} \in \mathbf{c}_j$  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： $\boldsymbol{\theta} = \{3, 3, (1, 2, 4, 6), 3, 6, (3, 5, 7), 6\}$
- 上流の近接クリーク： $\boldsymbol{\theta}^+ = \{3, 3, 6, 3, 6, \emptyset, 6\}$

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

$$\delta_{1 \rightarrow 3}(\text{Sep}_{1,3}) \leftarrow \sum_{\mathcal{C}_1 \setminus \text{Sep}_{1,3}} \psi_1(\mathcal{C}_1)$$

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

$$\delta_{2 \rightarrow 3}(\text{Sep}_{2,3}) \leftarrow \sum_{\mathcal{C}_2 \setminus \text{Sep}_{2,3}} \psi_2(\mathcal{C}_2)$$

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

$$\delta_{5 \rightarrow 6}(\text{Sep}_{5,6}) \leftarrow \sum_{\mathcal{C}_5 \setminus \text{Sep}_{5,6}} \psi_5(\mathcal{C}_5)$$

$$\psi_3(\mathcal{C}_3) \leftarrow \varphi_3 \delta_{4 \rightarrow 3} \delta_{1 \rightarrow 3} \delta_{2 \rightarrow 3}$$

$$\delta_{3 \rightarrow 6}(\text{Sep}_{3,6}) \leftarrow \sum_{\mathcal{C}_3 \setminus \text{Sep}_{3,6}} \psi_3(\mathcal{C}_3)$$

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

$$\delta_{4 \rightarrow 3}(\text{Sep}_{4,3}) \leftarrow \sum_{\mathcal{C}_4 \setminus \text{Sep}_{4,3}} \psi_4(\mathcal{C}_4)$$

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

$$\delta_{7 \rightarrow 6}(\text{Sep}_{7,6}) \leftarrow \sum_{\mathcal{C}_7 \setminus \text{Sep}_{7,6}} \psi_7(\mathcal{C}_7)$$

**Belief:**

$$\beta_6(\mathcal{C}_6) \leftarrow \varphi_6 \delta_{3 \rightarrow 6} \delta_{5 \rightarrow 6} \delta_{7 \rightarrow 6}$$

# Sum-Product Message Passing for Clique Trees

Input

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

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

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

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

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

$i \leftarrow \prec_c(j)$

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

$\delta_{i \rightarrow \partial_i^+}(\text{Sep}_{i, \partial i^+}) \leftarrow \sum_{\mathbf{C}_i \setminus \text{Sep}_{i, \partial i^+}} \psi_i(\mathbf{C}_i)$

end For

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

## 4. Junction Tree アルゴリズム

# 代表的なアルゴリズム

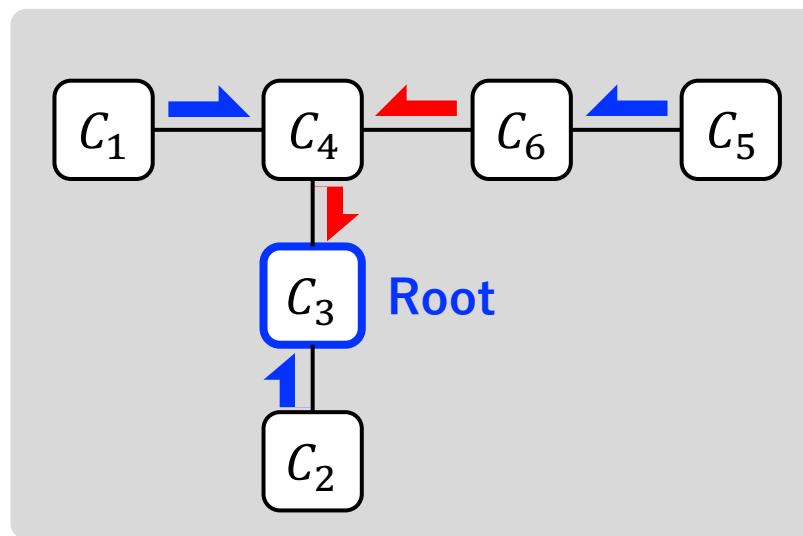
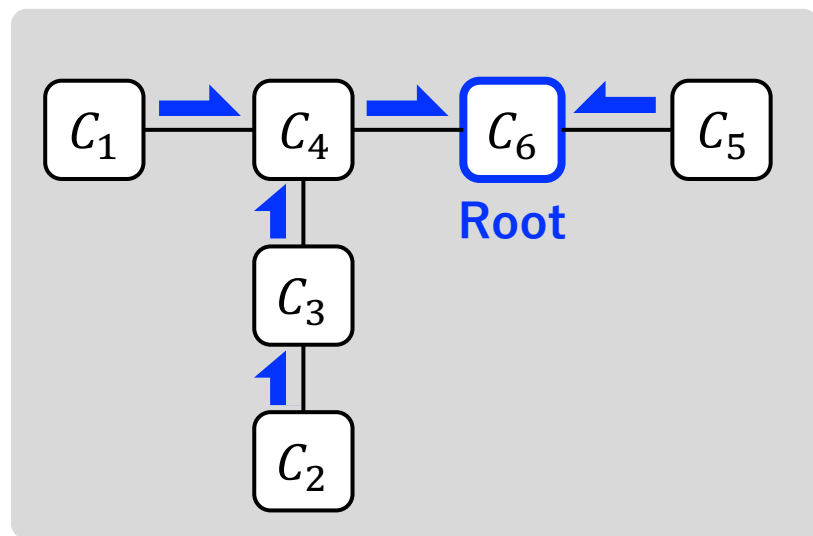
- Shafer-Shenoyアルゴリズム
  - ◆ Sum-product message passing
- Lauritzen-Spiegelhalterアルゴリズム
  - ◆ Belief update型
- Huginアルゴリズム
  - ◆ Separator potentialの導入

## 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

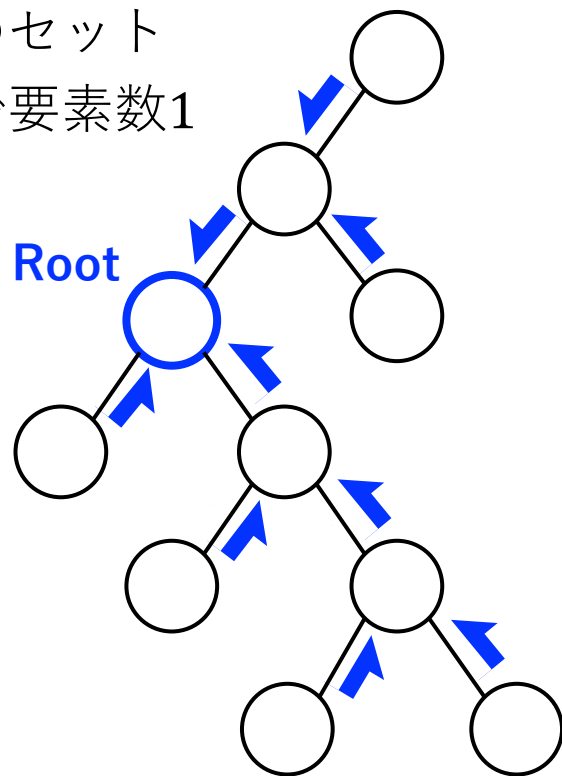


# 二つのプロトコルを定義

- COLLECT\_MESSAGE( $i$ )

