ISSN 0918-2802

Technical Report

記号的統計知識の表現と学習に関する研究

亀谷由隆

TR00-0015 November

Department of Computer Science
Tokyo Institute of Technology
Ôokayama 2-12-1 Meguro Tokyo 152-8552, Japan
http://www.cs.titech.ac.jp/

目 次

図 目 次

第1章　序論

現在までの情報技術の発展により，コンピュータシステムは身の回りのあらゆる場面で見かけるようになった．このように現実世界とコンピュータの接点が増すにつれ，データに含まれるノイズや欠損といった不確実性，あるいは複雑な対象領域に対し完璧にプログラミングできない場合の不完全性に対する対処法が情報処理における大きな課題となってきている．そのためには，これらの不確実性，不完全性を何らかの形で要約し，形式化する必要があるが，要約方法の一つとして確率概念が用いられてきた．確率概念の体系である確率論は不確実性の解釈・表現に関して，数多くの研究者によって議論され，また多くの人間の間で同意が得られている系統的な理論である．確率論に基づいて，自然界もしくは人間社会における確率現象の発生メカニズムの構造を統計モデル (statistical model) として記述し，モデルの確率パラメータを観測データから学習し，学習後のモデルをデータの分析，予測などに利用するという統計的推論が広く行なわれている．

伝統的な統計モデル，例えば正規分布，ポアソン分布などは数式言語によって確率密度関数の形で記述されているが，オートマトン，文法，グラフなどを表現言語とした統計モデリングが近年盛んに応用されている．例えば，**隠れマルコフモデル**（hidden Markov model; 以下 HMM）と**確率文脈自由文法**（probabilistic context-free grammar; 以下 PCFG）は，それぞれ Chomsky 階層における 3 型文法（正規文法）と 2 型文法（文脈自由文法）の規則選択に確率を導入したものである．HMM は音声認識 [49]，統計的自然言語処理 [8, 25, 33]，遺伝子情報処理 [2] などに応用され，その有効性が確認されている．また，PCFG やその拡張モデルは統計的自然言語処理 [8, 9, 24, 25, 33]，遺伝子情報処理 [2]，プラニング [48] などに用いられている．加えて，確率変数間の因果関係，独立性を有向非循環グラフ（directed acyclic graph; 以下 DAG）で表現した **Bayesian ネットワーク** [7, 43, 51, 36] も多くの応用分野に用いられている．

一方，我々は対象領域に対して数値では表せない記号的な知識をもつ場合がある．例えば，遺伝現象は抽象化された遺伝子に関する記号的規則によって理解される．また，選挙における有権者全体の投票行動は確率現象として捉えることができるが，例えば「有権者 A は政党 B を支持している」というのは記号的な事実である．人工知能研究の重要分野の一つである知識表現 (knowledge representation) 研究において不確実性を伴う問題領域において知識をどのように表現するかを研究するのは自然な成り行きであり，初期エキスパートシステムの頃からいくつかの方法が提案されてきた．しかし，不確実性を表現するというのはそれ自体多くの考察が必要であり，そのような考察なしに不確実性に対する処理手続きをコンピュータシステムに導入してしまうとそのシステムが（エラーを返すこともなく）人間にとって不自然な振る舞いを行ってしまう．一方，先述したように確率論は不確実性の解釈・表現に関する系統的な理論であるので，確率論に基づき知識表現を行うというのは合理的な選択といえる．確率とともに明示的に表現された記号的知識を本論文では**記号的統計モデル** (symbolic-statistical model) と呼ぶことにする．先述した HMM, PCFG（人間の持っている文法知識に確率を与えている）や，Bayesian ネットワークも記号的統計モデルの一つのクラスであると考えることができる．

また，知識表現形式として命題論理，一階述語論理といった論理形式が数多くの研究者によって議論されてきた．そして，一階述語論理の表現形式と導出原理に基づく証明機構から構成される計

算言語である論理プログラム [29] は人工知能言語の中心的な役割を果たしてきた．記号的統計モデルの表現言語として論理プログラムに確率を導入した場合，再帰述語や論理変数により統計知識が簡潔で理解しやすいものになり，我々は表現された統計知識を直ちに，かつ正確に捉えることができる．また，論理プログラムは 0 型文法と同等の生成能力（表現力）をもち，原理的には任意の複雑な確率現象を表現可能である．このような確率的な論理プログラムはこれまで数多く提案されているが [5, 13, 14, 18, 26, 30, 37, 39, 40, 41, 46, 50, 53]，その中で 1995 年に Sato が提案した**分布意味論** (distributional semantics) [53] は最小モデル意味論に基づき，任意の論理プログラムに対して確率的意味を宣言的に与えるものである．

また，統計知識の表現においては「どうやって数値（確率値）を与えたらよいのか？ / where do the numbers come from?」という問題が常につきまとう．人間にとって記号的な（論理的な）関係（例えば,「夕焼けが見えるとき次の日は晴れやすい」という，ある日の夕焼けと次の日の天気の因果関係）を記述，検証するのはそれほど難しくないが，具体的な確率パラメータ値（例えば，条件つき確率 $Pr(\text{夕焼け}_d = \texttt{yes} \mid \text{天気}_{d+1} = \text{晴れ})$ など）を人間の頭の中だけで客観的に規定するのは困難である．しかも，数百から数千という確率パラメータから成るモデルに対してこのような作業を行うのは現実的に見て不可能であり，先述したように通常の統計的推論では観測データから最尤推定 (maximum likelihood estimation) によりこれらの確率パラメータ値を求めている．その一方で，論理プログラムに確率を導入した先行研究のほとんどはこのようなパラメータ学習の機構については触れていない．構造と確率パラメータ値が規定されてはじめて統計モデルを規定したことになるのであるから，パラメータ学習アルゴリズムをもたない統計モデル表現言語は，客観的な統計知識に基づく統計的推論システムの一部には成り得ない．それに対し，Sato は BS プログラムという確率パラメータが付与された論理プログラムのクラスに対して，そのパラメータを観測データから最尤推定する **EM** (expectation-maximization) **アルゴリズム**を提案している [53]．以下では，EM アルゴリズムによるパラメータ学習を **EM 学習** と呼ぶことにする．次いで，Sato らは BS プログラムに基づく表現言語を PRISM (PRogramming In Statistical Modeling) と名付け，EM 学習機構を備えたプログラミング処理系（以下，PRISM 処理系と呼ぶ）を提案した [20, 54, 55]．更に [54] では PRISM で HMM や Bayesian ネットワークを記述し，PRISM プログラムがこれらの記号的統計モデルクラスの表現上の一般化であることが示されている．

しかし，PRISM 処理系の EM アルゴリズムは一般的な PRISM プログラムに対して動作するように設計されていたため，例えば HMM を PRISM プログラムとして記述したときに，その EM 学習には出力・受理記号列 L の指数オーダの計算量が必要であった．それに対し，HMM に特化して導出された EM アルゴリズムである **Baum-Welch アルゴリズム** [8, 25, 33, 49] は L に対して線形オーダで動作する．Baum-Welch アルゴリズムでは，HMM で仮定されているマルコフ性，つまりモデル固有の確率的性質を利用した動的計画法に基づいて効率化が図られている．同様に，PCFG には **Inside-Outside アルゴリズム**と呼ばれる専用 EM アルゴリズムが提案されており [3, 8, 25, 28, 33]，**単結合** (singly-connected) と呼ばれる Bayesian ネットワークの部分クラスにも効率的な EM アルゴリズムが用意されている [7]．逆にいえば，これらの統計モデルが有用とされるのは，専用の確率推論アルゴリズム（EM アルゴリズムを含む）を備えているためである．

そこで本論文では，効率的な専用 EM アルゴリズムの設計に欠かせない動的計画法のメカニズムを PRISM プログラム上で一般化することを目標とする．具体的には，ある PRISM プログラムのサブクラスに適用可能な**グラフィカル EM アルゴリズム**（graphical EM algorithm; 以下 gEM アルゴリズムと略す）と呼ばれる EM アルゴリズムを提案し，その計算量が上に挙げた各専用 EM アルゴリズムと同等であることを示す．その結果，PRISM およびその処理系は次の特長を有する．

- 最小モデル意味論の確率的な一般化である分布意味論 [53] に基づき，確率的意味が宣言的に規定されている．
- 既存の記号的統計モデルクラスである HMM, PCFG およびその拡張文法，Bayesian ネットワークの表現上の一般化になっている．
- 観測データからプログラム中に付与された確率パラメータを EM 学習できる．
- ある条件を満たす PRISM プログラムには gEM アルゴリズムを適用できる．gEM アルゴリズムは Baum-Welch アルゴリズム，Inside-Outside アルゴリズム，単結合 Bayesian ネットワーク用 EM アルゴリズムの一般化となっている．

4 番目の特長が本論文の成果によるものであるが，これは PRISM で HMM, PCFG, Bayesian ネットワークを記述すれば，gEM アルゴリズムがそれぞれ Baum-Welch アルゴリズム，Inside-Outside アルゴリズム，単結合 Bayesian ネットワーク用 EM アルゴリズムと同じ計算オーダで EM 学習できることを意味する．宣言的な意味論を備え，かつ表現上の一般性と EM 学習の効率性を同時に実現する点が，既存の統計知識表現の枠組には見られない特徴である．また，Baum-Welch アルゴリズムと同様，HMM でよく用いられる Viterbi アルゴリズムも PRISM プログラム上で一般化できる．

また，4 番目の特長は，我々がもし新しい統計モデルを考えたときに，それを PRISM プログラムとして適切に記述すれば，専用の EM アルゴリズムを新たに導出しなくとも，PRISM 処理系組込みの学習機構が充分高速に EM 学習を行うという保証を与えるものである．あるいは,（PRISM 処理系を使わなくても）本論文で提案する効率的な PRISM 用 EM アルゴリズムを理解していれば，新しい統計モデル専用の EM アルゴリズムを構築することができる．すなわち，

- PRISM プログラムを統計モデルの仕様書とすることで，新しいモデル専用の EM アルゴリズムの開発が容易になる

という点が 5 つ目の特長として挙げられる．本論文ではこれまで提案されていなかった PCFG の拡張文法に対する効率的な EM 学習法を PRISM プログラムを介して導出する様子を示す．その際に PCFG やその拡張文法が分かりやすく記述できるような**確率確定節文法** (probabilistic definite clause grammar) という形式を新たに提案する．これは自然言語処理における単一化文法 (unification grammar) の一つの実装である確定節文法 (definite clause grammar) [44, 57, 59] の確率化である．

加えて，先に gEM アルゴリズムは最悪計算量において既存の専用 EM アルゴリズムと同等であると述べたが，現実の日本語文法に基づく PCFG において gEM アルゴリズムが先に挙げた Inside-Outside アルゴリズムよりも平均文長においておよそ 1,000 倍高速に動作することが実験によって確かめられた．本論文ではその結果も合わせて示す．

1.1 本論文の構成

本論文は次のように構成されている．第 2 章では既存の有用な記号的統計モデルを説明し，一階述語論理と確率論の融合に関する先行研究について述べる．第 3 章では本論文が提案する記号的統計モデル言語 PRISM とその表現力について考察を行う．第 4 章では gEM アルゴリズムが適用可能な PRISM プログラムのサブクラスに対して議論し，gEM アルゴリズムを具体的に示す．第 5 章では第 4 章で示した PRISM プログラム用の高速 EM アルゴリズムを PCFG に適用した場合の計算時間を計測し，従来法である Inside-Outside アルゴリズムに比べて学習時間が大幅に短縮で

きることを示す．また，PCFG の拡張モデルに対する効率的な EM 学習法を PRISM プログラムから導出する．第 6 章で関連研究について述べ，第 7 章で本論文のまとめを行う．

1.2 本論文の記法に関する注意

本論文の論理プログラムの記法は Prolog の慣習に従う．またオブジェクトプログラムはタイプライタ書体で記述し，説明のためのメタ変数は斜体で記述する．また，本論文では以下に述べる記号を断りなく使う．

y_n を第 n 要素とするリストを $\langle y_1, y_2, \ldots \rangle$ で表現する．これはベクトル $\langle e_1, e_2, \ldots \rangle$ と同じ形であるが，違いは文脈から判断できる．またリスト $Y = \langle \ldots, y, \ldots \rangle$ に対して $y \in Y$ と書く．集合 X の要素数，記号列 ζ に含まれる記号数，リスト Y の要素数をそれぞれ $|X|, |\zeta|, |Y|$ で表す．これらはどれも見た目は同じだが文脈で違いを判断できる．集合の直積は “$\times$” で表す．$X_1 \times X_2 \times \cdots \times X_n$ を $\prod_{i=1}^{n} X_i$ と書く．また，X と Y の差集合は $X \setminus Y$ と書く．

確率変数の集合 $Y = \{X_1, X_2, \ldots, X_{|Y|}\}$ に対して，Y の要素を並べたベクトル $\langle X_1, X_2, \ldots, X_{|Y|} \rangle$ を Y の確率ベクトルと呼び，$\boldsymbol{Y}$ で表す．そして，各 X_i の実現値を x_i とおいたとき，$\boldsymbol{y} = \langle x_1, x_2, \ldots, x_{|Y|} \rangle$ を $\boldsymbol{Y}$ の実現値ベクトル，もしくは単に $\boldsymbol{Y}$ の実現値と呼ぶ．これらを用いて確率分布 $Pr(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_{|Y|}, \ldots)$ を $Pr(\boldsymbol{Y} = \boldsymbol{y}, \ldots)$ と略記する．更に，文脈より $\boldsymbol{y}$ を実現値とするのが $\boldsymbol{Y}$ であることが明らかなときには $Pr(\boldsymbol{Y} = \boldsymbol{y})$ を $Pr(\boldsymbol{y})$ と略す．また，$\mathbf{1}$ を要素がすべて 1 であるような実現値ベクトル，$\mathbf{0}$ を要素がすべて 0 であるような実現値ベクトルとする．そして，確率変数 X のとり得る実現値の集合を $\mathcal{D}(X)$ とおく．確率ベクトル $\boldsymbol{Y} = \langle X_1, X_2, \ldots, X_{|Y|} \rangle$ に対し，$\mathcal{D}(\boldsymbol{Y})$ は $\prod_{i=1}^{|Y|} \mathcal{D}(X_i)$ を表す．

述語を述語名 p と引数長 n で区別し，その組を p/n と表す．また，記号を節約するため，アトム（原子式）の集合 $\{A_1, A_2, \ldots\}$ とその要素から成る連言 $A_1 \wedge A_2 \wedge \ldots$ を式や説明文中で区別しないことがある．

アルゴリズムの記述で用いる疑似コード中の式 $x \mathrel{+}= y$, $x \mathrel{*}= y$ はそれぞれ $x := x + y$, $x := x \cdot y$ の略記である．また，疑似コード中のプログラム変数は適宜確保されているものとする．

第2章　研究の背景と目的

本章では，研究の背景としてはじめにいくつかの有用な記号的統計モデルクラスを導入し，それらの記述力やEM学習に関する考察を行う．また，先に述べたように，一階述語論理のような記述力の高い言語で統計モデルを記述する試みが近年盛んに行われている．その中でいくつかある系統のうち本研究と関連があるものについて説明を行う．そして，本章の最後で研究の目的を改めて示す．

2.1　既存の記号的統計モデル

本節では，まず既存の有用な記号的統計モデルクラスである隠れマルコフモデル (HMM)，確率文脈自由文法 (PCFG)，Bayesian ネットワークを説明する．それに加え，PCFG に文脈依存性を採り入れたいくつかの確率文法を導入する．これらのモデルクラスが有用なものになっているのは，各々が専用の効率的な EM アルゴリズムを備えているためである．本節ではこれらの EM アルゴリズムも合わせて説明する．加えて，本節では上に挙げたモデルクラスの表現力や専用 EM アルゴリズムの計算効率について考察する．Russell はモデルクラス間の表現力の関係を次のように定義した[1] [52].

定義 1（モデルクラス間の表現力の関係） 2 つのモデルクラス $\mathcal{M}_1, \mathcal{M}_2$ を考える．$\mathcal{M}_1$ に属する任意のモデル m_1 に対して，m_1 と同じ確率分布を表現するモデル $m_2 \in \mathcal{M}_2$ が必ず存在するとき，またそのときに限り，$\mathcal{M}_2$ は $\mathcal{M}_1$ に対して同等以上の表現力をもつ $\cdots(*)$. ■

以降では $(*)$ を「$\mathcal{M}_2$ は $\mathcal{M}_1$ の表現上の一般化である」といい，$\mathcal{M}_1 \prec_{\mathrm{rep}} \mathcal{M}_2$ と書くことにする．また，先に述べたように，モデルクラス $\mathcal{M}$ が効率的なパラメータ推定 (EM) アルゴリズムを有しているかどうかは $\mathcal{M}$ の有用性に大きく関わってくる．そこで次の関係を定義する．

定義 2（各モデルクラスの EM アルゴリズム間の関係） モデルクラス $\mathcal{M}_1, \mathcal{M}_2$ にそれぞれ専用の EM アルゴリズム $\mathcal{A}_1, \mathcal{A}_2$ が備えられていたとする．ただし，$\mathcal{M}_2$ が $\mathcal{M}_1$ の表現上の一般化であるとする．そして $\mathcal{M}_1$ に属する任意のモデルを m_1 とし，$m_2 \in \mathcal{M}_2$ を m_1 と同じ確率分布を表現するモデルとする．$\mathcal{A}_1$ と $\mathcal{A}_2$ が等価であり，かつ m_2 に $\mathcal{A}_2$ を適用したときの計算量と m_1 に $\mathcal{A}_1$ を適用したときの計算量が同オーダであるとき，またそのときに限り「$\mathcal{A}_2$ は $\mathcal{A}_1$ の一般化である」といい，$\mathcal{A}_2 \prec_{\mathrm{EM}} \mathcal{A}_1$ と書く．■

2 つの EM アルゴリズムの等価性の基準，および EM アルゴリズムの計算量の測り方は後に与える．我々は定義 1, 定義 2 の視点から HMM, PCFG, Bayesian ネットワークといった有用な記号的統計モデルクラスの一般性と有用性を議論する．詳しくは後に述べるが，HMM, PCFG, Bayesian ネットワークの表現上の一般化であり，かつその EM アルゴリズムが HMM, PCFG, Bayesian ネッ

[1] 原文は次の通り：Roughly speaking, a model family $\mathcal{L}_2$ is at least as expressive as $\mathcal{L}_1$ iff, for every e_1 in $\mathcal{L}_1$, there is some e_2 in $\mathcal{L}_2$ such that e_1 and e_2 denote "the same" distribution. 原文中の記号 $\mathcal{L}_1$, $\mathcal{L}_2$ は $\mathcal{M}_1$, $\mathcal{M}_2$ に，e_1, e_2 は m_1, m_2 に置き換えられている．

トワーク専用の EM アルゴリズムの一般化であるような記号的統計モデルクラスを提示するのが本研究の目的の一つである．はじめに，一般的な EM アルゴリズムを説明し，その後に既存の統計モデルおよびその専用 EM アルゴリズムを導入する．

2.1.1 EM アルゴリズム

まず，対象となる確率現象に関係するすべての確率変数の集合 X を考える．その確率ベクトル $\boldsymbol{X}$ の実現値を $\boldsymbol{x}$ とおく．また X のうち，その実現値を観測可能な確率変数の集合を Y とおき，その確率ベクトル $\boldsymbol{Y}$ の実現値を $\boldsymbol{y}$ とおく．$\boldsymbol{y}$ を**観測データ**と呼ぶ．ここで $\boldsymbol{y}$ の情報はすべて $\boldsymbol{x}$ にも含まれているが，その逆は成り立たない場合を考える[2]．このとき観測データ $\boldsymbol{y}$ は**不完全データ** (imcomplete data) と呼ばれる．完全データ $\boldsymbol{x}$ と不完全データ $\boldsymbol{y}$ の関係は many-to-one な関数 $\varphi : \mathcal{D}(\boldsymbol{X}) \to \mathcal{D}(\boldsymbol{Y})$ によって表現される．$\mathcal{D}(\boldsymbol{X})$ 上での $\boldsymbol{x}$ の確率密度 $p(\boldsymbol{x}|\boldsymbol{\theta})$ に対して（$\boldsymbol{\theta}$ は分布パラメータ），$\mathcal{D}(\boldsymbol{Y})$ 上での $\boldsymbol{y}$ の確率密度 $p(\boldsymbol{y}|\boldsymbol{\theta})$ は

$$p(\boldsymbol{y}|\boldsymbol{\theta}) = \int_{\boldsymbol{x}:\boldsymbol{y}=\varphi(\boldsymbol{x})} p(\boldsymbol{x}|\boldsymbol{\theta}) d\boldsymbol{x} \tag{2.1}$$

と計算される．一般に，独立な T 回の観測によって得られた観測データ（不完全とは限らない）$\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(T)}$ について**尤度** (likelihood) $\Lambda(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(T)}|\boldsymbol{\theta})$ を次のように定義する．

$$\Lambda(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(T)}|\boldsymbol{\theta}) \stackrel{\text{def}}{=} \prod_{t=1}^{T} p(\boldsymbol{y}^{(t)}|\boldsymbol{\theta}) \tag{2.2}$$

上の尤度を最大化するような $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ を見つけることを「$\boldsymbol{\theta}$ を**最尤推定**する」といい，$\boldsymbol{\theta}^*$ を $\boldsymbol{\theta}$ の**最尤推定値**という．統計モデルのパラメータ学習において，我々は常にこのような $\boldsymbol{\theta}$ の最尤推定を考えることにする．$\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)}, \ldots, \boldsymbol{y}^{(T)}$ が不完全データであるとき，EM アルゴリズム [17, 34, 35] は $\boldsymbol{\theta}$ の最尤推定において比較的簡単な手順で尤度最大化を図る数値計算技法であり，Dempster らによって一般的な形で形式化された [17]．

EM アルゴリズムでははじめに次の Q **関数**を導入する．簡単のため $T = 1$, $\boldsymbol{y} = \boldsymbol{y}^{(1)}$ とおく．

$$Q(\boldsymbol{\theta}', \boldsymbol{\theta}) \stackrel{\text{def}}{=} \int_{\boldsymbol{x}:\boldsymbol{y}=\varphi(\boldsymbol{x})} p(\boldsymbol{x}|\boldsymbol{y}, \boldsymbol{\theta}) \log p(\boldsymbol{x}, \boldsymbol{y}|\boldsymbol{\theta}') \tag{2.3}$$

そして，EM アルゴリズムでははじめに適当にパラメータを $\boldsymbol{\theta}^{(0)}$ に初期化し，$\Lambda(\boldsymbol{y}|\boldsymbol{\theta})$ が収束するまで下の **Expectation ステップ**（E ステップと呼ぶ）と **Maximization ステップ**（M ステップと呼ぶ）を交互に繰り返す．その時点の値 $\boldsymbol{\theta}$ を最尤推定値 $\boldsymbol{\theta}^*$ として終了する．

E ステップ: $Q(\boldsymbol{\theta}', \boldsymbol{\theta}^{(m)})$ を計算する．
M ステップ: $\boldsymbol{\theta}^{(m+1)} := \arg\max_{\boldsymbol{\theta}'} Q(\boldsymbol{\theta}', \boldsymbol{\theta}^{(m)})$ を計算し，$m := m + 1$ とする．

E ステップと M ステップにより $\boldsymbol{\theta}^{(m)}$ を $\boldsymbol{\theta}^{(m+1)}$ に置き換えることを以降では「パラメータを更新する」という．EM アルゴリズムにおいてはパラメータを更新する度に尤度 $\Lambda(\boldsymbol{y}|\boldsymbol{\theta})$ が単調増加する（従って最終的に収束する）ことが Dempster らによって証明されている [17, 34]．

2 つの EM アルゴリズムの等価性は次のように与えられる．等価な 2 つの EM アルゴリズムに同じ初期パラメータを与えたとき，同じパラメータ値に収束するのは定義より明らかである．

定義 3（EM アルゴリズムの等価性） 同一の統計モデルに適用される，異なる 2 つの EM アルゴリズム $\mathcal{A}_1, \mathcal{A}_2$ を考える．$\mathcal{A}_1, \mathcal{A}_2$ の m 回目の更新後にそれぞれで得られたパラメータが一致し

[2] 不完全データ $\boldsymbol{y}$ の別の形式化として，$Y \subset X$ であるという立場をとることもある．このとき，完全データは $\boldsymbol{x} = \langle \boldsymbol{y}, \boldsymbol{z} \rangle$ と書け，$\boldsymbol{z}$ は**補助データ** (augumented data) と見なせる．

たとき，$m+1$ 回目の更新後でもパラメータが一致するならば，$\mathcal{A}_1$ と $\mathcal{A}_2$ は等価な EM アルゴリズムである． ■

また，EM アルゴリズムは一種の山登り法であるため，尤度を大域的に最大化する保証はなく，その振舞いは初期パラメータ $\boldsymbol{\theta}^{(0)}$ に大きく依存する．また，パラメータ更新を何回繰り返せば尤度が収束するのかも $\boldsymbol{\theta}^{(0)}$ に依存するため，前もって知ることはできない．従って本論文では慣例に従って次の仮定を行なう．

仮定 1（EM アルゴリズムの計算量） EM アルゴリズムの計算量は一回のパラメータ更新に要する計算量で評価される． ■

2.1.2 隠れマルコフモデル

隠れマルコフモデル（hidden Markov model; 以下，HMM）は 5 つ組 $\langle S_{\rm hmm}, V_{\rm hmm}, \boldsymbol{\theta}_{\rm tr}, \boldsymbol{\theta}_{\rm out}, \boldsymbol{\theta}_{\rm init}\rangle$ で定義される[3]．$S_{\rm hmm} = \{q_1, q_2, \ldots, q_{|S_{\rm hmm}|}\}$ は状態集合，$V_{\rm hmm} = \{v_1, v_2, \ldots, v_{|V_{\rm hmm}|}\}$ は観測・出力記号の集合である．また，$\boldsymbol{\theta}_{\rm tr} = \{a_{qq'}\}$ は状態遷移確率行列で，$a_{qq'} = Pr(Q_{\ell+1} = q' \mid Q_\ell = q)$ である $(q, q' \in S_{\rm hmm})$．そして $\boldsymbol{\theta}_{\rm out} = \{b_q(v)\}$ は観測（出力）記号確率分布で，$b_q(v) = Pr(O_\ell = v | Q_\ell = q)$ である $(v \in V_{\rm hmm}, q \in S_{\rm hmm})$．さらに $\boldsymbol{\theta}_{\rm init} = \{a_q\}$ は初期状態分布で，$a_q = Pr(Q_1 = q)$ である $(q \in S_{\rm hmm})$．ここで Q_ℓ と O_ℓ はそれぞれ時刻 ℓ における状態，観測（出力）記号を値にとる確率変数である．2 つの確率ベクトル $\boldsymbol{O}$ と $\boldsymbol{Q}$ をそれぞれ $\boldsymbol{O} \stackrel{\rm def}{=} \langle O_1, O_2, \ldots, O_L\rangle$, $\boldsymbol{Q} \stackrel{\rm def}{=} \langle Q_1, Q_2, \ldots, Q_L\rangle$ と定める．式を簡単にするため以下では $\boldsymbol{\theta}_{\rm hmm} \stackrel{\rm def}{=} \langle \boldsymbol{\theta}_{\rm init}, \boldsymbol{\theta}_{\rm tr}, \boldsymbol{\theta}_{\rm out}\rangle$ とおく．

次に HMM を記号列 $\boldsymbol{o} = o_1 o_2 \cdots o_\ell$ の生成器として説明する．記号列生成器としての HMM は次のように振舞う．

1. 初期状態分布 $\boldsymbol{\theta}_{\rm init}$ に従い，確率 $Pr(Q_1 = q)$ で初期状態 $q_1 = q$ を選ぶ．
2. $\ell = 1$ とする．
3. 状態 q_ℓ の出力記号確率分布 $\boldsymbol{\theta}_{\rm tr}$ に従い，確率 $Pr(O_\ell = v \mid Q_\ell = q_\ell)$ で $o_\ell = v$ を選ぶ．
4. 状態 q_ℓ の状態遷移確率分布 $\boldsymbol{\theta}_{\rm out}$ に従い，確率 $Pr(Q_{\ell+1} = q \mid Q_\ell = q_\ell)$ で次の状態 $q_{\ell+1} = q$ を選ぶ．
5. $\ell := \ell + 1$ とする．$\ell < L$ ならばステップ 3 に戻る．そうでなければ終了する．

ただし L は記号列の長さであり，かつ $o_\ell \in V_{\rm hmm}, \ell = 1 \ldots L$ である．

2.1.2.1 HMM の EM 学習

よく知られた HMM の 3 つの問題 [8, 25, 33, 49] の 1 つに，「独立に得られた T 個の観測記号列 $\boldsymbol{o}^{(t)} \stackrel{\rm def}{=} o_1^{(t)} o_2^{(t)} \cdots o_L^{(t)}$ に対して $(t = 1 \ldots T)$，その尤度 $\prod_{t=1}^{T} Pr(\boldsymbol{O} = \boldsymbol{o}^{(t)} \mid \boldsymbol{\theta}_{\rm hmm})$ を最大にするような $\boldsymbol{\theta}^*_{\rm hmm} = \langle \boldsymbol{\theta}^*_{\rm init}, \boldsymbol{\theta}^*_{\rm tr}, \boldsymbol{\theta}^*_{\rm out}\rangle$ を求めよ（$\boldsymbol{\theta}_{\rm hmm}$ を最尤推定せよ）」というパラメータ推定問題がある．前節で述べた HMM の振る舞いから分かるように，観測 $\boldsymbol{o}^{(t)}$ に影響を与えている状態遷移

[3] ここでの定義は Rabiner のもの [49] に従うが，使っている記号は異なる．Rabiner の定義では Q_ℓ が与えられたとき $Q_{\ell+1}$ と O_ℓ は独立であると仮定しているが，Charniak の定義 [8] ではこのような独立性は仮定せず，2 つの条件つき確率 $Pr(Q_{\ell+1}|Q_\ell)$, $Pr(O_\ell|Q_\ell)$ の代わりに条件付確率 $Pr(O_\ell, Q_{\ell+1}|Q_\ell)$ を用いる．Charniak の定義による HMM は Rabiner の定義による HMM の表現上の一般化（定義 1）になっており，パラメータ数はそれぞれ $|S_{\rm hmm}|^2 + |S_{\rm hmm}| \cdot |V_{\rm hmm}|$ 個，$|S_{\rm hmm}|^2 |V_{\rm hmm}|$ 個である．

系列 $\boldsymbol{q}^{(t)} \stackrel{\text{def}}{=} \langle q_1^{(t)}, q_2^{(t)}, \ldots, q_L^{(t)} \rangle$ は直接観測できない[4]ので $(t = 1 \ldots T)$，一つの解決として我々は EM アルゴリズムによる $\boldsymbol{\theta}_{\text{hmm}}$ の（局所）最尤推定を行う．

効率的な HMM の EM 学習アルゴリズムとして，Baum-Welch アルゴリズムが知られている [8, 25, 33, 49]．Baum-Welch アルゴリズムでは**前向き確率** $\alpha_\ell^{(t)}(q)$ と**後向き確率** $\beta_\ell^{(t)}(q)$ という 2 つの確率値が重要な役割を果たす $(t = 1 \ldots T,\ \ell = 1 \ldots L,\ q \in S_{\text{hmm}})$．それぞれ式 2.4, 2.5 のように定義される[5]．ただし式中で用いられている確率ベクトル $\boldsymbol{O}_{\ell,\ell'}$ とその実現値ベクトル $\boldsymbol{o}_{\ell,\ell'}^{(t)}$ はそれぞれ $\boldsymbol{O}_{\ell,\ell'} \stackrel{\text{def}}{=} \langle O_\ell, O_{\ell+1}, \ldots, O_{\ell'} \rangle$ と $\boldsymbol{o}_{\ell,\ell'}^{(t)} \stackrel{\text{def}}{=} \langle o_\ell^{(t)}, o_{\ell+1}^{(t)}, \ldots, o_{\ell'}^{(t)} \rangle$ を意味する．

$$\alpha_\ell^{(t)}(q) \stackrel{\text{def}}{=} Pr(\boldsymbol{O}_{1,\ell-1} = \boldsymbol{o}_{1,\ell-1}^{(t)}, Q_\ell = q \mid \boldsymbol{\theta}_{\text{hmm}}) \tag{2.4}$$

$$\beta_\ell^{(t)}(q) \stackrel{\text{def}}{=} Pr(\boldsymbol{O}_{\ell,L} = \boldsymbol{o}_{\ell,L}^{(t)} \mid Q_\ell = q, \boldsymbol{\theta}_{\text{hmm}}) \tag{2.5}$$

各観測記号列 $\boldsymbol{o}^{(t)}$ に対する $(t = 1 \ldots T)$ 前向き確率 $\alpha_\ell^{(t)}(q)$ は次の式を使って $\ell = 1, 2, \ldots$ から順に $\ell = L$ まで計算することによって求められる：

$$\alpha_\ell^{(t)}(q) := \begin{cases} a_q & \text{if } \ \ell = 1, \\ \sum_{q' \in S_{\text{hmm}}} \alpha_{\ell-1}^{(t)}(q') a_{q'q} b_{q'}(o_{\ell-1}^{(t)}) & \text{otherwise} \end{cases} \tag{2.6}$$
for each $q \in S_{\text{hmm}}$.

同様に，後向き確率 $\alpha_\ell^{(t)}(q)$ は次の式を使って $\ell = L, L-1, \ldots$ から順に $\ell = 1$ まで計算される：

$$\beta_\ell^{(t)}(q) := \begin{cases} b_q(o_L^{(t)}) & \text{if } \ \ell = L, \\ \sum_{q' \in S_{\text{hmm}}} a_{qq'} b_q(o_\ell^{(t)}) \beta_{\ell+1}^{(t)}(q') & \text{otherwise} \end{cases} \tag{2.7}$$
for each $q \in S_{\text{hmm}}$.

そして，T 個の観測記号列 $\boldsymbol{o}^{(1)}, \boldsymbol{o}^{(2)}, \ldots, \boldsymbol{o}^{(T)}$ を生成するときに状態 q から状態 q' へ遷移した回数の期待値（$\boldsymbol{o}^{(1)}, \boldsymbol{o}^{(2)}, \ldots, \boldsymbol{o}^{(T)}$ の下での条件つき期待値）$\eta_{\text{tr}}(q, q')$ は次のように求められる．

$$\eta_{\text{tr}}(q, q') = \sum_{t=1}^{T} \frac{1}{P_t} \sum_{\ell=1}^{L-1} \alpha_\ell^{(t)}(q) a_{qq'} b_q(o_\ell^{(t)}) \beta_{\ell+1}^{(t)}(q') \tag{2.8}$$

ただし $P_t = Pr(\boldsymbol{O} = \boldsymbol{o}^{(t)} | \boldsymbol{\theta}_{\text{hmm}})$ とおいている．記号列 $\boldsymbol{o}^{(t)}$ の生起確率 $Pr(\boldsymbol{O} = \boldsymbol{o}^{(t)} | \boldsymbol{\theta}_{\text{hmm}})$ は前向き確率または後向き確率を使って，

$$Pr(\boldsymbol{O} = \boldsymbol{o}^{(t)} | \boldsymbol{\theta}_{\text{hmm}}) := \sum_{q \in S_{\text{hmm}}} \alpha_L^{(t)}(q) \tag{2.9}$$

$$Pr(\boldsymbol{O} = \boldsymbol{o}^{(t)} | \boldsymbol{\theta}_{\text{hmm}}) := \sum_{q \in S_{\text{hmm}}} a_q \beta_1^{(t)}(q) \tag{2.10}$$

と計算される．また，状態 q において記号 v の出力回数の期待値 $\eta_{\text{out}}(q, v)$, 初期状態が q になる回数の期待値（すなわち初期状態が q になる確率）$\eta_{\text{init}}(q)$ はそれぞれ次のようにして求められる．

$$\eta_{\text{out}}(q, v) := \sum_{t=1}^{T} \frac{1}{P_t} \sum_{\ell:\ \ell=1\ldots L,\ o_\ell^{(t)} = v} \sum_{q' \in S_{\text{hmm}}} \alpha_\ell^{(t)}(q) a_{qq'} b_q(o_\ell^{(t)}) \beta_{\ell+1}^{(t)}(q') \tag{2.11}$$

$$\eta_{\text{init}}(q) := \sum_{t=1}^{T} \frac{1}{P_t} \alpha_1^{(t)}(q) \beta_1^{(t)}(q) \tag{2.12}$$

[4] $\boldsymbol{O}, \boldsymbol{Q}$ の実現値をそれぞれ o, q とおくと，Dempster らの形式化 [17] では $\langle o, q \rangle$ が完全データ，o が観測される不完全データとなる．また，many-to-one の対応関数 $\varphi : \mathcal{D}(\boldsymbol{O}) \times \mathcal{D}(\boldsymbol{Q}) \to \mathcal{D}(\boldsymbol{O})$ は $\varphi(\langle o, q \rangle) = o$ として与えられる．

[5] ここでは，後の説明の都合上，Rabiner [49] とは異なる定義を与えている．それに伴って Baum-Welch アルゴリズムの記述も適宜変更している．ただし，最終的に計算される期待値 $\eta_{\text{tr}}(q, q')$, $\eta_{\text{out}}(q, v)$, $\eta_{\text{init}}(q)$ が Rabiner の記述した Baum-Welch アルゴリズムと等しくなるのは明らかで，従ってここで示す Baum-Welch アルゴリズムは Rabiner が与えたものと等価である．

Baum-Welch アルゴリズムでは，初期パラメータ $\boldsymbol{\theta}^{(0)}_{\rm hmm}$ を設定した後，期待値 $\eta_{\rm tr}(q,q')$, $\eta_{\rm out}(q,v)$, $\eta_{\rm init}(q)$ を使い次のように各 $q \in S_{\rm hmm}$, $v \in V_{\rm hmm}$ についてパラメータを更新する：

$$a_q := \eta_{\rm init}(q) \Big/ \textstyle\sum_{q' \in S_{\rm hmm}} \eta_{\rm init}(q') \ , \tag{2.13}$$

$$a_{qq'} := \eta_{\rm tr}(q,q') \Big/ \textstyle\sum_{q'' \in S_{\rm hmm}} \eta_{\rm tr}(q,q'') \ , \tag{2.14}$$

$$b_q(v) := \eta_{\rm out}(q,v) \Big/ \textstyle\sum_{v' \in V_{\rm hmm}} \eta_{\rm out}(q,v') \ . \tag{2.15}$$

そして，この更新を繰り返すと対数尤度 $\sum_{t=1}^{T} \log Pr(\boldsymbol{O}=\boldsymbol{o}^{(t)}|\boldsymbol{\theta}_{\rm hmm})$ は単調に増加していき最終的に収束する．収束したらその時点のパラメータ $\boldsymbol{\theta}^{*}_{\rm hmm}$ を推定パラメータとして Baum-Welch アルゴリズムは終了する．

2.1.2.2 Baum-Welch アルゴリズムの計算量

先に述べたように，EM アルゴリズムの計算量を 1 回のパラメータ更新に要する時間計算量で測ることにする．式 2.6, 2.7 より明らかなように，前向き確率 $\alpha^{(t)}_{\ell}(q)$ と後向き確率 $\beta^{(t)}_{\ell}(q)$ の計算には $O(|S_{\rm hmm}|^2 LT)$ 時間を要する．ただし，先に定めた通り，$|S_{\rm hmm}|$ は状態数，L は出力・受理する記号列の長さである．また，式 2.8, 2.11, 2.12 より，期待値 $\eta_{\rm tr}(q,q')$, $\eta_{\rm out}(q,v)$, $\eta_{\rm init}(q)$ の計算はそれぞれ $O(|S_{\rm hmm}|^2 LT)$, $O(|S_{\rm hmm}|^2 LT)$, $O(|S_{\rm hmm}|T)$ 時間かかる．更に，再計算を防ぐように式 2.13, 2.14, 2.15 を計算すれば，a_q, $a_{qq'}$, $b_q(v)$ の更新はそれぞれ $O(|S_{\rm hmm}|)$, $O(|S_{\rm hmm}|^2)$, $O(|S_{\rm hmm}| \cdot |V_{\rm hmm}|)$ 時間で済むことが分かる（$|V_{\rm hmm}|$ は出力記号パターン数）．

以上より，Baum-Welch アルゴリズムの計算量は $O\left(\max\left\{|S_{\rm hmm}|^2 LT,\ |S_{\rm hmm}| \cdot |V_{\rm hmm}|\right\}\right)$ であることが分かった．最尤推定は大標本（すなわち T は十分に大きいこと）を前提としており，$|S_{\rm hmm}|^2 LT$ の方が $|V_{\rm hmm}|$ よりも大きいと考えるのが普通である．よって以降では Baum-Welch アルゴリズムの計算量を $O(|S_{\rm hmm}|^2 LT)$ とする．

2.1.3 確率文脈自由文法

文脈自由文法 G を 4 つ組 $\langle V_{\rm n}, V_{\rm t}, R, S\rangle$ で定義する．ただし，$V_{\rm n}$ は非終端記号の集合，$V_{\rm t}$ は終端記号の集合，R は規則の集合，S は開始記号 $(S \in V_{\rm n})$ である．R 中の各規則 r は $A \rightarrow \zeta$ の形をしており，記号列 ζ' 中に非終端記号 A が出現するとき，A を ζ に置き換える（「A を展開する」,「r を適用する」などという）ことが可能であることを表す．我々は常に最左の非終端記号を置き換えるように規則を適用する（すなわち**最左導出**に固定する）．規則 r の適用により記号列 ζ が ξ に置き換えられるとき $\zeta \stackrel{r}{\Rightarrow} \xi$ と書く．r が略される場合もある．このような置き換えを 0 回以上行なって ζ から ξ が得られるとき，$\zeta \stackrel{*}{\Rightarrow} \xi$ と書く．特に，置換えが 1 回以上であることを強調する場合は $\zeta \stackrel{+}{\Rightarrow} \xi$ と書く．S から導出可能な非終端記号列 $\boldsymbol{w}$ を文と呼ぶ．CFG G における文の集合を G の言語と呼び，$\mathcal{L}_G$ と書く．

CFG G に基づく PCFG を $G(\boldsymbol{\theta}_{\rm pcfg})$ で表す．逆に G を「PCFG $G(\boldsymbol{\theta}_{\rm pcfg})$ の文法構造」と呼ぶ．$\boldsymbol{\theta}_{\rm pcfg}$ は $|R|$ 次元ベクトルであり，以降パラメータと呼ぶ．$\boldsymbol{\theta}_{\rm pcfg}$ の各要素は $\theta_{\rm pcfg}(r)$ で参照され，$0 \le \theta_{\rm pcfg}(r) \le 1$, $\sum_{\zeta:(A\rightarrow\zeta)\in R} \theta_{\rm pcfg}(A \rightarrow \zeta) = 1$ が成り立つとする $(A \in V_{\rm n},\ r \in R)$．考えているモデルが PCFG であると文脈から判断できる場合，$\theta_{\rm pcfg}$, $\boldsymbol{\theta}_{\rm pcfg}$ をそれぞれ単に θ, $\boldsymbol{\theta}$ と書く．PCFG では「最左の非終端記号 A に対し，適用する規則は他に影響を受けずに選択される」と仮定される．従って $\zeta_0 \stackrel{r_1}{\Rightarrow} \zeta_1 \stackrel{r_2}{\Rightarrow} \zeta_2 \stackrel{r_3}{\Rightarrow} \cdots \stackrel{r_K}{\Rightarrow} \zeta_K$ における規則の適用列 $\boldsymbol{r} = \langle r_1, r_2, \ldots, r_K\rangle$ の

出現確率 $Pr(\boldsymbol{r})$ は

$$Pr(\boldsymbol{r}) = \prod_{k=1}^{K} \theta(r_k) \tag{2.16}$$

と計算される．また，$\sigma(r, \boldsymbol{r})$ を適用規則列 $\boldsymbol{r}$ 中に 規則 r が出現する数とすると，

$$Pr(\boldsymbol{r}) = \prod_{r \in R} \theta(r)^{\sigma(r,\boldsymbol{r})} \tag{2.17}$$

と書くこともできる．$S \stackrel{*}{\Rightarrow} \boldsymbol{w}$ を実現する適用規則列すべてから成る集合を $\psi(\boldsymbol{w})$ とおく．文 $\boldsymbol{w}$ は適用規則列 $\boldsymbol{r}$ から一意に定まるので，

$$Pr(\boldsymbol{w}, \boldsymbol{r}) = \begin{cases} Pr(\boldsymbol{r}) & \text{if } \boldsymbol{r} \in \psi(\boldsymbol{w}) \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

が成り立つ．式 2.18 より S から $\boldsymbol{w}$ が導出される確率 $Pr(\boldsymbol{w})$ は次のように求まる：

$$Pr(\boldsymbol{w}) = \sum_{\text{all } \boldsymbol{r}} Pr(\boldsymbol{w}, \boldsymbol{r}) = \sum_{\boldsymbol{r} \in \psi(\boldsymbol{w})} Pr(\boldsymbol{r}) \ . \tag{2.19}$$

パラメータ $\boldsymbol{\theta}$ の下での確率値であることを強調するときは $Pr(\boldsymbol{r})$, $Pr(\boldsymbol{w})$ を $Pr(\boldsymbol{r}|\boldsymbol{\theta})$, $Pr(\boldsymbol{w}|\boldsymbol{\theta})$ と書く．上述した規則適用の独立性に加え，以降で考える PCFG $G(\boldsymbol{\theta})$ は次の 2 つの条件を満たすものとする．

- 右辺が ε である規則（ε 規則）や $A \stackrel{+}{\Rightarrow} A$ となるような $A \in V_{\mathrm{n}}$ が存在しない．
- $G(\boldsymbol{\theta})$ は整合的 (consistent) [64] である．すなわち $\sum_{\boldsymbol{w} \in \mathcal{L}_G} Pr(\boldsymbol{w}|\boldsymbol{\theta}) = 1$ が成り立つ．

Chi と Geman は，有限サイズの構文木や括弧なしの平文から訓練したパラメータ $\boldsymbol{\theta}^*$ の下で $G(\boldsymbol{\theta}^*)$ は整合的になることを示した [10].