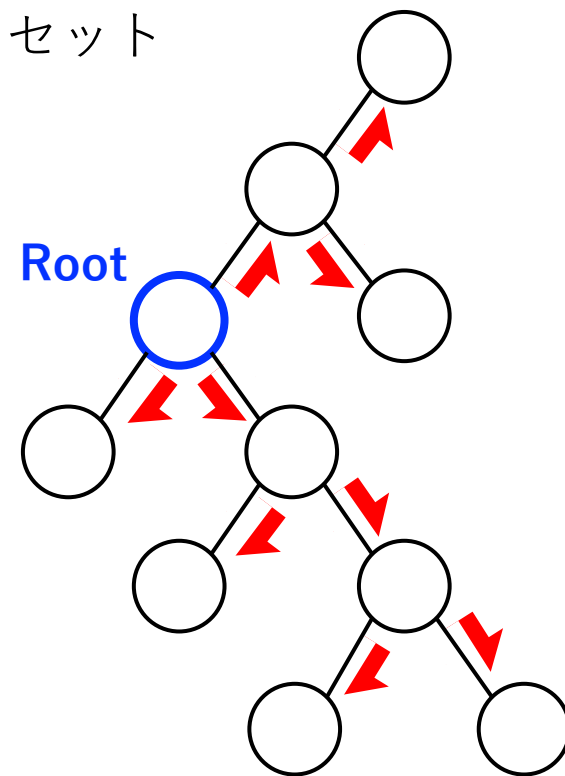
- 下流からノード $i$ に入るメッセージを”集めて”送る
- 出力は $i$ から上流ノードへのメッセージのセット

※ ツリーなので要素数1



- DISTRIBUTE\_MESSAGE( $i$ )

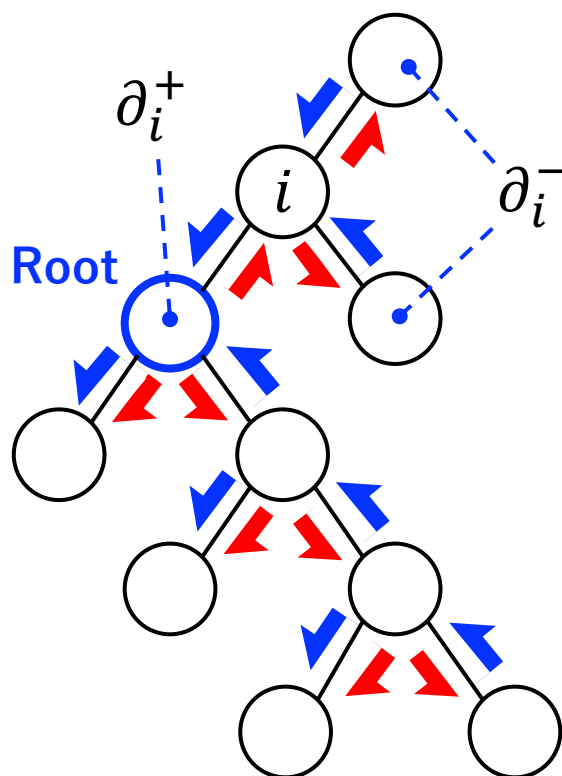
- 上流からノード $i$ に入るメッセージを”集めて”送る
- 出力は $i$ から下流ノードへのメッセージのセット



# 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.

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



- ◆  $\delta_{i \rightarrow \partial_i^+} = \text{COLLECT\_MESSAGE\_SS}(i, \varphi_i, \{\delta_{\gamma \rightarrow i} | \gamma \in \partial_i^-\})$   
 $\psi_{i \rightarrow \partial_i^+}(\mathcal{C}_i) \leftarrow \varphi_i \prod_{\gamma \in \partial_i^-} \delta_{\gamma \rightarrow i}$   
 $\delta_{i \rightarrow \partial_i^+}(\text{Sep}_{i, \partial_i^+}) \leftarrow \sum_{\mathcal{C}_i \setminus \text{Sep}_{i, \partial_i^+}} \psi_{i \rightarrow \partial_i^+}(\mathcal{C}_i)$
- ◆  $\{\delta_{i \rightarrow j} | j \in \partial_i^-\} = \text{DISTRIBUTE\_MESSAGE\_SS}(i, \varphi_i, \delta_{\partial_i^+ \rightarrow i})$   
 $\psi_{i \rightarrow j}(\mathcal{C}_i) \leftarrow \varphi_i \prod_{\gamma \in \partial_i \setminus j} \delta_{\gamma \rightarrow i}$   
 $\delta_{i \rightarrow j}(\text{Sep}_{i, j}) \leftarrow \sum_{\mathcal{C}_i \setminus \text{Sep}_{i, j}} \psi_{i \rightarrow j}(\mathcal{C}_i)$
- 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 •  $\Phi$ : 全てのファクターのセット  
•  $\mathcal{T}$ : Clique tree with  $k$  cliques  
•  $\alpha$ : ファクターのクリークへのassignment  $\triangleleft \alpha(\phi)$ : ファクター $\phi$ が対応するクリーク

Output •  $\beta_i(\mathbf{C}_i), i = 1, \dots, k$ : Belief  $\triangleleft$  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  $\mathbf{C}_i$ , set  $\varphi_i(\mathbf{C}_i) \leftarrow \prod_{\phi_j: \alpha(\phi_j)=i} \phi_j$

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

$\triangleleft$  Rootに向かってメッセージを更新

COLLECT\_MESSAGE\_SS( $i$ )

end For

For  $i = k, \dots, 2$

$\triangleleft$  Rootから逆方向にメッセージを更新

DISTRIBUTE\_MESSAGE\_SS( $i$ )

end For

For each clique  $\mathbf{C}_i$ , set  $\beta_i(\mathbf{C}_i) \leftarrow \varphi_i \prod_{k \in \partial i} \delta_{k \rightarrow i}$

$\triangleleft$  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  $\mathbf{c}_i$  と  $\mathbf{c}_j$  が  $\sum_{\mathbf{c}_i \setminus \text{Sep}_{ij}} \beta_i(\mathbf{c}_i) = \sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \beta_j(\mathbf{c}_j)$  を満たすとき  $\mathbf{c}_i$  と  $\mathbf{c}_j$  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

$$\bullet \beta_i(\mathbf{c}_i) = \varphi_i \prod_{k \in \partial i} \delta_{k \rightarrow i} = \delta_{j \rightarrow i} \varphi_i \prod_{k \in \partial i \setminus j} \delta_{k \rightarrow i}$$

$$\rightarrow \sum_{\mathbf{c}_i \setminus \text{Sep}_{ij}} \beta_i(\mathbf{c}_i) = \delta_{j \rightarrow i} \sum_{\mathbf{c}_i \setminus \text{Sep}_{ij}} \varphi_i \prod_{k \in \partial i \setminus j} \delta_{k \rightarrow i} = \delta_{j \rightarrow i} \delta_{i \rightarrow j}$$

$$\bullet \beta_j(\mathbf{c}_j) = \varphi_j \prod_{k \in \partial j} \delta_{k \rightarrow j} = \delta_{i \rightarrow j} \varphi_j \prod_{k \in \partial j \setminus i} \delta_{k \rightarrow j}$$

$$\rightarrow \sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \beta_j(\mathbf{c}_j) = \delta_{i \rightarrow j} \sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \varphi_j \prod_{k \in \partial j \setminus i} \delta_{k \rightarrow j} = \delta_{i \rightarrow j} \delta_{j \rightarrow i}$$

# 定義：sepset belief

Calibrated クリークツリーにおける  $\beta_i(\mathbf{c}_i)$  を clique belief とよび,  
また **sepset belief**  $\mu_{ij}(\text{Sep}_{ij})$  を次のように定義する.

$$\mu_{ij}(\text{Sep}_{ij}) = \sum_{\mathbf{c}_i \setminus \text{Sep}_{ij}} \beta_i(\mathbf{c}_i) = \sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \beta_j(\mathbf{c}_j)$$

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

# Reparametrization

Clique belief  $\beta_i(\mathbf{c}_i)$  と Sepset belief  $\mu_{ij}(\text{Sep}_{ij})$  により

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

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

- $\mathcal{V}_{\mathcal{T}}$  : 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 ◆

- $\beta_i = \varphi_i \prod_{j \in \partial_i} \delta_{j \rightarrow i}$  より,  $\prod_{i \in \mathcal{V}_{\mathcal{T}}} \beta_i = (\prod_{i \in \mathcal{V}_{\mathcal{T}}} \varphi_i) (\prod_{i \in \mathcal{V}_{\mathcal{T}}} \prod_{j \in \partial_i} \delta_{j \rightarrow i})$  ※  $\varphi_i = \prod_{\phi: \alpha(\phi)=i} \phi$
- $\mu_{ij} = \delta_{i \rightarrow j} \delta_{j \rightarrow i}$  より,  $\prod_{(ij) \in \mathcal{E}_{\mathcal{T}}} \mu_{ij} = \prod_{(ij) \in \mathcal{E}_{\mathcal{T}}} \delta_{i \rightarrow j} \delta_{j \rightarrow i}$

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 $\Phi$ のもとでのクリークツリーとする
- $\beta_i$ をCalibrated Belief,  $\mu_{ij}$ をSepset beliefとする
- $Q_{\mathcal{T}}(\mathbf{X}) = \frac{\prod_{i \in \mathcal{V}_{\mathcal{T}}} \beta_i(\mathbf{C}_i)}{\prod_{(ij) \in \mathcal{E}_{\mathcal{T}}} \mu_{ij}(\text{Sep}_{ij})}$ とする

このとき次の関係が成立する

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

## ▶ 証明の概要

- Running intersection propertyが成立するとき,  $\text{Sep}_{i, \partial_i^+}$ 所与のもとで  $\mathbf{C}_i$ と $\mathbf{C}_i$ 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．

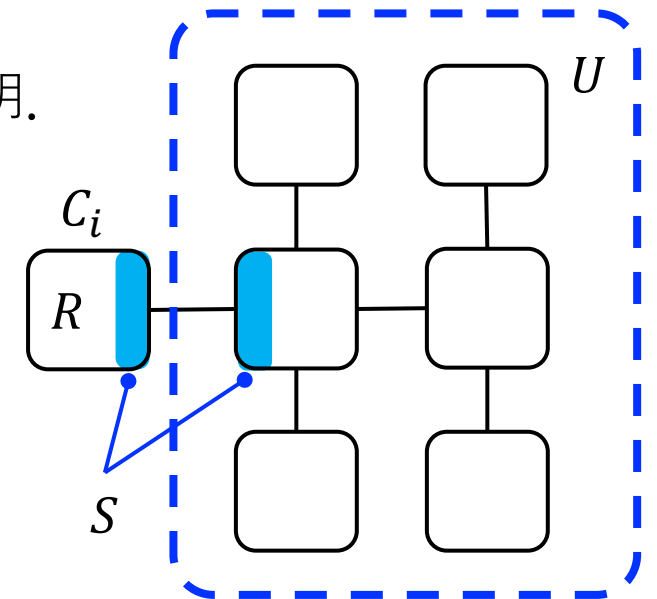
$C_i$ と，ある隣接クリーク $C_j$ 間のsepsetを $\text{Sep}_{ij}$ とする．

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

▶ 証明の概要： Running intersection propertyから，  
 $R_i$ と $U_i$ が $\text{Sep}_{ij}$ 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

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

$$\psi(\mathbf{X}, \mathbf{Y}) = \frac{\phi_1(\mathbf{X}, \mathbf{Y})}{\phi_2(\mathbf{Y})} \quad \text{※ } 0/0 \text{ の場合は } 0 \text{ とする.}$$

- 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

$$\delta_{i \rightarrow j} = \frac{\sum_{\mathbf{C}_i \setminus \text{Sep}_{ij}} \beta_i}{\delta_{j \rightarrow i}}$$

- 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

- $\Phi$ : 全てのファクターのセット
- $\mathcal{T}$ : Clique tree with  $k$  cliques
- $\alpha$ : ファクターのクリークへのassignment

◁  $\alpha(\phi)$ : ファクター  $\phi$  が対応するクリーク

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

Initialize

- For each clique  $\mathbf{C}_i$ , 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  $(i, j) \in \mathcal{E}_{\mathcal{T}}$

$$\sigma_{i \rightarrow j} \leftarrow \sum_{\mathbf{C}_i \setminus \text{Sep}_{ij}} \beta_i$$

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

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

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  $\beta_i$  について,  $(i, \partial_i)$  が選ばれても  $\beta_i$  に変化はない.