2.1.3.1 PCFG の EM 学習

HMM の場合と同様に，PCFG $G(\boldsymbol{\theta})$ に対してサイズ T のコーパス（例文集合）$\langle \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T \rangle$ が与えられたとき（ただし $\boldsymbol{w}_t \in \mathcal{L}_G, t = 1 \ldots T$），尤度 $\Lambda(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T|\boldsymbol{\theta}) \stackrel{\text{def}}{=} \prod_{t=1}^{T} Pr(\boldsymbol{w}_t \mid \boldsymbol{\theta})$ に対して $\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} \Lambda(\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T|\boldsymbol{\theta})$ なる最尤推定値 $\boldsymbol{\theta}^*$ を見つける問題を考える．文 $\boldsymbol{w}_t$ を定める規則適用列 $\boldsymbol{r}$ が複数ある場合（すなわち文法構造 G が曖昧な場合），そのどれから $\boldsymbol{w}_t$ が得られたのかは知り得ない．従って HMM の場合と同様，EM 学習によって局所的な最尤推定値 $\boldsymbol{\theta}^*$ を求めるという解決法が考えられる．PCFG の効率的な EM 学習法として Baker [3] が提案した Inside-Outside アルゴリズムが広く知られている [3, 8, 25, 28, 33]．Inside-Outside アルゴリズムは対象とする PCFG の文法構造 G を Chomsky 標準形[6] に制限した上で，CYK (Cocke-Younger-Kasami) パーザ [25, 38, 59] の三角行列をデータ構造として利用することによって効率化を図っている．

Inside-Outside アルゴリズムを以下に記述する．はじめにいくつかの用語と記号を定めておく．コーパス（例文の集合）$\mathcal{C}$ 中の各文 $\boldsymbol{w}^{(t)} = w_1^{(t)} w_2^{(t)} \cdots w_{L_t}^{(t)} \in \mathcal{L}_G$ に対して個々の $w_d^{(t)}$ は単語である $(t = 1 \ldots T,\ L_t > 0,\ d = 1 \ldots L_t)$．単語位置 d と d' の間にある部分単語列 $w_{d+1}^{(t)} \cdots w_{d'}^{(t)}$ を $\boldsymbol{w}_{d,d'}^{(t)}$ と書く（$\boldsymbol{w}^{(t)} = \boldsymbol{w}_{0,L_t}^{(t)}$ である）．また，(部分）単語列 $w_d^{(t)} \cdots w_{d'}^{(t)}$ をリスト $\langle w_d^{(t)}, \ldots, w_{d'}^{(t)} \rangle$

[6] CFG $G = \langle V_{\mathrm{n}}, V_{\mathrm{t}}, R, S \rangle$ において，規則集合 R に $A \to BC$, $A \to a$ の形をした規則しか現れないとき（ただし $A, B, C \in V_{\mathrm{n}}$, $a \in V_{\mathrm{t}}$），G は Chomsky 標準形であるという．

```
1: procedure GET-BETA() begin
2:     for t := 1 to T do begin
3:         β^(t)_{d,d'}(A) := 0 for each A ∈ V_n and d, d' such that 0 ≤ d < d' ≤ L_t;     /* 初期化 */
4:         for d := 0 to L_t − 1 do     /* 三角行列の対角要素について計算 */
5:             foreach A such that (A → w^(t)_{d+1}) ∈ R do
6:                 β^(t)_{d,d+1}(A) := θ(A → w^(t)_{d+1});
7:         for k := 2 to L_t do     /* 三角行列の非対角要素について計算 */
8:             for d := 0 to L_t − k do
9:                 foreach A ∈ V_n do
10:                    β^(t)_{d,d+k}(A) := Σ_{B,C:(A→BC)∈R} θ(A → BC) Σ_{k'=1}^{k−1} β^(t)_{d,d+k'}(B) β^(t)_{d+k',d+k}(C);
11:    end
12: end.
```

図 2.1: PCFG における内側確率の計算ルーチン GET-BETA.

で表すことがある．このとき，Inside-Outside アルゴリズムの中心は下のように定義される**内側確率** $\beta^{(t)}_{d,d'}(A)$, **外側確率** $\alpha^{(t)}_{d,d'}(A)$ という 2 種類の確率値の計算になる $(A \in V_{\rm n},\ 0 \le d < d' \le L_t)$.

$$\beta^{(t)}_{d,d'}(A) \stackrel{\rm def}{=} Pr(A \stackrel{*}{\Rightarrow} \boldsymbol{w}^{(t)}_{d,d'}) \tag{2.20}$$

$$\alpha^{(t)}_{d,d'}(A) \stackrel{\rm def}{=} Pr(S \stackrel{*}{\Rightarrow} \boldsymbol{w}^{(t)}_{0,d} A \boldsymbol{w}^{(t)}_{d',L_t}) \tag{2.21}$$

$(t = 1 \ldots T,\ 0 \le d < d' \le L_t,\ A \in V_{\rm n})$. これらの値は三角行列の要素 $T_{d,d'}$ 中に格納される．部分単語列の定義より $\boldsymbol{w}^{(t)}_{0,L_t} = \boldsymbol{w}_t$ となるので，$\beta^{(t)}_{0,L_t}(S)$ に文 $\boldsymbol{w}_t$ の生起確率 $Pr(S \stackrel{*}{\Rightarrow} \boldsymbol{w}_t)$ が格納される点に注意する．

内側確率と外側確率を計算する手続き GET-BETA, GET-ALPHA を各々図 2.1, 図 2.4 に示す．GET-BETA は CYK パーザにおいて部分木を組み上げるのと同じように，三角行列の対角要素から出発し，隅 $\beta^{(t)}_{0,n_\ell}(\cdot)$ に至るまで段階的に内側確率を計算していく．また，逆に GET-ALPHA では隅 $\alpha^{(t)}_{0,n_\ell}(\cdot)$ から対角要素に向かって外側確率を計算する．図 2.2 に $L = 6$ の三角行列に基づく Inside-Outside アルゴリズムの各確率計算のイメージを示す．GET-BETA の行 10 が内側確率計算の中心である．そこでは図 2.3 (1) で示された状況すべてを考慮して計算が進められる[7]．また，同様に GET-ALPHA の行 8–9 における右辺第 1 項，第 2 項ではそれぞれ図 2.3 (2), (3) という状況がすべて考慮されている．

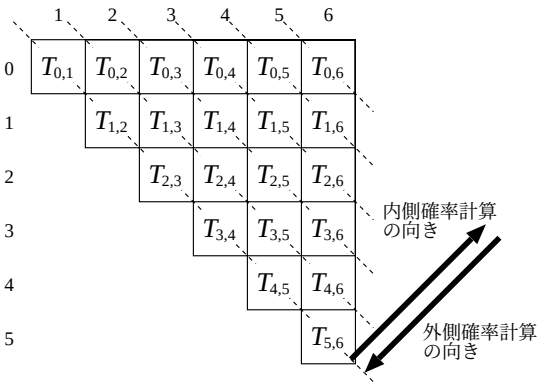

図 2.2: Inside-Outside アルゴリズムにおける内側確率，外側確率の計算イメージ．

[7]図 2.3 中では $L = L_t$ とおいている．

内側確率と外側確率を計算し終ったら，規則 $A \rightarrow BC$, $A \rightarrow a$ の適用回数の期待値が次のように計算される:

$$\eta(A \rightarrow BC) := \sum_{t=1}^{T} \frac{1}{\beta^{(t)}_{0,L_t}(S)} \sum_{k=2}^{L_t} \sum_{d=0}^{L_t-k} \sum_{k'=1}^{k-1} \theta(A \rightarrow BC) \alpha^{(t)}_{d,d+k}(A) \beta^{(t)}_{d,d+k'}(B) \beta^{(t)}_{d+k',d+k}(C), \quad (2.22)$$

$$\eta(A \rightarrow a) := \sum_{t=1}^{T} \frac{1}{\beta^{(t)}_{0,L_t}(S)} \sum_{d=0}^{L_t-1} \theta(A \rightarrow a) \alpha^{(t)}_{d,d+1}(A). \quad (2.23)$$

更に，上で計算された期待値からパラメータ $\theta(A \rightarrow \zeta)$ が更新（再推定）される:

$$\theta(A \rightarrow \zeta) := \eta[A \rightarrow \zeta] \Big/ \textstyle\sum_{\zeta' : (A \rightarrow \zeta') \in R} \eta(A \rightarrow \zeta'). \quad (2.24)$$

Inside-Outside アルゴリズムでは，Baum-Welch アルゴリズムと同様，はじめに $\boldsymbol{\theta}$ を適当な値に初期化し，次いで手続き Get-Beta, Get-Alpha および式 2.22, 2.23, 2.24 によって $\boldsymbol{\theta}$ を更新する．そして，この更新を繰り返すと対数尤度 $\sum_{t=1}^{T} \log Pr(\boldsymbol{w}_t) = \sum_{t=1}^{T} \log \beta^{(t)}_{0,L_t}(S)$ が単調増加しながら最終的には収束する．収束したら，そのときのパラメータの値を最終的な推定値として終了する（この点も Baum-Welch アルゴリズムと同じである）．

2.1.3.2 Inside-Outside アルゴリズムの計算量

Baum-Welch アルゴリズム同様，Inside-Outside アルゴリズムの計算量は 1 回のパラメータ更新に要する時間計算量で評価される．コーパス $\mathcal{C} = \langle \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T \rangle$ に出現する文の最大文長を L とすると，図 2.1, 2.4 の手続き Get-Beta, Get-Alpha を実行するのに $O(|R|L^3T)$ 時間必要なのは明らかである（$|R|$ は規則数）．また，期待値 $\eta(A \rightarrow \zeta)$ の計算（式 2.22, 2.23）に要する時間も同じく $O(|R|L^3T)$ である．式 2.24 のパラメータ更新には $O(|R|)$ 時間かかるが，これは無視できる．以上より，Inside-Outside アルゴリズムの計算量は $O(|R|L^3T)$ である．確率正規文法である HMM の EM 学習（Baum-Welch アルゴリズム）では記号列長 L に対して線形オーダの計算量であったのに対し，PCFG の EM 学習では文長 L に対して 3 乗オーダの計算量になる点に注意する．

また，計算量を非終端記号数 $|V_{\mathrm{n}}|$ に関して考察する場合には（最悪の場合として）Chomsky 標準形を満たす最大の規則集合

$$R_{\max}(V_{\mathrm{n}}, V_{\mathrm{t}}) \stackrel{\mathrm{def}}{=} \{A \rightarrow BC \mid A, B, C \in V_{\mathrm{n}}\} \cup \{A \rightarrow a \mid A \in V_{\mathrm{n}}, a \in V_{\mathrm{t}}\} \quad (2.25)$$

を考える（以下，$R_{\max}$ と略すことがある）．すると $|R_{\max}| = O(|V_{\mathrm{n}}|^3)$ となり，Inside-Outside アルゴリズムの計算量は $O(|V_{\mathrm{n}}|^3 L^3 T)$ となる．

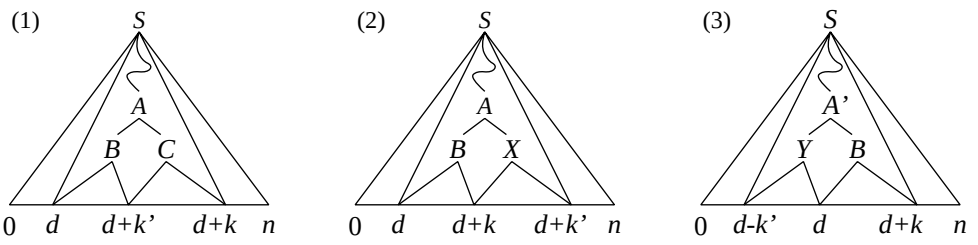

図 2.3: 内側確率，外側確率の計算において想定している状況．

```
1: procedure GET-ALPHA() begin
2:     for t := 1 to T do begin
3:         α^(t)_{d,d'}(A) := 0 for each A ∈ V_n and d, d' such that 0 ≤ d < d' ≤ L_t;     /* 初期化 */
4:         α^(t)_{0,L_t}(S) := 1;     /* ただし，最も右上の S については特別に 1 に初期化 */
5:         for k := L_t downto 2 do
6:             for d := 0 to L_t − k do
7:                 foreach B ∈ V_n do
8:                     α^(t)_{d,d+k}(B) := Σ_{A,X:(A→BX)∈R} θ(A→BX) Σ_{k'=k+1}^{L_t−d} α^(t)_{d,d+k'}(A) β^(t)_{d+k,d+k'}(X)
9:                                         + Σ_{A,Y:(A'→YB)∈R} θ(A→YB) Σ_{k'=1}^{d} α^(t)_{d−k',d+k}(A) β^(t)_{d−k',d}(Y)
10:    end
11: end.
```

図 2.4: PCFG における外側確率の計算ルーチン GET-ALPHA.

2.1.4 Bayesian ネットワーク

確率変数の集合が与えられたとき，Bayesian ネットワーク[8] [7, 43, 51] はその同時確率分布 (joint probability distribution; 以下，同時分布）[9]を有向非循環グラフ (DAG) で表現する．DAG の形状から導かれる（条件つき）独立性によってその同時確率分布が計算しやすい形に簡単化される．ここでは文献 [7, 51] に基づいて説明する．Bayesian ネットワークおよび関連アルゴリズムを理解する上で重要な概念に**条件つき独立性** (conditional independence) と d-**分離性** (d-separation) があるが，その詳細は付録で述べることにし，ここでは単結合 Bayesian ネットワークに対する EM アルゴリズムを簡単に述べることにする．

Bayesian ネットワークは 3 つ組 $\langle V_{\rm bn}, L_{\rm bn}, \boldsymbol{\theta}_{\rm bn}\rangle$ で表される．ネットワーク構造 $\langle V_{\rm bn}, L_{\rm bn}\rangle$ は DAG で，$V_{\rm bn} = \{X_1, X_2, \ldots, X_{|V_{\rm bn}|}\}$ は対象の確率現象を表現する確率変数に対応するノード集合である．また，$L_{\rm bn}$ は互いに直接の原因と結果になっている 2 つのノードについて (原因) → (結果) の向きに張られたリンクの集合である[10]．Π_i を X_i の親ノードの集合とする．また，$\boldsymbol{\theta}_{\rm bn}$ は $\{\theta_i(x|\boldsymbol{u}) \mid i = 1 \ldots |V_{\rm bn}|,\ x \in \mathcal{D}(X_i),\ \boldsymbol{u} \in \mathcal{D}(\boldsymbol{\Pi}_i)\}$ の要素を並べたベクトルである．ただし，各 $x_i \in \mathcal{D}(X_i)$, $\boldsymbol{u} \in \mathcal{D}(\boldsymbol{\Pi}_i)$ について

$$\theta_i(x_i|\boldsymbol{u}) \stackrel{\rm def}{=} Pr(X_i = x_i | \boldsymbol{\Pi}_i = \boldsymbol{u}) \tag{2.26}$$

である．特別に X_i が親をもたないノードであるとき ($\Pi_i = \emptyset$)，$\theta_i(x_i|\boldsymbol{u})$ は単に $\theta_i(x_i)$ と書かれる．各 $X_i \in V_{\rm bn}$ について $\theta_i(x_i|\boldsymbol{u})$ ($x_i \in \mathcal{D}(X_i)$, $\boldsymbol{u} \in \mathcal{D}(\boldsymbol{\Pi}_i)$) を格納した表を X_i の**条件つき確率表** (conditional probability table; CPT と略されることがある）と呼ぶ．$\boldsymbol{\theta}_{\rm bn}$ の値は人間が手で与えるか，**証拠** (evidence) と呼ばれる観測データから最尤推定することによって得られる．

このように定められた Bayesian ネットワークが表現する $V_{\rm bn} = \{X_1, X_2, \ldots, X_{|V_{\rm bn}|}\}$ の同時分布 $Pr(X_1, X_2, \ldots, X_{|V_{\rm bn}|})$ は次のように簡単化される．

$$Pr(X_1 = x_1, X_2 = x_2, \ldots, X_{|V_{\rm bn}|} = x_{|V_{\rm bn}|}) = \prod_{i=1}^{|V_{\rm bn}|} Pr(X_i = x_i | \boldsymbol{\Pi}_i = \boldsymbol{u}_i) = \prod_{i=1}^{|V_{\rm bn}|} \theta_i(x_i|\boldsymbol{u}_i) \tag{2.27}$$

(ただし $i = 1 \ldots |V_{\rm bn}|$ について $x_i \in \mathcal{D}(X_i)$ であり，$\boldsymbol{u}_i$ には対応する x_j ($x_j \in \mathcal{D}(X_j)$, $X_j \in \Pi_i$) が含まれる)．例えば，図 2.5 の Bayesian ネットワークにおいて $a, b, \ldots$ をそれぞれ $A, B, \ldots$ の

[8] 信念ネットワーク (belief network) など他にも多くの名称が与えられている．

[9] joint distribution は結合分布と訳されることもある．

[10] 2 つの変数が（間接的に）原因と結果の関係にあるかどうかはリンクをたどることで（d-分離性を満たすかどうかによって）判定される．

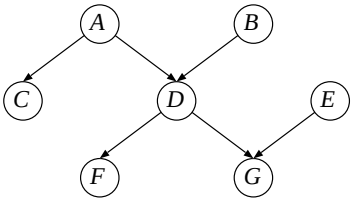

図 2.5: Bayesian ネットワークの例.

実現値（$a \in \mathcal{D}(A), b \in \mathcal{D}(B), \ldots$）とすれば，同時分布 $Pr(a,b,c,d,e,f,g)$ は次のように簡単化される：

$$\begin{aligned} Pr(a,b,c,d,e,f,g) &= Pr(a)Pr(b)Pr(c|a)Pr(d|a,b)Pr(e)Pr(f|d)Pr(g|d,e) \\ &= \theta_A(a)\theta_B(b)\theta_C(c|a)\theta_D(d|a,b)\theta_E(e)\theta_F(f|d)\theta_G(g|d,e). \end{aligned} \tag{2.28}$$

2.1.4.1 π-λ 計算法

Bayesian ネットワーク用 EM アルゴリズムを記述する前に，単結合 Bayesian ネットワーク[11]に対する効率のよい計算法として知られる Pearl による π 確率，λ 確率に基づく計算法 [43] を導入する[12]．本論文ではこの計算法を π-λ **計算法**と呼ぶ.

π-λ 計算の目標である条件つき確率分布 $Pr(X_i|\boldsymbol{E}=\boldsymbol{e})$ の計算は，各 $x_i \in \mathcal{D}(X_i)$ について $Pr(X_i=x_i|\boldsymbol{E}=\boldsymbol{e}) = Pr(x_i|\boldsymbol{e})$ を求めることにより実現される．$Pr(x_i|\boldsymbol{e})$ は次のように $Pr(x_i,\boldsymbol{e})$ を**正規化**することによって得られる：

$$Pr(x_i|\boldsymbol{e}) = \alpha Pr(x_i,\boldsymbol{e}). \tag{2.29}$$

ここで α は正規化定数である[13]．そして，$E^+_{X_i}$ ($E^-_{X_i}$) を X_i がその親（子）を通してアクセスできる E の部分集合とする ($E = E^+_{X_i} \cup E^-_{X_i}$)．それぞれの実現値を $\boldsymbol{e}^+_{X_i}$, $\boldsymbol{e}^-_{X_i}$ とおくと，式 2.29 右辺の $Pr(x_i,\boldsymbol{e})$ は条件つき確率の定義より $Pr(x_i,\boldsymbol{e}) = Pr(x_i,\boldsymbol{e}^+_{X_i}) \cdot Pr(\boldsymbol{e}^-_{X_i}|x_i)$ と分解される．ここで，

$$\pi_{X_i}(x_i) \stackrel{\text{def}}{=} Pr(x_i,\boldsymbol{e}^+_{X_i}) \tag{2.30}$$

$$\lambda_{X_i}(x_i) \stackrel{\text{def}}{=} Pr(\boldsymbol{e}^-_{X_i}|x_i) \tag{2.31}$$

を導入し，それぞれ X_i の π **確率**，λ **確率**と呼ぶ．すると，$Pr(x_i,\boldsymbol{e}) = \pi_{X_i}(x_i)\lambda_{X_i}(x_i)$ が成り立つ．そして，単結合ネットワークでは X_i から放射状にネットワークを分割していくことができるので（図 2.6 参照），条件つき確率 $Pr(x_i|\boldsymbol{e})$ の再帰的計算アルゴリズムが図 2.7 のように導出される．図 2.6 を見ると分かるように，この再帰的計算は X_i に近いノードから段階的に，かつ放射状に進んでいく．

[11] ある DAG において，任意の 2 ノード間の無向パスが一意に定まるとき，その DAG は単結合であるという.

[12] 本節では Castillo ら [7] に従い，Pearl が π 確率に与えた解釈を若干変更した π-λ 計算法を記述する（変更点および変更した理由は後述する）．また，Pearl は forward-chaining に基づく**証拠伝搬** (propagation of evidence) という計算法を提案しているが，ここで記述するのは Russell らの示した backward-chaining に基づく計算法 [51] である．すなわち，ここで述べる π-λ 計算では，π 確率の解釈は Castillo らに従い，入出力および計算手順は Russell らに従う.

[13] $x_i \in \mathcal{D}(X_i)$ について $Pr(x_i,\boldsymbol{e})$ をすべて計算し，次に $\alpha = 1/(\sum_{x_i \in \mathcal{D}(X_i)} Pr(x_i,\boldsymbol{e}))$ を計算する．$\alpha = Pr(\boldsymbol{e})$ であるので $Pr(x_i|\boldsymbol{e}) = \alpha Pr(x_i,\boldsymbol{e}) = Pr(x_i,\boldsymbol{e})/Pr(\boldsymbol{e})$ である．従って，式 2.29 からは $Pr(x_i|\boldsymbol{e})$ が $Pr(x_i,\boldsymbol{e})$ だけから計算できるように見えるが，実は $x_i \in \mathcal{D}(X_i)$ について $Pr(x_i,\boldsymbol{e})$ をすべて計算していなければならない.

2.1.4.2 Bayesian ネットワーク用 EM アルゴリズム

ここで考える Bayesian ネットワークにおけるパラメータ学習（最尤推定）問題においては，はじめに $V_{\rm bn}$ のうち観測可能な変数の集合 E を固定し，T 回の独立な観測において証拠 $\boldsymbol{E}=\boldsymbol{e}^{(1)}$, $\boldsymbol{E}=\boldsymbol{e}^{(2)}$, $\ldots$, $\boldsymbol{E}=\boldsymbol{e}^{(T)}$ が与えられたとき，尤度 $\prod_{t=1}^{T} Pr(\boldsymbol{E}=\boldsymbol{e}^{(t)} \mid \boldsymbol{\theta}_{\rm bn})$ を最大にするパラメータ $\boldsymbol{\theta}^{*}_{\rm bn}$ を見つける．t 回目の観測における非証拠変数 $X_i \in (V_{\rm bn} \setminus E)$ の実現値 x_i が観測できないため，計算が比較的容易な最尤推定法として EM アルゴリズムが考えられる．Bayesian ネットワーク用の EM アルゴリズムでは，Baum-Welch アルゴリズムや Inside-Outside アルゴリズム同様，はじめに適切な初期パラメータ値 $\boldsymbol{\theta}^{(0)}_{\rm bn}$ を与え，ノード X_i の親 $\boldsymbol{\Pi}_i$ が実現値 $\boldsymbol{u}$ をとり，X_i が x_i をとる回数の期待値 $\eta(X_i=x_i, \boldsymbol{\Pi}_i=\boldsymbol{u})$ を式 2.32 で計算し，その期待値を使って式 2.33 でパラメータを更新する．この更新を対数尤度が収束するまで繰り返し，収束したらその時のパラメータ値を推定値 $\boldsymbol{\theta}^{*}_{\rm bn}$ として終了する．

$$\eta^{(m)}(X_i=x_i, \boldsymbol{\Pi}_i=\boldsymbol{u}) \quad := \quad \sum_{t=1}^{T} Pr(X_i=x_i, \boldsymbol{\Pi}_i=\boldsymbol{u} | \boldsymbol{E}=\boldsymbol{e}^{(t)}, \boldsymbol{\theta}^{(m)}_{\rm bn}) \tag{2.32}$$

$$\theta^{(m+1)}_{X_i}(x_i|\boldsymbol{u}) \quad := \quad \frac{\eta^{(m)}(X_i=x_i, \boldsymbol{\Pi}_i=\boldsymbol{u})}{\sum_{x'_i \in \mathcal{D}(X_i)} \eta^{(m)}(X_i=x'_i, \boldsymbol{\Pi}_i=\boldsymbol{u})} \tag{2.33}$$

特に，単結合 Bayesian ネットワークにおいては式 2.32 中の $Pr(X_i=x_i, \boldsymbol{\Pi}_i=\boldsymbol{u} | \boldsymbol{E}=\boldsymbol{e}^{(t)}, \boldsymbol{\theta}^{(m)}_{\rm bn})$ を π-λ 計算法に基づいて式 2.34 のように計算する．パラメータ $\boldsymbol{\theta}^{(m)}_{\rm bn}$ と添字 $^{(t)}$ は省略しており，α は正規化定数である．また，再計算しなくて済むように途中の計算結果は保存しておく．

$$Pr(X_i=x_i, \boldsymbol{\Pi}_i=\boldsymbol{u} | \boldsymbol{E}=\boldsymbol{e}) \quad = \quad \alpha \lambda_{X_i}(x_i) \theta_{X_i}(x_i|\boldsymbol{u}) \prod_{j=1}^{M} \rho_{U_j \setminus X_i}(u_j). \tag{2.34}$$

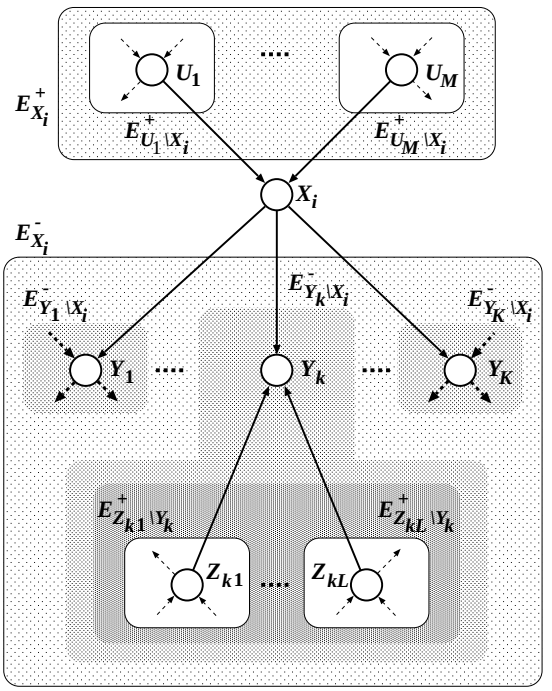

図 2.6: 証拠集合 E を分割する様子.

左辺を右辺で置き換えて計算する．ただし，$U_1, U_2, \ldots, U_M$ は X_i の親，$Y_1, Y_2, \ldots, Y_K$ は X_i の子，また各 $k = 1 \ldots K$ について，$Z_{k1}, Z_{k2}, \ldots, Z_{kN}$ は X_i を除く Y_k の親である．U_j, Y_k, $Z_{k\ell}$ の実現値を u_j, y_k, $z_{k\ell}$ と書く ($j = 1 \ldots M$, $k = 1 \ldots K$, $\ell = 1 \ldots N$)．

$$
\begin{aligned}
Pr(x_i|\boldsymbol{e}) &:= \alpha \rho_{X_i}(x_i) \\
\rho_{X_i}(x_i) &:= \pi_{X_i}(x_i)\lambda_{X_i}(x_i) \\
\pi_{X_i}(x_i) &:= \begin{cases} 0 & \text{if } X_i \in E \text{ かつ } X_i{=}x_i \text{ が } \boldsymbol{E}{=}\boldsymbol{e} \text{ に出現しない,} \\ \theta_i(x_i) & \text{if } X_i \text{ は根ノード,} \\ \displaystyle\sum_{\boldsymbol{u}=\langle u_1,\ldots,u_M\rangle} \theta_i(x_i|\boldsymbol{u}) \prod_{j=1}^{M} \rho_{U_j \setminus X_i}(u_j) & \text{otherwise} \end{cases} \\
\lambda_{X_i}(x_i) &:= \begin{cases} 1 & \text{if } X_i \text{ が葉ノード,} \\ 0 & \text{if } X_i \in E \text{ かつ } X_i{=}x_i \text{ が } \boldsymbol{E}{=}\boldsymbol{e} \text{ に出現しない,} \\ \displaystyle\prod_{k=1}^{K} \lambda_{Y_k \setminus X_i}(x_i) & \text{otherwise} \end{cases} \\
\rho_{X_i \setminus Y_k}(x_i) &:= \begin{cases} \pi_{X_i}(x_i) & \text{if } X_i \text{ が葉ノード,} \\ \pi_{X_i}(x_i) \displaystyle\prod_{k':1 \le k' \le K, k' \ne k} \lambda_{Y_{k'} \setminus X_i}(x_i) & \text{otherwise} \end{cases} \\
\lambda_{Y_k \setminus X_i}(x_i) &:= \begin{cases} \displaystyle\sum_{y_k} \lambda_{Y_k}(y_k)\theta_{Y_j}(y_j|x_i) & \text{if } X_i \text{ が } Y_k \text{ の唯一の親,} \\ \displaystyle\sum_{y_k} \lambda_{Y_k}(y_k) \sum_{\boldsymbol{z}_k=\langle z_{k1}, z_{k2}, \ldots, z_{kN}\rangle} \theta_{Y_j}(y_j|x_i, \boldsymbol{z}_k) \prod_{\ell=1}^{N} \rho_{Z_{k\ell} \setminus Y_k}(z_{k\ell}) & \text{otherwise} \end{cases}
\end{aligned}
$$

α は正規化定数である．

図 2.7: 単結合 Bayesian ネットワークにおける条件つき確率計算アルゴリズム．

2.1.4.3 Bayesian ネットワーク用 EM アルゴリズムの計算量

まず，単結合 Bayesian ネットワーク用 EM アルゴリズムの計算量を評価する．はじめに図 2.7 の条件つき確率計算アルゴリズムの計算量について考える．通常，暗黙のうちに行われているように，すべてのノード X_i ($i = 1 \ldots |V_{\rm bn}|$) に対して，その条件つき確率表 (CPT) のサイズがネットワーク中のノード数 $|V_{\rm bn}|$ と無関係な c 以下であると仮定する．すなわち，親ノード数，各変数がとり得る値の個数 $|\mathcal{D}(X_i)|$ ともに $|V_{\rm bn}|$ に対して定数オーダであるとする．このとき，図 2.7 のアルゴリズムは各ノードを 1 回ずつ X_i から放射状に訪問した後に終了するので，図 2.7 の計算量は $O(|V_{\rm bn}|)$ である．単結合 Bayesian ネットワークに対する EM アルゴリズムを考えると，式 2.34 の後は図 2.7 のアルゴリズムに従うため，式 2.34 の計算も $O(|V_{\rm bn}|)$ 時間で済む．更新に要する時間も（CPT のサイズが定数オーダであることから）$O(|V_{\rm bn}|)$ である．以上より，単結合 Bayesian ネットワークに対する EM アルゴリズムの計算量は $O(|V_{\rm bn}|)$ であることが分かった．

一方，単結合でないネットワーク，すなわち多重結合 (multiply-connected) ネットワークでは，パスが一意に定まらないようなノード対が存在する．従って，図 2.7 のような再計算が不要な放射状の再帰計算手続きを構築することができない．このような多重結合 Bayesian ネットワークは一般には $|V_{\rm bn}|$ に対して指数オーダである．

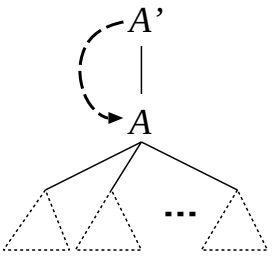

図 2.8: 擬似 PCSG における文脈 $A' = C(\boldsymbol{r}^{\mathrm{h}})$.

2.1.5 PCFG の拡張文法

PCFG はその名の通り，前の文脈とは無関係に句構造を決めてしまうため，自然言語などの複雑な（文脈依存性が要求されるような）確率現象を表現するモデルとしては問題が多いことが指摘されてきた．従って，自然言語処理などの応用においては文脈依存性を導入したモデルを考える必要があるが，Chomsky 階層の 1 型文法である文脈依存文法（context-sensitive grammar; 以下 CSG）に確率を導入するのは，計算機への実装や実行時間を考えると現実的ではない．そこで，文の生成手続きは CFG と同じであるが，規則の選択確率の分布に文脈依存性をもたせるような統計モデルを考える．すなわち，文脈依存文法のように文法構造によって文脈依存性を考慮する代わりに，統計モデルが表現する確率分布によって文脈依存性をもたせることを考える．本論文ではこのような統計モデルを扱うことにし，以降これらを「PCFG の拡張文法」と呼ぶ[14]．どのような文脈依存性をもたせるかについて多様性があるため，これまで数多くのモデルが提案されてきた．本節では最も簡単なモデルとして Kita らの規則 bigram モデル [24] と Charniak らの**擬似確率文脈依存文法**（pseudo probabilistic context-sensitive grammar; 以下，擬似 PCSG） [9] を紹介する．更に，いくつかの研究者によって提案されている PCFG に bigram を導入したモデルについても述べる．

後の説明を容易にするため，はじめに PCFG の拡張文法を形式化する．PCFG の拡張文法の文法構造は PCFG と同様，CFG $\langle V_{\mathrm{n}}, V_{\mathrm{t}}, R, S\rangle$ である．また，PCFG でも仮定したように，導出戦略を最左導出に固定する．PCFG における導出においては A を展開するときに規則 $r = (A \rightarrow \zeta)$ を確率 $\theta_{\mathrm{pcfg}}(r)$ で選択していたが，PCFG の拡張文法では確率 $\theta_{\mathrm{pcfg+}}(r|C(\boldsymbol{r}^{\mathrm{h}}))$ で選択することにする．ただし，$\boldsymbol{r}^{\mathrm{h}}$ はそれまでに適用された規則の系列（以下，**規則適用履歴**という）を表し[15]，$C(\boldsymbol{r}^{\mathrm{h}})$ は規則適用履歴 $\boldsymbol{r}^{\mathrm{h}}$ から一意に定まる**文脈**を表す．$\boldsymbol{r}^{\mathrm{h}}$ が空であるときは，モデル毎にあらかじめ固定された任意の記号 # を使って $C(\boldsymbol{r}^{\mathrm{h}}) = \#$ と定義する．規則適用履歴 $\boldsymbol{r}_1^{\mathrm{h}}$ と $\boldsymbol{r}_2^{\mathrm{h}}$ において A に関する規則の選択分布 $\theta_{\mathrm{pcfg+}}(A \rightarrow \zeta|C(\boldsymbol{r}_1^{\mathrm{h}}))$ と $\theta_{\mathrm{pcfg+}}(A \rightarrow \zeta|C(\boldsymbol{r}_2^{\mathrm{h}}))$ に十分な違いがあれば，この拡張文法は A の展開において 2 つの文脈 $c_1 = C(\boldsymbol{r}_1^{\mathrm{h}})$ と $c_2 = C(\boldsymbol{r}_2^{\mathrm{h}})$ の違いを捉えているといえる．

関数 C の選択にモデル設計者の意図が反映される．Kita らの提案した規則 bigram モデルは規則適用履歴を $\boldsymbol{r}^{\mathrm{h}} = r_1 r_2 \cdots r_k$ とおいたときに $C(\boldsymbol{r}^{\mathrm{h}}) = r_k$ としている．すなわち規則 bigram モデルは最左導出において直前に適用した規則を文脈とするモデルである．また，我々は最左導出に固定しているので，規則適用履歴 $\boldsymbol{r}^{\mathrm{h}}$ からそれまでに構築された部分構文木が一意に $\boldsymbol{t}^{\mathrm{h}}$ が定まる．Charniak らの提案した擬似 PCSG では，この部分構文木 $\boldsymbol{t}^{\mathrm{h}}$ において次の展開対象である非終端記号 A の親ノードのカテゴリ A' を $C(\boldsymbol{r}^{\mathrm{h}})$ にしている（図 2.8）．

[14] Magerman らはこのような統計モデルを “Context-free grammars with context-sensitive probability (CFGs with CSP)” と呼んでいる [32].
[15] $\boldsymbol{r}^{\mathrm{h}}$ の h は history の頭文字である．

```
1: procedure GET-BETA+() begin
2:     for t := 1 to T do begin
3:         β^(t)_{d,d'}(A|A') := 0 for each A, A' ∈ V_n and d, d' such that 0 ≤ d < d' ≤ L_t;
4:         for d := 0 to L_t − 1 do     /* 三角行列の対角要素について計算 */
5:             foreach A such that (A→w^(t)_{d+1}) ∈ R do
6:                 foreach A' ∈ V_n do
7:                     β^(t)_{d,d+1}(A|A') := θ(A→w^(t)_{d+1}|A');
8:         for k := 2 to L_t do     /* 三角行列の非対角要素について計算 */
9:             for d := 0 to L_t − k do
10:                foreach A, A' ∈ V_n do
11:                    β^(t)_{d,d+k}(A|A') := Σ_{B,C:(A→BC)∈R} θ(A→BC|A') Σ_{k'=1}^{k−1} β^(t)_{d,d+k'}(B|A) β^(t)_{d+k',d+k}(C|A);
12:    end
13: end.
```

図 2.9: 擬似 PCSG の内側確率の計算ルーチン GET-BETA$^+$.

PCFG の場合と同様，拡張文法 $G(\boldsymbol{\theta}_{\mathrm{pcfg}^+})$ とコーパス（例文の集合）$\langle \boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_T \rangle$ が与えられたとき，最尤推定値 $\boldsymbol{\theta}^*_{\mathrm{pcfg}^+}$ を見つける問題においては EM アルゴリズムが有効である．ただし，文脈 $C(\boldsymbol{r}^{\mathrm{h}})$ の与え方に多様性があるため，PCFG の拡張文法の EM アルゴリズムを一般形で書くことは難しい．本節では擬似 PCSG 用 EM アルゴリズムを簡単に述べる．

擬似 PCSG 用 EM アルゴリズムは Inside-Outside アルゴリズム（節 2.1.5）を素直に拡張したものである[16]．まず，内側確率と外側確率を親ノード A' に応じて細分化する．

$$\beta^{(t)}_{d,d'}(A|A') \stackrel{\mathrm{def}}{=} Pr(A \stackrel{*}{\Rightarrow} \boldsymbol{w}^{(t)}_{d,d'} | A' \text{ は } A \text{ の親}) \tag{2.35}$$

$$\alpha^{(t)}_{d,d'}(A|A') \stackrel{\mathrm{def}}{=} Pr(S \stackrel{*}{\Rightarrow} \boldsymbol{w}^{(t)}_{0,d} A \boldsymbol{w}^{(t)}_{d',L_t} | A' \text{ は } A \text{ の親}) \tag{2.36}$$

($t = 1 \ldots T, 0 \le d < d' \le L_t$ かつ $A, A' \in V_{\mathrm{n}}$)．図 2.9, 図 2.10 に示す手続き GET-BETA$^+$ および GET-ALPHA$^+$ は，それぞれ Inside-Outside アルゴリズムのサブルーチン GET-BETA, GET-ALPHA の拡張である．Inside-Outside アルゴリズム同様，GET-BETA$^+$ と GET-ALPHA$^+$ によって計算された内側確率，外側確率を使って，親が A' のときに規則 $A \to \zeta$ を適用した回数の期待値 $\eta(A \to \zeta | A')$ を

$$\eta(A \to BC|A') := \sum_{t=1}^{T} \frac{1}{\beta^{(t)}_{0,L_t}(S)} \sum_{k=2}^{L_t} \sum_{d=0}^{L_t-k} \sum_{k'=1}^{k-1} \theta(A \to BC|A') \cdot \alpha^{(t)}_{d,d+k}(A|A') \beta^{(t)}_{d,d+k'}(B|A) \beta^{(t)}_{d+k',d+k}(C|A) \tag{2.37}$$

$$\eta(A \to a|A') := \sum_{t=1}^{T} \frac{1}{\beta^{(t)}_{0,L_t}(S)} \sum_{d=0}^{L_t-1} \theta(A \to a|A') \alpha^{(t)}_{d,d+1}(A|A') \tag{2.38}$$

と計算する．その後，各非終端記号 A, A' についてパラメータ $\theta(A \to \zeta | A')$ を

$$\theta(A \to \zeta|A') := \eta(A \to \zeta|A') / \textstyle\sum_{\zeta':(A\to\zeta')\in R} \eta(A \to \zeta'|A') \tag{2.39}$$

によって更新する．擬似 PCSG 用の EM アルゴリズムの計算量は上より明らかに $O(|V_{\mathrm{n}}|^4 L^3 T)$ である[17]．すなわち，親ノード A' について細分化を図った分だけ計算時間が必要になる．

[16] Charniak らも擬似 PCSG 用の EM アルゴリズムを導出しているが [9]，式の形が異なる．彼らはアルゴリズムを完全に記述していないので，ここで記述するアルゴリズムと単純に比較できない．

[17] Charniak らは計算量については評価していない．

1: **procedure** GET-ALPHA$^+$() **begin**
2: **for** $t := 1$ **to** T **do begin**
3: $\alpha^{(t)}_{d,d'}(A|A') := 0$ for each $A, A' \in V_{\rm n}$ and d, d' such that $0 \leq d < d' \leq L_t$;
4: $\alpha^{(t)}_{0,L_t}(S|\#) := 1$; /* ただし，最も右上の S については特別に 1 に初期化 */
5: **for** $k := L_t$ **downto** 2 **do**
6: **for** $d := 0$ **to** $L_t - k$ **do**
7: **foreach** $A, B \in V_{\rm n}$ **do**
8: $\alpha^{(t)}_{d,d+k}(B|A) :=$
9: $$\sum_{A' \in V_{\rm n}} \sum_{X:(A \to BX) \in R} \theta(A \to BX|A') \sum_{k'=k+1}^{L_t - d} \alpha^{(t)}_{d,d+k'}(A|A') \beta^{(t)}_{d+k,d+k'}(X|A)$$
10: $$+ \sum_{A' \in V_{\rm n}} \sum_{Y:(A \to YB) \in R} \theta(A \to YB|A') \sum_{k'=1}^{d} \alpha^{(t)}_{d-k',d+k}(A|A') \beta^{(t)}_{d-k',d}(Y|A)$$
11: **end**
12: **end**.

図 2.10: 擬似 PCSG の外側確率の計算ルーチン GET-ALPHA$^+$.

表 2.1: 専用 EM アルゴリズムの計算量.

モデルクラス	計算量	
HMM	$O(\|S_{\rm hmm}\|^2 LT)$	$\|S_{\rm hmm}\|$: 状態数，L: 記号列長，T: データ数
PCFG	$O(\|V_{\rm n}\|^3 L^3 T)$	$\|V_{\rm n}\|$: 非終端記号数，L: 最大文長，T: データ数
Bayesian ネットワーク	$O(\|V_{\rm bn}\|T)$	$\|V_{\rm bn}\|$: ノード数，T: データ数
擬似 PCSG	$O(\|V_{\rm n}\|^4 L^3 T)$	$\|V_{\rm n}\|$: 非終端記号数，L: 最大文長，T: データ数

2.1.6 まとめ

隠れマルコフモデル (HMM)，確率文脈自由文法 (PCFG) およびその拡張文法，Bayesian ネットワークの EM アルゴリズムを記述した．いずれのアルゴリズムにおいても，モデル構造を 2 つに分割し，前向き確率・後向き確率，内側確率・外側確率，π 確率・λ 確率という 2 つの確率値を別々に計算し，後にそれらを掛け合わせパラメータ更新に必要な期待値を求める点が共通している．これは**分割統治法** (divide-and-conquer) の一種であると見ることができる．そして各々の確率値はある一方向に沿って段階的に計算される．このように一方向に沿って，すなわち後戻りせずに計算することによって再計算を防ぐ方法は**動的計画法** (dynamic programming) と呼ばれ，効率的なアルゴリズムを構築する上での基本的な考え方である．効率的な EM アルゴリズムを構築するには，分割統治法と動的計画法という 2 つの考えを採り入れる必要がある．本節で述べた各統計モデルクラス専用の EM アルゴリズムの計算量を表 2.1 にまとめておく．

2.2 既存の記号的統計モデル間の関係

本節では，先に述べた既存の記号的統計モデル（HMM, PCFG, Bayesian ネットワーク，PCFG の拡張文法）について，その表現上の関係および EM 学習における計算効率の関係について述べるが，次の点に注意しておく：一般に，有限個の確率変数に対し，Bayesian ネットワークはその任意の同時分布を表現できる．従って，対象ドメインの確率現象が有限個の確率変数 $X_1, X_2, \ldots, X_n$ で

表わされる（すなわちそのドメインが有限である）ならば，HMM, PCFG もしくはその拡張文法が表現する確率分布はどれも Bayesian ネットワークが表現する同時分布 $Pr(X_1, X_2, \ldots, X_n)$ と対応づけ可能である．すなわち，有限ドメインにおいては Bayesian ネットワークは HMM, PCFG およびその拡張文法の一般化になっている．また，このような知見のもと，HMM, PCFG およびその拡張文法を Bayesian ネットワークで記述する試みがいくつか為されており [4, 15, 47, 65]，以下の該当する節ではこれらの研究についても簡単に述べておく．

2.2.1 HMM と PCFG およびその拡張文法の関係

CFG の確率化である PCFG が正規文法の確率化である HMM の表現上の一般化になっているのは明らかである．しかし，PCFG 専用 EM アルゴリズムである Inside-Outside アルゴリズムでは Chomsky 標準形を満たさない PCFG は扱うことができない．HMM を正規文法の形で書き下しても，それは Chomsky 標準形でないので Inside-Outside アルゴリズムでは扱えないということになる．一つの解決として Chomsky 標準形変換アルゴリズム [25] によって HMM を Chomsky 標準形の PCFG に変換することはできるが，それに対する計算量は $O(L^3)$ になってしまう（L は文長・記号列長）．それに対し，HMM 専用 EM アルゴリズムである Baum-Welch アルゴリズムの計算量は $O(L)$ であるから，定義 2 の基準において Inside-Outside アルゴリズムは Baum-Welch アルゴリズムの一般化にはなっていない．

PCFG の拡張文法における文脈 $C(\boldsymbol{r}^{\mathrm{h}})$ を定数にすれば，PCFG とその拡張文法が同じ確率分布を表現するのは明らかである．そして，節 2.1.5 で説明した擬似 PCSG 用の EM アルゴリズムにおいて，文脈 $A' = C(\boldsymbol{r}^{\mathrm{h}})$ の **foreach** ループを取り除くと，それは Inside-Outside アルゴリズムと一致する．従って，PCFG の拡張文法は PCFG の表現上の一般化になっており，PCFG の拡張文法の一クラスである擬似 PCSG 用の EM アルゴリズムは Inside-Outside アルゴリズムの一般化になっている．

2.2.2 Bayesian ネットワークを用いた HMM の確率計算

ある長さ L 以下の文字列を受理する HMM は図 2.11 の Bayesian ネットワークで表現可能である[18]．各時刻 $\ell = 1 \ldots L$ において，Q_ℓ は ℓ における状態 $q_\ell \in S_{\mathrm{hmm}}$ を値にとる確率変数，O_ℓ は時刻 ℓ に出力する記号 $o_\ell \in V_{\mathrm{hmm}}$ を値にとる確率変数である．また，各 O_ℓ, $Q_{\ell'}$ ($\ell = 1 \ldots L$, $\ell' = 2 \ldots L$) に付与されている条件つき確率表 $\theta_{O_\ell}(o_\ell|q_\ell)$ と $\theta_{Q_{\ell'+1}}(q_{\ell'+1}|q_{\ell'})$ は各 ℓ, ℓ' で共通であるとする．Bayesian ネットワークの形状より，次の条件つき独立性がいえる：

- Q_ℓ が与えられたとき，$Q_{\ell'}$, $O_{\ell'}$ ($\ell' = 1 \ldots \ell - 1$) は O_ℓ と条件つき独立であり，かつ
- Q_ℓ が与えられたとき，$Q_{\ell'}$, $O_{\ell'}$ ($\ell' = 1 \ldots \ell - 1$) は $Q_{\ell+1}$ と条件つき独立である．

これは HMM における独立性の仮定に他ならず，従って図 2.11 の Bayesian ネットワークは HMM と同じ確率分布を表現している．一般に HMM は無限長の文字列を生成しうるが，文字列長の上限 T を固定すれば HMM を Bayesian ネットワークによって表現することができる．従って定義 1 より，Bayesian ネットワークは HMM の表現上の一般化になっている．

また，図 2.11 より明らかなように HMM を表現する Bayesian ネットワークは単結合であるので，Pearl の π-λ 計算法（節 2.1.4.1）が適用できる．その結果，節 2.1.4.2 で示した Bayesian ネッ

[18] 図 2.11 において，確率変数 Q_ℓ, O_ℓ はそれぞれ Q(ℓ), O(ℓ) と書かれている．

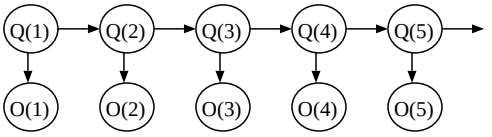

図 2.11: Bayesian ネットワークによって表現される HMM.

トワーク用 EM アルゴリズムが HMM 用の EM アルゴリズムである Baum-Welch アルゴリズム（節 2.1.2.1）と等価であり，かつ同じ計算オーダであることはことは容易に確認できる．以上と定義 2 より，Bayesian ネットワーク用の EM アルゴリズムは Baum-Welch アルゴリズムの一般化になっていることが分かる．

2.2.3 動的 Bayesian ネットワーク

図 2.11 の Bayesian ネットワークを発展させたものが 動的 Bayesian ネットワーク（dynamic Bayesian network; 以下，DBN） [4, 15, 51, 65] である．DBN ははじめ Dean と Kanazawa によって用いられ [15]，最近では AI の標準的教科書 [51] でも紹介されている統計モデルクラスである．HMM の一般化となっていることから，最近では DBN を音声認識に用いる試みもある [65].

DBN は 1 個の Bayesian ネットワークのコピーを時系列に従って連結したものである．各時刻 $\ell = 1 \ldots L$ の Bayesian ネットワークはスライス (slice) と呼ばれる．条件つき確率表は各スライスで共有される．通常の DBN においては，各スライス中の変数を常に観測可能な変数（証拠変数）と常に観測不可能な変数（状態変数）に分け，異なるスライス間で対応する状態変数同士をリンクで結ぶ．図 2.12（上）に DBN の一般形を示す[19]．State(ℓ) とあるのはスライス ℓ における状態変数の集まりである．一方 Percept(ℓ) はスライス ℓ の証拠変数の集まりである．図 2.12（下）に DBN の例を示す．A(ℓ), B(ℓ) は証拠変数，X(ℓ), Y(ℓ), Z(ℓ) が状態変数である．図 2.11 と比較して分かるように，HMM （を表現した Bayesian ネットワーク）は DBN の特殊形である．

DBN の EM アルゴリズムは，単純に考えると，節 2.1.4.2 の単体 Bayesian ネットワーク用 EM アルゴリズムに Baum-Welch アルゴリズムの考えを導入したものになる．すなわち，証拠変数に証拠を与えた上で，スライス $\ell = 1, 2, \ldots$ の順に前向きの条件つき確率を計算し，次にスライス $\ell = L, L-1, \ldots$ の順に後ろ向きの条件つき確率を計算する．そして前向きの条件つき確率と後ろ向きの条件つき確率を掛け合わせた値を使ってパラメータを更新する．この EM アルゴリズムは（Baum-Welch アルゴリズムのように）スライス数 L に対して $O(L)$ 時間で動作する．ただし，メモリ空間も $O(L)$ だけ必要になる．

応用問題においては DBN では 1 つのスライスに含まれる変数が非常に多くなるため，スライス数 L が大きくなると HMM に比べてメモリ空間が莫大になってしまう．そこで，計算時間を若干犠牲にして使用メモリ空間を抑えるアルゴリズムとして，Russell らは L に対して定数オーダのメモリ空間を使用して $O(L^2)$ 時間で動作するアルゴリズムを提案し [51]，その後 Binder らは $O(\log L)$ のメモリ空間を使用して $O(L \log L)$ 時間で動作するアルゴリズムを提案している [4].

[19] 図 2.12（上）は文献 [4, 51] の引用である．

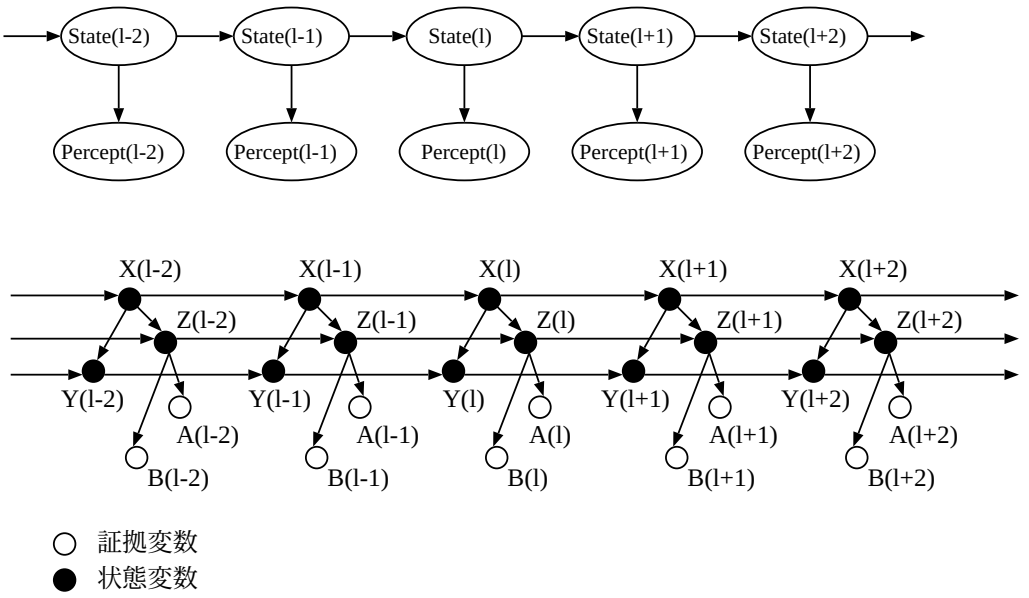

図 2.12: （上）DBN の一般形,（下）DBN の例.

2.2.4 Bayesian ネットワークを用いた PCFG の確率計算

Pydanath と Wellman は，文の生起確率や接頭辞（文頭からの部分文字列）の生起確率など，PCFG によって表現されている様々な（条件つき）確率を Bayesian ネットワークを用いて計算する方法を提案している [47]．彼らの方法では，例えば「文 “Time flies like an arrow” の生起確率を計算せよ」といった質問 (query) に対して，質問に応じた Bayesian ネットワークをはじめに構築し，そしてその Bayesian ネットワークを使って（条件つき）確率の計算を行う．先の HMM と同様，PCFG も無限長の文を生成しうるが，質問対象である文の長さが有限に固定されれば構文木の数および各構文木のサイズも有限になる．よって質問が与えられた下では扱うドメインが有限に制限されるので，無限ドメインを扱う PCFG においても Bayesian ネットワークが適用可能になる．長さ 4 の文に対して構築される Bayesian ネットワークを図 2.13 に示す（この説明においては元文法は重要ではないので省略する）．$N(i,j,k)$ は単語位置 i から始まる長さ j，レベル[20] k の非終端記号が n であるとき，$N(i,j,k)=n$ となる確率変数（ノード）である．$P(i,j,k)$ は $N(i,j,k)$ の非終端記号を展開するときに用いた規則を実現値とする確率変数（ノード）である．

一方，彼らは Charniak らの擬似 PCSG や PCFG + bigram モデルにおける確率計算が彼らの枠組みで可能であることを示した．例えば，図 2.14（左）は擬似 PCSG を表現したもので，親ノードが子ノードの非終端記号の規則選択に影響を与えている（影響を表すリンクが描かれている）．一方，図 2.14（右）は PCFG に bigram 確率を導入したモデル（以降では **PCFG+bigram** モデルと呼ぶ）を表現しており，非終端記号 A を展開するとき A が統治することになる部分木の直前の単語が A の規則選択に影響を与えている（点線のリンク）．Bayesian ネットワークにおける確率計算手続きには変化はない．このように，文脈依存性が簡潔に表現されるのは Bayesian ネットワークの記述力の高さを示すものである．

しかし，Pynadath と Wellman の方法は計算効率に問題がある．最初のステップである Bayesian

[20] おなじ (i,j) が存在するとき，すなわち $A \Rightarrow B$ という unit production が起こるとき，それらを区別するインデックスを彼らは「レベル」と呼んでいる．

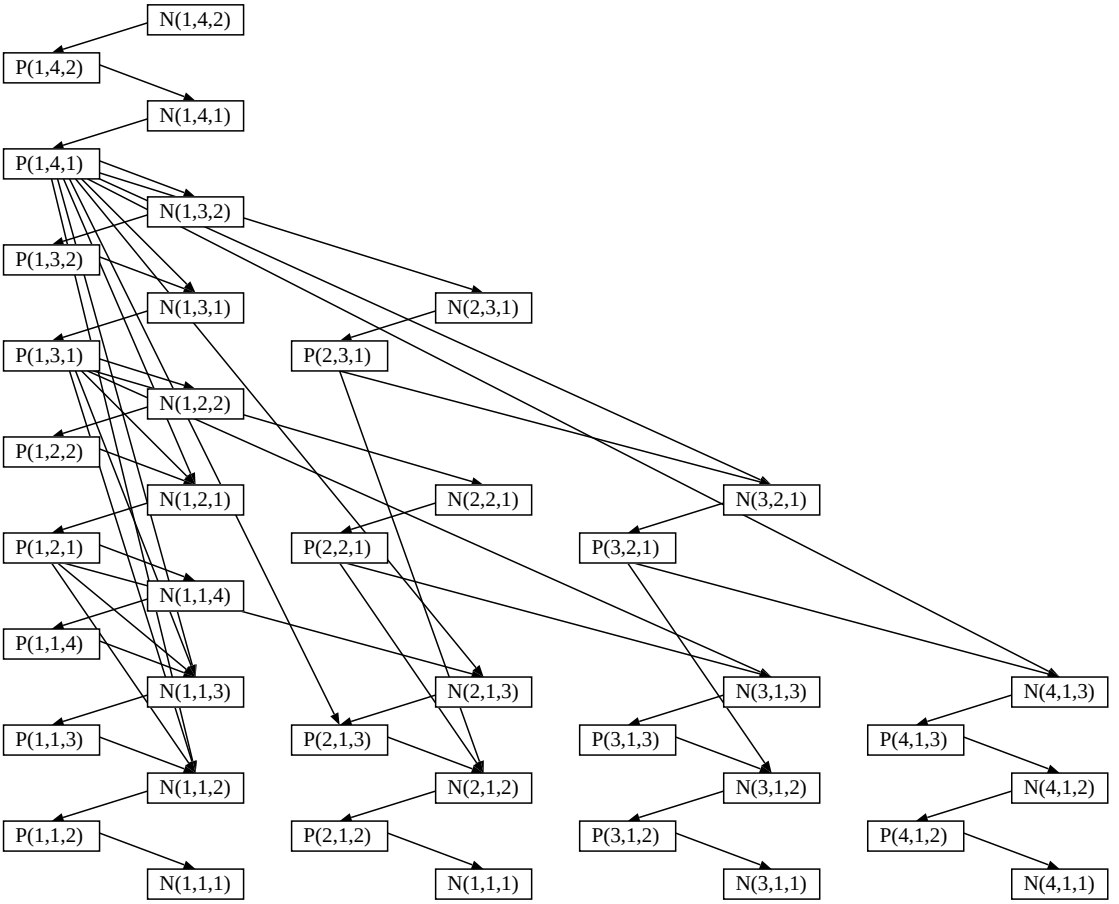

図 2.13: 長さ 4 の文に対して PCFG から構築される Bayesian ネットワーク.

ネットワークの構築は文長に対して多項式時間で可能であるが,(先の HMM の場合とは異なり) 構築された Bayesian ネットワークが単結合でないため，節 2.1.4 で述べた π-λ 計算法が適用できず，2 番目のステップである Bayesian ネットワークを使った確率計算が文長に対して（有限ではあるが）指数オーダになってしまう．文脈依存性を採り入れた場合も同様である．彼らはこれに対する解決法は述べていない．彼らの確率計算方法を利用した PCFG 用の EM アルゴリズムを導出できると思われるが（彼ら自身はその方法については述べていない)，その EM アルゴリズムは 定義 2 の計算量に関する条件を満たしておらず，この意味で彼らの枠組は Inside-Outside アルゴリズムや Charniak らの擬似 PCSG 用 EM アルゴリズムの一般化とはなっていない.

また，最近 Pynadath は確率状態依存文法（probabilistic state dependency grammar; 以下 PSDG）と名付けられたプラン認識用の確率文法を提案し,(PCFG から Bayesian ネットワークが構築されるように）PSDG から DBN （節 2.2.3）を構築し，その上で確率推論する方法を提案している [48]．プラナの心理状態 q が PSDG に導入されており，その心理状態 q を値にもつ確率変数 Q_ℓ $(\ell = 1, 2, \ldots)$ が（構築後の）DBN の状態変数を構成する.

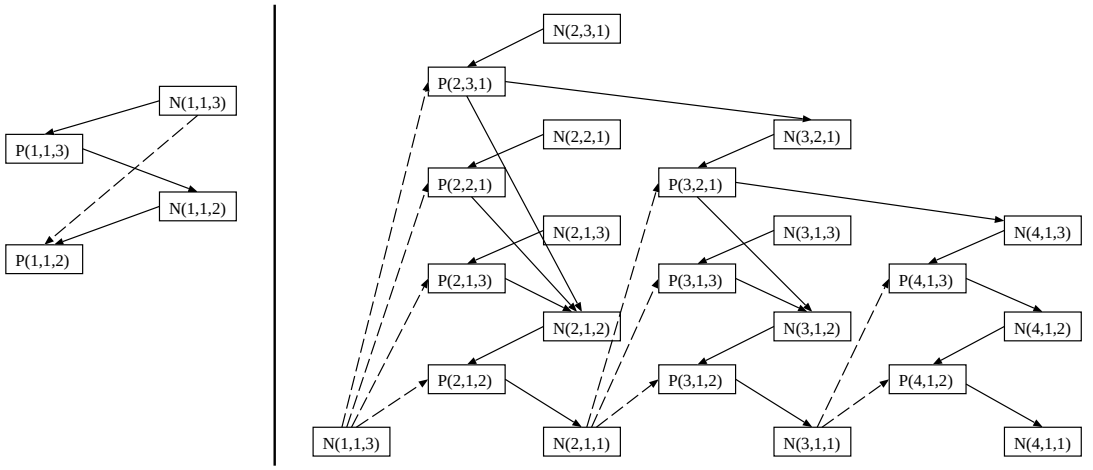

図 2.14: （左）擬似 PCSG と（右）PCFG+bigram モデルに対する Bayesian ネットワーク.

2.2.5 まとめ

広く知られている記号的統計モデルのクラスである隠れマルコフモデル (HMM)，確率文脈自由文法 (PCFG) およびその拡張，Bayesian ネットワークを簡単に形式化し，その EM アルゴリズムを記述した．そしてこれらのモデルクラス間の関係を調べた．その結果，定義 1 の基準における表現力について次の関係が得られた.

$$\text{(HMM)} \prec_{\text{rep}} \text{(PCFG)} \prec_{\text{rep}} \text{(PCFG の拡張文法)}$$
$$\text{(HMM)}, \text{(PCFG)}, \text{(PCFG の拡張文法)} \not\prec_{\text{rep}} \text{(Bayesian ネットワーク)}$$

しかし，有限ドメインにおいては（すなわち有限長の記号列・文のみを考える）次の関係が成り立つ.

$$\text{(HMM)} \prec_{\text{rep}} \text{(PCFG)} \prec_{\text{rep}} \text{(PCFG の拡張文法)} \prec_{\text{rep}} \text{(Bayesian ネットワーク)} \tag{2.40}$$

一方，定義 1 の基準に基づくと，上で記述した各専用 EM アルゴリズム（Baum-Welch アルゴリズム，Inside-Outside アルゴリズム，Charniak の擬似 PCSG 用 EM アルゴリズム，Pynadath らの方法から得られる EM アルゴリズム）の関係を以下にまとめる．我々は有限サイズのデータしか観測できない点に注意する．すなわちドメインは有限であるので，式 2.40 で関係 $\prec_{\text{rep}}$ を満たすモデルクラスの専用 EM アルゴリズムについて比較することができる．上に述べたように次の 2 つが成り立つことが分かった.

$$\text{(Inside-Outside アルゴリズム)} \prec_{\text{EM}} \text{(Charniak らの擬似 PCSG 用 EM アルゴリズム)}$$
$$\text{(Baum-Welch アルゴリズム)} \prec_{\text{EM}} \text{(Bayesian ネットワーク用 EM アルゴリズム)}$$

その一方で，上で見てきたように HMM に対して Inside-Outside アルゴリズムでは Baum-Welch アルゴリズムと同じ計算オーダが得られないことから，定義 2 より

$$\text{(Baum-Welch アルゴリズム)} \not\prec_{\text{EM}} \text{(Inside-Outside アルゴリズム)}$$

となる．また，Pynadath らが提案したのは確率計算法であって EM アルゴリズムではないため，Inside-Outside アルゴリズムおよび擬似 PCSG 用 EM アルゴリズムと Bayesian ネットワーク用

EM アルゴリズムの一般性に関する比較はできない．しかし，Pynadath らの確率計算自体が指数オーダになってしまうので，彼らの方法を EM アルゴリズムに利用したとしても，Inside-Outside アルゴリズムの一般化となるのは難しいと予想される．

以上から分かるように，各統計モデルクラスについて表現上の一般・特殊関係は成り立つこと多いが，ある統計モデルクラス専用 EM アルゴリズムは他クラスの専用 EM アルゴリズムの一般化になっていない場合が多い．

2.3 一階述語論理に基づく統計知識の表現

また，Bayesian ネットワークは HMM, PCFG およびその拡張文法の表現上の一般化となっており，非常に強力な統計モデリングのツールであるが，その問題点は命題論理の表現力しかもたない（従って有限ドメインの確率分布しか表現できない）点にある．例えば,「時刻 ℓ において状態 q_ℓ にあるとき，HMM は確率 $Pr(o_\ell|q_\ell)$ で記号 o_ℓ を出力し，確率 $Pr(q_{\ell+1}|q_\ell)$ で状態 $q_{\ell+1}$ に遷移する」という普遍的統計知識を表現するのに，節 2.2.2 の図 2.11 のように，$2 \times L$ 個の確率変数を用意し，しかも「各 $O_\ell, Q_{\ell'}$ $(\ell = 1 \ldots L, \ell' = 2 \ldots L)$ に付与されている条件つき確率表 $\theta_{O_\ell}(o_\ell|q_\ell)$ と $\theta_{Q_{\ell'+1}}(q_{\ell'+1}|q_{\ell'})$ は各 ℓ, ℓ' で共通である」という約束事を Bayesian ネットワークとは別にユーザが指定しなけなければならない．従って Bayesian ネットワークは図 2.11 のように個別の HMM を表現できるが，その Bayesian ネットワークが HMM の記号列出力器・受理器としての普遍的な性質を直接表現できているとは言い難い．

このような問題は Bayesian ネットワークが「すべての `X` について `p(X)` が成り立つ」,「`q(Y)` が成り立つような `Y` が存在する」といった全称限量子，存在限量子で表現される普遍的な統計知識をコンパクトに表現できないことに起因する．それに対し，このような普遍的知識の表現には古くから一階述語論理が用いられてきた．一階述語論理には全称限量子，存在限量子が備えられており,（可算）無限個のオブジェクトに関する性質，無限個のオブジェクト間の関係をたった一つの規則で表現することができる．従って，普遍的な統計知識が表現できるような一階述語論理に基づく統計知識記述言語を考えるのは自然かつ重要な流れである．これまで一階述語論理に確率要素を導入する研究が数多く行われており，特に最近その進展が目立つ [5, 13, 14, 18, 26, 30, 39, 40, 41, 46, 50, 53]. 中でも，先述したように，確率的意味を宣言的に与えることができる点，パラメータ学習アルゴリズムを備えている点から，本研究では Sato らが提案した PRISM を統計知識の記述言語，すなわち記号的統計モデル言語として採用する．

2.4 研究の目的

先にも述べたように，本論文では，記号的な統計モデリング言語 PRISM (PRogramming In Statistical Modeling) およびその効率的な EM アルゴリズムを提案する．PRISM は上記したような一階述語論理に基づく統計知識記述言語であるが，以下の特長をもつ．

- 最小モデル意味論の確率的な一般化である分布意味論 [53] に基づき，確率的意味が宣言的に規定されている．
- 既存の記号的統計モデルクラスである HMM, PCFG およびその拡張文法，Bayesian ネットワークの表現上の一般化になっている．
- 観測データからプログラム中に付与された確率パラメータを EM 学習できる．

- ある条件を満たすPRISMプログラムにはgEMアルゴリズムを適用できる．gEMアルゴリズムはBaum-Welchアルゴリズム，Inside-Outsideアルゴリズム，単結合Bayesianネットワーク用EMアルゴリズムの一般化となっている．

本論文の成果は4番目の特長であるが，これは定義2より，PRISMでHMM, PCFG, Bayesianネットワークを記述すれば，PRISM用EMアルゴリズムがそれぞれBaum-Welchアルゴリズム，Inside-Outsideアルゴリズム，単結合Bayesianネットワーク用EMアルゴリズムと同じ計算オーダでEM学習できることを意味する．このように一般性と効率性を同時に実現するというのが，既存の統計知識記述言語には見られない特徴である．

次章以降では，PRISMの統語論と意味論を示し，効率的なその専用EMアルゴリズムを記述する．そして，PRISMの表現力とそのEMアルゴリズムの効率性について考察を行う．そして，現実の言語データを用いてEMアルゴリズムが既存のInside-Outsideアルゴリズムより高速に動作する場合があることを実験的に示す．

2.5 第 2 章のまとめ

本章では，研究の背景と目的について述べた．節2.1では，はじめに複数の統計モデルクラス間での表現力の関係，および各モデルのEMアルゴリズム間でのEM学習の関係を定義し，Dempsterらが与えたEMアルゴリズムの一般形を記述した．そして，隠れマルコフモデル，確率文脈自由文法およびその拡張文法,（単結合）Baysianネットワークを紹介し，各々専用のEMアルゴリズムを形式的に説明した．また，これらの専用EMアルゴリズムに共通して見られる特徴として，2種類の確率値を別々に計算し，後にその積をとる一種の分割統治が行なわれていること，および各々の確率値は動的計画法に基づいて計算されていることを挙げ，高速なEMアルゴリズムを考案する上ではこれらの特徴を採り入れる必要があると述べた．次いで節2.2では，これらの統計モデルクラス間の表現上の一般・特殊関係と各専用EMアルゴリズムの計算効率の関係について述べた．そして，各統計モデルクラスについて表現上の一般・特殊関係は成り立つことが多いが，ある統計モデルクラス専用EMアルゴリズムは他クラスの専用EMアルゴリズムの一般化になっていない場合が多いことを指摘した．そして節2.3で，普遍的統計知識の記述には一階述語論理に基づく表現言語が適していると述べた．本研究の目的は前章で述べたが，本章最後の節2.4で改めて与えた．

第3章　記号的統計モデル言語 PRISM とその表現力

本章では確率的要素をもつ論理プログラミング言語である PRISM を導入する．そして有用とされている記号的統計モデルを PRISM プログラムとして記述し，PRISM プログラムがこれらの記号的統計モデルに対して表現上の一般化になっていることを示す．はじめに PRISM の基礎である分布意味論について述べる．

3.1　分布意味論

我々は分布意味論 [53] によって論理プログラム（以下，単にプログラムという）の確率的意味を宣言的に定めることができる．本節では分布意味論を論文 [53] の記述に基づき説明する．まず，我々が考えるプログラム DB は単節（ファクト）の集合 F とボディが空でない確定節（ルール）の集合 R から成る．すなわち $DB = F \cup R$ である．また，形式的には DB を高々可算無限個の基底確定節の集合であると考え，DB が論理変数を含む場合は常に基底代入してものを考える．また，分布意味論では R 中のヘッドと単一化するアトムが F 中に存在しない（「DB は**分離条件**を満たす」という）ことを仮定する．

分布意味論では，はじめに基礎確率分布と呼ばれる F のすべての解釈上の分布 P_F を与え，次に P_F からプログラム DB に出現するアトムに対するすべての解釈上の分布 P_{DB} を導く．そして，この P_{DB} をプログラム DB の確率的意味とする．すなわち，F の解釈に与えた確率分布（基礎分布 P_F）がプログラム P_{DB} の確率的意味 (P_{DB}) を定めると考える．この意味で先に行った分離条件の仮定（F のアトムの真偽は他のアトムの真偽から導かれることはない）は合理的なものである．

3.1.1　基礎確率分布 P_F

我々ははじめに F の可能な解釈の集合を標本空間 Ω_F と考え，Ω_F 上の確率分布 P_F を定義することを目指す．先に DB が論理変数を含む場合は常に基底代入してものを考えると述べた．従って F は可算（無限）集合に成り得るため，それに伴い Ω_F は一般には非可算（無限）集合になる．有限の標本空間 Ω に対しては，各標本点 $\omega \in \Omega$ に確率値 p_ω $(0 \leq p_\omega \leq 1, \sum_{\omega \in \Omega} p_\omega = 1)$ を与えておき，ある事象（Ω_F の部分集合）A の確率 $Pr(A)$ は $\sum_{\omega \in A} p_\omega$ によって定めることができたが，標本空間が非可算集合の場合にはこのように確率を定義することはできず，我々が望む確率（分布）が存在するかどうかも自明ではない．分布意味論では Kolmogorov に従い測度論に基づいて確率を与えており，以下でその手続きを簡単に述べる．なお，Sato の記法 [53] をいくつかの個所で変更している．

はじめに，$A_1, A_2, \ldots$ を F 中のアトムの数え上げとする．各 A_i に真のとき 1，偽のとき 0 を対

応させる．標本空間 Ω_F は $1, 0$ の無限次元直積空間 $\prod_{i=1}^{\infty}\{0,1\}_i$ と見なすことができる．そして Ω_F の部分集合 $[A_1^{x_1} \wedge A_2^{x_2} \wedge \cdots \wedge A_n^{x_n}]_F$ を次のように定義する．

$$[A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}]_F \stackrel{\text{def}}{=} \{\, \omega \mid \omega = (x_1, x_2, \ldots, x_n, *, *, \ldots) \in \Omega_F,\ * \text{ は } 0 \text{ または } 1\} \tag{3.1}$$

ただし，アトム A に対して記法 A^x が定義されている：

$$A^x = \begin{cases} A & \text{if } x = 1, \\ \neg A & \text{otherwise.} \end{cases}$$

以降の説明でもこの記法を用いることにする．

ところで，我々には次のような確率分布の系列 $P_F^{(n)}([A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}]_F)$ が与えられているとする $(n = 1, 2, \ldots; i = 1 \ldots n; x_i \in \{0, 1\})$:

$$0 \leq P_F^{(n)}([A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}]_F) \leq 1, \tag{3.2}$$

$$\sum_{x_1,\ldots,x_n} P_F^{(n)}([A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}]_F) = 1, \tag{3.3}$$

$$\sum_{x_{n+1}} P_F^{(n+1)}([A_1^{x_1} \wedge \cdots \wedge A_n^{x_n} \wedge A_{n+1}^{x_{n+1}}]_F) = P_F^{(n)}([A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}]_F). \quad (\textbf{両立条件}) \tag{3.4}$$

両立条件と Kolmogorov の拡張定理 [42] より各 $P_F^{(n)}$ の拡大となるような Ω_F 上の σ-加法的な確率測度 P_F が存在する．

$$P_F([A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}]_F) = P_F^{(n)}([A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}]_F) \tag{3.5}$$

このようにして得られた P_F を DB の**基礎確率分布** (basic probability distribution) と呼ぶ．

3.1.2 プログラムが表現する確率分布 P_{DB}

ここで，DB 中に現れるアトムの数え上げを $B_1, B_2, \ldots$ とおく（F のアトム $A_1, A_2, \ldots$ を含む点に注意）．これらのアトムに対する可能な解釈の集合を Ω_{DB}（**可能世界** (possible world) と呼ばれる）とおく．Ω_{DB} もまた $\{0,1\}$ の可算無限個の直積である．$\omega \in \Omega_F$ に対して ω から導かれる 最小 Herbrand モデルを $M_{DB}(\omega)$ と書く．そしてそれぞれ Ω_F, Ω_{DB} の部分集合である $[B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{M_{DB}}$, $[B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{DB}$ を

$$[B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{M_{DB}} \stackrel{\text{def}}{=} \{\omega \in \Omega_F \mid M_{DB}(\omega) \models B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}\} \tag{3.6}$$

$$[B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{DB} \stackrel{\text{def}}{=} \{\omega \in \Omega_{DB} \mid \omega \models B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}\} \tag{3.7}$$

と定める．$[\cdot]_{M_{DB}}$ は P_F-可測である．そして $n = 1, 2, \ldots$ について

$$P_{DB}^{(n)}([B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{DB}) \stackrel{\text{def}}{=} P_F([B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{M_{DB}}) \tag{3.8}$$

なる確率分布の系列 $P_{DB}^{(n)}$ を考える．$[\cdot]_{M_{DB}}$ の定義より，

- $[B_1^{y_1} \wedge \cdots \wedge B_n^{y_n} \wedge B_{n+1}]_{M_{DB}}$ と $[B_1^{y_1} \wedge \cdots \wedge B_n^{y_n} \wedge \neg B_{n+1}]_{M_{DB}}$ は互いに素である
- $[B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{M_{DB}} = [B_1^{y_1} \wedge \cdots \wedge B_n^{y_n} \wedge B_{n+1}]_{M_{DB}} \cup [B_1^{y_1} \wedge \cdots \wedge B_n^{y_n} \wedge \neg B_{n+1}]_{M_{DB}}$

は明らかである．これらと $P_{DB}^{(n)}$ の定義より，両立条件

$$\sum_{y_{n+1}} P_{DB}^{(n+1)}([B_1^{y_1} \wedge \cdots \wedge B_n^{y_n} \wedge B_{n+1}^{y_{n+1}}]_{DB}) = P_{DB}([B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{DB}) \tag{3.9}$$

が成り立つので，P_F の場合と同様，Ω_{DB} 上の σ-可法的な確率測度 P_{DB} が存在する．また，P_{DB} は P_F および各 $P_{DB}^{(n)}$ の拡大となっている．ここで各アトム A_i, B_j を真のとき 1, 偽のとき 0 をとる確率変数と見なす．すると，

$$P_F([A_1^{x_1} \wedge \cdots \wedge A_n^{x_n}]_F) = P_F(A_1 = x_1, \ldots, A_n = x_n) \tag{3.10}$$

$$P_{DB}([B_1^{y_1} \wedge \cdots \wedge B_n^{y_n}]_{DB}) = P_F(B_1 = y_1, \ldots, B_n = y_n) \tag{3.11}$$

と書くことができる．そして，我々はプログラム DB に確率的意味として P_{DB} を与える．特に DB の述語記号からなる任意の閉式（$\forall, \exists$ を含む）に対して $[G] \stackrel{\text{def}}{=} \{\omega \in \Omega_{DB} \mid \omega \models G\}$ とおけば $[G]$ は P_{DB}-可測で，G が真になる確率を $P_{DB}([G])$ によって定めることができる．例えば

$$\lim_{n\to\infty} P_{DB}([G(t_1) \wedge \ldots \wedge G(t_n)]) = P_{DB}([\forall x G(x)]) \tag{3.12}$$

$$\lim_{n\to\infty} P_{DB}([G(t_1) \vee \ldots \vee G(t_n)]) = P_{DB}([\exists x G(x)]) \tag{3.13}$$

が成り立つ（$t_1, t_2, \ldots$ は基底項の数え挙げ）．これは命題論理に基づく，あるいは（定数集合を有限とし，関数記号を禁止することにより）Herbrand 基底を有限に制限した一階述語論理に基づく統計知識の表現言語（例えば [30, 39, 46] など）にはない特徴である．また，F の可能な解釈から生じる最小不動点の集合を

$$\mathit{fix}(DB) \stackrel{\text{def}}{=} \{M_{DB}(\omega) \mid \omega \in \Omega_F\} \tag{3.14}$$

とおく．このとき，P_{DB} は以下の性質をもつことが証明される [53]．1. は確率が最小 Herbrand モデル上に分布すること，2. は分布意味論が最小モデル意味論の一般化になっていることを意味する．

1. $\mathit{fix}(DB)$ は P_{DB}-可測であり，P_{DB} は $\mathit{fix}(DB)$ のみに分布する．すなわち $P_{DB}(\mathit{fix}(DB)) = 1$ である．
2. P_F が一点 ω_0 のみに分布するなら P_{DB} は一点 $M_{DB}(\omega_0)$ のみに分布する．
3. P_{DB} は P_F の拡大である．

このように（有限の標本空間を考えるなど）表現形式に制限を加えることなく，最小モデル意味論に従って宣言的な確率的意味を定めるという点でも分布意味論は他の統計知識表現言語とは異なっている．

次節では，この分布意味論に基づく統計知識表現言語 PRISM について述べる．論理プログラムは知識表現言語でありながら実行可能であるという特長をもつが，PRISM にもその特長は引き継がれている．

3.2 PRISMプログラム

分布意味論における P_F は Ω_F を標本空間とする離散分布であればどのようなものを考えてもよいが，実用を考え，我々は次のように F と P_F に制限を加えて PRISM プログラムを定義する．

定義 4（PRISM プログラム） プログラム $DB = F \cup R$ を考える．ファクト集合 F と基礎確率分布 P_F が次の条件を満たすとき，DB は PRISM プログラムであるという．

表 3.1: P_F の制約の下での $\texttt{msw}(i,n,\cdot)$ の確率分布および解釈.