# Clique Tree Invariant

- Lauritzen-Spiegelhalter algorithmで,  
 $t$ 回更新したあとのBeliefを $\beta_i^{(\text{LS},t)}, \mu_{ij}^{(\text{LS},t)}$ 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,

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

$t' > t$ においても同様の関係が成立する.

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

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 $(k, \ell)$ を更新してみる

$$\sigma_{k \rightarrow \ell}^{(t+1)} = \sum_{\mathbf{c}_{k \setminus \text{Sep}_{k\ell}}} \beta_k^{(t)}$$

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

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

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

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

• 代入すると

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

よってステップ $t$ で  $\frac{\prod_{i \in \mathcal{V}_{\mathcal{T}}} \beta_i^{(\text{LS},t)}}{\prod_{(ij) \in \mathcal{E}_{\mathcal{T}}} \mu_{ij}^{(\text{LS},t)}} = \tilde{P}_{\Phi}(\mathbf{X})$  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  $\beta_i^{(\text{LS},0)} \leftarrow \prod_{\phi_j: \alpha(\phi_j)=i} \phi_j$ , また  $\mu_{ij}^{(\text{LS},0)} \leftarrow 1$  とすると,

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

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 $\varphi_i$  ( $i \in \mathcal{V}_{\mathcal{T}}$ ),  $\delta_{i \rightarrow j}^{(SS,0)}$ ,  $\delta_{j \rightarrow i}^{(SS,0)}$  ( $(i,j) \in \mathcal{E}_{\mathcal{T}}$ )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

$$\beta_i^{(LS,0)} = \varphi_i \prod_{k \in \partial_i} \delta_{k \rightarrow i}^{(SS,0)}, \quad \mu_{ij}^{(LS,0)} = \delta_{i \rightarrow j}^{(SS,0)} \delta_{j \rightarrow i}^{(SS,0)} \quad \text{※ Calibrateな表式}$$

- 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

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



# 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)

$$\text{SP: } \delta_{i \rightarrow j}^{(\text{SS},1)} = \sum_{\mathbf{c}_i \setminus \text{Sep}_{ij}} \varphi_i \prod_{\gamma \in \partial_i \setminus j} \delta_{\gamma \rightarrow i}^{(\text{SS},0)} \quad \dots (1)$$

$$\text{LS: } \beta_j^{(\text{LS},0)} = \varphi_j \prod_{k \in \partial_j} \delta_{k \rightarrow j}^{(\text{SP},0)} \quad \dots (2)$$

$$\mu_{ij}^{(\text{LS},0)} = \delta_{i \rightarrow j}^{(\text{SP},0)} \delta_{j \rightarrow i}^{(\text{SP},0)} \quad \dots (3)$$

$$\sigma_{i \rightarrow j}^{(\text{LS},1)} = \sum_{\mathbf{c}_i \setminus \text{Sep}_{ij}} \beta_i^{(\text{LS},0)} \quad \dots (4)$$

$$\beta_j^{(\text{LS},1)} = \beta_j^{(\text{LS},0)} \cdot \frac{\sigma_{i \rightarrow j}^{(\text{LS},1)}}{\mu_{ij}^{(\text{LS},0)}} \quad \dots (5)$$

$$\mu_{ij}^{(\text{LS},1)} = \sigma_{i \rightarrow j}^{(\text{LS},1)} \quad \dots (6)$$

- (4)に(2)を代入すると

$$\begin{aligned} \sigma_{i \rightarrow j}^{(\text{LS},1)} &= \sum_{\mathbf{c}_i \setminus \text{Sep}_{ij}} \varphi_i \prod_{k \in \partial_i} \delta_{k \rightarrow i}^{(\text{SS},0)} \\ &= \delta_{j \rightarrow i}^{(\text{SS},0)} \sum_{\mathbf{c}_i \setminus \text{Sep}_{ij}} \varphi_i \prod_{k \in \partial_i \setminus j} \delta_{k \rightarrow i}^{(\text{SS},0)} \\ &= \delta_{j \rightarrow i}^{(\text{SS},0)} \delta_{i \rightarrow j}^{(\text{SS},1)} \quad \dots (4') \end{aligned}$$

- (5)に(2), (3), (4')を代入すると

$$\begin{aligned} \beta_j^{(\text{LS},1)} &= \varphi_j \prod_{k \in \partial_j} \delta_{k \rightarrow j}^{(\text{SS},0)} \cdot \frac{\delta_{j \rightarrow i}^{(\text{SS},0)} \delta_{i \rightarrow j}^{(\text{SS},1)}}{\delta_{i \rightarrow j}^{(\text{SS},0)} \delta_{j \rightarrow i}^{(\text{SS},0)}} \\ &= \varphi_j \delta_{i \rightarrow j}^{(\text{SP},1)} \prod_{k \in \partial_j \setminus i} \delta_{k \rightarrow j}^{(\text{SP},0)} \end{aligned}$$

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

$$\begin{aligned} \beta_j^{(\text{LS},1)} &= \varphi_j \prod_{k \in \partial_j} \delta_{k \rightarrow j}^{(\text{SS},1)} \\ \mu_{ij}^{(\text{LS},1)} &= \delta_{j \rightarrow i}^{(\text{SS},1)} \delta_{i \rightarrow j}^{(\text{SS},1)} \end{aligned}$$

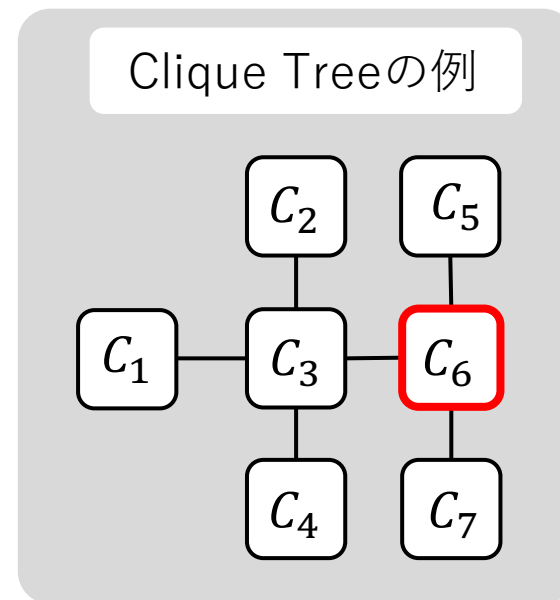
# Incremental Update

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

(例) 右図で $\text{Scope}[\phi'] \subseteq \mathbf{C}_6$ 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： $\mathbf{C}_6$ より上流が変化
- Downward pathについて： $\mathbf{C}_6$ 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 $\Phi$ においてすでに得られているmessageを $\delta$ とする.