$\texttt{msw}(i,n,\cdot)$ の実現値（真のとき 1, 偽のとき 0）				P_F の制約下での確率	解釈： スイッチ i が試行 n で $\cdots$
$\texttt{msw}(i,n,v_1)$	$\texttt{msw}(i,n,v_2)$	$\cdots$	$\texttt{msw}(i,n,v_k)$		
1	0	$\cdots$	0	θ_{i,v_1}	値 v_1 をとる
0	1	$\cdots$	0	θ_{i,v_2}	値 v_2 をとる
$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$	$\vdots$
0	0	$\cdots$	1	θ_{i,v_k}	値 v_k をとる
otherwise				0	どの値もとらない, または V_i 以外の値をとる
total				1	

1. F のアトムは $\texttt{msw}(i,n,v)$ の形をしている．i,n,v はいずれも基底項であり，i,n を各々グループ id, 試行 id と呼ぶ．各 i に対して，i の値集合と呼ばれる基底項の有限集合 V_i が与えられているものとする．ただし，$\texttt{msw}(i,n,v)$ が F に出現しているとき（$v \in V_i$），$v' \in V_i$ かつ $v' \neq v$ なる $\texttt{msw}(i,n,v')$ も必ず F に出現しなければいけない．

2. 任意の基底項 i に対して $V_i = \{v_1, v_2, \ldots, v_k\}$ とおく．このとき $\texttt{msw}(i,n,v_1)$, $\texttt{msw}(i,n,v_2)$, $\ldots$, $\texttt{msw}(i,n,v_k)$ のうち，常にいずれか一つのみが真になる．$\theta_{i,v}$ は $v \in V_i$ に関して $\texttt{msw}(i,n,v)$ が真になる確率値を表し，$\sum_{v \in V_i} \theta_{i,v} = 1$ が成り立つ．

3. $n \neq n'$ または $i \neq i'$ のとき $\texttt{msw}(i,n,\cdot)$ と $\texttt{msw}(i',n',\cdot)$ は独立な確率変数である． ■

基底アトム $\texttt{msw}(i,n,v)$ は「名前 i をもつスイッチが試行 n において値 v をとる」という事実を表現している．そして，P_F に制約を与えることにより各スイッチ i の確率的振舞いを定めている．具体的には，

- 条件2 の制約により，スイッチ i および値集合 V_i に対して，スイッチ $\texttt{msw}(i,n,v)$ $(v \in V_i)$ の真偽の組み合わせに対して表 3.1 のような分布の制約を与える．そして，スイッチ i が試行 n においてあたかも確率 $\theta_{i,v}$ で値 v をとっているかのように見せる．あるいは，$\texttt{msw}(i,n,\texttt{V})$ においては，論理変数 $\texttt{V}$ がパラメータ $\langle \theta_{i,v_1}, \ldots, \theta_{i,v_n} \rangle$ の多項分布 (multinomial distribution) に従う確率変数のように振舞うと考えることも可能である（ただし $V_i = \{v_1, \ldots, v_n\}$）．

- 条件 2, 条件 3（$n \neq n'$ の場合）の制約より，スイッチ i の試行は同一の分布に従った独立試行 (i.i.d.) とする．

- 条件 3（$i \neq j$ の場合）の制約より，スイッチ i の試行の結果は他のスイッチ j の試行結果とは独立であるとする．

前節で述べた分布意味論において $P_F^{(n)}$ は与えられたものであると仮定していたが，PRISM プログラムでは例えば次のようにして $P_F^{(n)}$ を構築する．

- ファクトを $A_1, A_2, \ldots$ と重複なしに数え上げ，以下を $n = 1, 2, \ldots$ について行う．

- $n = 1$ で $A_1 = \texttt{msw}(i,\cdot,v)$ とおいたとき，$P_F^{(1)}([A_1]_F) \stackrel{\text{def}}{=} \theta_{i,v}$, $P_F^{(1)}([\neg A_1]_F) \stackrel{\text{def}}{=} 1 - \theta_{i,v}$ と定める．

- $n \geq 2$ のとき $A_n = \texttt{msw}(i,m,v)$ とおく．そして $U_{i,m}^{(n)} \stackrel{\text{def}}{=} \{v' \mid \texttt{msw}(i,m,v') \in \{A_1, \ldots, A_{n-1}\}\}$

を導入する．$U^{(n)}_{i,m}$ は $A_1,\ldots,A_{n-1}$ 中に出現する $\mathtt{msw}(i,m,\cdot)$ がもつ第 3 引数項（スイッチ値）の集合である．そして $x_1,\ldots,x_{n-1}\in\{1,0\}$ の全ての組合せについて以下を行なう．

- $A_{n'}=\mathtt{msw}(i,m,v')$ であり[1]，かつ $x_{n'}=1$（$A_{n'}$ が真）であるような $n'<n$ が存在するとき，次を定める．

$$\left\{\begin{array}{rcl} P_F^{(n)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}\wedge A_n]_F) & \stackrel{\rm def}{=} & 0 \\ P_F^{(n)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}\wedge\neg A_n]_F) & \stackrel{\rm def}{=} & P_F^{(n-1)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}]_F) \end{array}\right. \tag{3.15}$$

- そうでなく，$U^{(n)}_{i,m}=V_i-\{v\}$ であるとき（すなわち A_n 以外の $\mathtt{msw}(i,m,\cdot)$ が全て $A_1,\ldots,A_{n-1}$ の中に出現しているとき），次を定める．

$$\left\{\begin{array}{rcl} P_F^{(n)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}\wedge A_n]_F) & \stackrel{\rm def}{=} & P_F^{(n-1)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}]_F) \\ P_F^{(n)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}\wedge\neg A_n]_F) & \stackrel{\rm def}{=} & 0 \end{array}\right. \tag{3.16}$$

- そうでないとき，$U=U^{(n)}_{i,m}$ とおいて次を定める．

$$\left\{\begin{array}{l} P_F^{(n)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}\wedge A_n]_F) \\ \qquad\qquad \stackrel{\rm def}{=} \frac{\theta_{i,v}}{1-\sum_{v'\in U}\theta_{i,v'}}\cdot P_F^{(n-1)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}]_F) \\ P_F^{(n)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}\wedge\neg A_n]_F) \\ \qquad\qquad \stackrel{\rm def}{=} \frac{1-\left(\theta_{i,v}+\sum_{v'\in U}\theta_{i,v'}\right)}{1-\sum_{v'\in U}\theta_{i,v'}}\cdot P_F^{(n-1)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}]_F) \end{array}\right. \tag{3.17}$$

式 3.15〜3.17 のいずれも

$$\left\{\begin{array}{rcl} P_F^{(n)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}\wedge A_n]_F) & = & p\cdot P_F^{(n-1)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}]_F) \\ P_F^{(n)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}\wedge\neg A_n]_F) & = & (1-p)\cdot P_F^{(n-1)}([A_1^{x_1}\wedge\cdots\wedge A_{n-1}^{x_{n-1}}]_F) \end{array}\right. \tag{3.18}$$

$(0\le p\le 1)$ という形をしていることに注意する．従って，上のように構築された $P_F^{(n)}$ が式 3.2 〜 3.4 を満たし，故に PRISM プログラムの基礎分布 P_F が存在することが分かる．

次に，$P_F^{(n)}$（すなわち P_F）が PRISM プログラムの条件 1, 2, 3 を満たすことを例によって示す．簡単のため，論理変数を含まない PRISM プログラム DB_1 を考える．そして DB_1 に出現するファクトは $\mathtt{msw}(i,m,v_1)$, $\mathtt{msw}(i,m,v_2)$, $\mathtt{msw}(i,m,v_3)$, $\mathtt{msw}(i,m,v_4)$ 4 つのみであるとする（i, m, $v_1,\ldots,v_4$ は基底項）．i の値集合 $V_i=\{v_1,\ldots,v_4\}$ である．また，$A_n=\mathtt{msw}(i,m,v_n)$ としてファクトを数え上げる $(n=1\ldots4)$．更に記号を簡単にするため $p_n=\theta_{i,v_n}$ とおく（当然 $p_1+\cdots+p_4=1$ である）．

この DB_1 に対して，$P_F^{(n)}$ を表現する図 3.1 の**意味木** (semantic tree) を考える．通常通り，各ノードにはファクト A_n もしくはその否定 $\neg A_n$ がラベルづけされている．加えて，図 3.1 の意味木では各有向辺に式 3.18 中の係数 p または $1-p$ がラベルづけされている．根ノードから深さ n' のノード $(0<n'\le n)$ に至る有向パス中に出現するファクトを $A_1,\ldots,A_{n'}^{x_{n'}}$ とし，同じパス中に出現する係数を $p_1,\ldots,p_{n'}$ とおくと，この有向パスは確率

$$P_F^{(n')}([A_1^{x_1}\wedge\cdots\wedge A_{n'}^{x_{n'}}]_F)=p_1p_2\cdots p_{n'}$$

[1] $A_1, A_2,\ldots$ は重複なしの数え上げなので当然 $v\neq v'$ である．

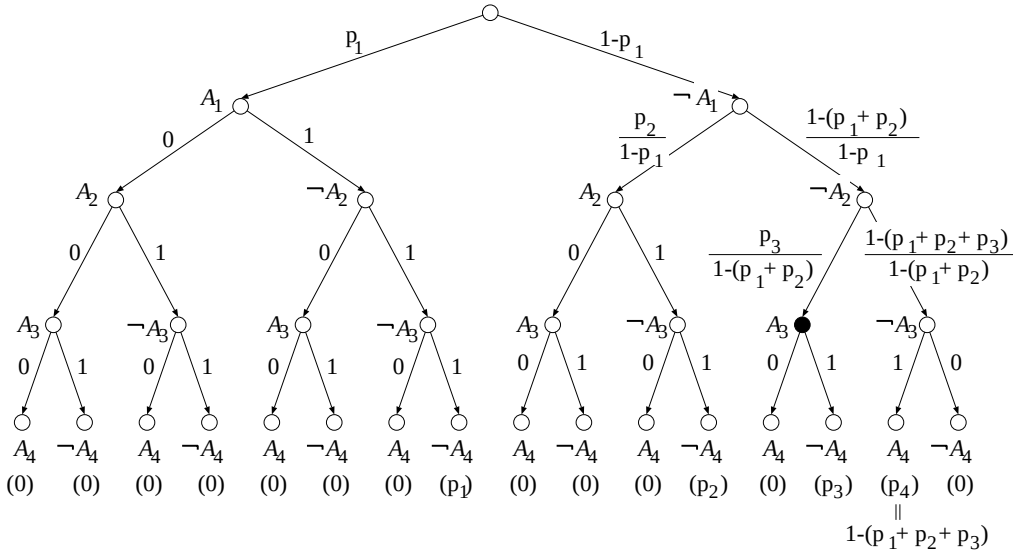

図 3.1: $P_F^{(n)}$ を表現する意味木.

を表現している．例えば，図 3.1 において根ノードから • で表されたノードに至る有向パスは

$$P_F^{(3)}([\neg A_1 \wedge \neg A_2 \wedge A_3]_F) = (1-p_1) \cdot \frac{1-(p_1+p_2)}{1-p_1} \cdot \frac{p_3}{1-(p_1+p_2)} = p_3$$

を表わす．このように根ノードから有向パスを辿ることと，先に述べた PRISM プログラムにおける $P_F^{(n)}$ の構築法を一対一に対応づけることが可能である．図 3.1 の葉ノードそれぞれに対し，その葉ノードに至る有向パスが表現する確率を (　) の中に書いた．これを見ると $P_F^{(n)}$ が PRISM プログラムの条件 1, 2, 3 を満たすことが分かる．そして $P_F^{(n)}$ の拡大である P_F も条件 1, 2, 3 を満たす.

最後に，分布意味論の一般的手順（節 3.1.2）により，PRISM プログラム DB に確率的意味 P_{DB} を与えることができる．

3.3 PRISM の表現力

PRISM プログラムは Chomsky 階層における 0 型文法（Turing マシン）の一つの確率化であり，先述した記号的統計モデル（HMM, PCFG およびその拡張文法, Bayesian ネットワーク）の一般化となっている．それを示すために本節ではこれらのモデルを実際に PRISM プログラムとして書き下してみる．

```
(1) target(hmm/1).
(2) data('hmm.dat').
(3) table([hmm/1,hmm/3]).
(4) values(init,[q0,q1]).
(5) values(out(_),[a,b]).
(6) values(tr(_),[q0,q1]).
(7) hmm(Cs):-                % 記号列 Cs を出力するには…
        msw(init,once,Qi),   %     Qi を初期状態にセットして,
        hmm(1,Qi,Cs).        %     カウンタを 1 にしてループに入る.
(8) hmm(L,Q,[C|Cs]):-        % ループ:
        L =< 3,              %     カウンタが 3 以下だったら,
        msw(out(Q),L,C),     %     状態 Q で記号 C を出力し,
        msw(tr(Q),L,NextQ),  %     Q から NextQ に遷移して,
        L1 is L+1,           %     時計を一つ進め,
        hmm(L1,NextQ,Cs).    %     ループを繰り返す(再帰).
(9) hmm(L,_,[]):- L>3.       % カウンタが 3 を超えたらループを終了する.
```

図 3.2: HMM プログラム.

3.3.1 隠れマルコフモデル

状態集合 $S_{\mathrm{hmm}} = \{\mathtt{q0}, \mathtt{q1}\}$ で，出力記号集合 $V_{\mathrm{hmm}} = \{\mathtt{a}, \mathtt{b}\}$ で，長さ 3 の記号列を出力・受理する HMM を PRISM プログラムで図 3.2 のように記述する[2]．各節番号 (j) は実際には書かれない．以降ではこの PRISM プログラムを単に HMM プログラムと呼ぶ．PRISM プログラムでは効率的な実行が可能になるようにユーザがいくつか付加的な情報をシステムに知らせる必要がある．HMM プログラムの場合は節 (1) ～ (6) がそれにあたる．節 (1) はその真偽を観測可能なアトムが hmm(·) であることを指定する．節 (2) は後に述べる EM 学習に必要な観測データが hmm.dat というファイルに格納されていることを示す[3]．節 (3) は後に述べるテーブル述語を指定するものである．テーブル述語をもつサブルーチンの計算結果を保存しながら計算を進めることにより，EM 学習が高速になるという効果をもつ．節 (4) ～ (6) は PRISM プログラム中の各スイッチ msw(i,·,·) の値集合 V_i を定めるものである．例えば節 (5) では $V_{\mathtt{tr}(\cdot)} = \{\mathtt{q0}, \mathtt{q1}\} = S_{\mathrm{hmm}}$ を指定している．一方，節 (7) ～ (9) は HMM を表現する統計モデルの部分である．これらの節は手続き的，宣言的の 2 通りで理解できる．まず，msw(init,·,q), msw(out(q),ℓ,v), msw(tr(q),ℓ,q') それぞれが真になる確率 $\theta_{\mathtt{init},q}$, $\theta_{\mathtt{out}(q),v}$, $\theta_{\mathtt{tr}(q),q'}$ を $Pr(Q_1 = q)$, $Pr(O_\ell = v \mid Q_\ell = q)$, $Pr(Q_{\ell+1} = q' \mid Q_\ell = q)$ と対応づける．

[2]分かりやすさのため，この HMM プログラムではカウンタとして非負整数を表わす項 0, 1, 2, ...を使い，論理プログラムの拡張である算術述語 is/2, >/2 などを用いているが，カウンタ L を廃して下のように書けば論理プログラムの範囲になる．L の代わりに msw/3 の第 2 引数に部分文字列のリストを与えており，このリストの「長さ」がカウンタ L と同じく，異なるループ ℓ, ℓ' における msw(tr(q),ℓ,·) と msw(tr(q),ℓ',·) の独立性，および msw(out(q),ℓ,·) と msw(out(q),ℓ',·) の独立性を保証する（後の手続き的理解を参照）．また，true は恒真命題である．

```
(7) hmm(Cs):- msw(init,once,Qi),hmm(Qi,Cs).
(8) hmm(Q,[C|Cs]):- msw(out(Q),[C|Cs],C), msw(tr(Q),[C|Cs],NextQ), hmm(NextQ,Cs).
(9) hmm(_,[]):- true.
```

[3]観測データの形式は次のようになる．count(G,F) はゴール G を F 回観測したことを表す．

```
count(hmm([a,a,a]),12).
count(hmm([b,a,a]),21).
count(hmm([a,b,b]),5).
count(hmm([a,b,a]),19).
   :
```

◇ 手続き的理解

コメント文のように手続き的に読めば，このHMMプログラムが生成器としてのHMMの確率的振舞いをそのまま記述していることが分かる．手続き的理解においては，`msw(`i`,`n`,V)` の論理変数 `V` の値が（ちょうどサイコロを振るように）確率的に決められると考えると分かりやすい．節 `(7)` 中の `msw(init,once,·)` は確率 $Pr(Q_1 = q) = a_q$ で初期状態 q を選択する．また，適切な ℓ と q が与えられたときに節 `(8)` 中の `msw(out(`q`),`ℓ`,·)` は確率 $Pr(O_\ell = v \mid Q_\ell = q) = b_q(v)$ で出力記号 v を値にとる．そして，`msw(tr(`q`),`ℓ`,·)` は確率 $Pr(Q_{\ell+1} = q' \mid Q_\ell = q) = a_{qq'}$ で次状態 q' を値にとる．第2引数に ℓ を与えることで，各ループにおけるスイッチ `out(·)`, `tr(·)` の動作は独立であることが保証される．一方，スイッチ `init` は一回しか動作しない（初期状態を決めるのは1回だけ）ので，第2引数に特別な定数記号 `once` を与えている．また，再帰呼び出しによってループが実現され，プログラムが非常にコンパクトになっていることが分かる．節 `(9)` はループの終了に対応する．

◇ 宣言的理解

`hmm(`$\boldsymbol{o}_{1,L}$`)` が真になる確率を $Pr(\boldsymbol{O}_{1,T} = \boldsymbol{o}_{1,L})$, `hmm(`$\ell$`,`$q$`,`$\boldsymbol{o}_{\ell,L}$`)` が真になる確率を $Pr(\boldsymbol{O}_{\ell,L} = \boldsymbol{o}_{\ell,L} \mid Q_\ell = q)$ と解釈する．また，プログラム中のリスト `[`o_ℓ`,`$o_{\ell+1}$`,...,`o_L`]` を実現値ベクトル $\boldsymbol{o}_{\ell,L} = \langle o_\ell, o_{\ell+1}, \ldots, o_L \rangle$ をと見なす．PRISMプログラムを宣言的に理解するにはClarkの完備化 [11] を施すと分かりやすい．例えば述語 `hmm/1` を定義しているのは節 `(7)` だけであるから，`hmm/1` の完備化は次のようになる．

$$\forall \mathtt{Y}\Big(\mathtt{hmm(Y)} \Leftrightarrow \exists \mathtt{Qi},\mathtt{Cs}\Big(\mathtt{Y} = \mathtt{Cs} \wedge \mathtt{msw(init,once,Qi)} \wedge \mathtt{hmm(1,Qi,Cs)}\Big)\Big) \tag{3.19}$$

ここで `Y` に任意の基底項を代入して，

1. 同値な2つの閉式 G, G' について $P_{DB}([G]) = P_{DB}([G'])$ は等しい（定義より明らか）．
2. P_{DB} は最小Herbrandモデル上にしか分布しない（節 3.1.2 を参照）．
3. `msw(init,once,`v`)` が v にとる値は S_{hmm} の要素 `q0`, `q1` のみである（他をとる確率が0）．また，`msw(init,once,q0)`, `msw(init,once,q1)` は確率的に排反である（同時に真になる確率が0）．
4. 任意の $q, \boldsymbol{o}$ について `msw(init,once,`q`)` と `hmm(1,`q`,`$\boldsymbol{o}$`)` は独立，すなわち下が成り立つ（アトムの独立性については後に定義を与える）：

$$\begin{aligned} &P_{DB}(\mathtt{msw(init,once,}q\mathtt{)}=1, \mathtt{hmm(1,}q\mathtt{,}\boldsymbol{o}\mathtt{)}=1) \\ &\qquad = P_{DB}(\mathtt{msw(init,once,}q\mathtt{)}=1) \cdot Pr(\mathtt{hmm(1,}q\mathtt{,}\boldsymbol{o}\mathtt{)}=1). \end{aligned}$$

の4点に注意すると，節 `(7)` は確率関係式

$$Pr(\boldsymbol{O}_{1,L}=\boldsymbol{o}_{1,L}) = \textstyle\sum_{q\in S_{\mathrm{hmm}}} Pr(Q_1=q) Pr(\boldsymbol{O}_{1,L}=\boldsymbol{o}_{1,L} \mid Q_1=q) \tag{3.20}$$

を意味することが分かる．同様に節 `(8)`, `(9)` から任意の $\ell, q, v, \boldsymbol{o}_{\ell,L}$ に関する確率関係式

$$\begin{aligned} &Pr(\boldsymbol{O}_{\ell,L}=\boldsymbol{o}_{\ell,L} \mid Q_\ell=q) \\ &\quad = \begin{cases} \sum_{q'\in S_{\mathrm{hmm}}} Pr(O_\ell=o_\ell \mid Q_\ell=q) Pr(Q_{\ell+1}=q' \mid Q_\ell=q) \cdot \\ \qquad\qquad Pr(\boldsymbol{O}_{\ell+1,L}=\boldsymbol{o}_{\ell+1,L} \mid Q_{\ell+1}=q') & \text{if } \ell \le L \\ 1 & \text{if } \ell > L \end{cases} \end{aligned} \tag{3.21}$$

が得られる．式 3.20 は確率の公理を満たしており，また式 3.21 はマルコフ仮定の下では正しい．従って，先に記述した HMM プログラムは正しいことが分かる．このように我々は記述した統計知識の理解，もしくは検証を宣言的に（つまり，プログラムの実行手続きとは無関係に）行うことができる．従って，統計知識の表現として非常に見通しのよいものとなる．

上では `hmm(`ℓ`,`q`,`$\boldsymbol{o}_{\ell,L}$`)` が真になる確率を $Pr(\boldsymbol{O}_{\ell,L} = \boldsymbol{o}_{\ell,L} \mid Q_\ell = q)$ と解釈しているが，$\boldsymbol{o}_{\ell,L}$ を t 回目の観測部分列 $\boldsymbol{o}_{\ell,L}^{(t)}$ に置き換えると，この確率値は Baum-Welch アルゴリズム（節 2.1.2.1）で用いる後向き確率（式 2.5）に他ならない．更に，式 3.21 は後向き確率の計算式（式 2.7）と一致し，式 3.20 は後向き確率による記号列 $\boldsymbol{o}^{(t)}$ の生起確率 $Pr(\boldsymbol{O}=\boldsymbol{o}^{(t)})$ の計算式（式 2.10）と一致する．

3.3.2 Bayesian ネットワーク

図 2.5 の Bayesian ネットワークを PRISM プログラムとして記述する．また，観測可能な変数を C, G としている．簡単のため，すべての変数 A, B, C, D, E, F, G が `yes`, `no` の 2 つの値をとるものとする．

```
(B1) target(world).
(B2) data('world.dat').
(B3) values(_,[yes,no]).

(B4) world(C,G):- world(_,_,C,_,_,G).
(B5) world(A,B,C,D,E,F,G):-
         msw(a,A),msw(b,B),msw(c(A),C),msw(d(A,B),D),
         msw(e,E),msw(f(E),F),msw(g(D,E),G).
```

節 `(B1)` で観測可能な述語として `world/2` が指定されている．例えば

```
count(world(yes,yes),10).
count(world(yes,no),30).
count(world(no,yes),55).
count(world(no,no),5).
```

などという 100 $(= 10 + 30 + 55 + 5)$ 個の観測データが与えられる．節 `(B5)` が Bayesian ネットワークを表現している部分である．Bayesian ネットワークは非循環な有向グラフであり，HMM のときのように同じスイッチ `msw(`i`,`$\cdot$`,`$\cdot$`)` を繰り返し使うことがないため，第 2 引数は省略している[4]．図 2.5 が表現する同時分布 $Pr(a, b, c, d, e, f, g)$ は次のように簡単化されることは先に述べた．

$$Pr(a, b, c, d, e, f, g) = \theta_A(a)\theta_B(b)\theta_C(c|a)\theta_D(d|a, b)\theta_E(e)\theta_F(f|d)\theta_G(g|d, e)$$

節 `(B5)` は上の式，すなわち Bayesian ネットワークが表現する同時分布を一つの節によって記述できている．節 `(B5)` のボディ（右辺）中の各 `msw(`i`,`v`)` は条件つき確率表 (CPT) のエントリ $\theta_{X_i}(x_i|\boldsymbol{u})$ を表現していると見ることができる．Bayesian ネットワークに限らず，一般的に PRISM プログラムでは条件つき確率 $Pr(X = x|U_1 = u_1, U_2 = u_2, \ldots, U_M = u_M)$ を `msw(`f_X`(`u_1`,`u_2`,...,`u_M`),`$\cdot$`,`x`)` や `msw(cpt(`t_X`,`u_1`,`u_2`,...,`u_M`),`$\cdot$`,`x`)` で表現することができる．ただし，f_X は X に対応づけた関数記号，t_X は X に対応づけた項である．もちろん，節 `B3` のように前もって $V_{f_X(u_1,u_2,\ldots,u_M)} = \mathcal{D}(X)$ を定めておかなければならない．

[4] 処理系がプログラム読み込み時に `msw(`i`,`v`)` を `msw(`i`,once,`v`)` に翻訳すると考えてよい．

3.3.3 確率文脈自由文法およびその拡張文法

本節では，確率文脈自由文法およびその拡張文法を PRISM プログラムで記述するが，その前に**確率確定節文法**（probabilistic definite clause grammar; 以下 PDCG）という記述形式を提案する．これは，**確定節文法**（definite clause grammar; 以下 DCG） [44, 57, 59] の確率化である．PDCG を用いて PCFG とその拡張文法を記述する．

3.3.3.1 確率確定節文法

PDCG は規則選択に関するスイッチ `msw(·,·,·)` を暗黙のうちにもつ PRISM プログラムと見ることができる．ちょうど DCG を Prolog プログラムに翻訳するように，PDCG は PRISM プログラムに翻訳される．先の PCFG の記述で行ったように，我々は ε 規則をもたず，$A \stackrel{+}{\Rightarrow} A$ が起こらない文法構造を仮定する．はじめに DCG を説明し，それから PDCG を記述する．

◇ 確定節文法

PDCG における各生成規則（以下，DCG 規則と呼ぶ）の基本構文は次の 2 ついずれかの形をしている ($k \geq 1$).

A `-->` B_1`,`B_2`,...,`B_k`.`

ただし，A は p`(`x_1`,`x_2`,...,`x_m`)` の形をし ($m \geq 0$)[5]，$B_1, \ldots, B_k$ の各々は q`(`y_1`,`y_2`,...,`$y_{m'}$`)` または `[`w_1`,`w_2`,...,`$w_{m''}$`]` の形をしている ($m' \geq 0$, $m'' \geq 1$)．p, q は非終端記号を表す述語記号であり，w_j $(1 \leq j \leq m'')$ は終端記号を表す定数記号である．以上の構文をもつ規則を「(非終端記号) p に関する生成規則」という．また $x_1, \ldots, x_m$ および $y_1, \ldots, y_{m'}$ は任意の項である．

DCG 規則は対応する Prolog プログラムに翻訳される．例えば，

```
(R1) s --> n(Num,Psn),v(Num,Psn),[you].
(R2) n(sing,1) --> [i].
(R3) n(sing,3) --> [she].
(R4) v(sing,1) --> [love].
(R5) v(sing,3) --> [loves].
```

という DCG は下の Prolog プログラムに変換される．

```
(R1') s(Z0,Z1):- n(Num,Psn,Z0,Z2),v(Num,Psn,Z2,Z3),Z3=[you|Z1].
(R2') n(sing,1,Z0,Z1):- Z0=[i|Z1].
(R3') n(sing,3,Z0,Z1):- Z0=[she|Z1].
(R4') v(sing,1,Z0,Z1):- Z0=[love|Z1].
(R5') v(sing,3,Z0,Z1):- Z0=[loves|Z1].
```

q`(`y_1`,`y_2`,...,`$y_{m'}$`)` という形をした (非終端記号に対応する) 述語には引数の最後に**差分リスト**と呼ばれるデータ構造を構成する 2 つの引数が追加される．長さ L の文 $\boldsymbol{w}$ に対し，`[`w_{d+1}`,...,`$w_{d'}$`,...,`$w_{d''}$`]` と `[`$w_{d'+1}$`,...,`$w_{d''}$`]` の組で表される差分リストは部分文 $\boldsymbol{w}_{d,d'}$ を表している．例えば $\boldsymbol{w} = \langle$`she,loves,you`$\rangle$ に対して，`[she,loves,you]` と `[you]` の組からなる差分リストは $\boldsymbol{w}_{0,2}$ と対応づけできる．$\boldsymbol{w} = \langle$`she,loves,you`$\rangle$ が正しい文かどうかを知りたい場合は Prolog インタプリタのトップゴールに

```
?- s([she,loves,you],[]).
```

として実行し，成功すれば $\boldsymbol{w}$ が正しい文だと分かる．

[5] p は p`(`x_1`,`x_2`,...,`x_m`)` の $m = 0$ の場合に対応する．

◇ 確率確定節文法の形式

PDCG における生成規則（以下，PDCG 規則と呼ぶ）は次の 2 ついずれかである ($k \geq 1$).

A `==>` B_1`,`B_2`,...,`B_k`.`

ただし，A は p`(`x_1`,`x_2`,...,`x_m`)` の形をし ($m \geq 0$)，$B_1, \ldots, B_k$ の各々は q`(`y_1`,`y_2`,...,`$y_{m'}$`)@`i`:`n または `[`w_1`,`w_2`,...,`$w_{m''}$`]` の形をしている ($m' \geq 0, m'' \geq 1$)．DCG 同様，p, q は非終端記号を表す述語記号であり，w_j ($1 \leq j \leq m''$) は終端記号を表す定数記号である．また $x_1, \ldots, x_m$ および $y_1, \ldots, y_{m'}$ そして i, n は任意の項である．“`@`i`:`n” という構文要素をもつ点が DCG と異なってる．

先に述べたように，PDCG は規則選択に関するスイッチを暗黙のうちにもつ PRISM プログラムである．上で述べた q`(...)@`i`:`n は「非終端記号 q を左辺にもつ規則は，スイッチ i の試行 n で得られた値に基づき選択せよ」ということを表している．例えば，`pp` に関する生成規則

```
(R6) pp(X,Y,W1,W2) ==>
         p(W1,W3)@sw1:X,
         [the],
         n(W3,W2)@sw2(W3):Y.
```

が選ばれた後，`sw1` という名のスイッチが試行 `X` で出力した値により `p` に関する規則のうちいずれかを選択し，“`the`” を読み込んだ後，スイッチ `sw2(W3)` の `Y` という試行において得られた値で `n` の規則選択が行われる．

◇ 確率確定節文法の翻訳

一方，規則 `(R6)` が `pp` に関する 3 番目の規則であったとき，規則 `(R6)` は次の PRISM プログラム節 `(R6')` に翻訳される．

```
(R6') pp(I,N,X,Y,W1,W2,Z0,Z1):-
          msw(I,N,3),
          p(sw1,X,W1,W3,Z0,Z2),
          Z2=[the|Z3],
          n(sw2(W3),Y,W3,W2,Z3,Z1).
```

差分リストが元述語の引数の末尾に追加されるのは DCG と同じであるが，PDCG では更に規則選択に用いるスイッチ `I` とその試行名 `N` が元述語の引数の先頭に追加される．そして，ボディの先頭に追加された `msw(I,N,3)` が真のとき，それに伴って他の `pp` に追加された `msw` は偽になり，`(R6)` が唯一選択される．

◇ 省略時の翻訳

PDCG で新たに導入された構文要素 “`@`i`:`n” において，i, n（あるいはその両方）が省略された場合を考える．q`(...)@`i`:`n において i が省略されたとき，PRISM プログラムへの翻訳器は $i =$ `sw_`q を補う．同様に，q`(...)@`i`:`n において n が省略されたとき，PRISM プログラムへの翻訳器は $n =$ `Z/Z'` を補う．ただし，`Z, Z'` はこの順で q`(...)` に付け加えられる差分リストの組で

ある．我々は ε-規則や $A \stackrel{+}{\Rightarrow} A$ となるような文法構造がないと仮定しているので，一つの構文木に $A \stackrel{*}{\Rightarrow} \boldsymbol{w}_{d,d'}$ となるような (A, d, d') の組は唯一つである．そして DCG の説明で見たように差分リストの組 `Z/Z'` と $\boldsymbol{w}_{d,d'}$ は対応づけできるので，スイッチの試行 id として `Z/Z'` を使えば，そこで行われた選択が他と必ず独立になることが保証される．

◇ 補強項の導入

PDCG では DCG と同様，**補強項**を許している．PDCG では補強項の中にスイッチ `msw(·,·,·)` を入れてもよい．例えば，数 (`Num`) や人称 (`Psn`) などの素性が出現する確率を考慮することもできる．

```
(R10) s --> { msw(number,once,Num), msw(person,once,Psn) },
            n(Num,Psn),v(Num,Psn),[you].
(R11) n(sing,1) --> [i].
(R12) n(sing,3) --> [she].
(R13) v(sing,1) --> [love].
(R14) v(sing,3) --> [loves].
```

統計的言語モデルから発展した一階論理に基づく統計知識表現言語の多く [13, 14, 37, 50] は，(規則選択の拡張として）節ごとに確率を割り当てているため，PDCG のような形で確率選択を入れるには，文法記述者の工夫が必要になる．

3.3.3.2 確率文脈自由文法

PCFG を PDCG で記述する．省略法を上のように定めたので，DCG で CFG を記述するのと同じ形で PCFG を記述することができる．

```
s ==> np, vp.
s ==> vp.
np ==> det, n.
np ==> pron.
np ==> n.
vp ==> v, n.
vp ==> v.
n ==> [car].
    :
```

上の PDCG は次の PRISM プログラムに翻訳される．

```
s(I,N,Z0,Z1):- msw(I,N,1), np(sw_np,Z0/Z2,Z0,Z2), vp(sw_vp,Z2/Z1,Z2,Z1).
s(I,N,Z0,Z1):- msw(I,N,2), vp(sw_vp,Z0/Z1,Z0,Z1).
np(I,N,Z0,Z1):- msw(I,N,1), det(sw_det,Z0/Z2,Z0,Z2), n(sw_n,Z2/Z1,Z2,Z1).
np(I,N,Z0,Z1):- msw(I,N,2), pron(sw_pron,Z0/Z1,Z0,Z1).
np(I,N,Z0,Z1):- msw(I,N,3), n(sw_n,Z0/Z1,Z0,Z1).
vp(I,N,Z0,Z1):- msw(I,N,1), v(sw_v,Z0/Z2,Z0,Z2), n(sw_n,Z2/Z1,Z2,Z1).
vp(I,N,Z0,Z1):- msw(I,N,2), v(sw_v,Z0/Z1,Z0,Z1).
n(I,N,Z0,Z1):- msw(I,N,1), Z0=[car|Z1].
    :
```

この翻訳から分かるように非終端記号 A の規則選択には各 A に対して設けられたスイッチ `sw_`A を使う．すなわち規則選択に関して文脈自由であることが保証される．同様に，同じ A に関する選択が複数回あっても（ε-規則をもたない，$A \stackrel{+}{\Rightarrow} A$ が現れないという仮定から）差分リスト `Z/Z'` は異なった値をもつので，各選択間の独立性は保証されている．

3.3.3.3 確率文脈自由文法の拡張文法

本節では擬似PCSG（節 2.1.5）と PCFG+bigram モデル（節 2.2.4）を PDCG で記述し，PDCG（PRISM プログラムの記述力の高さ）を確認する．はじめに擬似 PCSG を PDCG で記述する．

```
s ==> np@sw_np(s), vp@sw_vp(s).
s ==> vp@sw_vp(s).
np ==> det@sw_det(np), n@sw_n(np).
np ==> pron@sw_pron(np).
np ==> n@sw_n(np).
vp ==> v@sw_v(vp), n@sw_n(vp).
vp ==> v@sw_v(vp).
n ==> [car].
    :
```

3 番目と 6 番目の規則における右辺の非終端記号 `n` に注目する．`n` の規則選択において 3 番目の規則ではスイッチ `sw_n(np)` を指定し，6 番目の規則ではスイッチ `sw_n(vp)` を指定している．すなわち，構文木中の親ノードである左辺の終端記号（それぞれ `np`, `vp`）に応じて別のスイッチ（分布）に基づいて規則を選択する．これは擬似 PCSG で行われている規則選択に他ならない．よって，上のように非常に簡潔な形で擬似 PCSG を表現できることが分かった．

一方，PCFG+bigram モデルは次のように記述する．各非終端記号 A に対して `W, W'` という 2 つの引数が付け加えられている．A`(W,W')` は A の統治する部分単語列の直前の単語が `W` で，A の統治する部分単語列の最後の単語が `W'` であることを表現する．PCFG+bigram モデルの規則ではこの単語情報を非終端記号の引数で伝播させている．

```
s(W0,W1) ==> np(W0,W2)@sw_np(W0), vp(W2,W1)@sw_vp(W2).
s(W0,W1) ==> vp(W0,W1)@sw_vp(W0).
np(W0,W1) ==> det(W0,W2)@sw_det(W0), n(W2,W1)@sw_n(W2).
np(W0,W1) ==> pron(W0,W1)@sw_pron(W0).
np(W0,W1) ==> n(W0,W1)@sw_n(W0).
vp(W0,W1) ==> v(W0,W2)@sw_v(W0), n(W2,W1)@sw_n(W2).
vp(W0,W1) ==> v(W0,W1)@sw_v(W0).
n(_,car) ==> [car].
    :
```

そして，A`(W,W')` に対して `sw_`A`(W)` というスイッチで選択する．先に述べたように `W` は A`(W,W')` が統治する部分単語列の直前の単語であるので，上は PCFG+bigram モデルにおける規則選択を簡潔な形で記述している．以上より，PDCG を用いると PCFG の拡張文法が非常に簡潔な形で記述できることが分かった．

3.4 第 3 章のまとめ

本章では，Sato によって提案された分布意味論とそれに基づく記号的統計モデル言語である PRISM (PRogramming In Statistical Modeling) を紹介した．そして，前章で述べた隠れマルコフモデル，確率文脈自由文法およびその拡張文法，Bayesian ネットワークを PRISM で実際に記述し，PRISM がこれらの統計モデルの表現上の一般化となることを示した．更に，確率文脈自由文法の拡張文法を簡潔に記述する上での PRISM プログラムの略記法として確率確定節文法という形式を提案した．

第4章　EMアルゴリズムとその高速化

前章では分布意味論に基づきPRISMプログラムの定義を行ない，PRISMの表現力の高さを例示した．しかし，序論で述べたように客観的な統計知識を記述するためにはその表現言語だけでなく，観測データからモデルの分布パラメータを学習する機構があることが望ましい．本章ではPRISMプログラムの分布パラメータ（各ファクト `msw(`i`,`n`,`v`)` に付与された $\theta_{i,v}$）の最尤推定法としてEMアルゴリズムを導入する．また，このEMアルゴリズムに（ある条件の下で）高速化を施し，効率的とされている既存の統計モデル専用EMアルゴリズムの一般化を行なう．

4.1　EMアルゴリズム

4.1.1　準備

PRISM用EMアルゴリズムを記述するための準備としていくつかの定義を行なう．最初に我々は学習に必要な観測データの形式とその観測方式を定める必要がある．まず，観測可能なアトムの集合を固定する．

定義5（観測可能なアトムの集合） PRISMプログラム $DB = F \cup R$ を考える．R のヘッドに出現するアトムの集合を $head(R)$ とおく．プログラマ（モデル記述者）は DB を記述すると同時に，高々可算無限であるような $head(R) \cup F$ の部分集合 O_{DB} を観測可能なアトムの集合として定めておくものとする． ■

O_{DB} の各要素は基底アトムであり，**観測可能アトム** (observable atom) あるいは**ゴール** (goal) と呼ばれる．また，$head(R) \cup F$ に含まれないアトムは最小Herbrandモデルに含まれない．従って，P_{DB} の性質（節3.1.2）よりそのようなアトムは確率1で偽になるため観測する意味がなく，観測対象から外して構わない．また，プログラマは O_{DB} を任意に定めることができる．従って定義5はプログラムの実質的な制限にはなっていないことに注意する．次に，EMアルゴリズムを構成する上で重要な概念である極小支持集合について説明する．その定義には先に触れた R に対するClarkの完備化 [11] $comp(R)$ を用いる．

定義6（極小支持集合） PRISMプログラム $DB = F \cup R$ を考える．各 $B \in head(R)$ に対し

$$comp(R) \models B \leftrightarrow S_1 \vee S_2 \vee \cdots \vee S_m \tag{4.1}$$

を満たす $S_1, S_2, \ldots, S_m \subseteq F$ $(m \geq 1)$ を B の**極小支持集合** (minimal support set) と呼ぶ．また，上が成り立つとき，B の極小支持集合の集合を $\psi_{DB}(B) \stackrel{\text{def}}{=} \{S_1, S_2, \ldots, S_m\}$ とおく．また，ファクト $A \in F$ の極小支持集合は唯一 $\{A\}$ であるとする．つまり $\psi_{DB}(A) = \{\{A\}\}$ である． ■

例えばHMMプログラムの場合，下のファクト集合は ψ_{DB}`(hmm([a,b,a]))` に含まれる．これは記号列 `aba` を出力するのに行なった確率的選択の記録と見ることができる．例えば `msw(tr(`q`),·,`q'`)`

という形のファクトから状態遷移パスが構成される．

$$\left\{\begin{array}{l}\texttt{msw(init,once,q0)},\\ \texttt{msw(out(q0),1,a)},\texttt{msw(out(q1),2,b)},\texttt{msw(out(q1),3,a)},\\ \texttt{msw(tr(q0),1,q1)},\texttt{msw(tr(q1),2,q1)},\texttt{msw(tr(q1),3,q0)}\end{array}\right\} \tag{4.2}$$

定義 7（さまざまなファクト集合） PRISM プログラム $DB = F \cup R$ を考える．そのとき，以下の F の部分ファクト集合を定める．

$$relv(G) \stackrel{\text{def}}{=} \bigcup_{S \in \psi_{DB}(G)} S \quad \text{for each } G \in O_{DB} \tag{4.3}$$

$$\Sigma_{DB}(G) \stackrel{\text{def}}{=} relv(G) \cup \left\{\texttt{msw}(i,n,v) \mid v \in V_i, \exists v'\big(\texttt{msw}(i,n,v') \in relv(G), v' \neq v\big)\right\} \quad \text{for each } G \in O_{DB} \tag{4.4}$$

$$\Sigma_{DB} \stackrel{\text{def}}{=} \bigcup_{G \in O_{DB}} \Sigma_{DB}(G) \tag{4.5}$$

■

Σ_{DB} はゴール $G \in O_{DB}$ のいずれかの真偽に関連するファクトの全体である．以降では Σ_{DB} のファクトを考えれば充分である．次に，この Σ_{DB} を使って PRISM プログラム DB のグループ id 集合とパラメータをそれぞれ下のように定める．

定義 8（グループ id 集合） PRISM プログラム DB を考える．そのとき，

$$I_{DB} \stackrel{\text{def}}{=} \left\{\, i \;\middle|\; \exists n, v\big(\texttt{msw}(i,n,v) \in \Sigma_{DB}\big) \,\right\} \tag{4.6}$$

と定め，DB のグループ id 集合と呼ぶ． ■

定義 9（パラメータ，パラメータ空間） PRISM プログラム DB を考える．そのとき，$\left\{\theta_{i,v} \;\middle|\; i \in I_{DB}, v \in V_i\right\}$ の要素を並べたベクトルを $\boldsymbol{\theta}_{DB}$ と書き，DB のパラメータと呼ぶ．パラメータ $\boldsymbol{\theta}_{DB}$ の下での確率分布 P_{DB}, P_F をそれぞれ $P_{DB}(\cdot|\boldsymbol{\theta}_{DB})$, $P_F(\cdot|\boldsymbol{\theta}_{DB})$ と書く．また，$\Theta_{DB} \stackrel{\text{def}}{=} \left\{\boldsymbol{\theta}_{DB} \mid 0 \leq \theta_{i,v} \leq 1, \sum_{v \in V_i} \theta_{i,v} = 1 \text{ for } i \in I\right\}$ を DB のパラメータ空間と呼ぶ． ■

混乱がない限り $I_{DB}, \boldsymbol{\theta}_{DB}, \Theta_{DB}$ をそれぞれ $I, \boldsymbol{\theta}, \Theta$ と略す．ところで，我々はプログラム DB の分布を定める確率変数（すなわち DB 中に現れるすべてのアトム）のうち，その一部 $\boldsymbol{O}_{DB}$ 中の確率変数しか観測できない．確率ベクトル $\boldsymbol{O}_{DB}$ の実現値 $\boldsymbol{y}$ を観測したとき，これは不完全データである．我々の目標はプログラム DB の分布（確率的意味）を規定するパラメータ $\boldsymbol{\theta}$ に客観的な値を観測値データ $\boldsymbol{y}$ から与えることである．そのために $\boldsymbol{y}$ に対してパラメータ $\boldsymbol{\theta}$ の最尤推定，すなわち尤度 $P_{DB}(\boldsymbol{O}_{DB} = \boldsymbol{y}|\boldsymbol{\theta})$ を最大化する $\boldsymbol{\theta} \in \Theta$ を見つけることを考える．節 2.1.1 で見てきたように，EM アルゴリズムはこのような状況に有効な最尤推定アルゴリズムである．

ここで節 2.1.1 で示した Dempster らの形式化を考える．まず，最小 Herbrand モデルを考えれば，Σ_{DB} 中のファクトに対して真偽割り当て $\boldsymbol{x}$ を与えると O_{DB} 中の各ゴール G の真偽は一意に決まる．すなわち $\boldsymbol{x}$ に対して $\boldsymbol{O}_{DB}$ の真偽割り当て $\boldsymbol{y}$ が一意に定まる（もちろん逆は必ずしも成り立たない）．従って，この関係は many-to-one 関数 $\varphi : \mathcal{D}(\boldsymbol{\Sigma}_{DB}) \to \mathcal{D}(\boldsymbol{O}_{DB})$ により $\boldsymbol{y} = \varphi(\boldsymbol{x})$ と表現できる．Σ_{DB} が有限集合である（従って $\mathcal{D}(\boldsymbol{\Sigma}_{DB})$ も有限）と仮定したとき，Q 関数は次のように構成される．

$$Q(\boldsymbol{\theta}', \boldsymbol{\theta}) = \sum_{\boldsymbol{x}:\boldsymbol{y}=\varphi(\boldsymbol{x})} P_{DB}(\boldsymbol{\Sigma}_{DB} = \boldsymbol{x}|\boldsymbol{O}_{DB} = \boldsymbol{y}, \boldsymbol{\theta}) \log P_{DB}(\boldsymbol{\Sigma}_{DB} = \boldsymbol{x}, \boldsymbol{O}_{DB} = \boldsymbol{y}|\boldsymbol{\theta}') \tag{4.7}$$

簡単のため 1 つのデータ $\boldsymbol{y}$ を観測したとしている．この Q 関数に基づけば $P_{DB}(\boldsymbol{O}_{DB}=\boldsymbol{y}|\boldsymbol{\theta})$ は（局所的に）最大化するような EM アルゴリズムが導出される．ところで，式 4.7 中の $\boldsymbol{x}\,(\boldsymbol{y})$ のとり得る値の数は $\Sigma_{DB}\,(O_{DB})$ のファクトに対する真偽割り当てパターン数であるから $O(2^{|\Sigma_{DB}|})$ $\left(\,O(2^{|O_{DB}|})\,\right)$ である．従って一般には現実的な時間での EM 学習は難しい．そこで，PRISM プログラムにいくつかの条件を与えることで Q 関数（そしてそこから導出される EM アルゴリズム）をより簡単な形にすることを試みる．

4.1.2 PRISM プログラムに与える条件

PRISM プログラムに与える条件を述べる前に，準備としていくつかの言葉を定義しておく．まず Σ_{DB} の部分ファクト集合に関して確率的矛盾性および確率的排反性を定義する．頭に「確率的」と付くのはその定義が 基礎確率分布 P_F に基づくためである．

定義 10（確率的に矛盾するファクト 集合） PRISM プログラム DB と空でないファクト集合 $S \subseteq \Sigma_{DB}$ を考える．任意のパラメータ $\boldsymbol{\theta} \in \Theta$ について $P_F(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta}) = 0$ が成り立つとき，またそのときに限り S は確率的に矛盾するという． ■

定義 11（確率的に排反なファクト集合） PRISM プログラム DB を考える．また，空でなく，かつ確率的に無矛盾な 2 つのファクト集合 $S_1, S_2 \subseteq \Sigma_{DB}$ を考える．$S_1 \cup S_2$ が確率的に矛盾するとき，またそのときに限り S_1 と S_2 は確率的に排反であるという． ■

PRISM プログラムの基礎確率分布 P_F の条件（定義 4）より，上の 2 つの性質に関する必要十分条件が得られる．証明は容易なので省略する．

命題 1（ファクト 集合が確率的に矛盾するための必要十分条件） PRISM プログラム DB と空でないファクト集合 $S \subseteq \Sigma_{DB}$ を考える．すると，$\mathtt{msw}(i,n,v), \mathtt{msw}(i,n,v') \in S$ かつ $v \neq v'$ であるような基底項 i, n および基底項 $v, v' \in V_i$ が存在するとき，またそのときに限り S は確率的に矛盾する． ■

命題 2（ファクト 集合が確率的に排反になるための必要十分条件） PRISM プログラム DB を考える．また，空でなく，かつ確率的に無矛盾な 2 つのファクト集合 $S_1, S_2 \subseteq \Sigma_{DB}$ を考える．すると，$\mathtt{msw}(i,n,v) \in S_1, \mathtt{msw}(i,n,v') \in S_2$ かつ $v \neq v'$ であるような基底項 i, n および基底項 $v, v' \in V_i$ が存在するとき，またそのときに限り S_1 と S_2 は確率的に排反である． ■

ファクト集合に関する排反性を使って，確率的に排反なアトムについて定義し，その必要十分条件を得る．定義より $O_{DB} \subseteq \mathit{head}(R) \cup F$ であるから，ゴール（O_{DB} のアトム）間の確率的排反性は定義 12 もしくは命題 3 に基づいて判定されることに注意する．

定義 12（確率的に排反なアトム） PRISM プログラム DB と 2 つのアトム $B_1, B_2 \in \mathit{head}(R) \cup F$ を考える．任意のパラメータ $\boldsymbol{\theta} \in \Theta$ に対して $P_{DB}(B_1=1, B_2=1|\boldsymbol{\theta}) = 0$ であるとき，B_1 と B_2 は確率的に排反であるという． ■

命題 3（アトムが確率的に排反になるための必要十分条件） PRISM プログラム DB と 2 つのアトム $B_1, B_2 \in O_{DB}$ を考える．B_1 の任意の極小支持集合 $S_1 \in \psi_{DB}(B_1)$ と B_1 の任意の極小支持集合 $S_2 \in \psi_{DB}(B_2)$ について S_1 と S_2 が確率的に排反であるとき，またそのときに限り B_1 と

B_2 は確率的に排反である．■

以上の準備の下，PRISM に与える 3 つの条件である**有限支持条件** (finite support condition)，**排反性条件** (exclusiveness condition)，**唯一性条件** (uniqueness condition) について述べる．まず，下に述べる有限支持条件を仮定することによって確率計算や EM 学習が有限時間に終了することが保証される．PRISM プログラムは知識表現としては可算無限個のオブジェクトが扱えるが，その上での推論は有限時間で行なわれなければならないので，有限支持条件は実用上必要である．他方，意味論の範疇にまで有限性を持ち込んでいる他の統計知識表現言語 [30, 39, 46] とは区別する必要がある．

仮定 2（有限支持条件） PRISM プログラム $DB = F \cup R$ を考える．各 $B \in head(R)$ に対し $\psi_{DB}(B)$ は有限で，かつ $\psi_{DB}(B)$ に属する B の極小支持集合の各々も有限である．■

有限支持条件の下では，O_{DB} が有限であることと定義 7 より Σ_{DB} が有限集合になる．従って，それぞれ定義 8 と定義 9 より I_{DB} が有限集合，$\boldsymbol{\theta}_{DB}$ が有限次元ベクトルになる．また，次の排反性条件を仮定すると，各アトム $B \in head(R)$ の確率計算（後述）が容易になる．そして，唯一性条件は O_{DB} 中の一つのゴール G が唯一真になることを意味するものである．

仮定 3（排反性条件） PRISM プログラム DB を考える．このとき，任意の $B \in head(R)$ について $\psi_{DB}(B)$ 中の極小支持集合 S は確率的に無矛盾であり，かつ他の極小支持集合 $S' \in \psi_{DB}(B)$ と確率的に排反である $(S \neq S')$．■

仮定 4（唯一性条件） 有限支持条件を満たす PRISM プログラム DB を考える．O_{DB} 中の各ゴールは他と確率的に排反である．更に，任意のパラメータ $\boldsymbol{\theta} \in \Theta$ の下で O_{DB} 中のゴールの確率和が 1 になる，すなわち $\sum_{G \in O_{DB}} P_{DB}(G = 1|\boldsymbol{\theta}) = 1$ が成り立つ．■

$O_{DB} = \{G^{(1)}, G^{(2)}, \ldots, G^{(|O_{DB}|)}\}$ および $\psi_{DB}(G^{(k)}) = \{S_1^{(k)}, S_2^{(k)}, \ldots, S_{m_k}^{(k)}\}$ とおき $(k = 1 \ldots |O_{DB}|)$，排反性条件と唯一性条件を同時に仮定したときのイメージを図 4.1 に示す．排反性条件は，隠れマルコフモデル (HMM) における「1 回のサンプリングで生じる状態遷移はただ 1 通りである」ことに対応する．また，唯一性条件は HMM の「1 回のサンプリングでは，観測可能な文字列のうち，ただ 1 つが選ばれる」ことに対応する．図 4.1 においては $G^{(k)}$ が観測文字列，$S_j^{(k)}$ が状態遷移パスと記号出力結果に対応する（式 4.2 を参照）．この例から分かるように，排反性条件と唯一性条件は従来の統計モデルでは当然のものとして仮定されている場合が多い．従って，モデリングをする上でこれら 2 つの条件はそれほど大きな制約にはならないものと考えられる．

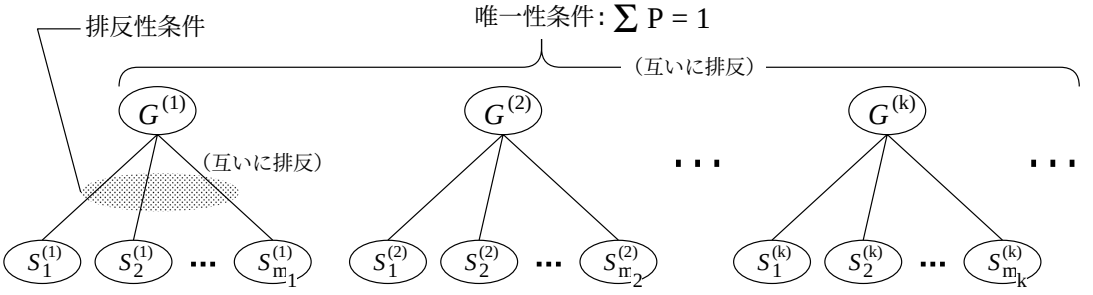

図 4.1: 排反性条件と唯一性条件を同時に仮定したときのイメージ図．

表 4.1: 唯一性条件下でのゴールの真偽割り当てに関する確率分布.

$G^{(1)}$	$G^{(2)}$	$\cdots$	$G^{(k)}$	$\cdots$	確率	解釈	観測変数 Y_{DB}
1	0	$\cdots$	0	$\cdots$	$P_{DB}(G^{(1)}=1\|\boldsymbol{\theta})$	$G^{(1)}$ を観測（$G^{(1)}$ のみ真）	$Y_{DB}=1$
0	1	$\cdots$	0	$\cdots$	$P_{DB}(G^{(2)}=1\|\boldsymbol{\theta})$	$G^{(2)}$ を観測（$G^{(2)}$ のみ真）	$Y_{DB}=2$
$\vdots$	$\vdots$	$\ddots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$
0	0	$\cdots$	1	$\cdots$	$P_{DB}(G^{(k)}=1\|\boldsymbol{\theta})$	$G^{(k)}$ を観測（$G^{(k)}$ のみ真）	$Y_{DB}=k$
$\vdots$	$\vdots$	$\ddots$	$\vdots$		$\vdots$	$\vdots$	$\vdots$
otherwise					0	$G^{(1)}, G^{(2)}, \ldots, G^{(k)}$ のいずれも偽，もしくは 2 つ以上が真	対応なし
total					1		

4.1.3 観測

唯一性条件の下で観測データの形式と観測方式を以下のように考える．この仮定はプログラム DB の記述に関する制約ではないことに注意する．

仮定 5（観測） 唯一性条件を満たす PRISM プログラム DB を考える．我々は O_{DB} 中の唯一真なるゴールを観測する．これを T 回繰り返し（ただし T は有限），t 回目 $(1 \leq t \leq T)$ に観測されたゴールを「t 番目の**観測ゴール** (observed goal)」と呼び，G_t とおく．各々の観測行為は他と独立に行なわれる．また，$\mathcal{G} \stackrel{\text{def}}{=} \langle G_1, G_2, \ldots, G_T \rangle$ を**観測データ**と呼ぶ． ■

明らかに $G_t \in O_{DB}$ が成り立つ $(t = 1 \ldots T)$．そして,(唯一観測可能な) Ω_{DB} 上の確率変数 Y_{DB} を導入し,「DB の**観測変数**」と呼ぶことにする ．これは唯一性条件により O_{DB} 中の各ゴールの真偽割り当てがあたかも一つの確率変数のように振舞うことから導入されたものである．唯一性条件下でのゴールの真偽割り当てに関する確率分布を表 4.1 に示す（$O_{DB} = \left\{G^{(1)}, G^{(2)}, \ldots, G^{(|O_{DB}|)}\right\}$ とおいている）．

定義 13（観測変数 Y_{DB}） 唯一性条件を満たす PRISM プログラム DB を考える．そして O_{DB} の数え上げを $G^{(1)}, G^{(2)}, \ldots, G^{(|O_{DB}|)}$ とおく．$k = 1 \ldots |O_{DB}|$ について $G^{(k)}$ が真のとき実現値 k をとる Ω_{DB} 上の確率変数 Y_{DB} を考え，我々は Y_{DB} の実現値を観測することにする．このとき $Y_{DB}(G_k) \stackrel{\text{def}}{=} k$ と書く．仮定 5 の観測方式をとったとき，**観測データ** $\mathcal{G} \stackrel{\text{def}}{=} \langle G_1, G_2, \ldots, G_T \rangle$ に対し，$y_t \stackrel{\text{def}}{=} Y_{DB}(G_t)$ とおく $(t = 1 \ldots T)$. ■

次に，上で行なった仮定 5 の下で尤度を定義する．

定義 14（PRISM プログラムにおける尤度） 唯一性を満たす PRISM プログラム DB と観測データ $\mathcal{G} = \langle G_1, G_2, \ldots, G_T \rangle$ が与えられたとき，尤度 $\Lambda_{DB}(\mathcal{G}|\boldsymbol{\theta})$ は各観測 G_t $(1 \leq t \leq T)$ の確率積，すなわち $\Lambda_{DB}(\mathcal{G}|\boldsymbol{\theta}) \stackrel{\text{def}}{=} \prod_{t=1}^{T} P_{DB}(G_t = 1|\boldsymbol{\theta})$ で与えられる．また尤度の対数をとったものを**対数尤度**と呼び，$\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta})$ と書く． ■

$\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta}) = \sum_{t=1}^{T} \log P_{DB}(G_t = 1|\boldsymbol{\theta})$ であることは定義より明らかである．PRISM プログラムにおいて以下の統計量はゴールの確率計算と EM アルゴリズムに用いられる（どちらも後述）．

定義 15（極小支持集合に関する統計量） PRISM プログラム DB を考える．このとき，各アトム

$B \in head(R)$ の極小支持集合 $S \in \psi_{DB}(B)$ それぞれについて以下の統計量を定める：

$$\sigma_{i,v}(S) \stackrel{\text{def}}{=} \left|\{n \mid \texttt{msw(}i\texttt{,}n\texttt{,}v\texttt{)} \in S\}\right| \quad \text{for each } i \in I, v \in V_i. \tag{4.8}$$

■

ここで PRISM 用の EM アルゴリズムを定義する前にアトムが真になる確率の計算式を示す．排反性条件を仮定すると次のように簡単な式で計算できるようになる．

命題 4（アトムの確率計算） 排反性を満たす PRISM プログラム DB を考える．任意のパラメータ $\boldsymbol{\theta} \in \Theta$ について次が成り立つ：

- $B \in head(R) \cup F, S \in \psi_{DB}(B)$ なる各 S について

$$P_F(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta}) = \textstyle\prod_{i \in I, v \in V_i} \theta_{i,v}^{\sigma_{i,v}(S)}, \tag{4.9}$$

- 各 $B \in head(R) \cup F$ について

$$P_{DB}(B=1|\boldsymbol{\theta}) = \textstyle\sum_{S \in \psi_{DB}(B)} P_F(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta}). \tag{4.10}$$

■

排反性条件を仮定することにより計算式は簡単化されたが，例えば HMM プログラム（排反性条件を満たしている）では $G \in O_{DB}$（例えば $G = \texttt{hmm([a,b,a])}$）に対して，個々の $S \in \psi_{DB}(G)$ は一つの状態遷移パスに対応する．従って $|\psi_{DB}(G)|$ は可能な状態遷移パスに対応し，出力・受理記号列長 L の指数オーダとなる．よって確率計算は計算量的に見て依然困難である．

4.1.4 PRISM 用 EM アルゴリズム

本節では節 2.1.1 で述べた方法に従いながら PRISM 用 EM アルゴリズムを記述するが，その前にいくつかの準備を行なう．

定義 16（可能支持集合） 排反性，唯一性条件を共に満たす PRISM プログラム DB を考える．そのとき，$\Delta_{DB} \stackrel{\text{def}}{=} \bigcup_{G \in O_{DB}} \bigcup_{S \in \psi_{DB}(G)} S$ と定め，Δ_{DB} の各要素を DB の**可能支持集合** (possible support set) と呼ぶ. ■

Δ_{DB} に現れるファクト集合は O_{DB} 中のどれかのゴールに対する極小支持集合となっている．Σ_{DB} と Δ_{DB} の定義より明らかに $\Sigma_{DB} = \bigcup_{S \in \Delta_{DB}} S$ が成り立つ．その他，Δ_{DB} について次の性質が成り立つ．

命題 5（可能支持集合の性質） PRISM プログラム DB を考える．DB が排反性，唯一性をみたすとき，またそのときに限り (i) Δ_{DB} に現れる任意の 2 つのファクト集合 S, S' は互いに排反であり，(ii) 任意の $\boldsymbol{\theta} \in \Theta$ の下で $\sum_{S \in \Delta_{DB}} P_F(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta}) = 1$ が成り立つ. ■

（証明） はじめに必要性を示す．まず，排反性条件よりすべての $G \in O_{DB}$ について $\psi_{DB}(G)$ は互いに排反な極小支持集合から成る $\cdots(*)$．また，唯一性条件より O_{DB} 中の任意の 2 つのゴール G, G' は排反である．従って命題 3（アトムが確率的に排反になるための必要十分条件）より，G

の任意の極小支持集合 S と G' の任意の極小支持集合 S' は互いに排反である $\cdots(**)$．よって $(*)$ と $(**)$ より $G \in O_{DB}, S \in \psi_{DB}(G)$ を満たす任意の S は，$G' \in O_{DB}, S' \in \psi_{DB}(G')$ を満たす任意の S' と互いに排反である（ただし $S \neq S'$）．これと Δ_{DB} の定義より (i) は成り立つ．

再び唯一性条件より，任意の $\boldsymbol{\theta} \in \Theta$ について $\sum_{G \in O_{DB}} P_{DB}(G = 1|\boldsymbol{\theta}) = 1$ が成り立つ．命題 4 より，排反性条件の下では式 4.10 が成り立つから，$\sum_{G \in O_{DB}} \sum_{S \in \psi_{DB}(G)} P_{DB}(\boldsymbol{S} = \mathbf{1}|\boldsymbol{\theta}) = \sum_{S \in \Delta_{DB}} P_{DB}(\boldsymbol{S} = \mathbf{1}|\boldsymbol{\theta}) = 1$. よって (ii) も成り立つ．十分性は以上の議論より明らかに成り立つ． □

命題 5 に基づき，排反性，唯一性条件を満たす PRISM プログラム $DB = F \cup R$ に対して 先に導入した Y_{DB} と同じように Ω_F 上の新しい確率変数 X_{DB} を導入する．X_{DB} を「DB の**支持変数**」と呼ぶことにする．

定義 17（支持変数 X_{DB}） 排反性，唯一性条件を満たす PRISM プログラム DB を考える．また，Δ_{DB} の数え上げを $S^{(1)}, S^{(2)}, \ldots, S^{(|\Delta_{DB}|)}$ とする．そのとき，X_{DB} は $\boldsymbol{S}^{(j)} = \mathbf{1}$ のとき $X_{DB} = j$ となる Ω_F 上の確率変数である．この関係を $X_{DB}(S_j) \stackrel{\text{def}}{=} j$ と書く． ■

排反性条件と唯一性条件の下では Dempster ら [17] による EM アルゴリズムの一般形（節 2.1.1）で用いられる many-to-one 関数 φ を定義することができる．

定義 18（対応関数 φ） 排反性，唯一性条件を満たす PRISM プログラム DB, および DB の観測変数 Y_{DB} と支持変数 X_{DB} を考える．$S \in \Delta_{DB}, G \in O_{DB}$ について，$S \in \psi_{DB}(G)$ かつ $x = X_{DB}(S)$ および $y = Y_{DB}(G)$ が成り立つとき，またそのときに限り $y = \varphi(x)$ とする． ■

式を簡単にするため，以降では X_{DB}, Y_{DB} をそれぞれ X, Y とおく．ここで EM アルゴリズムの形式化に基づき次の Q **関数**を定義する．Q 関数の値が ∞ にならないように排反性条件，唯一性条件と合わせて有限支持条件を仮定する．

定義 19（Q 関数） 有限支持，排反性，唯一性条件を満たす PRISM プログラム DB と観測データ $\mathcal{G}$ が与えられたとする．$\mathcal{G} = \langle G_1, G_2, \ldots, G_T \rangle$ とおいたとき，関数 $Q : \Theta \times \Theta \to \boldsymbol{R}$ を次のように定義し，DB と $\mathcal{G}$ の下での Q 関数と呼ぶ．

$$Q(\boldsymbol{\theta}', \boldsymbol{\theta}) \stackrel{\text{def}}{=} \sum_{t=1}^{T} \sum_{x : y_t = \varphi(x)} P_{DB}(X = x | Y = y_t, \boldsymbol{\theta}) \log P_{DB}(X = x, Y = y_t | \boldsymbol{\theta}') \tag{4.11}$$

■

X のとり得る値の数は $|\Delta_{DB}|$ であるので，式 4.7 に比べ，計算が簡単になっていることが分かる．そして，後に与える PRISM 用 EM アルゴリズムの正当化を行なう上で次の Q 関数の性質は重要である．これまで見てきたように，有限支持，排反性，唯一性条件の下では観測変数 Y_{DB}, 支持変数 X_{DB} を導入することにより，PRISM の EM 学習を Dempster らの形式化 [17] の中で捉えることができるので，補題 1 の証明は文献 [17, 34] に与えられているものをそのまま使えばよい．

補題 1（Q 関数と尤度の関係） 排反性，唯一性条件を満たす PRISM プログラム DB と観測データ $\mathcal{G}$, およびそれらの下での Q 関数を考える．そのとき $Q(\boldsymbol{\theta}', \boldsymbol{\theta}) \geq Q(\boldsymbol{\theta}, \boldsymbol{\theta})$ ならば $\lambda_{DB}(\boldsymbol{\theta}'; \mathcal{G}) \geq \lambda_{DB}(\mathcal{G}|\boldsymbol{\theta})$ が成り立つ． ■

ここで PRISM 用 EM アルゴリズムを記述する．この EM アルゴリズムは有限支持，排反性，唯

一性条件を満たすすべての PRISM プログラムに対して適用可能である．アルゴリズム中で計算されている値

$$\eta(i,v|\boldsymbol{\theta}) \stackrel{\text{def}}{=} \sum_{t=1}^{T} \frac{1}{P_{DB}(G_t=1|\boldsymbol{\theta})} \sum_{S\in\psi_{DB}(G_t)} P_{DB}(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(S) \tag{4.12}$$

は，観測データ $\mathcal{G}=\langle G_1,G_2,\ldots,G_T\rangle$ が与えられた下で，`msw(i,·,v)` の形をしたファクトのうち真になっているものの個数の期待値である．

定義 20（PRISM 用 EM アルゴリズム） 有限支持，排反性，唯一性条件を満たす PRISM プログラム DB と観測データ $\mathcal{G}$ を入力とし，パラメータ $\boldsymbol{\theta}$ を出力とする，図 4.2 の手続き Learn-PRISM-Naive を PRISM 用 EM アルゴリズムとする． ■

先に導入した確率変数 X_{DB}, Y_{DB} は Learn-PRISM-Naive の導出のために一時的に利用したもので，導出されたアルゴリズムの中には現れなくなる点に注意する．Learn-PRISM-Naive は次の定理によって正当化される．

定理 1（PRISM 用 EM アルゴリズムの正当性） 有限支持，排反性，唯一性条件を満たす PRISM プログラム DB と観測データ $\mathcal{G}$ を考える．そのとき，手続き Learn-PRISM-naive は尤度 $\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta})$ を局所的に最大にする $\boldsymbol{\theta}\in\Theta$ を出力する． ■

（証明） 式 4.11 の Q 関数を以下のように変形する．

$$\begin{aligned}
Q(\boldsymbol{\theta}',\boldsymbol{\theta}) &= \sum_{t=1}^{T}\sum_{x:y_t=\varphi(x)} P_{DB}(X=x|Y=y_t,\boldsymbol{\theta})\log P_{DB}(X=x,Y=y_t|\boldsymbol{\theta}') \\
&= \sum_{t=1}^{T}\frac{1}{P_{DB}(Y=y_t|\boldsymbol{\theta})}\sum_{x:y_t=\varphi(x)} P_{DB}(X=x,Y=y_t|\boldsymbol{\theta})\log P_{DB}(X=x,Y=y_t|\boldsymbol{\theta}') \\
&= \sum_{t=1}^{T}\frac{1}{P_{DB}(Y=y_t|\boldsymbol{\theta})}\sum_{x:y_t=\varphi(x)} P_{DB}(X=x|\boldsymbol{\theta})\log P_{DB}(X=x|\boldsymbol{\theta}') \\
&= \sum_{t=1}^{T}\frac{1}{P_{DB}(G_t=1|\boldsymbol{\theta})}\sum_{S\in\psi_{DB}(G_t)} P_{DB}(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})\log P_{DB}(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta}') \\
&= \sum_{t=1}^{T}\frac{1}{P_{DB}(G_t=1|\boldsymbol{\theta})}\sum_{S\in\psi_{DB}(G_t)} P_{DB}(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})\log\prod_{i\in I,v\in V_i}\{\theta'_{i,v}\}^{\sigma_{i,v}(S)} \qquad \left(\text{式 4.9 より}\right) \\
&= \sum_{t=1}^{T}\frac{1}{P_{DB}(G_t=1|\boldsymbol{\theta})}\sum_{S\in\psi_{DB}(G_t)} P_{DB}(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})\left(\sigma_{i,v}(S)\sum_{i,v}\log\theta'_{i,v}\right) \\
&= \sum_{i,v}\left(\sum_{t=1}^{T}\frac{1}{P_{DB}(G_t=1|\boldsymbol{\theta})}\sum_{S\in\psi_{DB}(G_t)} P_{DB}(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(S)\right)\log\theta'_{i,v} \\
&= \sum_{i,v}\eta(i,v|\boldsymbol{\theta})\log\theta_{i,v} \;=\; \sum_{i}\left(\sum_{v}\eta(i,v|\boldsymbol{\theta})\log\theta'_{i,v}\right) \\
&\leq \sum_{i}\left(\sum_{v}\eta(i,v|\boldsymbol{\theta})\log\frac{\eta(i,v|\boldsymbol{\theta})}{\sum_{v'\in V_i}\eta(i,v'|\boldsymbol{\theta})}\right)
\end{aligned}$$

ただし $\eta(i,v|\boldsymbol{\theta}) \stackrel{\text{def}}{=} \sum_{t=1}^{T} \frac{1}{P_{DB}(G_t=1|\boldsymbol{\theta})} \sum_{S\in\psi_{DB}(G_t)} P_{DB}(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(S)$ である．上より

$$\theta' = \frac{\eta(i,v|\boldsymbol{\theta})}{\sum_{v'\in V_i}\eta(i,v'|\boldsymbol{\theta})}$$

とおけば，任意の $\boldsymbol{\theta}''$ に対して $Q(\boldsymbol{\theta}',\boldsymbol{\theta}) \geq Q(\boldsymbol{\theta}'',\boldsymbol{\theta})$ が成り立ち，従って $Q(\boldsymbol{\theta}',\boldsymbol{\theta}) \geq Q(\boldsymbol{\theta},\boldsymbol{\theta})$ が成り立つ．Learn-PRISM-Naive で更新されるパラメータの列を $\boldsymbol{\theta}^{(0)},\boldsymbol{\theta}^{(1)},\boldsymbol{\theta}^{(2)},\ldots$ とおくと，Learn-PRISM-Naive における処理の様子と補題 1 より対数尤度は $\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta}^{(0)}) \leq \lambda_{DB}(\mathcal{G}|\boldsymbol{\theta}^{(1)}) \leq \lambda_{DB}(\mathcal{G}|\boldsymbol{\theta}^{(2)}) \leq \ldots$ と単調増加することが分かる．すべての $t=1\ldots T$ について $P_{DB}(G_t=1|\boldsymbol{\theta}^{(0)})$ なる $\boldsymbol{\theta}^{(0)}$ を選べば，$m=1,2,\ldots$ について $\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta}^{(m)})$ は有界である．よって $\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta}^{(m)})$ は必ず収束し，収束先は $\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta})$ を局所的に最大にする $\boldsymbol{\theta}^*$ である． □

4.2 高速な EM アルゴリズム

前節で述べた PRISM 用 EM アルゴリズムは有限支持，排反性，唯一性条件を満たすすべての PRISM プログラムに対して適用可能なものであった．しかし，手続き Learn-PRISM-Naive の第 10, 11 行における $\eta(i,v|\boldsymbol{\theta}^{(m)})$ の最悪計算量は $\max_{t=1\ldots T}|\psi_{DB}(G_t)|$ である．例えば，先に触れたように HMM プログラムの場合 $|\psi_{DB}(\texttt{hmm([a,b,a])})|$ は出力・受理記号列長 L に対して指数オーダである．従って，HMM プログラムに対して Learn-PRISM-Naive を現実的な時間で実行させるのは一般に難しい．

本節では HMM の Baum-Welch アルゴリズム（L に対して線形オーダで計算できる）などと等価で同等の計算効率をもつ PRISM プログラム用 EM アルゴリズムを示す．この EM アルゴリズムを我々は**グラフィカル** *EM* **アルゴリズム**（以下，gEM アルゴリズム）と呼ぶ．すなわち，gEM アルゴリズムは PRISM で HMM を記述すると Baum-Welch アルゴリズムになり，PRISM で PCFG を記述すると Inside-Outside アルゴリズムになる．しかし，gEM アルゴリズムは先に

1: **procedure** Learn-PRISM-Naive($DB,\mathcal{G}$) **begin**
2: /* パラメータを初期化 */
3: Select some $\boldsymbol{\theta}$ from Θ such that $P_{DB}(G_t=1|\boldsymbol{\theta}) > 0$ for all $t=1\ldots T$;
4: $\lambda^{(0)} := \sum_{t=1}^{T} \log P_{DB}(G_t=1|\boldsymbol{\theta}^{(0)})$; /* 式 4.9 で計算する */
5: $m := 0$;
6: **repeat**
7: **foreach** $i\in I, v\in V_i$ **do** /* E ステップ */
8: $\eta(i,v|\boldsymbol{\theta}^{(m)}) := \sum_{t=1}^{T} \frac{1}{P_{DB}(G_t=1|\boldsymbol{\theta}^{(m)})} \sum_{S\in\psi_{DB}(G_t)} P_F(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta}^{(m)})\sigma_{i,v}(S)$;
9: $m := m+1$;
10: **foreach** $i\in I, v\in V_i$ **do** /* M ステップ */
11: $\theta_{i,v}^{(m)} := \frac{\eta(i,v|\boldsymbol{\theta}^{(m)})}{\sum_{v'\in V_i}\eta(i,v'|\boldsymbol{\theta}^{(m)})}$;
12: $\lambda^{(m+1)} := \sum_{t=1}^{T} \log P_{DB}(G_t=1|\boldsymbol{\theta}^{(m)})$
13: **until** $\lambda^{(m)} - \lambda^{(m-1)}$ becomes sufficiently small; /* 対数尤度が収束したらループを終了 */
14: **return** $\boldsymbol{\theta}^{(m)}$ /* 局所的な最尤推定値 $\boldsymbol{\theta}$ を返す */
15: **end**.

図 4.2: PRISM 用 EM アルゴリズム Learn-PRISM-Naive.

議論したような PRISM プログラムすべてに対して働くわけではない．次節では gEM アルゴリズムが適用できるための条件を示す．

4.2.1 グラフィカル EM アルゴリズムが適用できるための条件

ルール R の完備化 $comp(R)$ に基づく式 4.1 の関係式は観測事象であるゴールと基本事象である `msw(·,·,·)` 間の関係を定めていた．ここで我々は中間事象を表現する**テーブル述語** (table predicate) と**テーブルアトム** (table atom) を導入する．テーブル述語はあらかじめユーザが PRISM プログラム DB 中で指定するものとする．「テーブル」という接頭辞は後に述べる OLDT 探索に由来する．テーブルアトムは HMM の Baum-Welch アルゴリズムにおける trellis 図中のノード，PCFG の Inside-Outside アルゴリズムにおける部分木を一般化したものであり，後に記述する gEM アルゴリズムはテーブルアトムを利用して動的計画法に基づき PRISM プログラム中のパラメータ $\boldsymbol{\theta}$ を推定する．

定義 21（テーブル述語とテーブルアトム） PRISM プログラム DB でユーザによって指定されたテーブル述語の集合を $table(DB)$ とおく．また，$table(DB)$ 中の述語をもつ（基底）アトムをテーブルアトムと呼び，その集合を τ^*_{DB} とおく． ■

更に準備として独立なファクト集合および独立な DB 中のアトム（すなわちファクトまたは $head(R)$ のアトム）について定義し，PRISM プログラムの P_F の条件の下での必要十分条件（明らかなので証明は略）を得ておく．

定義 22（独立なファクト集合） PRISM プログラム $DB = F \cup R$ を考える．また，空でなく，かつ確率的に無矛盾な 2 つのファクト集合 $S_1, S_2 \subseteq F$ を考える．任意のパラメータ $\boldsymbol{\theta} \in \Theta$ および $x_1 \in \{0,1\}^{|S_1|}$, $x_2 \in \{0,1\}^{|S_2|}$ について $P_F(\boldsymbol{S}_1 = \boldsymbol{x}_1, \boldsymbol{S}_2 = \boldsymbol{x}_2 | \boldsymbol{\theta}) = P_F(\boldsymbol{S}_1 = \boldsymbol{x}_1 | \boldsymbol{\theta}) \cdot P_F(\boldsymbol{S}_2 = \boldsymbol{x}_2 | \boldsymbol{\theta})$ が成り立つとき，S_1 と S_2 は独立であるという． ■

定義 23（独立なアトム） PRISM プログラム $DB = F \cup R$ と DB に出現するアトム B_1, B_2 を考える．任意のパラメータ $\boldsymbol{\theta} \in \Theta$ および $y_1, y_2 \in \{0,1\}$ について $P_{DB}(B_1 = y_1, B_2 = y_2 | \boldsymbol{\theta}) = P_{DB}(B_1 = y_1 | \boldsymbol{\theta}) \cdot P_{DB}(B_2 = y_2 | \boldsymbol{\theta})$ が成り立つとき，B_1 と B_2 は独立であるという． ■

命題 6（ファクト集合が独立になるための必要十分条件） PRISM プログラム $DB = F \cup R$ を考える．また，空でなく，かつ確率的に無矛盾な 2 つのファクト集合 $S_1, S_2 \subseteq F$ を考える．$S_1 \cap S_2 = \emptyset$ であり，かつ S_1 と S_2 が確率的排反でないとき，またそのときに限り S_1 と S_2 は独立である． ■

命題 7（アトムが独立になるための必要十分条件） PRISM プログラム $DB = F \cup R$ と DB に出現するアトム B_1, B_2 を考える．B_1 の任意の極小支持集合 $S_1 \in \psi_{DB}(B_1)$ と B_2 の任意の極小支持集合 $S_2 \in \psi_{DB}(B_2)$ について S_1 と S_2 が独立であるとき，またそのときに限り B_1 と B_2 は独立である． ■

我々はこれまで PRISM プログラム DB に分離条件，有限支持条件，排反性条件，唯一性条件を仮定してきた．gEM アルゴリズムを使うためには，排反性を強め，更に**非循環支持条件** (acyclic support condition) と**独立支持条件** (independent support condition) を加える必要がある．もちろん，gEM アルゴリズムを使わないのならこの 2 つの条件は必要ない．式 4.1 と同様に $comp(R)$

を考える．

仮定 6（非循環支持条件と独立支持条件） 有限支持条件を満たす PRISM プログラム DB を考える．$t = 1 \ldots T$ について次の条件を満たすような τ^*_{DB} の 順序付きの 有限部分集合 $\tau^{(t)}_{DB} = \langle \tau^{(t)}_1, \tau^{(t)}_2, \ldots, \tau^{(t)}_{K_t} \rangle$ が存在するとする：

$$\begin{aligned} comp(R) \quad \models \quad & \left(G_t \leftrightarrow \tilde{S}^{(t)}_{0,1} \vee \cdots \vee \tilde{S}^{(t)}_{0,m_0} \right) \\ & \wedge \left(\tau^{(t)}_1 \leftrightarrow \tilde{S}^{(t)}_{1,1} \vee \cdots \vee \tilde{S}^{(t)}_{1,m_1} \right) \wedge \cdots \wedge \left(\tau^{(t)}_{K_t} \leftrightarrow \tilde{S}^{(t)}_{K_t,1} \vee \cdots \vee \tilde{S}^{(t)}_{K_t,m_{K_t}} \right) \end{aligned} \tag{4.13}$$

ただし，G_t を特別なテーブルアトム $\tau^{(t)}_0$ と見なしたとき，$k = 0 \ldots K_t$ について $\tilde{S}^{(t)}_{k,1}, \ldots, \tilde{S}^{(t)}_{k,m_k}$ の各々は $F \cup \{\tau^{(t)}_{k+1}, \ldots, \tau^{(t)}_{K_t}\}$ の部分集合であり**（非循環支持条件）**，かつ独立な基底アトムから成る**（独立支持条件）**．また，$\tilde{S}^{(t)}_{k,1}, \ldots, \tilde{S}^{(t)}_{k,m_k}$ はいずれも確率的に無矛盾であり，かつ確率的に他と排反である**（強排反性条件）**．$\tilde{S}^{(t)}_{k,1}, \ldots, \tilde{S}^{(t)}_{k,m_k}$ の各々は $\tau^{(t)}_k$ に対する**テーブル化された極小支持集合**（以下，t-極小支持集合と略す）と呼ばれる．t-極小支持集合の集合を $\tilde{\psi}_{DB}(\tau^{(t)}_k) \stackrel{\text{def}}{=} \{\tilde{S}^{(t)}_{k,1}, \ldots, \tilde{S}^{(t)}_{k,m_k}\}$ とおく． ■