## 1. Upward pathの修正

- $i$ から親クリークについて

$$\psi'_{i \rightarrow \partial_i^+}(\mathbf{c}_i) \leftarrow \varphi'_i \prod_{\gamma \in \partial_i \setminus \partial_i^+} \delta_{\gamma \rightarrow i}$$

$i$ に入ってくるmessageに変更はない.

$$\delta'_{i \rightarrow \partial_i^+}(\text{Sep}_{i, \partial_i^+}) \leftarrow \sum_{\mathbf{c}_i \setminus \text{Sep}_{i, \partial_i^+}} \psi'_{i \rightarrow \partial_i^+}(\mathbf{c}_i)$$

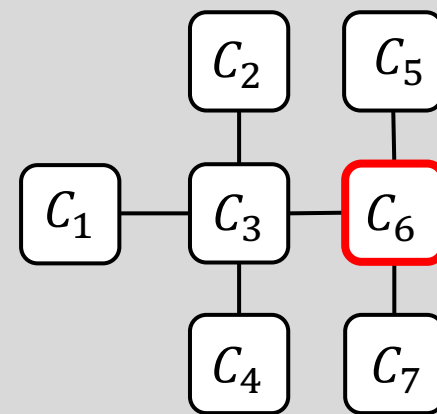
- $j = \partial_i^+$ から上流について

$$\psi'_{j \rightarrow \partial_j^+}(\mathbf{c}_i) \leftarrow \varphi_j \delta'_{i \rightarrow j} \prod_{\gamma \in \partial_j \setminus \partial_j^+, i} \delta_{\gamma \rightarrow j}$$

$i$ 以外から入ってくるmessageに変更はない.

$$\delta'_{j \rightarrow \partial_j^+}(\text{Sep}_{j, \partial_j^+}) \leftarrow \sum_{\mathbf{c}_j \setminus \text{Sep}_{j, \partial_j^+}} \psi'_{j \rightarrow \partial_j^+}(\mathbf{c}_j)$$

Clique Treeの例



# 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 $\Phi$ においてすでに得られているmessageを $\delta$ とする.

## 2. Downward pathの修正

- $i$ から近接下流クリークについて

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

$i$ に入ってくるmessageに変更はない.

$$\delta'_{i \rightarrow \partial_i^-}(\text{Sep}_{i, \partial_i^-}) \leftarrow \sum_{\mathbf{C}_i \setminus \text{Sep}_{i, \partial_i^-}} \psi'_{i \rightarrow \partial_i^-}(\mathbf{C}_i)$$

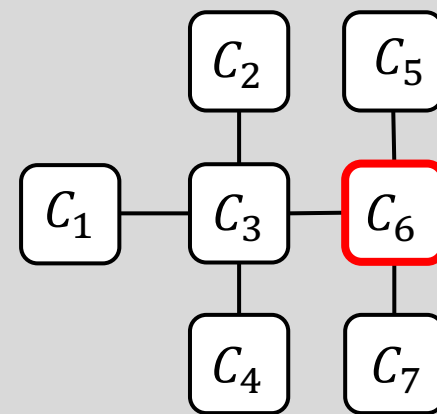
- $j \in \partial_i^-$ から下流について

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

$i$ 以外から入ってくるmessageに変更はない.

$$\delta'_{j \rightarrow \partial_j^-}(\text{Sep}_{j, \partial_j^-}) \leftarrow \sum_{\mathbf{C}_j \setminus \text{Sep}_{j, \partial_j^-}} \psi'_{j \rightarrow \partial_j^-}(\mathbf{C}_j)$$

Clique Treeの例



# Incremental Update: Evidenceの印加

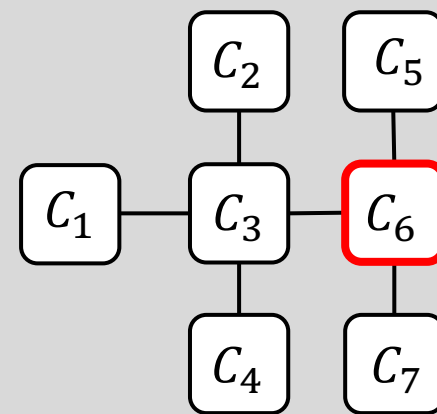
- ファクター集合 $\Phi$ のもとでのCalibrated clique treeに新しいファクター $\phi'$ を加えることを考える.
  - ここでは $\text{Scope}[\phi'] \subseteq \mathbf{C}_i$ とする
- 新しいファクター集合 $\Phi' = \Phi \cup \phi'$ のもとで $\tilde{P}_{\Phi'}(\mathbf{X})$ を得たい.
- Evidenceを与えることも、新しいファクターの追加と見なすことができる

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

$\phi'$ 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

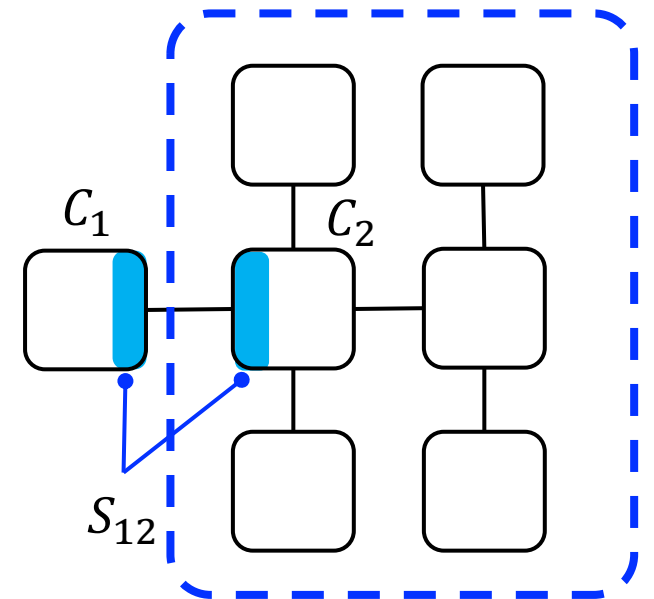
- Running intersection propertyのもとでの条件付き独立性より