有限支持条件を仮定しているので，各 m_k は有限で $(k = 0 \ldots K_t)$，G_t および $\tau^{(t)}_1, \ldots, \tau^{(t)}_{K_t}$ に対する t-説明はいずれも有限集合である．また，強排反性条件を満たす PRISM プログラムは排反性条件も満たすことに注意する．例えば，HMM プログラムにおいて $G_t =$ `hmm([a,b,a])` を観測したとき $(\exists t = 1 \ldots T)$，式 4.14 は次のような形になる．

$$\begin{aligned} comp(R) \quad \models \quad & \left(\begin{array}{l} \texttt{hmm([a,b,a])} \leftrightarrow \quad \left(\texttt{msw(init,once,q0)} \wedge \texttt{hmm(1,q0,[a,b,a])} \right) \\ \qquad\qquad\qquad\quad \vee \left(\texttt{msw(init,once,q1)} \wedge \texttt{hmm(1,q1,[a,b,a])} \right) \end{array} \right) \\ & \wedge \left(\begin{array}{l} \texttt{hmm(1,q0,[a,b,a])} \leftrightarrow \\ \quad \left(\texttt{msw(out(q0),1,a)} \wedge \texttt{msw(tr(q0),1,q0)} \wedge \texttt{hmm(2,q0,[b,a])} \right) \\ \vee \left(\texttt{msw(out(q0),1,a)} \wedge \texttt{msw(tr(q0),1,q1)} \wedge \texttt{hmm(2,q1,[b,a])} \right) \end{array} \right) \\ & \wedge \ldots \end{aligned} \tag{4.14}$$

そして，$\tau^{(t)}_{DB} = \langle$ `hmm([a,b,a])`, `hmm(1,q0,[a,b,a])`, `hmm(1,q1,[a,b,a])`, `hmm(2,q0,[b,a])`, $\ldots \rangle$ となり，また

$$\begin{aligned} \tilde{\psi}_{DB}\left(\texttt{hmm([a,b,a])}\right) \quad &= \quad \Big\{ \{ \texttt{msw(init,once,q0)}, \texttt{hmm(1,q0,[a,b,a])} \} \\ & \qquad \{ \texttt{msw(init,once,q1)}, \texttt{hmm(1,q1,[a,b,a])} \} \Big\} \\ \tilde{\psi}_{DB}\left(\texttt{hmm(1,q0,[a,b,a])}\right) \quad &= \quad \Big\{ \{ \texttt{msw(out(q0),1,a)} \wedge \texttt{msw(tr(q0),1,q0)} \wedge \texttt{hmm(2,q0,[b,a])} \} \\ & \qquad \{ \texttt{msw(out(q0),1,a)} \wedge \texttt{msw(tr(q0),1,q1)} \wedge \texttt{hmm(2,q1,[b,a])} \} \Big\} \end{aligned} \tag{4.15}$$

などが成り立つ．上の仮定で用いた $\tilde{\psi}_{DB}$ と $\tau^{(t)}_{DB}$ $(t = 1 \ldots T)$ をどのように求めるかが次の問題になる．一つの解決として我々は Tamaki と Sato によって提案された **OLDT** (*Ordered Linear resolution for Definite clauses with Tabulation*) [58, 63] を用いることにする．

4.2.2 OLDT 探索

OLDT は任意の論理プログラムに適用可能な汎用の証明手続きである．OLDT を実現する OLDT インタプリタは**解テーブル** (solution table) と**参照テーブル** (lookup table) をもち，与えられたゴール G に対してにサブゴールとその解の組を解テーブルに格納しながら G の全解探索を行う．途中ですでに出現したサブゴール G'（その時点では解が見つかっているとは限らない）のインスタンス G'' を見つけた場合はそれを参照テーブルに登録しておく．もし G' の解が見つかったら，その解を DB 中に追加された単位節と見なし，G'' と resolution を行う．最初に出現したサブゴール G' が解を生み出す生産者に相当し，後に出現する G' のインスタンス G'' はその解を消費するだけの消費者と見ることができる．このような生産と消費を SLD 木（resolution の対象になるのが最左アトムに固定されているので正確には OLD 木）上で繰り返す．

G'' が G' のインスタンスであるとき，G'' の探索空間は G' の探索空間の一部となる．従って，G' で解を生産し，G'' でその解を消費するという「生産・消費」手続きは健全であることが分かる．また，通常の Prolog インタプリタ（resolution の対象になるのが最左アトムに固定）では G'' が G' の子孫であり，かつ variant であるときプログラムが停止しなくなってしまうが，OLDT では G'' が単なる消費者となり，それ以上探索が行われないので Prolog インタプリタでは停止しないようなプログラムでも OLDT インタプリタでは停止する場合がある．また，multi-stage depth-first という探索戦略を用いたときの OLDT の完全性，定数集合が有限で関数記号なしの論理プログラム[1] に対する任意の探索戦略における完全性および停止性が証明されている [58]．確率の文脈で考えると，すべての解を見つけなければ正しい確率計算が出来なくなるので完全性は重要な性質である．

加えて，一旦探索したサブゴール G' の解はテーブルに保存しておき，そのインスタンスに対しては再探索（再計算）を行わないため，OLDT は副次的に（効率的な構文解析器のような）効率的な全解探索手続きとなっている．PRISM プログラムの効率的な EM 学習を目標とする我々はこの副次的な性質，すなわち OLDT では再計算が行われないことに注目し，この性質を次の 2 点で利用する．

1. 全解探索器として利用する．
2. 全解探索した後の解テーブルから gEM アルゴリズムの計算に必要なデータ構造を抽出する．

すなわち，我々は OLDT を単に全解探索器として使うだけでなく，探索途中の結果も後の gEM アルゴリズムで利用する．また，2. で抽出されるデータ構造を**支持グラフ** (support graph) と呼ぶ．gEM アルゴリズムはこの支持グラフのコンパクトさを利用し，効率的に動作する．

4.2.3 支持グラフの獲得

支持グラフは仮定 6 で与えた 順序集合 $\tau_{DB}^{(t)}$ $(t = 1 \ldots T)$ および $\tilde{\psi}_{DB}$ をグラフで表現したものである．支持グラフは次節で記述することにし，ここでは基となる $\tau_{DB}^{(t)}$ および $\tilde{\psi}_{DB}$ を OLDT によって獲得する方法を記述する．

はじめに，我々は PRISM プログラムを対応する別の論理プログラムに翻訳する．この翻訳は確定節文法（definite clause grammar; 以下 DCG）[44, 57] や Poole の仮説推論システム Theorist [45] で行われている翻訳と同様に，差分リストを形成する 2 つの引数を各アトムに付け加える．この差分リストに t-極小支持集合が格納される．例として HMM プログラム（節 3.3.1）と翻訳後のプ

[1] Datalog と呼ばれることがある．

```
(1) target(hmm/1).
(2) data('hmm.dat').
(3) table([hmm/1,hmm/3]).


(4) values(init,[q0,q1]).

(5) values(out(_),[a,b]).

(6) values(tr(_),[q0,q1]).

(7) hmm(Cs):-
        msw(init,once,Qi),
        hmm(1,Qi,Cs).
(8) hmm(L,Q,[C|Cs]):-
        L =< 3,
        msw(out(Q),L,C),
        msw(tr(Q),L,NextQ),
        L1 is L+1,
        hmm(L1,NextQ,Cs).
(9) hmm(L,_,[]):- L>3.
```

```
(T1)  top_hmm(Z,X):- tab_hmm(Z,X,[]).
(T2)  （対応する節は翻訳されない）
(T3)  tab_hmm(Z,[hmm(Z)|X],X):- hmm(Z,_,[]).
(T3') tab_hmm(Z1,Z2,Z3,[hmm(Z1,Z2,Z3)|X],X):-
              hmm(Z1,Z2,Z3,_,[]).
(T4)  e_msw(init,Y,q0,[msw(init,Y,q0)|X],X).
(T4') e_msw(init,Y,q1,[msw(init,Y,q1)|X],X).
(T5)  e_msw(out(Z),Y,a,[msw(out(Z),Y,a)|X],X).
(T5') e_msw(out(Z),Y,b,[msw(out(Z),Y,b)|X],X).
(T6)  e_msw(tr(Z),Y,q0,[msw(tr(Z),Y,q0)|X],X).
(T6') e_msw(tr(Z),Y,q1,[msw(tr(Z),Y,q1)|X],X).
(T7)  hmm(Cs,X0,X1):-
          e_msw(init,once,Qi,X0,X2),
          tab_hmm(1,Qi,Cs,X2,X1).
(T8)  hmm(L,Q,[C|Cs],X0,X1):-
          L =< 3,
          e_msw(out(Q),L,C,X0,X2),
          e_msw(tr(Q),L,NextQ,X2,X3),
          L1 is L+1,
          tab_hmm(L1,NextQ,Cs,X3,X1).
(T9)  hmm(L,_,[],X,X):- L>3.
```

図 4.3: （左）翻訳前の HMM プログラム,（右）翻訳後の HMM プログラム.

ログラムを図 4.3 に並べて示す．節 (Tj) と節 (Tj') はそれぞれ元の節 (j) から翻訳されたものである．また，翻訳後のプログラムでも対応する変数には同じ変数名を与えているが，X, Y, Z で始まる変数は翻訳中に自動的に付け加えられたものである．特に，先ほど述べた差分リストを形成する引数では X で始まる変数を使っている．観測ゴール hmm(L) （L は a または b から成る長さ 3 のリスト）に対して，OLDT では top_hmm(L,X) をトップゴールとする[2]．X には観測ゴール hmm(L) の解が返ってくる．また，p/n がテーブル述語であると指定された場合，翻訳されたプログラムでは $p/(n+2)$ がテーブル述語になる．その他，2 つの点に注意する．

- テーブルアトムを差分リストに格納するために，テーブル述語 p/n に対応して，tab_$p/(n+2)$ という述語に関する節が（節 (T3), (T3') のように）翻訳プログラムに加えられる．
- 元プログラムにおける述語引数間の束縛関係はそのまま保存される．

前者を説明する．図 4.3 の節 (T7) や (T8) を見て分かるように，元のプログラムにおけるテーブルアトム $p(X_1,X_2,\ldots,X_n)$ の呼び出しが翻訳後のプログラムでは tab_$p(X_1,X_2,\ldots,X_n$,X0,X1) に置き換えられている（X0, X1 は差分リストのために付け加えられた変数であるが，説明のため X0, X1 と名付けただけで，図 4.3 中の X0, X1 とは無関係である）．そして，節 (T3), (T3') から分かるように，その後 tab_$p(X_1,X_2,\ldots,X_n$,X0,X1) から $p(X_1,X_2,\ldots,X_n$,_,[]) が呼び出され，$p(X_1,X_2,\ldots,X_n)$ が証明された（成功した）とき，X0 は [$p(X_1,X_2,\ldots,X_n)$|X1] と単一化する．以上より，元プログラムでテーブルアトム $p(X_1,X_2,\ldots,X_n)$ が証明されれば，翻訳後のプログラムにおいて差分リストに $p(X_1,X_2,\ldots,X_n)$ が格納される．一方，後者については図 4.3 の例より明らかである．

[2] トップゴールの述語 hmm/1 もテーブル述語にしている（これは必須である）ので，その解は解テーブルに保存される．また，テーブル述語については途中の解はすべて解テーブルに保存されるので，図 4.3 の節 (T3) のように後ろから 2 つ目の変数を無名変数 (anonymous variable) にしても解は失われない．この点は DCG や Theorist の場合とは異なる．

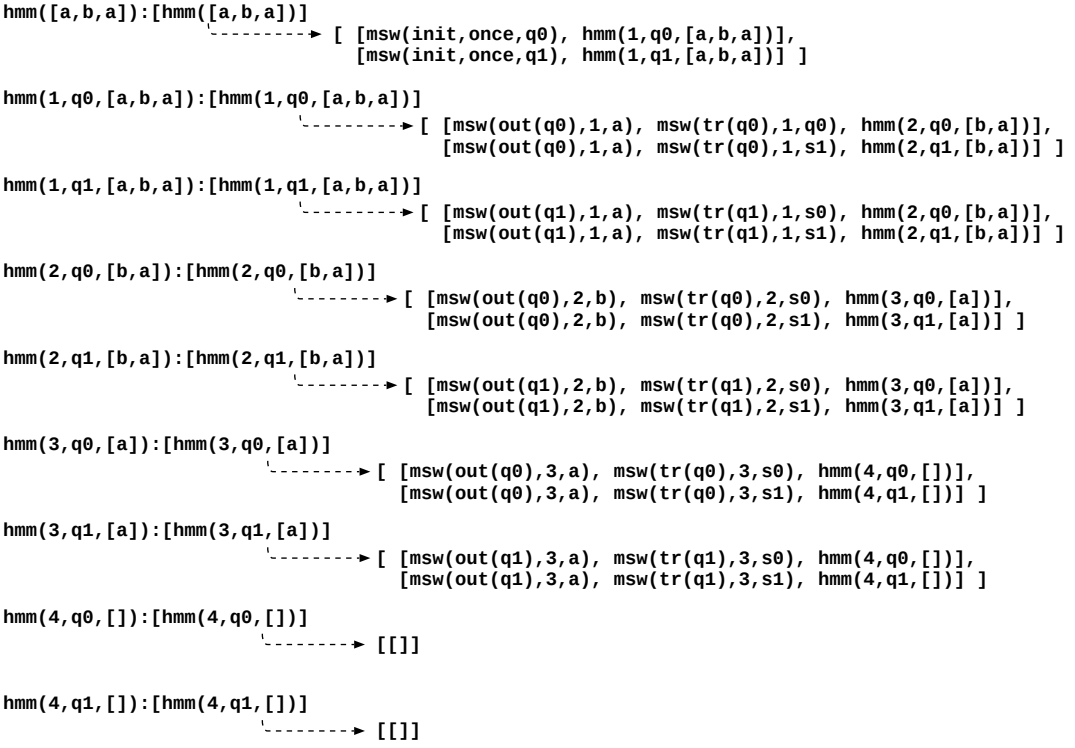

図 4.4: HMM プログラムに OLDT を適用した後の解テーブル.

このように翻訳したプログラムに対して OLDT 探索を行う．そのときに我々は元の OLDT 探索手続きに手を加え,「解テーブル中の部分解にその t-極小支持集合のリストを付与する」ことにする．また，翻訳によって付け加えられた差分リストが翻訳前のプログラムの OLDT 探索に影響を与えない，すなわち探索空間が翻訳前と翻訳後で同じになる点に注意する．図 4.4 は HMM プログラムの翻訳に OLDT を適用し，すべて解を求め終わった後の解テーブルである．解 τ からその t-極小支持集合にポインタ（点線矢印）が張られている点に注意する．先述したように，我々はこの解テーブルから $\tau_{DB}^{(t)}$ $(t = 1 \ldots T)$ と $\tilde{\psi}_{DB}$ を抽出する．まず，これまで説明した PRISM プログラムの翻訳方法と OLDT 探索の仕方から次の命題が成り立つ.

命題 8（解テーブルからの $\tilde{\psi}_{DB}$ の抽出） 有限支持，強排反性，非循環支持，独立支持条件を満たす PRISM プログラム DB とその翻訳 DB' を考える．DB' とゴール G に対して OLDT を適用し，OLDT 終了後に得られる解テーブルを $\mathcal{T}$ とする．そのとき，$\mathcal{T}$ における各部分解 τ（G の解を含む）と τ に付与されたリスト中の t-極小支持集合を $\tilde{S}_1, \tilde{S}_2, \ldots, \tilde{S}_m$ とすると，$\tilde{\psi}_{DB}(\tau) = \{\tilde{S}_1, \tilde{S}_2, \ldots, \tilde{S}_m\}$ が成り立つ. ■

PRISM プログラム DB の観測アトム $p(X_1,\ldots,X_n)$ に対して，その翻訳プログラムのトップゴール `top_`$p(X_1,\ldots,X_n$`,X)` に OLDT を適用すると，最後の引数 `X` に元プログラムにおける観測アトム $p(X_1,\ldots,X_n)$ の解が返ってくる．次に，トポロジカルソーティング (topological sorting) [19] を行えばよい．一般にトポロジカルソーティングは半順序関係 $\prec$ が与えられたときに $\prec$ を満たすような全順序関係 $\prec_{\text{total}}$ を定めるものである．もともと我々はプログラム DB に非循環支持条件（仮定 6）を仮定しているので，トポロジカルソーティングは必ず成功し，全順序関係の順に並べた $\tau_{DB}^{(t)}$ を返す.

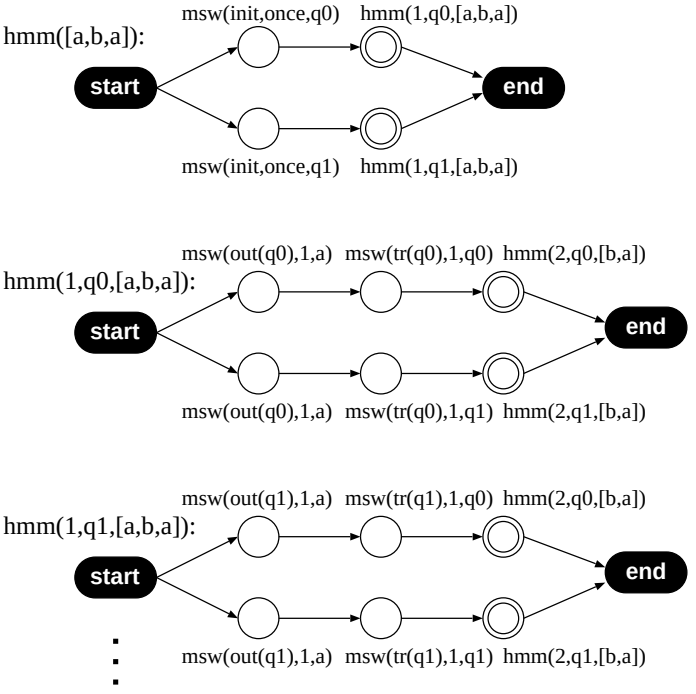

図 4.5: HMM プログラムにおいて観測 `hmm([a,b,a])` に対する支持グラフ.

4.2.4 支持グラフの性質

$\tau_{DB}^{(t)}$ $(t = 1 \ldots T)$ と $\tilde{\psi}_{DB}$ を支持グラフというデータ構造で捉えると gEM アルゴリズムが理解しやすくなる[3]. 支持グラフは各観測ゴール G_t ごとに用意される $(t = 1 \ldots T)$. 式を簡単にするため, 本節では添字の $\cdot^{(t)}$ や $\cdot_t$ を省略する.

まず, 前節で示した HMM プログラムのゴール $G =$ `hmm([a,b,a])` の場合 (式 4.14, 4.15, 図 4.4 参照) に得られる支持グラフを図 4.5 に示す. 支持グラフは**再帰遷移ネットワーク** (recursive transition network; 以下 RTN) に似た構造をもつ非循環有向グラフ (DAG) であり, 共通の辺をもたない幾つかの部分グラフの集まりである. 部分グラフは各テーブルアトム $\tau \in \tau_{DB}$ ごとに用意され, その各々を τ の**部分支持グラフ**と呼び, $\langle \tau, \tilde{\psi}_{DB}(\tau) \rangle$ で参照する. τ の部分支持グラフは $comp(R)$ に基づく τ の同値式 $comp(R) \models \tau \leftrightarrow \tilde{S}_1 \vee \tilde{S}_2 \vee \cdots \vee \tilde{S}_m$ (式 4.14 参照) の表現と見ることができる.

各部分支持グラフは**開始ノード**, **終了ノード**と呼ばれる 2 つの特殊なノードをもち (図 4.5 では各々 `start`, `end` とラベルづけされている), 各 $\tilde{S} \in \tilde{\psi}_{DB}(\tau)$ に対して開始ノード, $\tilde{S}$ の各要素がラベルづけされたノード, 終了ノードが一列に連結されている. 有向辺はすべて開始ノードから終了ノードに向かっている. 開始ノードから終了ノードに至るパスを**局所パス**と呼び, $\tilde{S}$ で参照する. 局所パスにおいて, ファクト `msw(`i`,`n`,`v`)` のラベルをもつノードを**ファクトノード**, テーブルアトム τ' のラベルをもつノードを**テーブルノード**と呼ぶ. 図 4.5 ではそれぞれ ○ と ◎ で表している. また, 支持グラフ中には同じラベルをもつ複数のノードが存在し得る点に注意する. τ の部分支持グラフの開始ノード (終了ノード) を単に τ の開始ノード (終了ノード) と呼ぶことがある. 支持グラフは次の特徴をもつ.

[3] 「グラフィカル」という単語が頭につくのはこの点に由来する.

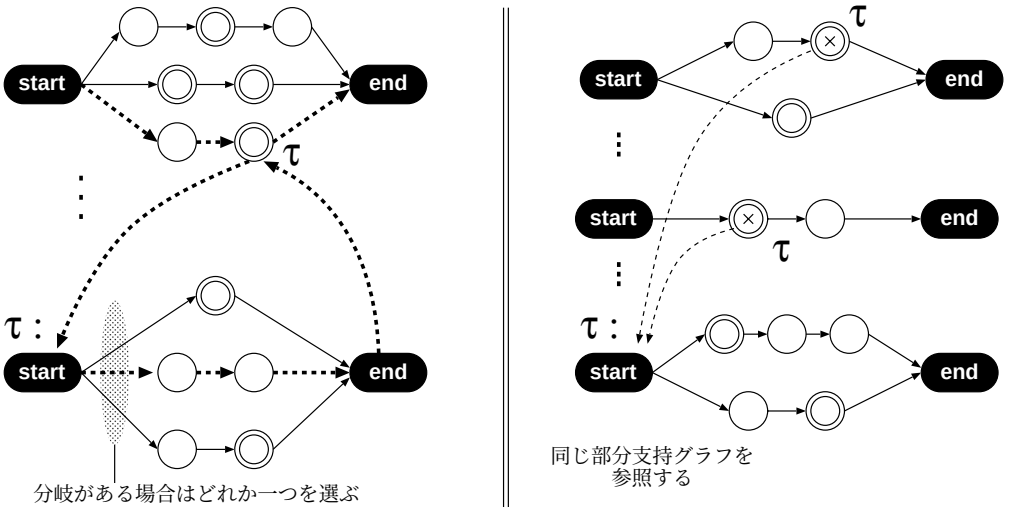

図 4.6: （左）支持グラフの再帰的巡回,（右）再帰的巡回パスの共有.

1. 支持グラフに対して RTN のように再帰的な巡回を行なうことができる.
2. 複数の巡回パスの一部が共有される.
3. $\tau_{DB} = \langle \tau_1, \tau_2, \ldots \rangle$, $\tau_0 = G$ とおく. $k = 0 \ldots K$ について τ_k の部分支持グラフにラベル τ' をもつテーブルノードが出現するとき, $\tau' = \tau_{k'}$ とおくと $k < k'$ である.

1 つ目の特徴である再帰的な巡回は次のようにして行なわれる. G の開始ノードから出発し, 辺に沿って各ノードを訪問していくが, 途中に τ がラベルづけされているテーブルノードがあったら, 部分支持グラフ τ の開始ノードにジャンプする. そしてその終了ノードに至ったらジャンプ元のノードに戻る（図 4.6 左の点線部分）. これを再帰的に繰り返し, G の終了ノードに至ったら一回の巡回を終了する. 分岐がある場合はその中のどれかを選ぶ. このような巡回の途中でファクトノードにされる `msw(`i`,`n`,`v`)` を集めると G の極小支持集合 S が得られる ($S \in \psi_{DB}(G)$). 再帰的巡回を全通り行なえば G の極小支持集合をすべて見つけることができる.

2 つ目の特徴が得られるのは, 同じラベル τ をもつノードでは同じ（τ の）部分支持グラフにジャンプするためである（図 4.6 右の $\times$ 印のノード）. このような共有構造により支持グラフのサイズが圧縮される. 従って, 我々は gEM アルゴリズムをこの支持グラフの上で動作させることによって効率的な確率計算を実現する.

3 つ目の特徴は PRISM プログラムの非循環支持条件より得られる. 後に述べるように, この特徴により我々は gEM アルゴリズムを動的計画法に基づいて設計できる.

4.2.5 グラフィカル EM アルゴリズム

定理 1 によって, 手続き Learn-PRISM-Naive （図 4.2）は有限支持, 排反性, 唯一性条件を満たす PRISM プログラムの（局所的な）最尤推定アルゴリズムであることが示された. しかし, パラメータ $\boldsymbol{\theta}$ と観測データ $\mathcal{G}$ が与えられた下での `msw(`i`,`$\cdot$`,`v`)` の出現回数の条件つき期待値（以降, **期待出現回数**という）$\eta(i, v|\boldsymbol{\theta})$ の計算時間が例えば HMM の出力・受理記号列長 L の指数オーダになるため, 一般には Learn-PRISM-Naive の現実時間での実行は困難であった.

本節で示す gEM アルゴリズムは, 上記の条件に加え, 強排反性, 非循環支持, 独立支持条件

```
1: procedure LEARN-GEM (DB, G) begin
2:     Select some θ from Θ such that P_DB(G_t=1|θ) > 0 for all t = 1...T;
3:     GET-INSIDE-PROBS(DB, G);
4:     λ^(0) := Σ_{t=1}^{T} log P[t, G_t];
5:     repeat
6:         GET-EXPECTATIONS(DB, G);
7:         foreach i ∈ I, v ∈ V_i do
8:             θ_{i,v} := η[i, v]/ Σ_{v'∈V_i} η[i, v'];
9:         GET-INSIDE-PROBS(DB, G);
10:        λ^(m) := Σ_{t=1}^{T} log P[t, G_t]
11:    until λ^(m) − λ^(m−1) becomes sufficiently small;
12:    return θ
13: end.
```

図 4.7: gEM アルゴリズムのメインルーチン LEARN-GEM.

を満たす PRISM プログラムに適用可能な EM アルゴリズムである．同じ PRISM プログラムと観測データを与えたとき，LEARN-PRISM-NAIVE と gEM アルゴリズムは等価である．すなわち m 回目のパラメータ更新後のパラメータが両者で等しく $\boldsymbol{\theta}^{(m)}$ であったとき，更に更新して得られたパラメータ $\boldsymbol{\theta}^{(m+1)}$ もまた両者で等しくなることが証明できる．また，前節で述べた支持グラフのコンパクトさを利用して効率的に期待値 $\eta(i, v|\boldsymbol{\theta})$ の計算を行なうことができる．特に，PRISM で HMM を記述したとき，gEM アルゴリズムの計算量と HMM 専用 EM アルゴリズムである Baum-Welch アルゴリズムの計算量が同じオーダになる．同様に PCFG を記述すれば，gEM アルゴリズムと Inside-Outside アルゴリズムが同じ計算量となる．

Inside-Outside アルゴリズムと同様に，gEM アルゴリズムでも**内側確率**，**外側確率**という 2 つの確率値の計算が中心になる．各々の確率値は $\mathcal{P}[t, \tau^{(t)}]$, $\mathcal{Q}[t, \tau^{(t)}]$ という配列変数に格納される $(t = 1 \ldots T,\ \tau^{(t)} \in \tau_{DB}^{(t)})$．これは各 τ の部分支持グラフが保持すると考えると分かりやすい．同様に，各 $\tau^{(t)}$ の部分支持グラフは各局所パス $\tilde{S} \in \tilde{\psi}_\ell(\tau)$ ごとに配列変数 $\mathcal{R}[t, \tau^{(t)}, \tilde{S}]$ が用意されている $(t = 1 \ldots T,\ \tau^{(t)} \in \tau_{DB}^{(t)},\ \tilde{S} \in \tilde{\psi}_{DB}(\tau^{(t)}))$．また，`msw(i,·,v)` の期待出現回数 $\eta(i, v|\boldsymbol{\theta})$ を格納する配列変数として $\eta[i, v]$ を用意する．LEARN-GEM は 2 つのサブルーチン GET-INSIDE-PROBS, GET-EXPECTATIONS をもつ．前者が内側確率，後者が外側確率と $\eta(i, v|\boldsymbol{\theta})$ を計算する．

gEM アルゴリズムのメインルーチン LEARN-GEM を図 4.7 に示す．LEARN-GEM では，はじめにパラメータを初期化する（行 2）．そして，GET-INSIDE-PROBS, GET-EXPECTATIONS, パラメータの更新（行 7–8）をこの順に繰り返す．対数尤度 λ が収束したら（行 11），その時点でのパラメータ値 $\boldsymbol{\theta}$ を推定値 $\boldsymbol{\theta}^*$ として終了する．$\mathcal{P}[t, G_t]$ にゴール G_t が真になる確率 $P_{DB}(G_t=1|\boldsymbol{\theta})$ が格納されており，対数尤度の計算にはこの値を使う（行 4, 10）．図 4.8, 図 4.9 にサブルーチン GET-INSIDE-PROBS, GET-EXPECTATIONS を示す．また，図 4.10 は支持グラフ上における各サブルーチンの計算イメージである．以降ではこの図を使って gEM アルゴリズムを説明する．

◇ 内側確率の計算

式を簡単にするため，以下では $t = 1 \ldots T$ を固定し，添字の $\cdot^{(t)}$ や $\cdot_t$ は省略する．GET-INSIDE-PROBS における内側確率 $\mathcal{P}[t, \tau]$ の計算は τ_{DB} の末尾要素の部分支持グラフから順に行なう．τ_k の部分支持グラフ $(k = 1 \ldots K_t)$ の各局所パス $\tilde{S} \in \tilde{\psi}_{DB}(\tau_k)$ ではパス中の各ノードの確率積を計算し，

```
 1: procedure GET-INSIDE-PROBS (DB, G) begin
 2:     for t := 1 to T do begin
 3:         Put G_t = τ_0^(t);
 4:         for k := K_t downto 0 do begin
 5:             P[t, τ_k^(t)] := 0;
 6:             foreach S̃ ∈ ψ̃_DB(τ_k^(t)) do begin
 7:                 Put S̃ = {A_1, A_2, ..., A_|S̃|};
 8:                 R[t, τ_k^(t), S̃] := 1;
 9:                 for ℓ := 1 to |S̃| do
10:                     if A_ℓ = msw(i,·,v) then R[t, τ_k^(t), S̃] *= θ_{i,v}
11:                     else R[t, τ_k^(t), S̃] *= P[t, A_ℓ];
12:                 P[t, τ_k^(t)] += R[t, τ_k^(t), S̃]
13:             end /* foreach S̃ */
14:         end /* for k */
15:     end /* for t */
16: end.
```

図 4.8: 内側確率を計算するサブルーチン GET-INSIDE-PROBS.

```
 1: procedure GET-EXPECTATIONS (DB, G) begin
 2:     for i ∈ I, v ∈ V_i do η[i, v] := 0;
 3:     for t := 1 to T do begin
 4:         Put G_t = τ_0^(t);
 5:         Q[t, τ_0^(t)] := 1;
 6:         for k := 1 to K_t do Q[t, τ_k^(t)] := 0;
 7:         for k := 0 to K_t do
 8:             foreach S̃ ∈ ψ̃_DB(τ_k^(t)) do begin
 9:                 Put S̃ = {A_1, A_2, ..., A_|S̃|};
10:                 for ℓ := 1 to |S̃| do
11:                     if A_ℓ = msw(i,·,v) then η[i, v] += Q[t, τ_k^(t)] · R[t, τ_k^(t), S̃]/P[t, G_t]
12:                     else if P[t, A_ℓ] > 0 then Q[t, A_ℓ] += Q[t, τ_k^(t)] · R[t, τ_k^(t), S̃]/P[t, A_ℓ]
13:             end /* foreach S̃ */
14:     end /* for t */
15: end.
```

図 4.9: 外側確率と出現期待値を計算するサブルーチン GET-EXPECTATIONS.

$\mathcal{R}[t, \tau_k, \tilde{S}]$ に格納する（行 9–11, 図 4.10 (1)）．その際，msw$(i,\cdot,v)$ でラベルづけされているファクトノードに対しては $\theta_{i,v}$ を乗じ，τ' でラベルづけされているテーブルノードに対しては $\mathcal{P}[t, \tau']$ を乗じる[4]（図 4.10 (2)）．最後に $\mathcal{R}[t, \tau_k, \tilde{S}]$ の和をとることによって $\mathcal{P}[\tau_k]$ を計算する（行 12, 図 4.10 (3)）．

◇ 外側確率および msw$(i,\cdot,v)$ の期待出現回数の計算

一方，GET-EXPECTATIONS では GET-INSIDE-PROBS とは逆に G の部分支持グラフから順に計算を進める．はじめに配列変数 $\mathcal{Q}$ と η を初期化しておく．特に外側確率 $\mathcal{Q}[t, \tau]$ について $\tau = G$ のみを 1, 他は 0 にする点に注意する（行 6）．次に τ_k $(k = 1 \ldots K_t)$ の部分支持グラフの局所

[4]節 4.2.4 で述べた支持グラフの 3 つ目の特徴より $\tau' = \tau_{k'}$ とおくと必ず $k < k'$ であることと，$\mathcal{P}$ の計算は O_ℓ の最後尾から順に行なわれることから，$\mathcal{P}[\ell, \tau']$ は参照されるとき既に計算済みになっている点に注意する．

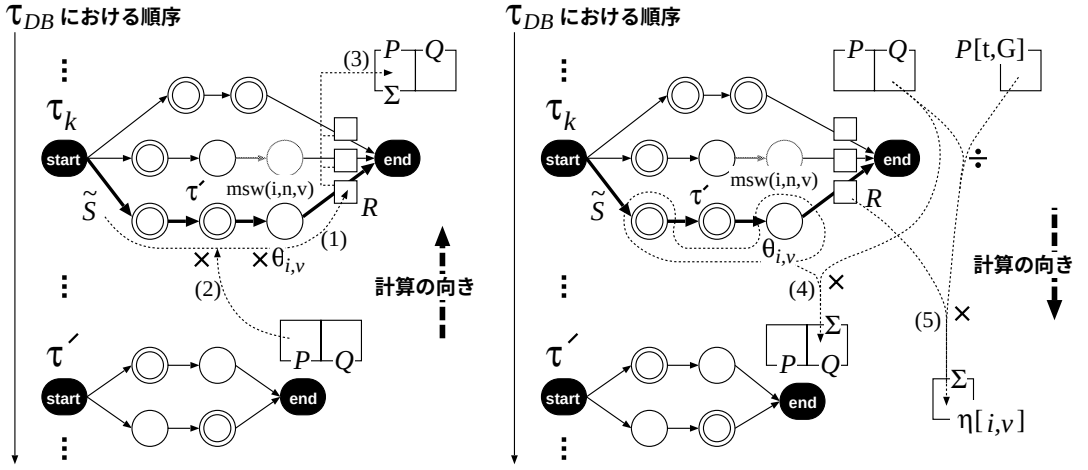

図 4.10: サブルーチン GET-INSIDE-PROBS（左）と GET-EXPECTATIONS（右）の計算イメージ．

パス $\tilde{S}$ を考える（行 8）．更に，行 12 で外側確率 $\mathcal{Q}$ が書き換えられる $\tau' \in \tilde{S}$ を考える．また，行 12 の式で $\mathcal{Q}[t,\tau']$ に加算されるのは，$\tilde{S}$ における τ' の局所的な外側確率（τ' 以外のノードの確率積）と τ' の親部分木 τ_k の外側確率 $\mathcal{Q}[t,\tau_k]$ の積である[5]（図 4.10 (4)）．また，行 11 において，$\texttt{msw}(i,\cdot,v)$ のラベルをもつファクトノードに対しては局所パスの確率 $\mathcal{R}[t,\tau_k,\tilde{S}]$ と親部分木 τ_k の外側確率 $\mathcal{Q}[t,\tau_k]$ の積をゴール G が真になる確率 $P_{DB}(G=1|\boldsymbol{\theta})$ で割って[6] $\eta[i,v]$ に足し込む（図 4.10 (5)）．こうして $\eta[i,v]$ の内容を書き換えていくと，GET-EXPECTATIONS の終了時には $\eta[i,v]$ に $\texttt{msw}(i,\cdot,v)$ の出現回数の期待値が格納されている．

4.2.6 グラフィカル EM アルゴリズムの正当性

gEM アルゴリズムは次のように正当化される．まず補題 2 によって（より広いクラスの）PRISM プログラムの最尤推定アルゴリズムである LEARN-GEM と gEM アルゴリズムの等価性が示さる．そして，補題 2 によって定理 2 が導かれる．補題 2, 定理 2 の証明は付録 B に与える．

補題 2（手続き LEARN-PRISM-NAIVE とグラフィカル EM アルゴリズムの等価性） 有限支持，強排反性，唯一性条件，非循環支持，独立支持条件を満たす PRISM プログラム DB と観測データ $\mathcal{G}$ を考える．そして，2 つの手続き LEARN-PRISM-NAIVE と LEARN-GEM に同じ初期値 $\boldsymbol{\theta}^{(0)}$ を与え，収束基準を同じにすると，両者が返す収束値 $\boldsymbol{\theta}^*$ は一致する． ■

定理 2（グラフィカル EM アルゴリズムの正当性） 有限支持，強排反性，唯一性条件，非循環支持，独立支持条件を満たす PRISM プログラム DB と観測データ $\mathcal{G}$ を考える．そのとき，手続き LEARN-GEM は尤度 $\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta})$ を局所的に最大にする $\boldsymbol{\theta} \in \Theta$ を出力する． ■

[5] $\tau' = \tau_{k'}$ とおくと，支持グラフの 3 つ目の特徴より必ず $k < k'$ が成り立つので，τ' は τ_{DB} では常に τ_k より後ろに現れる．逆にいえば $k'' = k \ldots K_t$ なる $\tau_{k''}$ の部分支持グラフでは $\mathcal{Q}[t,\tau_k]$ の値は書き換えられることはない．従って行 12 の式の右辺に現れる $\mathcal{Q}[t,\tau_k]$ は既に計算済みである．

[6] EM アルゴリズムの性質より，パラメータ更新を行なう度に個々の $t = 1 \ldots T$ について $P_{DB}(G_t=1|\boldsymbol{\theta})$ は増大するので，LEARN-GEM 行 2 のように，すべての $t = 1 \ldots T$ について $P_{DB}(G_t=1|\boldsymbol{\theta}) > 0$ となる $\boldsymbol{\theta}$ に初期化しておけば，以降 $\boldsymbol{\theta}$ の更新が行なわれても $P_{DB}(G_t=1|\boldsymbol{\theta}) = 0$ となることはない．

4.3 計算量

本節では，先に提案したPRISMプログラムの高速なEM学習法の計算量を評価し，既存の効率的専用EMアルゴリズムであるBaum-Welchアルゴリズム，Inside-Outsideアルゴリズム，単結合Bayesianネットワーク用EMアルゴリズムと同じオーダであることを示す．提案手法はOLDT探索とgEMアルゴリズムを連結することによって実現されている．従って，EM学習全体の計算量としては両者の計算量の和をとればよい．まず，gEMアルゴリズムの計算量を一般的に評価する．

4.3.1 グラフィカルEMアルゴリズムの計算量

先に述べたように，EMアルゴリズムの計算量は一回のパラメータ更新に要する計算量で測る．従ってgEMアルゴリズムの計算量はメインルーチンLearn-gEMの**repeat**ループ1回の計算量である．**repeat**ループではGet-Inside-Probs（図4.8），Get-Expectations（図4.9）が呼び出され，パラメータが更新される．

アルゴリズムの記述から分かるように，サブルーチンGet-Inside-Probs, Get-Expectationsは支持グラフ中の各ノードを定数回しか訪れない．従って大まかにいうと，これらのサブルーチンの計算量は支持グラフのサイズ，すなわちグラフ中のリンク数またはノード数に比例する[7]．また，パラメータ更新の計算量はパラメータ数に比例する．よって，

$$\Xi_t \stackrel{\text{def}}{=} \bigcup_{\tau \in \tau_{DB}^{(t)}} \tilde{\psi}_{DB}(\tau) \tag{4.16}$$

$$\xi_{\text{num}} \stackrel{\text{def}}{=} \max_{1 \le t \le T} |\Xi_t| \tag{4.17}$$

$$\xi_{\text{maxsize}} \stackrel{\text{def}}{=} \max_{\tilde{S}: 1 \le t \le T, \tilde{S} \in \Xi_t} |\tilde{S}| \tag{4.18}$$

とおいたとき，gEMアルゴリズムの計算量は $O\left(\max\left\{|\boldsymbol{\theta}_{DB}|, \xi_{\text{num}}\xi_{\text{maxsize}}T\right\}\right)$ である．T は観測データのサイズ，すなわち $T = |\mathcal{G}|$ である．最尤推定は大標本を前提としているので，通常 $|\boldsymbol{\theta}_{DB}| < \xi_{\text{num}}\xi_{\text{maxsize}}T$ としてよい．そのとき gEMアルゴリズムの計算量は $O(\xi_{\text{num}}\xi_{\text{maxsize}}T)$ となる．

4.3.2 OLDTの計算量

OLDT探索の計算量は個々のプログラムによって異なるので，その都度考察する必要がある．ここでは，例としてChomsky標準形を満たすCFGの最大規則集合 $R_{\max}$ を表現する論理プログラム（PRISMプログラムではない）を考える．

$$\begin{aligned}&\left\{\texttt{q}(i,d,d')\texttt{:-}\ \texttt{q}(j,d,d''),\texttt{q}(k,d'',d') \;\middle|\; i,j,k = 1\ldots N,\ 0 \le d < d'' < d' \le L\right\}\\ &\qquad \cup \left\{\texttt{q}(i,d,d') \;\middle|\; i = 1\ldots N,\ 0 \le d \le L-1,\ d' = d+1\right\}.\end{aligned} \tag{4.19}$$

$\texttt{q}(i,d,d')$ は i 番目の非終端記号が部分文 $\boldsymbol{w}_{d,d'}$ を統治することを表す．また，各プログラム節 $\texttt{q}(i,d,d')\texttt{:-}\ \texttt{q}(j,d,d''),\texttt{q}(k,d'',d')$ はプログラムの上から (i,j,k,d,d',d'') の辞書順に従って出現しているとする．テーブル操作が $O(1)$ で可能であったとすると，OLDT探索の計算量はOLDTにおける探索木（OLDT木と呼ばれる）のサイズによって測ることが出来る．また，OLDTの探索戦略をmulti-stage depth-first戦略 [58] に固定する．

[7] 支持グラフの形状より，各ノード数はリンク数と同じオーダである．

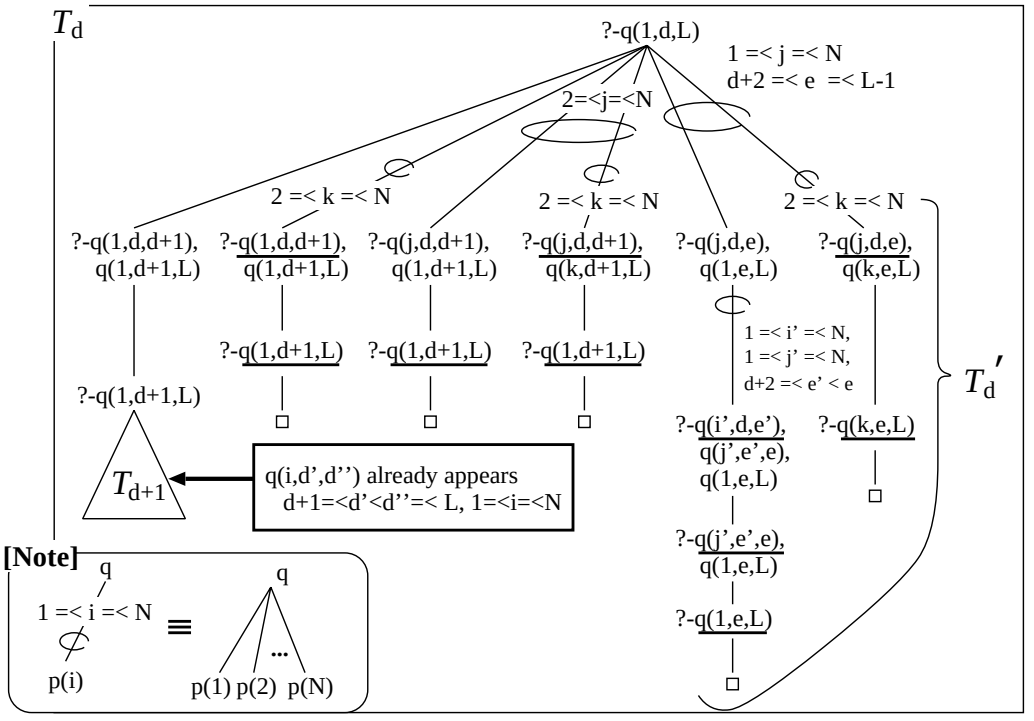

図 4.11: トップゴール "`?-q(1,`d`,`L`).`" に対する OLDT 木 T_d.

ここでトップゴール `?- q(1,`d`,`L`)` に対する OLDT 木を考え，T_d とおく．図 4.11 に $0 \leq d \leq L-3$ の場合の T_d を示す．元プログラムが単純なので似たような構造の部分木が多く現れるので，そのような部分木の集まりを図左下の [**Note**] のようにまとめて表記している[8]．また，下線を引いたノードは参照ノード (lookup node) である．このノードは他の場所で得られた解を単に消費するだけであり，探索空間は広がらない．

我々は depth-first 戦略をとっているので，図 4.11 において T_d は自身の中に T_{d+1} が現れるような再帰構造を持つ．ここで，T_d 中に現れるが T_{d+1} 中に現れないノードの数を h_d とおき，図 4.11 の T_d' に注目する．T_d' の各 j, i', j' は 1 から N までの値をとり得る．また，可能な (e, e') の組の数 $|\{(e,e') \mid d+2 \leq e' < e \leq L-1\}| = O((L-d)^2)$ であるから，T_d' 中のノード数は $O(N^3(L-d)^2)$ である．T_{d+1} にも T_d' にも含まれないノード数は無視できるので，$h_d = O(N^3(L-2)^2)$ である．開始記号 S を 1 番目の非終端記号とすると，トップゴール `?- q(1,0,`L`)` が文全体 $\boldsymbol{w}$ の解析に相当する．そのときの計算量は $\sum_{d=0}^{L-3} h_d = O(N^3L^3)$ になる（$d = L-1, L-2$ のときは無視できる）．よって規則集合 $R_{\max}$ に対して OLDT 探索を用いると，文を解析するのに $O(N^3L^3)$ 時間必要である．

また，DCG 形式から翻訳した Prolog プログラムでも $O(N^3L^3)$ 時間必要であることが確認できる．更に，PDCG 形式から翻訳された PRISM プログラムは DCG 形式から翻訳された Prolog プログラム（`msw(·,·,·)` を含まない）と同じ探索空間になるように工夫されているため，PDCG 形式から翻訳した PRISM プログラムでも同じ $O(N^3L^3)$ 時間で処理できる．

[8] 単にまとめて表記しているだけで，OLDT 探索としては個々の部分木を訪れている点に注意.

4.3.3 Baum-Welch アルゴリズムとの比較

節 2.1.2.1 で記述した HMM の専用 EM アルゴリズムである Baum-Welch アルゴリズムと gEM アルゴリズムの計算量を比較する．節 3.3.1 で記述した HMM プログラムで状態数を $|S_{\mathrm{hmm}}|$, 出力・受理記号列長を L とした場合，上と同じようにして OLDT 探索の計算量を測ると $O(|S_{\mathrm{hmm}}|^2 L)$ であることが分かった．T 個の観測データについては単純に T 倍して $O(|S_{\mathrm{hmm}}|^2 LT)$ になる．また，例として与えた図 4.5 などから，$t = 1 \ldots T$ について $\tau_{DB}^{(t)}$ は，

$$\tau_{DB}^{(t)} = \{\, \mathtt{hmm(}\ell\mathtt{,}q\mathtt{,[}o_\ell\mathtt{,}\ldots\mathtt{,}o_L\mathtt{])} \mid \ell = 1 \ldots L, q \in S_{\mathrm{hmm}} \,\} \cup \{\, \mathtt{hmm([}o_1\mathtt{,}\ldots\mathtt{,}o_L\mathtt{])} \,\} \tag{4.20}$$

であり，$|\tau_{DB}^{(t)}| = O(|S_{\mathrm{hmm}}|L)$ が成り立つ．また，各 $\mathtt{hmm(}\ell\mathtt{,}q\mathtt{,[}o_\ell\mathtt{,}\ldots\mathtt{,}o_L\mathtt{])}$ $(\ell = 1 \ldots L-1,$ $q \in S_{\mathrm{hmm}})$ について

$$\tilde{\psi}_{DB}(\mathtt{hmm(}\ell\mathtt{,}q\mathtt{,[}o_\ell\mathtt{,}\ldots\mathtt{,}o_L\mathtt{])}) = \left\{ \left\{ \begin{array}{c} \mathtt{msw(tr(}q\mathtt{),}\ell\mathtt{,}o_\ell\mathtt{),msw(out(}q\mathtt{),}\ell\mathtt{,}q'\mathtt{),} \\ \mathtt{hmm(}\ell+1\mathtt{,}q'\mathtt{,[}o_{\ell+1}\mathtt{,}\ldots\mathtt{,}o_L\mathtt{])} \end{array} \right\} \,\middle|\, q' \in S_{\mathrm{hmm}} \right\} \tag{4.21}$$

である．節 4.3.1 で与えた $\xi_{\mathrm{num}}, \xi_{\mathrm{maxsize}}$ の定義より $\xi_{\mathrm{num}} = O(|S_{\mathrm{hmm}}|^2 L)$, $\xi_{\mathrm{maxsize}} = 3 = O(1)$ であるが成り立つ（$\tilde{\psi}_{DB}(\mathtt{hmm([}o_1\mathtt{,}\ldots\mathtt{,}o_L\mathtt{])})$ は無視できる）．よって gEM アルゴリズムの計算量は $O(|S_{\mathrm{hmm}}|^2 LT)$ となる．一方，節 2.1.2.2 で議論したように HMM の専用 EM アルゴリズムである Baum-Welch アルゴリズムの計算量も $O(|S_{\mathrm{hmm}}|^2 LT)$ である．

以上より，PRISM で HMM を記述したとき，提案手法で EM 学習を行なうときの計算量は Baum-Welch アルゴリズムのものと同オーダである．

4.3.4 確率文脈自由文法との比較

Inside-Outside アルゴリズムの計算量評価と同様に文法構造に Chomsky 標準形を仮定する．PDCG 形式（節 3.3.3.1）で記述した PCFG を翻訳した後の PRISM プログラム（節 3.3.3.2）で計算量を評価する．まず，この PRISM プログラムに OLDT 探索を適用すると，非基底呼び出しによる将来的には成功しない部分木が発生する．しかし，これらのサイズは無視できるので取り除くと，図 4.11 のような再帰的な構造をもつ OLDT 木が残る．この OLDT 木のサイズは節 4.3.2 で評価したように $O(|V_{\mathrm{n}}|^3 L^3)$ である．T 個の観測データについては単純に T 倍することができるので，PCFG に対する OLDT 探索の計算量は $O(|V_{\mathrm{n}}|^3 L^3 T)$ になる．

次に gEM アルゴリズムの計算量を評価する．L を文長とすると，$t = 1 \ldots T$ について $\tau_{DB}^{(t)}$ は

$$\tau_{DB}^{(t)} = \{\, A\mathtt{(sw\_}A\mathtt{,}\boldsymbol{w}_{dL}\mathtt{/}\boldsymbol{w}_{d'L}\mathtt{,}\boldsymbol{w}_{dL}\mathtt{,}\boldsymbol{w}_{d'L}\mathtt{)} \mid A \in V_{\mathrm{n}},\ 0 \le d < d' \le L \,\} \tag{4.22}$$

となる．$\mathtt{[}w_{d+1}\mathtt{,}\ldots\mathtt{,}w_{d'}\mathtt{]}$ を $\boldsymbol{w}_{dd'}$ と略記している．上より明らかに $|\tau_{DB}^{(t)}| = O(|V_{\mathrm{n}}|L)$ が成り立つ．また，各 $A\mathtt{(sw\_}A\mathtt{,}\boldsymbol{w}_{dL}\mathtt{/}\boldsymbol{w}_{d'L}\mathtt{,}\boldsymbol{w}_{dL}\mathtt{,}\boldsymbol{w}_{d'L}\mathtt{)}$ $(A \in V_{\mathrm{n}},\ 0 \le d < d' \le L)$ について

$$\tilde{\psi}_{DB}(A\mathtt{(sw\_}A\mathtt{,}\boldsymbol{w}_{dL}\mathtt{/}\boldsymbol{w}_{d'L}\mathtt{,}\boldsymbol{w}_{dL}\mathtt{,}\boldsymbol{w}_{d'L}\mathtt{)}) = \left\{ \left\{ \begin{array}{l} \mathtt{msw(sw\_}A\mathtt{,}\boldsymbol{w}_{dL}\mathtt{/}\boldsymbol{w}_{d'L}\mathtt{,}j\mathtt{),} \\ B\mathtt{(sw\_}B\mathtt{,}\boldsymbol{w}_{dL}\mathtt{/}\boldsymbol{w}_{d''L}\mathtt{,}\boldsymbol{w}_{dL}\mathtt{,}\boldsymbol{w}_{d''L}\mathtt{)),} \\ C\mathtt{(sw\_}C\mathtt{,}\boldsymbol{w}_{d''L}\mathtt{/}\boldsymbol{w}_{d'L}\mathtt{,}\boldsymbol{w}_{d''L}\mathtt{,}\boldsymbol{w}_{d'L}\mathtt{))} \end{array} \right\} \,\middle|\, \begin{array}{l} B, C \in V_{\mathrm{n}}, \\ d < d'' < d', \\ A \to BC \text{ は } A \text{ の } j \text{ 番目の規則} \end{array} \right\} \tag{4.23}$$

である．先ほどと同様に $\xi_{\mathrm{num}} = O(|V_{\mathrm{n}}|^3 L^3)$, $\xi_{\mathrm{maxsize}} = 3 = O(1)$ であることが分かる．よって gEM アルゴリズムの計算量は $O(|V_{\mathrm{n}}|^3 L^3 T)$ である．一方，節 2.1.3.2 で議論したように PCFG の専用 EM アルゴリズムである Inside-Outside アルゴリズムの計算量も $O(|V_{\mathrm{n}}|^3 L^3 T)$ である．

従って PDCG 形式で PCFG を記述したとき，提案手法による EM 学習と Inside-Outside アルゴリズムの計算オーダは等しい．

4.3.5 単結合 Bayesian ネットワークとの比較

節 3.3.2 で述べた，図 2.5 の単結合 Baysian ネットワークを表現する PRISM プログラムをもう一度取り上げる．このプログラムは節 (B5) のように msw(i,n,v) を並べた非常に簡潔なもので，同時分布 $Pr(a,b,c,d,e,f,g)$ を表現する world/6 にサブルーチンは存在しない．

```
(B1) target(world/2).
(B2) data('world.dat').
(B3) values(_,[yes,no]).

(B4) world(C,G):- world(_,_,C,_,_,G).
(B5) world(A,B,C,D,E,F,G):-
         msw(a,A),msw(b,B),msw(c(A),C),msw(d(A,B),D),
         msw(e,E),msw(f(E),F),msw(g(D,E),G).
```

一方，本論文で提案している PRISM の EM 学習法では，OLDT がサブルーチンの計算記録を保存し，gEM アルゴリズムがその計算記録に埋め込まれた共有データ構造を利用することで高速化を図っている．そこで，HMM プログラムなどと同様，EM 学習が高速化できるようテーブル述語によるサブルーチンを記述することを考える．これはユーザの負担が増えることにつながるが，プログラミングにおいて高速な実行が行なわれるようにユーザが努力を払うのは珍しいことではない．テーブル述語を導入してもプログラムの宣言的な確率的意味を与えることができ，Prolog で使われるカット記号（プログラムの宣言的意味を保証しない）と対比すると，知識表現言語に対する高速化の工夫としては無理が少ない．

4.3.5.1 テーブル述語を用いた単結合 Bayesian ネットワークの記述

例えば，同じ図 2.5 の単結合 Bayesian ネットワークをテーブル述語を使って記述してみる．そのときの基本的な考え方は節 2.1.4.1 で説明した π-λ 計算法である．我々ははじめにネットワーク中の任意のノード X_i を選び，π-λ 計算法と同様に X_i から放射状に計算が進むようにプログラムを記述する．一例として D から放射状に計算が進むように記述する．また，あらかじめノードの訪問順も決めておく．はじめに以下の宣言を行なう．

```
(B'1)   target(mpd/1).
(B'2)   data(user).
(B'3)   table([mpd/1,rho/4,pi/4,lmd/4,rho_except/5,lmd_except/5]).
(B'4)   values(cpt(_,_),[yes,no]).
```

節 (B'1) によって述語 mpd/1 が観測可能であることが宣言される．証拠 $\langle C,G\rangle = \langle$yes,no$\rangle$ が観測されたとき，この証拠はリスト [c:yes,g:no] で表現される．このリストにおける各証拠はノードの訪問順と同じ順で並んでいるものとする．観測ゴールは mpd([c:yes,g:no]) になる．

テーブル述語として指定する rho(X_i,x_i,Es,Es') は $\rho_{X_i}(x_i)$ を表現する．2 つの引数 Es, Es' は証拠のリストを伝播するための差分リストを形成する．同様に pi(X_i,x_i,Es,Es') は $\pi_{X_i}(x_i)$ を，lmd(X_i,x_i,Es,Es') は $\lambda_{X_i}(x_i)$ を表現する．また，rho_except(X_i,Y_k,x_i,Es,Es') と lmd_except(Y_k,X_i,x_i,Es,Es') はそれぞれ $\rho_{X_i\backslash Y_k}(x_i)$ と $\lambda_{Y_k\backslash X_i}(x_i)$ を表現する．また，次に記述する theta(X_i,x_i,$\boldsymbol{u}$,Ein,Eout) が条件つき確率表 $\theta_{X_i}(x_i)$ を表現している．

```
(B'5)   theta(X,VX,VUs,Ein,Eout):-
            ( Ein=[X:VX|Eout]
            ; Ein \== [X:_|_],
              Eout=Ein,
            ),
            msw(cpt(X,VUs),VX).
```

Ein と [X:VX'|Eout] が単一化可能で，VX' と VX が単一化不可能なとき，theta(X_i,x_i,$\boldsymbol{u}$,Ein,Eout) は **false** となる．これはちょうど図 2.7 の π-λ 計算において確率 0 となる計算に対応する．

確率和は ∃ で縛られた変数（ヘッドに現れない変数），確率積は連言で表現していることに注意すれば，以降の節が図 2.7 の各計算式を表現していることが分かる．

```
%%%% 証拠 Es の周辺分布 (marginal probability distribution)
(B'6)   mpd(Es):- rho(d,D,Es,[]).
(B'7)   rho(d,D,Es0,Es1):- pi(d,D,Es0,Es2),lmd(d,D,Es2,Es1).

%%%% D の上流側

(B'8)   pi(d,D,Es0,Es1):-
            rho_except(a,d,A,Es0,Es2),
            rho_except(b,d,B,Es2,Es3),
            theta(d,D,[A,B],Es3,Es1).

(B'9)   rho_except(a,d,A,Es0,Es1):- pi(a,A,Es0,Es2), lmd_except(a,c,A,Es2,Es1).
(B'10)  rho_except(b,d,B,Es0,Es1):- pi(b,B,Es0,Es1).

(B'11)  lmd_except(c,a,A,Es0,Es1):- theta(c,C,[A],Es0,Es2), lmd(c,C,Es2,Es1).

%%%% D の下流側

(B'12)  lmd(d,D,Es0,Es1):- lmd_except(f,d,D,Es0,Es2), lmd_except(g,d,D,Es2,Es1).

(B'13)  lmd_except(f,d,D,Es0,Es1):-
            theta(f,F,[D],Es0,Es2), lmd(f,F,Es2,Es1).  % D が F の唯一の親

(B'14)  lmd_except(g,d,G,Es0,Es1):-
            rho_except(e,g,E,Es0,Es2),
            theta(g,G,[D,E],Es2,Es3),
            lmd(g,G,Es3,Es1).

(B'15)  rho_except(e,g,E,Es0,Es1):- pi(e,E,Es0,Es1).

%%%% 根ノード (A,B,E)

(B'16)  pi(a,A,Es0,Es1):- theta(a,A,[],Es0,Es1).
(B'17)  pi(b,B,Es0,Es1):- theta(b,B,[],Es0,Es1).
(B'18)  pi(e,E,Es0,Es1):- theta(e,E,[],Es0,Es1).

%%%% 葉ノード (C,F,G)

(B'19)  lmd(c,_,Es,Es):- true.
(B'20)  lmd(f,_,Es,Es):- true.
(B'21)  lmd(g,_,Es,Es):- true.
```

4.3.5.2 計算量の評価

はじめに，OLDT 探索の計算量を評価する．上で述べたようなプログラムの書き方により，OLDT は Bayesian ネットワークのノード数 $|V_{\rm bn}|$ に対して線形時間で探索できることが分かる．次に，gEM アルゴリズムの計算量を評価する．上で述べたようなプログラムの書き方により，手続き GET-INSIDE-PROBS による内側確率の計算は π-λ 計算法と同じ手順で行なわれるため，計算オーダは π-λ 計算法と同じく $O(|V_{\rm bn}|)$ である．また，外側確率・期待値の計算を行なう GET-EXPECTATIONS の計算オーダは GET-INSIDE-PROBS と等しく $O(|V_{\rm bn}|)$ であり，従って gEM アルゴリズムの計算オーダもまた $O(|V_{\rm bn}|)$ である．よって以上より，単結合 Bayesian ネットワークに対しては，提案手法と単結合 Bayesian ネットワーク用 EM アルゴリズムの計算量は等しいことが分かる．

4.3.6 PCFG の拡張文法との比較

PCFG の拡張文法として，節 3.3.3.3 において PDCG 形式で記述した 2 つのモデル，疑似 PCSG と PCFG+bigram モデルを取り上げ，それぞれを翻訳した後の PRISM プログラム（節 3.3.3.2）を考える．

4.3.6.1 疑似 PCSG の場合

疑似 PCSG を翻訳した PRISM プログラムに対して OLDT 探索を適用すると，解テーブルに登録するゴールのパターンが親 A' の分だけ増え，PCFG では参照ノードとして探索を止めていた部分でも探索しなければならなくなる．A' のパターンの分だけ，すなわち PCFG に比べて $|V_{\rm n}|$ 倍余計に探索時間がかかることになる．従って OLDT の探索は $O(|V_{\rm n}|^4L^3T)$ 時間かかる．

次に gEM アルゴリズムの計算量を評価する．L を文長とすると，$t=1\ldots T$ について $\tau_{DB}^{(t)}$ は

$$\tau_{DB}^{(t)} = \{\ A(\texttt{sw\_}A(A'),\boldsymbol{w}_{dL}/\boldsymbol{w}_{d'L},\boldsymbol{w}_{dL},\boldsymbol{w}_{d'L})\ \mid A,A'\in V_{\rm n},\ 0\le d<d'\le L\ \} \tag{4.24}$$

となる．先ほど同様 $[w_{d+1},\ldots,w_{d'}]$ を $\boldsymbol{w}_{dd'}$ と略記する．また，各 $A(\texttt{sw\_}A(A'),\boldsymbol{w}_{dL}/\boldsymbol{w}_{d'L},\boldsymbol{w}_{dL},\boldsymbol{w}_{d'L})$ $(A,A'\in V_{\rm n},\ 0\le d<d'\le L)$ について

$$\tilde{\psi}_{DB}(A(\texttt{sw\_}A(A'),\boldsymbol{w}_{dL}/\boldsymbol{w}_{d'L},\boldsymbol{w}_{dL},\boldsymbol{w}_{d'L})) = \left\{ \left\{ \begin{array}{l} \texttt{msw(sw\_}A(A'),\boldsymbol{w}_{dL}/\boldsymbol{w}_{d'L},j), \\ B(\texttt{sw\_}B(A),\boldsymbol{w}_{dL}/\boldsymbol{w}_{d''L},\boldsymbol{w}_{dL},\boldsymbol{w}_{d''L})), \\ C(\texttt{sw\_}C(A),\boldsymbol{w}_{d''L}/\boldsymbol{w}_{d'L},\boldsymbol{w}_{d''L},\boldsymbol{w}_{d'L})) \end{array} \right\} \middle| \begin{array}{l} B,C\in V_{\rm n}, \\ d<d''<d', \\ A\to BC \text{ は } A \text{ の } j \text{ 番目の規則} \end{array} \right\} \tag{4.25}$$

である．よって $\xi_{\rm num}=O(|V_{\rm n}|^4L^3)$, $\xi_{\rm maxsize}=3=O(1)$ であることが分かり，gEM アルゴリズムの計算量は $O(|V_{\rm n}|^4L^3T)$ となる．以上より，提案手法による EM 学習の 計算量は $O(|V_{\rm n}|^4L^3T)$ である[9]．

4.3.6.2 PCFG+bigram モデルの場合

PCFG+bigram モデルを翻訳した PRISM プログラムに対して OLDT 探索を適用すると，先ほどの疑似 PCSG とは異なり，加えた文脈が部分文字列情報の一部であるため，探索空間は PCFG と同じである．従って OLDT の探索は $O(|V_{\rm n}|^3L^3T)$ 時間かかる．

[9]先にも述べたように疑似 PCSG の計算量については Charniak らは評価を行なっていない．

次に gEM アルゴリズムの計算量を評価する．L を文長とすると，$t = 1 \ldots T$ について $\tau_{DB}^{(t)}$ は

$$\tau_{DB}^{(t)} = \{\ A(\mathtt{sw\_}A(w_d), \boldsymbol{w}_{dL}/\boldsymbol{w}_{d'L}, \boldsymbol{w}_{dL}, \boldsymbol{w}_{d'L})\ \mid A \in V_{\mathrm{n}},\ 0 \le d < d' \le L\ \} \tag{4.26}$$

となる．また，各 $A(\mathtt{sw\_}A(w_d), \boldsymbol{w}_{dL}/\boldsymbol{w}_{d'L}, \boldsymbol{w}_{dL}, \boldsymbol{w}_{d'L})$ $(A \in V_{\mathrm{n}},\ 0 \le d < d' \le L)$ について

$$\begin{aligned}
&\tilde{\psi}_{DB}(A(\mathtt{sw_}A(w_d), \boldsymbol{w}_{dL}/\boldsymbol{w}_{d'L}, \boldsymbol{w}_{dL}, \boldsymbol{w}_{d'L})) \\
&= \left\{ \left\{ \begin{array}{l} \mathtt{msw}(\mathtt{sw_}A(w_d), \boldsymbol{w}_{dL}/\boldsymbol{w}_{d'L}, j), \\ B(\mathtt{sw_}B(w_d), \boldsymbol{w}_{dL}/\boldsymbol{w}_{d''L}, \boldsymbol{w}_{dL}, \boldsymbol{w}_{d''L})), \\ C(\mathtt{sw_}C(w_{d''}), \boldsymbol{w}_{d''L}/\boldsymbol{w}_{d'L}, \boldsymbol{w}_{d''L}, \boldsymbol{w}_{d'L})) \end{array} \right\} \;\middle|\; \begin{array}{l} B, C \in V_{\mathrm{n}}, \\ d < d'' < d', \\ A \to BC \text{ は } A \text{ の } j \text{ 番目の規則} \end{array} \right\}
\end{aligned} \tag{4.27}$$

である．よって $\xi_{\mathrm{num}} = O(|V_{\mathrm{n}}|^3 L^3)$, $\xi_{\mathrm{maxsize}} = 3 = O(1)$ であることが分かり，gEM アルゴリズムの計算量は $O(|V_{\mathrm{n}}|^3 L^3 T)$ となる．以上より，文脈を加えたにも関わらず，提案手法による EM 学習の計算量は PCFG の場合と同じ $O(|V_{\mathrm{n}}|^3 L^3 T)$ である．

4.4 第 4 章のまとめ

本章では，PRISM プログラムに対し，初歩的な EM 学習法と高速化された EM 学習法をそれぞれ節 4.1, 節 4.2 で提案し，後者に対して Baum-Welch アルゴリズム，Inside-Outside アルゴリズムなどの既存の専用 EM アルゴリズムとの計算量の比較を節 4.3 で行なった．その結果，本章で提案した PRISM プログラム用の高速 EM 学習法は，これらの専用 EM アルゴリズムを（第 2 章，定義 2 の意味で）一般化したものであることを示した．

ただし，初歩的 EM 学習法が適用可能になるためには，PRISM プログラムは有限支持条件，排反性条件，唯一性条件の 3 つの条件を満たさなければならない．高速 EM 学習法を適用するには，更に強排反性条件，非循環支持条件，独立支持条件を満たすことが前提となる．

高速 EM 学習法は Tamaki と Sato による OLDT 探索と新たに導出されたグラフィカル EM アルゴリズムの 2 段階から成る．OLDT では解テーブルと参照テーブルという 2 つのテーブルを参照，更新しながら探索することにより，再計算（再探索）を防ぐことができる．更に，本章で提案した高速 EM 学習法では，解テーブルから支持グラフという中間データを抽出し，その支持グラフ上でグラフィカル EM アルゴリズムを動作させる．支持グラフは HMM における trellis 図や Inside-Outside アルゴリズムにおける三角行列に対応するコンパクトなデータ構造であり，グラフィカル EM アルゴリズムは，第 2 で述べた分割統治と動的計画法の考え方に基づき，支持グラフを定数回走査するだけで E ステップが計算できるように設計されている．

第5章　WFSTに基づく確率文脈自由文法の高速学習

前章ではPRISMプログラムに対して効率的な専用EMアルゴリズム（例えばInside-Outsideアルゴリズム）に対して同じ計算オーダでEM学習を行なう方法を提案した．提案手法ではOLDT探索とグラフィカルEMアルゴリズム（gEMアルゴリズム）を組合せている．すなわちOLDT探索で用いた解テーブルから支持グラフという構造をもつデータを抽出し，支持グラフ上でgEMアルゴリズムを動作させる．OLDT探索は汎用の探索手法であるが，対象が確率文脈自由文法(PCFG)のとき，OLDTの代わりに構文解析器（パーザ）を用いることができる．効率的とされているパーザはどれもWFST (well-formed substring table)という途中の解析結果を保存するデータ構造を備えており，これはOLDTにおける解テーブルに相当する．従って，PCFGに対しては

1. 観測データであるコーパス（例文集）のすべての文について構文解析を行ない，
2. そのときに用いたWFSTから支持グラフを構築し，
3. gEMアルゴリズムを適用してEM学習を行なう

方法が考えられる．この場合，ボトムアップパーザであるCYK法や一般化LR法を用いることができ，トップダウンパーザに分類されるOLDTに比べ，上の1の部分の効率化が期待できる．

更に，この方法では最終的な構文木にならなかった部分は支持グラフの構築時に自然と取り除かれるので，文法の制約により曖昧性が減っている場合は上記の2, 3の部分についても高速化が期待できる．ATR対話コーパス(SLDB)に対して実験を行なった結果，Inside-Outsideアルゴリズムに比べ，平均文長においておよそ1,000倍の速度向上が得られた．

5.1　基本的考え

図5.1の例文法$G1$に基づいて先ほど述べたPCFGのEM学習法の基本的考えを説明する[1]．Inside-Outsideアルゴリズムは CYKパーザのWFSTである三角行列に基づいて確率計算を行なう．Inside-Outsideアルゴリズムと対比して説明するため，我々はCYKパーザとgEMアルゴリ

$G1$:

(1)	S → PP V	(6)	NP → V N	(10)	N → 一郎
(2)	S → ADV VP	(7)	PP → NP P	(11)	P → を
(3)	VP → PP V	(8)	PP → N P	(12)	V → 走る
(4)	VP → ADV V	(9)	ADV → 急いで	(13)	V → 見た
(5)	NP → VP N				

図 5.1: 文脈自由文法の例 $G1$.

[1] $G1$ は文献 [38] より引用している．

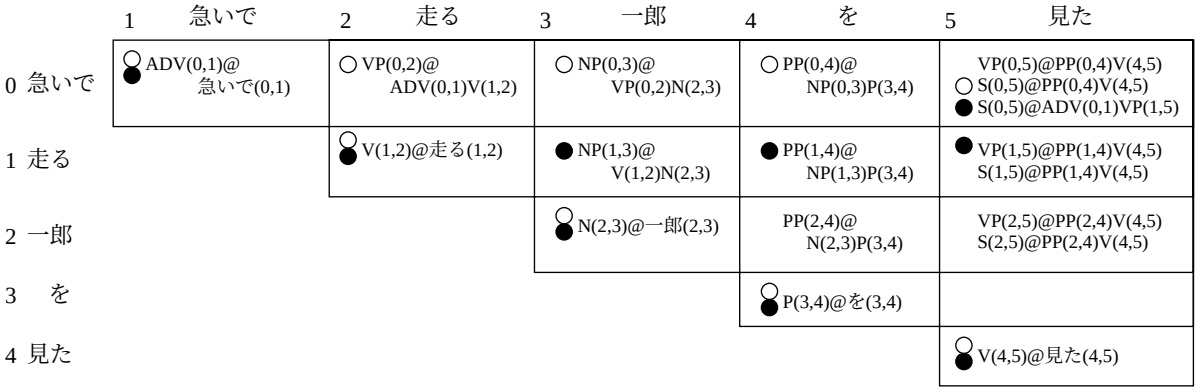

図 5.2: $G1$ と文 ⟨ 急いで, 走る, 一郎, を, 見た ⟩ に対する三角行列.

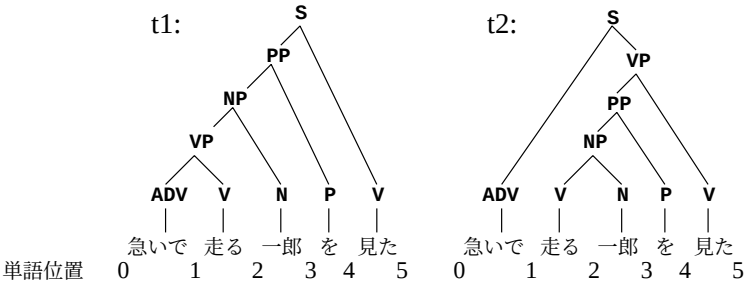

図 5.3: 三角行列から取り出された 2 つの構文木.

ズムの組合せを考える．図 5.1 の文法 $G1$ に対し，文 ⟨ 急いで, 走る, 一郎, を, 見た ⟩ を CYK パーザで構文解析すると図 5.2 の三角行列が得られる．導出木が求められるよう，三角行列の d 行 d' 列に $A(d,d')@B(d,d'')C(d'',d')$ という部分木の親子を格納する．図 5.2 の場合は○と●印をつけた部分木の親子からそれぞれ図 5.3 の構文木 $t1, t2$ が取り出される．

我々は，最終的に構文木になる部分木の親子（図 5.2 では○と●印をつけた部分木の親子）に注目する．例えば，0 行 5 列では

$$\begin{aligned}&\mathtt{S}(0,5)@\mathtt{PP}(0,4)\mathtt{V}(4,5)\\&\mathtt{S}(0,5)@\mathtt{ADV}(0,1)\mathtt{VP}(1,5)\end{aligned}$$

という 2 つが該当するが，これを論理的には

$$\begin{aligned}(\mathtt{S} \overset{*}{\Rightarrow} \boldsymbol{w}_{0,5}) \quad \leftrightarrow \quad & (\mathtt{S} \rightarrow \mathtt{PP\ V}) \wedge (\mathtt{PP} \overset{*}{\Rightarrow} \boldsymbol{w}_{0,4}) \wedge (\mathtt{V} \overset{*}{\Rightarrow} \boldsymbol{w}_{4,5}) \\ & \vee (\mathtt{S} \rightarrow \mathtt{ADV\ VP}) \wedge (\mathtt{ADV} \overset{*}{\Rightarrow} \boldsymbol{w}_{0,1}) \wedge (\mathtt{VP} \overset{*}{\Rightarrow} \boldsymbol{w}_{1,5}) \end{aligned} \tag{5.1}$$

と読むことができ，さらに PCFG においては

$$\begin{aligned}Pr(\mathtt{S} \overset{*}{\Rightarrow} \boldsymbol{w}_{0,5}) \quad = \quad & \theta(\mathtt{S} \rightarrow \mathtt{PP\ V}) Pr(\mathtt{PP} \overset{*}{\Rightarrow} \boldsymbol{w}_{0,4}) Pr(\mathtt{V} \overset{*}{\Rightarrow} \boldsymbol{w}_{4,5}) \\ & + \theta(\mathtt{S} \rightarrow \mathtt{ADV\ VP}) Pr(\mathtt{ADV} \overset{*}{\Rightarrow} \boldsymbol{w}_{0,1}) Pr(\mathtt{VP} \overset{*}{\Rightarrow} \boldsymbol{w}_{1,5}) \end{aligned} \tag{5.2}$$

という確率関係式を表していると見ることができる．式 5.1 中の $A \overset{*}{\Rightarrow} \boldsymbol{w}_{d,d'}$ をテーブルアトム，規則 $A \rightarrow \zeta$ をファクト，すなわち F のアトム $\mathtt{msw}(\cdot,\cdot,\cdot)$ と見れば，式 5.1 から支持グラフを作ることができる．よって，我々は

1. (CYK) パーザで構文解析し，
2. 三角行列から支持グラフ（○●の部分）を抜き出し，
3. その支持グラフに対して gEM を適用して PCFG のパラメータを学習する

ことが可能になる．図 5.2 を見れば分かるように文法の制約により三角行列中に構文木を成す（○●でマークされている）部分木の親子が出現しない行列要素がある．よって支持グラフのサイズが三角行列も小さくなることが期待できる．先に述べたように Inside-Outside アルゴリズムは三角行列上で動作するアルゴリズムであり，その計算量は三角行列のサイズに依存する．一方，gEM アルゴリズムは支持グラフのサイズに比例して計算時間がかかる．よって支持グラフのサイズが三角行列よりもコンパクトであれば，gEM アルゴリズムが Inside-Outside アルゴリズムより高速に動作することが期待できる．

5.2 実験

我々は ATR 対話コーパス (SLDB) と人間によって記述された CFG に基づき，PCFG の EM 学習に要する計算時間（以下，訓練時間と呼ぶ）を計測した．本節ではその計測結果について述べる．

5.2.1 準備

対象 PCFG の元になる CFG は 860 規則から成る，田中らが開発した音声認識用日本語文法 [60] に手が加えられたものである．ATR 対話コーパスもこの文法に対応して手が加えられている．以降ではこの CFG を G^* で参照することにする．G^* は品詞を細分化したカテゴリを終端記号とした CFG であり，非終端記号数 173, 終端記号数 441 である．ATR 対話コーパス中の文では,（実際の単語ではなく）上記カテゴリの列を対象とした．文長は平均 9.97, 最短 2, 最長 49 である．また，先ほどの説明では CYK パーザと gEM アルゴリズムの組合せを説明したが，G^* は Chomsky 標準形ではないので，2 つの実験ではともに一般化 LR (GLR) パーザ [62] との組合せを採用した[2]．GLR パーザの WFST は**共有圧縮統語森** (packed shared parse forest) と呼ばれる．

5.2.2 手順

本実験では G^* が与えられた場合の訓練時間を提案手法と I-O アルゴリズムの間で比較する．具体的には，我々は文長 L を変化させたときにパラメータを一回更新するのに要する計算時間（更新時間と呼ぶ）が変化する様子を比較する．この実験ではパラメータの質については言及しない．まず，我々は ATR コーパス $\mathcal{C}$ の中で文長 $L-1$ と L の文をグループ化し，各々から無作為に取り出した 100 文を $\mathcal{C}_L$ とする $(L = 2, 4, \ldots 26)$[3]．そして，各 $\mathcal{C}_L$ を一つの訓練コーパスとし，各々に対して更新時間を計測する．I-O アルゴリズムは Chomsky 標準形でしか動作しないので，あらかじめ G^* を Chomsky 標準形に変換した．その結果 860 規則が 2,308 規則（非終端記号数 210, 終端記号数 441）の文法になった．

[2]具体的には，東京工業大学 田中・徳永研究室で開発・公開されている MSLR (Morphological and Syntactic LR) パーザに支持グラフ抽出ルーチンと gEM アルゴリズムのルーチンを連結した．MSLR パーザは `http://tanaka-www.cs.titech.ac.jp/pub/mslr/` で公開されている．MSLR パーザは形態素解析と構文解析を同時に行なう機能を有するが，今回の実験では構文解析機能のみを使用した．MSLR パーザを含め，実験で用いたプログラムはすべて C 言語で実装されている．C コンパイラは gcc 2.8.1 を使用した．また，実験で使用した計算機の CPU と OS はそれぞれ Sun UltraSPARC-II 296 MHz と Solaris 2.6 である．

[3]長さ 27 以上の 176 文（全体の 1.6 %）はデータ不足のため，この実験では考慮しなかった．

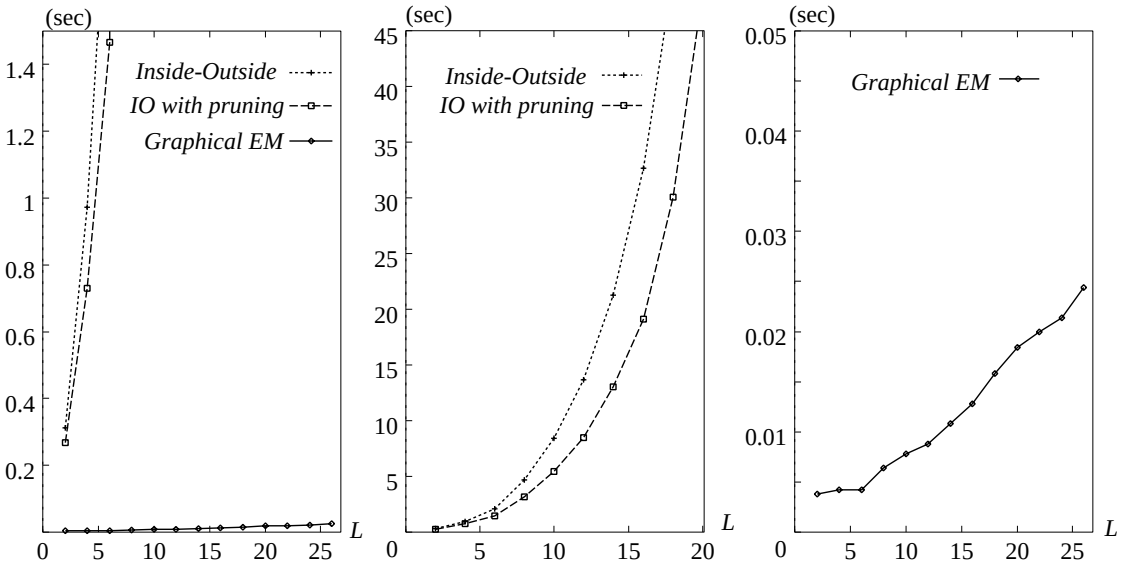

図 5.4: （左）Inside-Outside アルゴリズムとその枝刈り版，および gEM アルゴリズムにおける更新時間 (sec) の変化,（中央）縦軸を縮小したもの,（右）縦軸を拡大したもの．

5.2.3 結果

更新時間を計測した結果を図 5.4 左に示す．縦軸が更新時間 (sec)，横軸 L が使用した訓練コーパス $\mathcal{C}_L$ を表す．“*Inside-Outside*” は I-O アルゴリズムの更新時間，“*IO with pruning*” は [25] で説明されている，I-O アルゴリズムの外側確率の計算において無駄な計算部分を枝刈りするように改良したものである（以下，枝刈り版 I-O アルゴリズムと呼ぶ）．“*Graphical EM*” は gEM アルゴリズムの更新時間を示す．また，変化の様子を見やすくするために，図 5.4 左の縦軸を拡大，縮小したものをそれぞれ図 5.4 中央，図 5.4 右に示す．図 5.4 中央において “*Graphical EM*” は見にくいため省略した．

図 5.4 左のグラフから分かるように，gEM アルゴリズムは I-O アルゴリズムやその枝刈り版に比べてはるかに高速な計算が行なわれていることが分かる．また，図 5.4 中央のグラフから分かるように I-O アルゴリズムはアルゴリズム（図 2.1, 図 2.4, 式 2.22~2.24）の解析から得られる理論値どおり L^3 の曲線を描く．枝刈り版 I-O アルゴリズムは枝刈りした分高速であるものの，仮説駆動型である（$L \times L$ の三角行列の全要素をスキャンする）点は変わらないので，枝刈りが最も効率良く行なわれた場合でも L^2 を下回ることはない．収束まで数 100 回の更新を要すること，および **再出発法**[4] (random restart hill-climbing) [51] を採用することを考慮すると，$L = 20$ を越える訓練コーパス $\mathcal{C}_L$ に 対して I-O アルゴリズムおよびその枝刈り版を収束するまで動作させるのは現実的ではない．それに対し，提案手法における $L = 2, 4, \ldots, 26$ の範囲では L に対してほぼ線形に計算できており（図 5.4 右），最悪計算量 $O(|V_{\mathrm{n}}|^3 L^3)$ とは大きな差があることが分かった．これは文法の制約により，WFST に格納される部分木の数が抑えられたためと考えられる．ATR コーパスにおける文長平均 9.97 に近い $L = 10$ では I-O アルゴリズムに対しておよそ 1,000 倍（枝刈り版に対してはおよそ 700 倍）の速度向上が得られた．

[4] 再出発法では，良質なパラメータを得る目的で初期値をランダムに選び，収束するまでパラメータ更新するという作業を複数回（$h > 1$ 回）繰り返し，もっとも尤度の高い収束パラメータを最終的な学習結果とする．

再出発法を採用したとき，提案手法における訓練時間の内訳は次のようになる：

$$\begin{aligned}(\text{全体の訓練時間}) \quad &= \quad (\text{構文解析時間}) + (\text{支持グラフ抽出に要する時間}) \\ &\quad + (\text{gEM アルゴリズム実行時間}), \\ (\text{gEM アルゴリズム実行時間}) \quad &= \quad (\text{更新時間}) \times (\text{収束までの更新回数}) \times (\text{再出発回数 } h).\end{aligned}$$

先に述べた文長毎の訓練コーパス $\mathcal{C}_L$ $(L = 2, 4, \ldots, 26)$ を使って，訓練時間の内訳（構文解析時間，支持グラフ抽出時間，gEM 実行時間）を計測した．その結果を図 5.5 に示す．横軸が L, 縦軸が処理時間 (sec) である．図 5.5（左）は再出発なし $(h = 1)$ の場合，図 5.5（右）は再出発回数 $h = 10$ の場合である．また，収束までの更新回数はコーパス $\mathcal{C}_L$ によって異なるため，ここでは 100 に固定した．構文解析時間 (“*Parsing*”)，支持グラフ抽出時間 (“*Support graph*”)，gEM 実行時間 (“*Graphical EM*”) はいずれも文長 L に対してほぼ線形になっていることが分かる．更に図 5.5（右）より，再出発法を採用した場合は構文解析時間と支持グラフ抽出時間が訓練時間全体に占める割合は非常に小さい．構文解析と支持グラフ抽出は再出発の度に繰り返す必要がないからである．構文解析と支持グラフ抽出を gEM アルゴリズムの前処理と捉えれば，わずかな前処理（図 5.5）で大きな速度向上（図 5.4）が得られているということもでき，構文解析と EM 学習を分離したメリットが現れている．

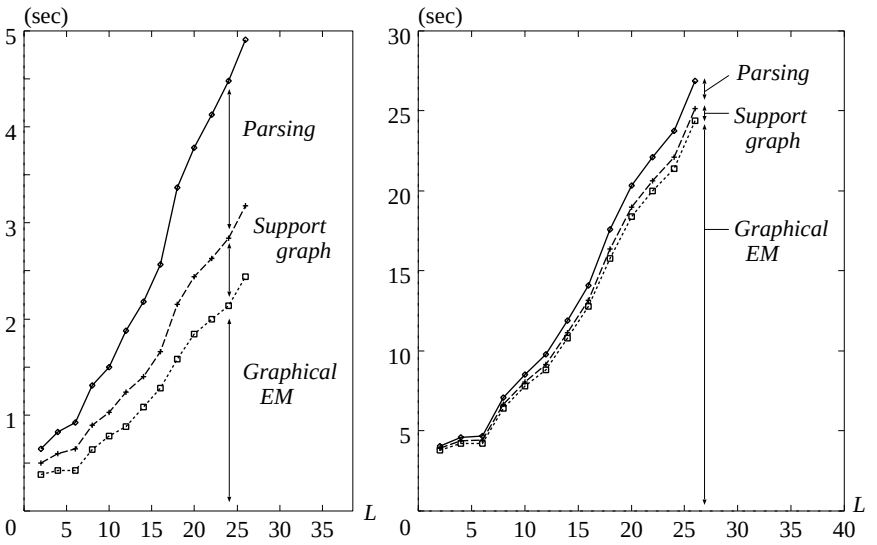

図 5.5: 訓練時間全体に占める各過程の処理時間の内訳.（左）再出発なしの場合 $(h = 1)$,（右）再出発回数 $h = 10$ の場合.