$$\begin{aligned} p(C_1 \setminus \text{Sep}_{12}, \text{Sep}_{12}, \mathcal{V} \setminus C_1) &= p(C_1 \setminus \text{Sep}_{12} | \text{Sep}_{12}, \mathcal{V} \setminus C_1) p(\text{Sep}_{12} | \mathcal{V} \setminus C_1) p(\mathcal{V} \setminus C_1) \\ &\rightarrow p(C_1 \setminus \text{Sep}_{12} | \text{Sep}_{12}) p(\text{Sep}_{12} | \mathcal{V} \setminus C_1) p(\mathcal{V} \setminus C_1) \\ &= \frac{p(C_1 \setminus \text{Sep}_{12}, \text{Sep}_{12}) p(\text{Sep}_{12}, \mathcal{V} \setminus C_1)}{p(\text{Sep}_{12})} = \frac{p(C_1) p((\mathcal{V} \setminus C_1) \cup \text{Sep}_{12})}{p(\text{Sep}_{12})} \end{aligned}$$

$$\begin{aligned} p((\mathcal{V} \setminus C_1) \cup \text{Sep}_{12}) &= p(C_2 \setminus \text{Sep}_{23}, \text{Sep}_{23}, \mathcal{V} \setminus (C_1 \cup C_2)) \\ &\rightarrow \frac{p(C_2) p((\mathcal{V} \setminus (C_1 \cup C_2)) \cup \text{Sep}_{23})}{p(\text{Sep}_{23})} \end{aligned}$$

- これ続けると、次の表現を得る.

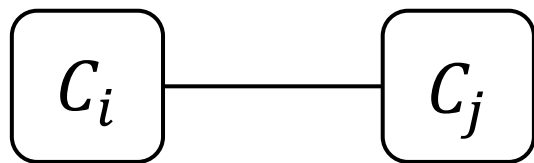
$$p(\mathbf{X}) = \frac{\prod_i p(\mathbf{C}_i)}{\prod_{(i,j) \in \mathcal{E}_{\mathcal{T}}} p(\text{Sep}_{ij})}$$



# Hugin architecture

- Separator node と Separator potential  $\phi$  を導入

Clique tree



Hugin architecture



- $Sep_{ij} = C_i \cap C_j$

- $p(\mathbf{X}) = \frac{\prod_{i \in \mathcal{V}_{\mathcal{T}}} \psi_i(\mathbf{x}_{C_i})}{\prod_{(i,j) \in \mathcal{E}_{\mathcal{T}}} \phi_{ij}(\mathbf{x}_{Sep_{ij}})}$  という形を満たす  $\psi_i(\mathbf{x}_{C_i})$  と  $\phi_{ij}(\mathbf{x}_{Sep_{ij}})$  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

- $\{\phi_{ij}^*, \psi_j^*\} \leftarrow \text{COLLECT\_MESSAGE\_HUGIN}(i, \psi_i, \phi_{ij}, \psi_j)$

$$\phi_{ij}^*(\text{Sep}_{ij}) \leftarrow \sum_{\mathbf{C}_i \setminus \text{Sep}_{ij}} \psi_i(\mathbf{C}_i)$$

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

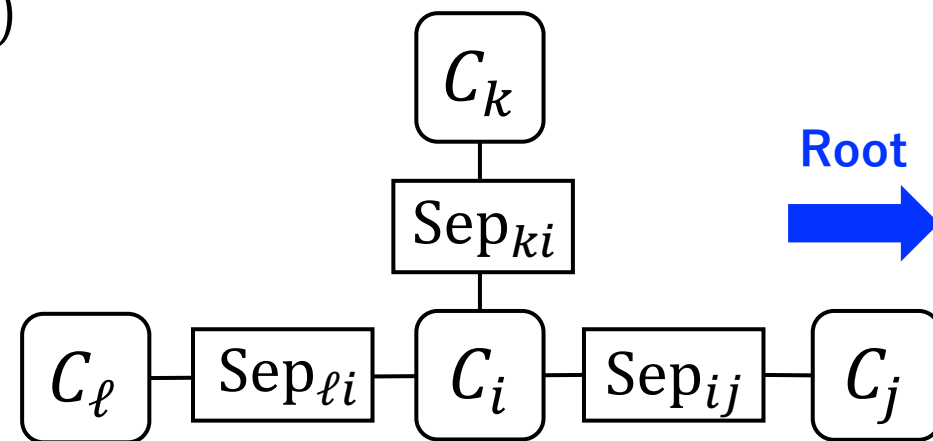
$$\psi_j(\mathbf{C}_j) \leftarrow \psi_j^*(\mathbf{C}_j)$$

- $\{\phi_{ik}^{**}, \psi_k^{**} | k \in \partial_i^-\} \leftarrow \text{DISTRIBUTE\_MESSAGE\_HUGIN}(i, \psi_i^*, \{\phi_{ki}^*, \psi_k^* | k \in \partial_i^-\})$

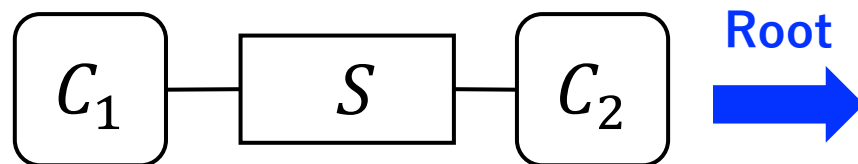
$$\phi_{ki}^{**}(\text{Sep}_{ki}) \leftarrow \sum_{\mathbf{C}_k \setminus \text{Sep}_{ki}} \psi_k^*(\mathbf{C}_k)$$

$$\psi_k^{**}(\mathbf{C}_k) \leftarrow \frac{\phi_{ki}^{**}(\text{Sep}_{ki})}{\phi_{ki}^*(\text{Sep}_{ki})} \psi_k^*(\mathbf{C}_k)$$

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



# 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

$$\phi_S^{(t+1)}(S) \leftarrow \sum_{C_i \setminus S} \psi_1^{(t)}(C_1)$$

$$\psi_2^{(t+1)}(C_2) \leftarrow \frac{\phi_S^{(t+1)}(S)}{\phi_S^{(t)}(S)} \psi_2^{(t)}(C_2)$$



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

※  $\psi_1^{(t+1)} = \psi_1^{(t)}$  とした.

- あるステップ $t + 2$ でのDISTRIBUTE\_MESSAGE\_HUGIN

$$\phi_S^{(t+2)}(S) \leftarrow \sum_{C_2 \setminus S} \psi_2^{(t+1)}(C_2)$$

$$\psi_1^{(t+2)}(C_1) \leftarrow \frac{\phi_S^{(t+2)}(S)}{\phi_S^{(t+1)}(S)} \psi_1^{(t+1)}(C_1)$$



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

※  $\psi_2^{(t+2)} = \psi_2^{(t+1)}$  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  $\psi_i^{**}(\mathbf{c}_i) = \sum_{\mathbf{X} \setminus \mathbf{c}_i} p(\mathbf{X})$ ,  $\phi_{ij}^{**}(\text{Sep}_{ij}) = \sum_{\mathbf{X} \setminus \text{Sep}_{ij}} p(\mathbf{X})$  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 ◆

- $\mathbf{C}_i$ をleaf cliqueとし, 近接クリークを $\mathbf{C}_j$ とする.
- クリークツリーにおける条件付き独立性より $p(\mathbf{X}) = p(\mathbf{C}_i, \text{Sep}_{ij}, \mathbf{X}_{\setminus \mathbf{C}_i}) = p(\mathbf{C}_i | \text{Sep}_{ij}) p(\text{Sep}_{ij}, \mathbf{X}_{\setminus \mathbf{C}_i})$
- $\mathbf{C}_i$ のないクリークツリーで $p(\text{Sep}_{ij}, \mathbf{X}_{\setminus \mathbf{C}_i}) = \frac{\prod_{k \neq i} \psi_k^{**}(\mathbf{C}_k)}{\prod_{(k, \ell) \in \mathcal{E}_{T \setminus \mathbf{C}_i}} \phi_{S_{k\ell}}^{**}(\text{Sep}_{k\ell})}$ , また $\psi_k^{**}(\mathbf{C}_k) = p(\mathbf{C}_k)$ ,  $\phi_{k\ell}^{**}(\text{Sep}_{k\ell}) = p(\text{Sep}_{k\ell})$ が成立しているとする.
- $\mathbf{C}_i$ を含むクリークツリーでのHuginアルゴリズムにより $p(\mathbf{C}_i | \text{Sep}_{ij}) = \frac{\psi_i^{**}(\mathbf{C}_i)}{\phi_{ij}^{**}(\text{Sep}_{ij})}$ を得る.  
ここで $\sum_{\mathbf{C}_j \setminus \text{Sep}_{ij}} \psi_j^{**}(\mathbf{C}_j) = \phi_{ij}^{**}(\text{Sep}_{ij}) = p(\text{Sep}_{ij})$ なので $\psi_i^{**}(\mathbf{C}_i) = p(\mathbf{C}_i)$

# 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

$$\delta_{j \rightarrow i} \leftrightarrow \frac{\phi_{ij}^{**}(\text{Sep}_{ij})}{\phi_{ij}^*(\text{Sep}_{ij})}, \quad \forall i \in \partial_j^-$$

$$\delta_{i \rightarrow j} \leftrightarrow \frac{\phi_{ij}^*(\text{Sep}_{ij})}{\phi_{ij}(\text{Sep}_{ij})}, \quad \forall j \in \partial_i^+$$

## ● Shafer-Shenoy アルゴリズム

### ◆ COLLECT\_MESSAGE\_SS

$$\psi_{i \rightarrow \partial_i^+}(\mathbf{C}_i) \leftarrow \varphi_i \prod_{\gamma \in \partial_i^-} \delta_{\gamma \rightarrow i} \quad \triangleleft \text{Computational cost} \uparrow$$

$$\delta_{i \rightarrow \partial_i^+}(\text{Sep}_{i, \partial_i^+}) \leftarrow \sum_{\mathbf{C}_i \setminus \text{Sep}_{i, \partial_i^+}} \psi_{i \rightarrow \partial_i^+}(\mathbf{C}_i)$$

### ◆ DISTRIBUTE\_MESSAGE\_SS

$$\psi_{i \rightarrow j}(\mathbf{C}_i) \leftarrow \varphi_i \prod_{\gamma \in \partial_i \setminus j} \delta_{\gamma \rightarrow i} \quad \triangleleft \text{Computational cost} \uparrow$$

$$\delta_{i \rightarrow j}(\text{Sep}_{i,j}) \leftarrow \sum_{\mathbf{C}_i \setminus \text{Sep}_{i,j}} \psi_{i \rightarrow j}(\mathbf{C}_i)$$

### ◆ 周辺化分布の評価 △ Computational cost ↑

$$\beta_i(\mathbf{C}_i) \leftarrow \varphi_i \prod_{k \in \partial_i} \delta_{k \rightarrow i}$$

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

## ● Hugin アルゴリズム

### ◆ COLLECT\_MESSAGE\_HUGIN

$$\phi_{ij}^*(\text{Sep}_{ij}) \leftarrow \sum_{\mathbf{C}_i \setminus \text{Sep}_{ij}} \psi_i(\mathbf{C}_i)$$

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

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

### ◆ DISTRIBUTE\_MESSAGE\_HUGIN

$$\phi_{ki}^{**}(\text{Sep}_{ki}) \leftarrow \sum_{\mathbf{C}_k \setminus \text{Sep}_{ki}} \psi_k^*(\mathbf{C}_k)$$

$$\psi_k^{**}(\mathbf{C}_k) \leftarrow \frac{\phi_{ki}^{**}(\text{Sep}_{ki})}{\phi_{ki}^*(\text{Sep}_{ki})} \psi_k^*(\mathbf{C}_k) \quad \triangleleft \text{Memory cost} \uparrow$$

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

- どちらも  $O(\exp(\text{width}))$  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

# $\tilde{P}_\Phi(\mathbf{X}) \propto Q_{\mathcal{T}}(\mathbf{X}) \Leftarrow \beta_i(\mathbf{C}_i) \propto \tilde{P}_\Phi(\mathbf{C}_i)$ 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

$$\tilde{P}_\Phi(\mathbf{X}) = \tilde{P}_\Phi(\mathbf{C}_r) \prod_{i \neq r} \tilde{P}_\Phi(\mathbf{C}_i | C_{i+1}, \dots, C_r) \rightarrow \tilde{P}_\Phi(\mathbf{C}_r) \prod_{i \neq r} \tilde{P}_\Phi(\mathbf{C}_i | \text{Sep}_{i, \partial_i^+})$$

◆  $Q_{\mathcal{T}}(\mathbf{X})$  を変形すると

$$Q_{\mathcal{T}}(\mathbf{X}) = \beta_r(C_r) \prod_{i \neq r} \frac{\beta_i(\mathbf{C}_i)}{\mu_{i \partial_i^+}(\text{Sep}_{i, \partial_i^+})} = \beta_r(C_r) \prod_{i \neq r} \frac{\beta_i(\mathbf{C}_i)}{\sum_{C_i \setminus \text{Sep}_{i, \partial_i^+}} \beta_i(C_i)} = \beta_r(C_r) \prod_{i \neq r} \beta_i(\mathbf{C}_i | \text{Sep}_{i, \partial_i^+})$$

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

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

- 適当なルートクリーク  $\mathbf{C}_r$  について考える.

◆ Chain-rule と Running Intersection Property から

$$P_\Phi(\mathbf{X}) = \frac{1}{Z} \tilde{P}_\Phi(\mathbf{C}_r) \prod_{i \neq r} \tilde{P}_\Phi(\mathbf{C}_i | \mathbf{C}_{i+1}, \dots, \mathbf{C}_r) \rightarrow \frac{1}{Z} \tilde{P}_\Phi(\mathbf{C}_r) \prod_{i \neq r} \tilde{P}_\Phi(\mathbf{C}_i | \text{Sep}_{i, \partial_i^+})$$

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

- ◆  $Q_{\mathcal{T}}$  について  $p_\beta(\mathbf{C}_i \setminus \text{Sep}_{ij} | \text{Sep}_{ij}) = \frac{\beta_i(\mathbf{C}_i)}{\sum_{\mathbf{C}_i \setminus \text{Sep}_{ij}} \beta_i(\mathbf{C}_i)}$  と定義するとこれは確率分布である.

$$Q_{\mathcal{T}}(\mathbf{C}_r) = \beta_r(\mathbf{C}_r) \sum_{\mathbf{X} \setminus \mathbf{C}_r} \prod_{i \neq r} p_\beta(\mathbf{C}_i \setminus \text{Sep}_{ij} | \text{Sep}_{ij}) = \beta_r(\mathbf{C}_r)$$

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

$$\phi_{ki}^*(\text{Sep}_{ki}) = \sum_{\mathbf{c}_k \setminus \mathbf{s}_{ki}} \psi_k(\mathbf{c}_k), \quad \forall k \in \partial_i^-$$

$$\psi_i^*(\mathbf{c}_i) = \psi_i(\mathbf{c}_j) \prod_{k \in \partial_i^-} \frac{\phi_{ki}^*(\text{Sep}_{ki})}{\phi_{ki}(\text{Sep}_{ki})}$$

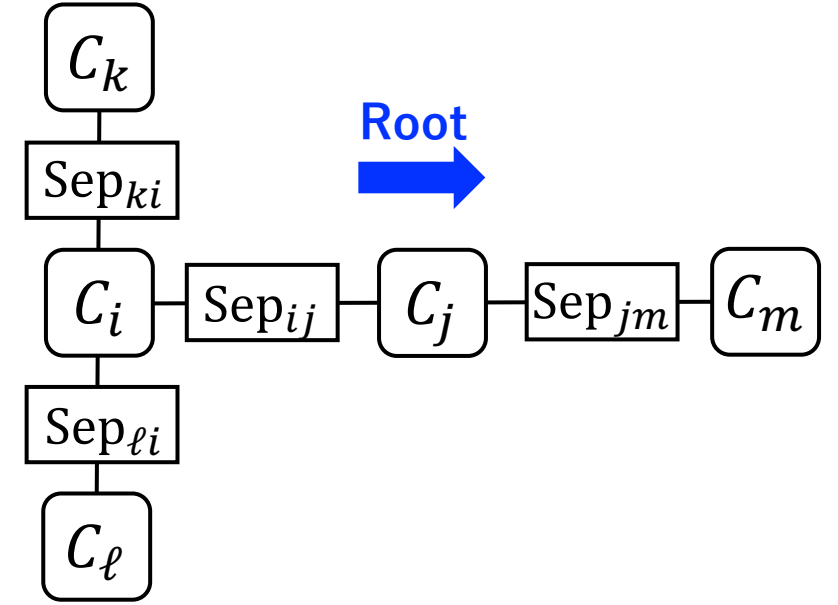
- HUGINでの下流へのメッセージの伝搬

$$\phi_{jm}^{**}(\text{Sep}_{jm}) = \sum_{\mathbf{c}_m \setminus \mathbf{s}_{jm}} \psi_m^*(\mathbf{c}_m) \quad \forall m \in \partial_j^+,$$

$$\psi_j^{**}(\mathbf{c}_i) = \psi_j^*(\mathbf{c}_j) \prod_{m \in \partial_j^+} \frac{\phi_{jm}^{**}(\text{Sep}_{jm})}{\phi_{jm}^*(\text{Sep}_{jm})}$$

$$= \psi_j(\mathbf{c}_j) \frac{\phi_{ij}^*(\text{Sep}_{kj})}{\phi_{ij}(\text{Sep}_{kj})} \prod_{k \in \partial_j^- \setminus i} \frac{\phi_{kj}^*(\text{Sep}_{kj})}{\phi_{kj}(\text{Sep}_{kj})} \prod_{m \in \partial_j^+} \frac{\phi_{jm}^{**}(\text{Sep}_{jm})}{\phi_{jm}^*(\text{Sep}_{jm})}$$

$$\text{よって} \quad \sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \psi_j^{**}(\mathbf{c}_j) = \phi_{ij}^*(\text{Sep}_{kj}) \sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \psi_j(\mathbf{c}_j) \prod_{k \in \partial_j^- \setminus i} \frac{\phi_{kj}^*(\text{Sep}_{kj})}{\phi_{kj}(\text{Sep}_{kj})} \prod_{m \in \partial_j^+} \frac{\phi_{jm}^{**}(\text{Sep}_{jm})}{\phi_{jm}^*(\text{Sep}_{jm})}$$



$$\sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \psi_j^{**}(\mathbf{c}_j) = \phi_{ij}^{**}(\text{Sep}_{ij}) \text{なので}$$

$$\phi_{ij}^{**}(\text{Sep}_{ij}) = \phi_{ij}^*(\text{Sep}_{ij}) \sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \psi_j(\mathbf{c}_j) \prod_{k \in \partial_j^- \setminus i} \frac{\phi_{kj}^*(\text{Sep}_{kj})}{\phi_{kj}(\text{Sep}_{kj})} \prod_{m \in \partial_j^+} \frac{\phi_{jm}^{**}(\text{Sep}_{jm})}{\phi_{jm}^*(\text{Sep}_{jm})} \dots \square$$

● Shafer-Shenoy より

$$\mu_{j \rightarrow i}(\text{Sep}_{ij}) = \sum_{\mathbf{c}_j \setminus \text{Sep}_{ij}} \psi_j(\mathbf{c}_j) \prod_{k \neq i} \mu_{k \rightarrow j} \dots \blacksquare$$

$\square$ と $\blacksquare$ の比較から,

$$\mu_{j \rightarrow i} \leftrightarrow \frac{\phi_{ij}^{**}(\text{Sep}_{ij})}{\phi_{ij}^*(\text{Sep}_{ij})}, \quad \forall i \in \partial_j^-$$

$$\mu_{i \rightarrow j} \leftrightarrow \frac{\phi_{ij}^*(\text{Sep}_{ij})}{\phi_{ij}(\text{Sep}_{ij})}, \quad \forall j \in \partial_i^+$$

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