5.3 第 5 章のまとめ

前章では,（いくつかの制約が課せられているものの）かなり広いクラスの PRISM プログラムに対する高速 EM 学習法を示したのに対し，本章では学習対象を確率文脈自由文法に絞ったときの高速 EM 学習法を提示した．この場合，探索エンジンとして OLDT インタプリタの代わりに構文解析器（パーザ）を用いることができる．ATR 対話コーパスと田中らによって開発された日本語文法に対して，パラメータ学習時間を計測した結果，従来手法である Inside-Outside アルゴリズ

ムに比べて，平均文長でおよそ 1,000 倍の速度向上が得られた．Inside-Outside アルゴリズムによる確率文脈自由文法のパラメータ学習はその計算効率の悪さが度々指摘されているが [33]，実際のコーパス・文法においては本章で提案した手法が計算効率の問題に対する一つの解決になり得ることを示した．

第6章　関連研究

先にも述べたように，普遍的な統計知識が表現できるような一階述語論理に基づく統計モデリング言語を考えるのは知識表現研究における重要な分野であり [51]，特に最近その進展が目立つ。本章では Sato らの調査 [23, 36, 56] を参考に，そのうちの一部 [5, 13, 14, 18, 26, 30, 37, 39, 40, 41, 46, 50, 53] を確率的制約アプローチ，確率的ホーンアブダクション，KBMC (knowledge-based model construction) アプローチ，対数線形モデルの 4 つに分類し，紹介する。以下では，一階述語論理に基づく統計モデリング言語によって記述された統計知識を KB で表す。

確率的制約アプローチ [30, 39, 41] では，確率値もしくは確率値の範囲が付与された論理式によって我々の統計知識を表現する。興味のある論理式が満たすべき確率値または確率値の範囲は，線形方程式または線形不等式を解くことによって得られる。確率 Pr を KB 中の論理式に与えられる可能な解釈の集合，すなわち **可能世界** (possible world) Ω_{KB} から $[0, 1]$ への関数と捉える点も共通である。その意味で，可能世界に確率値を付与し，その制約を行列形式で記述する Nilsson の Probabilistic Logic [41] も確率的制約アプローチに含まれる。

例えば，Lukasiewicz [30] の枠組みでは各論理式 f が $(H|B)[c_1, c_2]$ という形をしている。ただし $0 \leq c_1 \leq c_2 \leq 1$ で，かつ H, B は **true**, **false**, アトム（原子式）の連言である。f は $c_1 \cdot Pr(B) \leq Pr(B \wedge H) \leq c_2 \cdot Pr(B)$ のとき，またその時に限り $(H|B)[c_1, c_2]$ は Pr の下で真であると解釈する。

共通する問題点としては，有限個のオブジェクトのみが使用でき，関数記号は使用できない点である[1]。このように可能なアトムを有限個に制限すると，原理的な記述力は命題論理と同等になってしまい，一階述語論理としての利点は論理変数による表現のコンパクト化だけになる。また，このアプローチでは論理式の確率値，確率値の範囲の計算に議論が限られており，観測データから確率パラメータを推定する方法は提示されていない。

KBMC アプローチははじめ Breese [5] によって提案され，これまでいくつかの研究が為されている [18, 26, 40]。KBMC では，はじめに普遍的な統計知識を記述した論理式（条件つき確率表が付与されている）から成る KB が用意される。そして，興味ある現象に対して質問 Q を行うと，Q の対応する Bayesian ネットワークが構築される。そして，節 2.1.4.1 で述べたような確率計算アルゴリズムによって Q の（条件つき）確率値が計算される。Pynadath らの Bayesian ネットワークによる PCFG の確率計算法（節 2.2.4）は対象モデルクラスを PCFG に限った場合の KBMC であると見ることができる。

Koller らが記述した KB の例を示す [26]。式 6.1, 6.2 の規則は遺伝子の伝搬を表現している。述語 `genotype(P,G)` は人間 `P` が遺伝子 `G` をもつことを表し，`parent(P,Q)` は `Q` が `P` の親であることを表す。そして，`phenotype(P,G)` は遺伝子 `G` が人間 `P` の表現型中に観測されることを表す。

$$\texttt{genotype(P,G)} \xleftarrow{0.5} \texttt{parent(P,Q)}, \texttt{genotype(Q,G)}. \tag{6.1}$$

[1] 従って，可能なアトムは有限個であり，可能世界 Ω_{KB} も有限になる。Ng らの場合 [39] は次の理由による：論理式の確率的意味を定めるのに確率範囲に関する線形不等式を用いている。従って，その式の確率的意味を知るためには線形計画法によって線形不等式を解く必要がある。しかし，線形計画法を無限個の制約に対しては適用できない。このため Ng らは有限個のオブジェクトと関数記号を禁止している。

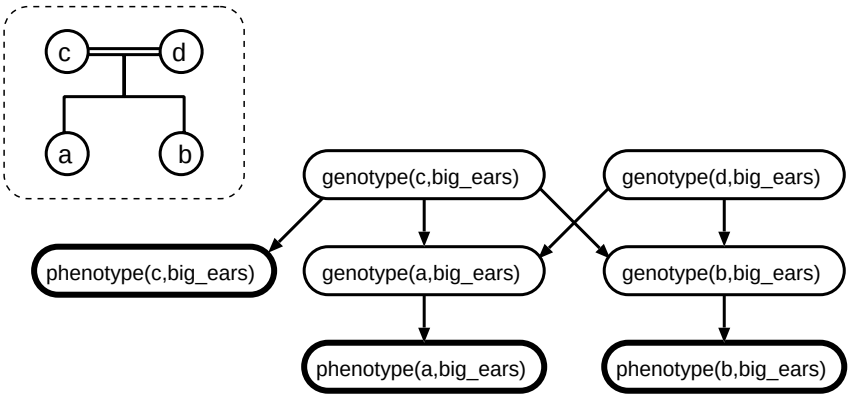

図 6.1: 家系樹と構築された Bayesian ネットワーク.

$$\texttt{phenotype(P,G)} \xleftarrow{0.75} \texttt{genotype(P,G)}. \tag{6.2}$$

式 6.1 は「子 P の親 Q が遺伝子 G をもつとき，P が G を受け継ぐ条件つき確率は 0.5 である」ことを表し，式 6.2 は「ある人が遺伝子 G を持っていたとき，G が表現型中に観測される条件つき確率は 0.75 である」ことを表す。phenotype(P,G) と genotype(P,G) が Bayesian ネットワークの確率変数になり得る．一方，parent(P,Q) は文脈変数と呼ばれ，Bayesian ネットワークを構築するときの論理的制約として用いられる．例えば図 6.1 に特定の家系樹（点線の囲み内）を与えたときに式 6.1, 6.2 を用いて構築された Bayesian ネットワークを示す．更に Koller らは構築された Bayesian ネットワーク上で EM アルゴリズム（節 2.1.4.2）を用いてパラメータ学習することを提案した [26]。しかし，どのような確率空間を考えて EM 学習を行なったのかなど，節 2.1.1 で説明した Dempster らの形式化の中での議論が為されていない．

Poole の**確率的ホーンアブダクション** (probabilistic Horn abduction) は表現形式として我々の枠組に近いが，Sato らが指摘しているように [56]，意味論において本質的ではない仮定をおいている．また，パラメータ学習（EM 学習）については触れられていない．

対数線形モデルは最近の統計的自然言語処理研究と大きな関わりがある。Abney は Brew が提案していた**確率的 HPSG** (stochastic head-driven phrase structure grammar) [6] において正しいパラメータ学習が行なわれていないことを改めて指摘し，**確率属性文法** (stochastic attribute-value grammar) と**改良反復スケーリング法** (improved iterative scaling) [16] を用いたパラメータ学習法を提案した [1]．ただし，Abney も Brew も規則適用に関するすべての情報，すなわち解析済みコーパスが与えられていると仮定している．それに対し Riezler は統計的言語モデリングの道具として**確率的制約論理プログラム** (probabilistic constraint logic program) を提案し，反復改良スケーリング法を（EM アルゴリズムのように）不完全データを扱えるように拡張した *Iterative Maximization* というパラメータ学習法を提案した [50]．一方，Cussens は Muggleton の *Stochastic Logic Programs* (SLP) [37] を対数線形モデルの表現に用いることを提案した [13, 14].

Ω 上の対数線形モデルは $\omega \in \Omega$ に対して次の確率

$$Pr(\omega) = \frac{1}{Z} \exp\left(\sum_i \lambda_i f_i(\omega)\right) \tag{6.3}$$

を与えるものである．ただし，f_i はこの分布の素性 (feature) であり，λ_i はモデルの確率パラメー

タである．そして

$$Z = \sum_{\omega \in \Omega} \exp\left(\sum_i \lambda_i f_i(\omega)\right) \tag{6.4}$$

は正規化定数である．Riezler や Cussens の枠組においては Ω を可能な証明木の集合と考え，Abney らの枠組においては可能な構文木の集合と考える．式 6.3 を見て分かるように，対数線形モデルは非常に一般的な形をしており，自由なモデリングが可能であるが，反面，証明木（構文木）の数は組合せ的に増えるので，式 6.4 による Z 項の計算は一般には困難である．Abney や Cussens らはサンプリングにより Z を計算することを提案しているが，サンプリング誤差などの議論はない．それに対し，本論文の提案手法では可能な限り高速に計算できるように OLDT と gEM アルゴリズムという道具を用意している．

また，式 6.3 から分かるように Riezler や Cussens は可能な証明木の集合上に分布を与えている．従ってこれらを統計知識表現の観点から見たとき，証明手続きを介さなければプログラムに確率的意味を与えることができない．すなわち宣言的な確率的な意味は与えることができない．一方，PRISM プログラムでは最小モデル意味論の確率化である分布意味論に基づいているので，節 3.3.1 の HMM プログラムで見たように，記述した統計モデルを宣言的に理解・検証することが可能である．

本論文では，知識表現言語として一階述語論理に焦点を絞ったが，Koller らは関数的プログラムで確率現象をモデル化する方法を提案し，その上での確率推論アルゴリズムを示した [27]．その関数プログラムで PCFG を記述すればその確率計算アルゴリズムは Inside アルゴリズム（図 2.1 の手続き Get-Beta）と同効率で動作すると述べている．記述力の高い言語（モデルクラス）$\mathcal{M}_1$ で特定のモデルクラス $\mathcal{M}_2$ に属するモデルを記述したときに $\mathcal{M}_1$ の確率推論アルゴリズムが $\mathcal{M}_2$ に専用の確率推論アルゴリズムと同じオーダで計算できるという主張は，本論文と同じであるが，彼女らは EM アルゴリズムは提案していない（つまり Inside-Outside アルゴリズムでいえば EM 学習に必要な Get-Alpha と Get-Beta のうち，Get-Beta の方しか提案していない）点で我々と異なる．

第7章　結論

統計的知識を表現するための論理プログラムに基づく表現言語 PRISM に対する EM (expectation-maximization) アルゴリズムによるその効率的なパラメータ学習法を提案した．パラメータを観測データから学習することにより，統計的知識のより客観的な記述が可能になる．また，PRISM は Chomsky 階層の 0 型文法の確率化であるので，隠れマルコフモデル (HMM)，確率文脈自由文法 (PCFG)，Bayesian ネットワークといった従来の統計モデル記述言語では表現できなかった複雑な確率現象も PRISM により記述できる．

また，この効率的な EM アルゴリズム（グラフィカル EM アルゴリズムと呼ばれる）が PRISM プログラムに適用可能になる（自明ではない）条件を見つけ，提案した EM アルゴリズムが効率的とされてきた既存の専用 EM アルゴリズムと同オーダで計算できることを HMM, PCFG, 単結合 Bayesian ネットワークのすべてにおいて示した．これはモデル記述者がモデルに（マルコフ性などの）独立性を仮定すればその分だけ EM 学習の高速化が可能になることを意味し，モデル記述者は設計しようとする統計モデルに対して表現の一般性と学習の効率性のバランスをとることが可能になる．

そして，本論文では統計的言語モデルを容易に記述できるように確定節文法 (DCG) の確率化である確率確定節文法 (PDCG) を提案した．この形式により，確率分布に文脈依存性をもたせた CFG に基づく確率文法 (CFGs with context-sensitive probability) を容易に記述することができる．PDCG で記述した確率文法についてもその計算量を評価した．更に，対象モデルを PCFG に絞った場合，高速な構文解析器（パーザ）とグラフィカル EM アルゴリズムを組み合わせる方法を提案した．現実のコーパス（学習データ）と文法を用いた実験では，効率的とされてきた Inside-Outside アルゴリズムに比べて，全く同じ計算をしているにも関わらず平均文長でおよそ 1,000 倍の速度向上が得られた．

今後の課題としては, まず PRISM 処理系の実装が挙げられる．現在，グラフィカル EM アルゴリズムは既に実装されているので，OLDT インタプリタ（もしくはその Prolog エミュレータ）を実装する必要がある．また，PRISM の高い記述力と PDCG などに見られるような統計的言語モデルとの相性の良さ，EM 学習の効率性を考えると，HPSG (head-driven phrase structure grammar) などの単一化文法を PRISM に基づき確率化するのは興味深い課題である．

謝辞

本研究にあたりまして，終始変わらぬ熱心な指導を頂いた指導教官の佐藤泰介教授に心より感謝致します．本論文の実験で用いました ATR 対話コーパスと日本語文法の改訂版は東京工業大学田中・徳永研のご厚意により提供頂きました．記して感謝致します．また，同研究室白井清昭助手には上記コーパス・文法に関する情報やテキスト処理プログラムの提供，文献紹介など大変お世話になりました．重ねて感謝申し上げます．また，東京工業大学 佐藤 (泰) 研究室の皆様には大変お世話になりました．特に，上田展久氏には本研究に関しまして論文紹介を含め，貴重なコメントを数多く頂きました．そして，森高志氏には本論文の実験において大きな御助力を頂きました．また，研究室の計算環境の整備に関しまして船田悟志氏に大変お世話になりました．この場を借りてお礼申し上げます．また，PRISM 処理系（旧版）を使用して下さった佐藤 (泰) 研究室 OB の萩原，大山，古川，荻林，角瀬，石丸，熊倉各氏には貴重なコメントを頂きました．感謝致します．

参考文献

[1] Abney, S. (1997). Stochastic attribute-value grammars. *Computational Linguistics*, Vol.23, No.4, pp.597–618.

[2] 浅井潔 (2000). 配列情報と確率モデル. 人工知能学会誌, Vol.15, No.1, pp.35–42.

[3] Baker, J. K. (1979). Trainable grammars for speech recognition, *Proc. of Spring Conf. of the Acoustical Society of America*, pp.547–550.

[4] Binder, J., Murphy, K. and Russell, S. (1997). Space-efficient inference in dynamic probabilistic networks. *Proc. of the 15th Intl. Joint Conf. on Artificial Intelligence*, pp.1292–1296.

[5] Breese, J. S. (1992). Construction of belief and decision networks. *Computational Intelligence*, Vol.8, No.4, pp.624–647.

[6] Brew, C. (1995). Stochastic HPSG. *Proc. of the 7th Conf. of European Chapter of the Association for Computational Linguistics*, pp.83–89.

[7] Castillo, E., Gutierrez, J.M. and Hadi, A.S. (1997). *Expert Systems and Probabilistic Network Models*. Springer-Verlag.

[8] Charniak, E. (1993). *Statistical Language Learning*. The MIT Press.

[9] Charniak, E. and Carroll, G. (1994). Context-sensitive statistics for improved grammatical language models. *Proc. of the 12th National Conf. on Artificial Intelligence*, pp.728–733.

[10] Chi, Z. and Geman, S. (1998). Estimation of probabilistic context-free grammars. *Computational Linguistics*, Vol.24, No.2, pp.299–305.

[11] Clark, K. (1978). Negation as failure. In Gallaire, H., and Minker, J. (eds), *Logic and Databases*, pp.293–322, Plenum Press.

[12] Console, L., Theseider Dupre, D. and Torasso, P. (1991). On the relationship between abduction and deduction, *Logic Computation*, Vol.1, No.5, pp.661–690.

[13] Cussens, J. (1999). Loglinear models for first-order probabilistic reasoning. *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence*, pp.126–133.

[14] Cussens, J. (2000). Stochastic logic programs: sampling, inference and applications. *Proc. of the 16th Conf. on Uncertainty in Artificial Intelligence*, to appear.

[15] Dean, T. (1989). A model for reasoning about persistence and causation. *Computational Intelligence*, Vol.5, No.3, pp.142–150.

[16] Della Pietra, S., Della Pietra, V. and Lafferty, J. (1997). Inducing features of random fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.19, No.4, pp.380–393.

[17] Dempster, A. P., Laird, N. M. and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm (with discussion), *Royal Statistical Society*, Series B, Vol.39, No.1, pp.1–38.

[18] Glesner, S. and Koller, D. (1995). Constructing flexible dynamic belief networks from first-order probabilistic knowledge bases. *Proc. of European Conf. on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, LNCS 946, Springer, pp.217–226.

[19] 石畑清 (1989). アルゴリズムとデータ構造. 岩波書店.

[20] 亀谷由隆, 佐藤泰介 (1997). 記号的統計モデル言語 PRISM. 電子情報通信学会技術報告, Vol.97, No.373, AI97-33, pp.71–78.

[21] Kameya, Y., Ueda, N. and Sato, T. (1999). A graphical method for parameter learning of symbolic-statistical models. *Proc. of the 2nd Intl. Conf. on Discovery Science*, LNAI 1721, Springer, pp.264–276.

[22] 亀谷由隆, 佐藤泰介 (2000). 統計的記号処理言語 PRISM, 発見科学とデータマイニング（bit 別冊）, 第 2 章, 共立出版.

[23] Kameya, Y. and Sato T. (2000). Efficient EM learning for parameterized logic programs. *Proc. of the 1st Conf. on Computational Logic*, LNAI Vol.1861, pp.269–294.

[24] Kita, K., Morimoto, T., Ohkura, K., Sagayama, S. and Yano, Y. (1994). Spoken sentence recognition based on HMM-LR with hybrid language modeling. *IEICE Trans. on Information & Systems*, Vol.E77-D, No.2.

[25] 北研二 (1999). 確率的言語モデル. 東京大学出版会.

[26] Koller, D. and Pfeffer, A. (1997). Learning probabilities for noisy first-order rules. *Proc. of the 15th Intl. Joint Conf. on Artificial Intelligence*, pp.1316–1321.

[27] Koller, D., McAllester, D. and Pfeffer, A. (1997). Effective Bayesian Inference for Stochastic Programs. *Proc. of 15th National Conf. on Artificial Intelligence*, pp.740–747.

[28] Lari, K. and Young, S. J. (1990). The estimation of stochastic context-free grammars using the Inside-Outside algorithm. *Computer Speech and Language*, Vol.4, pp.35–56.

[29] Lloyd, J. W. (1984). *Foundations of Logic Programming*. Springer-Verlag, 1984.

[30] Lukasiewicz, T. (1998). Probabilistic logic programming. *Proc. of the 13th European Conference on Artificial Intelligence*, pp.388–392.

[31] Magerman, D. and Marcus, M. (1991). $\mathcal{P}$earl: a probabilistic chart parser, *Proc. of the 5th Conf. on the European Chapter of the ACL*, pp.15–20.

[32] Magerman, D. and Weir, C. (1992). Efficiency, robustness and accuracy in $\mathcal{P}$icky chart parsing, *Proc. of the 30th Annual Meeting of the ACL*, pp.40–47.

[33] Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*. The MIT Press.

[34] McLachlan, G. J. and Krishnan, T. (1997). *The EM Algorithm and Extensions*. Wiley Interscience.

[35] 宮川雅巳 (1987). EM アルゴリズムとその周辺. 応用統計学, Vol.16, No.1.

[36] 本村陽一, 佐藤泰介 (2000). ベイジアンネットワーク — 不確定性のモデリング技術 —. 人工知能学会誌, Vol.15, No.4, pp.575–582.

[37] Muggleton, S. (1996). Stochastic logic programs. In *Advances in Inductive Logic Programming* (Raedt, L. De ed.), OSP Press, pp.254–264.

[38] 永田 昌明 (1999). “形態素, 構文解析.” 自然言語処理 —基礎と応用—（田中穂積監修）第 1 章 (pp.2–45), 電子情報通信学会.

[39] Ng, R. and Subrahmanian, V. S. (1992). Probabilistic logic programming. *Information and Computation*, Vol.101, pp.150–201.

[40] Ngo, L. and Haddawy, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, Vol.171, pp.147–177.

[41] Nilsson, N. J. (1986). Probabilistic logic. *Artificial Intelligence*, Vol.28, pp.71–87.

[42] 西尾真喜子 (1978). 確率論. 実教出版.

[43] Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann.

[44] Pereira, F. C. N. and Warren, D. H. D. (1980). Definite clause grammars for language analysis — a survey of the formalism and a comparison with augumented transition networks. *Artificial Intelligence*, Vol.13.

[45] Poole, D. (1991). Compiling a default reasoning system into Prolog. *New Generation Computing*, Vol.9, No.1.

[46] Poole, D. (1993). Probabilistic Horn abduction and Bayesian networks. *Artificial Intelligence*, Vol.64, pp.81–129.

[47] Pynadath, D. V. and Wellman, M. P. (1998). Generalized queries on probabilistic context-free grammars, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.20, No.1, pp.65–77.

[48] Pynadath, D. V. (1999). *Probabilistic Grammars for Plan Recognition*. Ph. D. dissertation, University of Michigan.

[49] Rabiner, L. and Juang, B. (1993). *Foundations of Speech Recognition*. Prentice-Hall.

[50] Riezler, S. (1997). *Probabilistic constraint logic programming*. Arbeitsberichte des SFB 340 Bericht Nr. 117, Universität Tübingen.

[51] Russell, S. and Norvig, P. (1995). *Artificial Intelligence: A Modern Approach*. Prentice-Hall.

[52] Russell, S. (1999). Expressive probability models in science. *Proc. of the 2nd Intl. Conf. on Discovery Science*, LNAI 1721, Springer, pp.13–16.

[53] Sato, T. (1995). A statistical learning method for logic programs with distribution semantics. *Proc. of the 12th Intl. Conf. on Logic Programming*, pp.715–729.

[54] Sato, T. and Kameya, Y. (1997). PRISM: a language for symbolic-statistical modeling. *Proc. of IJCAI'97*, pp.1330–1335.

[55] Sato, T. (1998). Modeling scientific theories as PRISM programs. *ECAI-1998 Workshop on Machine Discovery*, pp.37–45.

[56] Sato, T. and Kameya, Y. (2000). A Viterbi-like algorithm and EM learning for statistical abduction. *Proc. of UAI-2000 Workshop on Fusion of Domain Knowledge with Data for Decision Support*.

[57] Sterling, L. and Shapiro, E. (1986). *The Art of Prolog*. The MIT Press.

[58] Tamaki, H. and Sato, T. (1986). OLD resolution with tabulation. *Proc. of the 3rd Intl. Conf. on Logic Programming*, LNCS 225, Springer, pp.84–98.

[59] 田中穂積 (1988). 自然言語処理の基礎. 産業図書.

[60] 田中穂積, 竹澤寿幸, 衛藤 純司 (1997). MSLR 法を考慮した音声認識用日本語文法 — LR 表工学 (3) —. 音声言語情報処理研究会, 情報処理学会, Vol.97.

[61] Tanner, M. (1993). *Tools for Statistical Inference* (2nd ed.). Springer-Verlag.

[62] Tomita, M. and Ng, S. (1991). The generalized LR parsing algorithm. In M. Tomita (ed.), *Generalized LR Parsing*, Kluwer Academic Publishers.

[63] Warren, D. S. (1992). Memoing for logic programs. *Communications of the ACM*, Vol.35, No.3, pp.93–111.

[64] Wetherell, C.S. (1980). Probabilistic languages: a review and some open questions. *Computing Surveys*, Vol.12, No.4, pp.361–379.

[65] Zweig, G. and Russell, S. (1998). Speech recognition with dynamic Bayesian networks. *Proc. of the 16th National Conf. on Artificial Intelligence*, pp.173–180.

付 録A　Bayesian ネットワークの記述

節 2.1.4 で Bayesian ネットワークとその条件つき確率計算アルゴリズムおよび EM アルゴリズムを簡単に記述したが，本付録では，それらに対しより詳細な記述を与える。読みやすさを考え，節 2.1.4 の説明と重複させたり，説明の順番を入れ替えた個所もある。

確率変数の集合が与えられたとき，Bayesian ネットワーク[1] [7, 43, 51] はその同時確率分布（joint probability distribution; 以下，同時分布）[2]を有向非循環グラフ (DAG) で表現する。DAG の形状から導かれる（条件つき）独立性によってその同時確率分布が計算しやすい形に簡単化される。ここでは文献 [7, 51] に基づいて説明する。

A.1　Bayesian ネットワークの構築

節 2.1.4 では天下り的に Bayesian ネットワークを記述したが，Bayesian ネットワークの構築は以下の手順で行われる。(i) 対象となる確率現象を表現する確率変数の有限集合 $V_{\rm bn} = \{X_1, X_2, \ldots, X_{|V_{\rm bn}|}\}$ を定め，Bayesian ネットワークのノードと同一視する。次に (ii) 各 $X_i \in V_{\rm bn}$ に対して，X_i に直接の影響を与える確率変数の集合 $\Pi_i \subset V_{\rm bn}$ を考え，(iii) Π_i 中の各ノードから X_i に有向リンクを張る。そして (iv) 各ノード X_i に**条件つき確率表**（conditional probability table; 以下 CPT）を付与する。X_i に付与される CPT には，親ノード（の確率ベクトル）$\mathbf{\Pi}_i$ が実現値 $\boldsymbol{u} \in \mathcal{D}(\mathbf{\Pi}_i)$ をとったとき X_i が値 $x_i \in \mathcal{D}(X_i)$ をとる条件つき確率値

$$\theta_i(x_i|\boldsymbol{u}) = Pr(X_i = x_i | \mathbf{\Pi}_i = \boldsymbol{u}) \tag{A.1}$$

が格納されている。X_i が根ノード（親をもたないノード）であるとき ($\Pi_i = \emptyset$)，$\theta_i(x_i|\boldsymbol{u})$ は単に $\theta_i(x_i)$ と書かれる。

このように構築された Bayesian ネットワークを改めて 3 つ組 $\langle V_{\rm bn}, L_{\rm bn}, \boldsymbol{\theta}_{\rm bn}\rangle$ で表す。ただし，$L_{\rm bn} \stackrel{\rm def}{=} \{(U, X_i) \mid i = 1 \ldots |V_{\rm bn}|,\ U \in \Pi_i\}$ であり，かつネットワーク構造 $\langle V_{\rm bn}, L_{\rm bn}\rangle$ は DAG でなければならない。また，$\boldsymbol{\theta}_{\rm bn}$ は $\{\theta_i(x|\boldsymbol{u}) \mid i = 1 \ldots |V_{\rm bn}|,\ x \in \mathcal{D}(X_i),\ \boldsymbol{u} \in \mathcal{D}(\mathbf{\Pi}_i)\}$ の要素を並べたベクトルである。$\boldsymbol{\theta}_{\rm bn}$ の値は人間が手で与えるか，**証拠** (evidence) と呼ばれる観測データから最尤推定することによって得られる。

図 2.5 に $|V_{\rm bn}| = 7$ かつ $X_1 = A,\ X_2 = B,\ X_3 = C,\ X_4 = D,\ X_5 = E,\ X_6 = F,\ X_7 = G$ なる Bayesian ネットワークを示す。例えば D の親ノードの集合 Π_4 は $\{A, B\}$ である。次節では Bayesian ネットワークおよび関連アルゴリズムを理解する上で最も重要な概念である**条件つき独立性** (conditional independence) と **d-分離性** (d-separation) を導入する。

[1] 信念ネットワーク (belief network) など他にも多くの名称が与えられている。
[2] joint distribution は結合分布と訳されることもある。

A.2 条件つき独立性と d-分離性

条件つき独立性は，Bayesian ネットワークとは独立に定義される概念である。はじめに，3 つの互いに素な確率変数集合 X, Y, Z を考える[3]。任意の $\boldsymbol{x} \in \mathcal{D}(\boldsymbol{X})$, $\boldsymbol{y} \in \mathcal{D}(\boldsymbol{Y})$, $\boldsymbol{z} \in \mathcal{D}(\boldsymbol{Z})$ について

$$Pr(\boldsymbol{X}=\boldsymbol{x}, \boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{Z}=\boldsymbol{z}) = Pr(\boldsymbol{X}=\boldsymbol{x}|\boldsymbol{Z}=\boldsymbol{z}) \cdot Pr(\boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{Z}=\boldsymbol{z}) \tag{A.2}$$

もしくはこれの両辺を $Pr(\boldsymbol{X}=\boldsymbol{x}|\boldsymbol{Z}=\boldsymbol{z})$ で割って得られる

$$Pr(\boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{X}=\boldsymbol{x}, \boldsymbol{Z}=\boldsymbol{z}) = Pr(\boldsymbol{Y}=\boldsymbol{y}|\boldsymbol{Z}=\boldsymbol{z}) \tag{A.3}$$

が成り立つとき，またそのときに限り「Z を与えたとき X と Y は条件つき独立 (conditionally independent) である」といい，$X \perp\!\!\!\perp Y \mid Z$ と書く。X と Y の（無条件な）独立性は，上で $Z=\emptyset$ とおいた特別な場合である[4]。

また，d-分離性は有向グラフに関係する概念である。まず，*head-to-head* **ノード** という考えを導入する。有向グラフ $\langle V = \{X_1, X_2, \ldots, X_{|V|}\}, L\rangle$ 中の無向パス（有向リンクの方向性を除いたときに得られるパス）$-X_i - X_j - X_k -$ を考える。そして L 中にリンク $X_i \rightarrow X_j$ および $X_j \leftarrow X_k$ が存在するとき,「X_j はその無向パスにおける head-to-head ノードである」という。次に，3 つの互いに素な確率変数（ノード）集合 $X, Y, Z \subset V$ を考える。有向グラフ $\langle V, L\rangle$ において，X 中のノードから Y 中のノードに至るような無向パスすべてについて

1. A がそのパスの head-to-head ノードであり，かつ A およびその子孫は Z に含まれない
2. A がそのパスの head-to-head ノードでなく，かつ A が Z に含まれる

のいずれかが成り立つようなノード $A = X_i$ が存在するとき，またそのときに限り「$\langle V_{\rm bn}, L_{\rm bn}\rangle$ において Z は X と Y を d-分離する」という。$\langle V_{\rm bn}, L_{\rm bn}\rangle$ において Z が X と Y を d-分離するとき，またそのときに限り $X \perp\!\!\!\perp Y \mid Z$ であるとする[5]。

すると，先述した Bayesian ネットワーク $\langle V_{\rm bn}, L_{\rm bn}, \boldsymbol{\theta}_{\rm bn}\rangle$ の構築法から，各 $X_i \in V_{\rm bn}$ について Π_i は $\{X_i\}$ と $V_{\rm bn} \setminus (\{X_i\} \cup \Pi_i)$ を d-分離することは明らかである。すなわち各 $X_i \in V_{\rm bn}$ について条件つき独立性 $\{X_i\} \perp\!\!\!\perp \left(V_{\rm bn} \setminus (\{X_i\} \cup \Pi_i)\right) \mid \Pi_i$ が得られる。従って，条件つき独立性の定義より，Bayesian ネットワークは $V_{\rm bn} = \{X_1, X_2, \ldots, X_{|V_{\rm bn}|}\}$ の同時分布 $Pr(X_1, X_2, \ldots, X_{|V_{\rm bn}|})$ を次のような簡単な形で規定する ：

$$Pr(X_1=x_1, X_2=x_2, \ldots, X_{|V_{\rm bn}|}=x_{|V_{\rm bn}|}) = \prod_{i=1}^{|V_{\rm bn}|} Pr(X_i=x_i|\boldsymbol{\Pi}_i=\boldsymbol{u}_i) = \prod_{i=1}^{|V_{\rm bn}|} \theta_i(x_i|\boldsymbol{u}_i) \tag{A.4}$$

(ただし $i = 1 \ldots |V_{\rm bn}|$ について $x_i \in \mathcal{D}(X_i)$ であり，$\boldsymbol{u}_i$ には対応する x_j $(x_j \in \mathcal{D}(X_j),\ X_j \in \Pi_i)$ が含まれる)。例えば，図 2.5 の Bayesian ネットワークにおいて $a, b, \ldots$ をそれぞれ $A, B, \ldots$ の実現値（$a \in \mathcal{D}(A),\ b \in \mathcal{D}(B),\ \ldots$）とすれば，同時分布 $Pr(a, b, c, d, e, f, g)$ は次のように簡単化される：

$$Pr(a, b, c, d, e, f, g) = Pr(a)Pr(b)Pr(c|a)Pr(d|a, b)Pr(e)Pr(f|d)Pr(g|d, e). \tag{A.5}$$

以下の節では，単結合 Bayesian ネットワークにおける条件つき確率計算法と Bayesian ネットワーク用の EM アルゴリズムを順に記述する。

[3]少し紛らわしいが，特に断らない限り，正の整数 i, j, k, ℓ などで添え字づけされた X_i, U_j, Y_k, Z_ℓ などは確率変数を表し，それ以外の X, U, Y, Z などは確率変数の集合を表すものとする。

[4]これは次の理由による：$Z = \emptyset$ のとき $\boldsymbol{Z}=\boldsymbol{z}$ は恒真な命題を表すことに注意すれば，式 A.2 は任意の $\boldsymbol{x} \in \mathcal{D}(\boldsymbol{X})$, $\boldsymbol{y} \in \mathcal{D}(\boldsymbol{Y})$ について $Pr(\boldsymbol{X}=\boldsymbol{x}, \boldsymbol{Y}=\boldsymbol{y}) = Pr(\boldsymbol{X}=\boldsymbol{x}) \cdot Pr(\boldsymbol{Y}=\boldsymbol{y})$ であるといっている。これは X と Y が独立であることの定義式である.

[5]文面より明らかなように，この文は他から導かれるものではなく，有向グラフを条件つき独立性の表現として解釈するための仮定であることに注意する。

A.3 π-λ 計算法

一旦，Bayesian ネットワークが記述されたら，次の重要な仕事は観測された証拠から新たな確率的推論，すなわち証拠の下での条件つき確率を計算することである。単結合 Bayesian ネットワークに対する効率のよい計算法として Pearl による π 確率，λ 確率（後に定義する）に基づく計算法 [43] が知られている。本論文ではこの計算法を π-λ **計算法**と呼び，以下に記述する[6]。

ある状況において，$\boldsymbol{E}=\boldsymbol{e}$ であると観測された変数集合 $E \subset X$ 中の各変数を**証拠変数** (evidential variable) と呼び，式 $\boldsymbol{E}=\boldsymbol{e}$ を証拠と呼ぶ。残りの変数から成る集合 $X \setminus E$ 中の各変数を**非証拠変数** (nonevidential variable) と呼ぶ。π-λ 計算の目標は，証拠 $\boldsymbol{E}=\boldsymbol{e}$ が与えられ，確率変数 X_i が指定されたときに，$\boldsymbol{E}=\boldsymbol{e}$ の下での X_i の条件つき確率分布 $Pr(X_i|\boldsymbol{E}=\boldsymbol{e})$ [7] を計算し，出力することである。

条件つき確率分布 $Pr(X_i|\boldsymbol{E}=\boldsymbol{e})$ の計算手順は X_i の周辺分布 $Pr(X_i)$ の計算手順（$E=\emptyset$ とおいた場合）と同じである。従って $Pr(X_i|\boldsymbol{E}=\boldsymbol{e})$ を高速に計算するには $Pr(X_i)$ が高速に計算できなくてはならない。π-λ 計算法では単結合ネットワーク特有の d-分離性から導かれる条件つき独立性を利用して，周辺分布の計算式 $\sum_{\boldsymbol{y}\in\mathcal{D}(\boldsymbol{Y})} Pr(X_i, \boldsymbol{Y}=\boldsymbol{y})$ （ただし $Y=V_{\rm bn}-\{X_i\}$）を因数分解 (factorization) することによって計算の高速化を図っている[8]。

π-λ 計算の目標である条件つき確率分布 $Pr(X_i|\boldsymbol{E}=\boldsymbol{e})$ の計算は，各 $x_i \in \mathcal{D}(X_i)$ について $Pr(X_i=x_i|\boldsymbol{E}=\boldsymbol{e}) = Pr(x_i|\boldsymbol{e})$ を求めることにより実現される。$Pr(x_i|e)$ は次のように $Pr(x_i, \boldsymbol{e})$ を**正規化**することによって得られる：

$$Pr(x_i|\boldsymbol{e}) = \alpha Pr(x_i, \boldsymbol{e}). \tag{A.6}$$

ここで α は正規化定数である。そして，$E^+_{X_i}$ ($E^-_{X_i}$) を X_i がその親（子）を通してアクセスできる E の部分集合とする ($E = E^+_{X_i} \cup E^-_{X_i}$)。単結合ネットワークの定義より，任意の 2 ノード間のパスは一意に決まる（そして当然 X_i を通る）ため，X_i の祖先から X_i の子孫へのパスは存在しない。従って $E^+_{X_i}$ と $E^-_{X_i}$ は互いに素である。これを「E は $E^+_{X_i}$ と $E^-_{X_i}$ に分割された」という。$\boldsymbol{E}^+_{X_i}$, $\boldsymbol{E}^-_{X_i}$ の実現値をそれぞれ $\boldsymbol{e}^+_{X_i}$, $\boldsymbol{e}^-_{X_i}$ とおくと，式 A.6 右辺の $Pr(x_i, \boldsymbol{e})$ は次のように変形される：

$$\begin{aligned} Pr(x_i, \boldsymbol{e}) &= Pr(x_i, \boldsymbol{e}^+_{X_i}, \boldsymbol{e}^-_{X_i}) = Pr(x_i, \boldsymbol{e}^+_{X_i}) \cdot Pr(\boldsymbol{e}^-_{X_i}|x_i, \boldsymbol{e}^+_{X_i}) \\ &= Pr(x_i, \boldsymbol{e}^+_{X_i}) \cdot Pr(\boldsymbol{e}^-_{X_i}|x_i). \end{aligned} \tag{A.7}$$

---

[6] 本節では Castillo ら [7] に従い，Pearl が π 確率に与えた解釈を若干変更した π-λ 計算法を記述する（変更点および変更した理由は後述する）。また，Pearl は forward-chaining に基づく**証拠伝搬** (propagation of evidence) という計算法を提案しているが，ここで記述するのは Russell らの示した backward-chaining に基づく計算法 [51] である。すなわち，ここで述べる π-λ 計算の記述においては，π 確率の解釈は Castillo らに従い，入出力および計算手順は Russell らに従う。

[7] 具体的には，例えば各 $x \in \mathcal{D}(X_i)$ について $Pr(X_i=x|\boldsymbol{E}=\boldsymbol{e})$ の値が書かれた表。

[8] 例えば，図 2.5 の例では周辺分布 $Pr(D=d)$ は次式から得られる：

$$\begin{aligned} Pr(d) &= \textstyle\sum_{a,b,c,e,f,g} Pr(a)Pr(b)Pr(c|a)Pr(d|a,b)Pr(e)Pr(f|d)Pr(g|d,e) \qquad \cdots (*) \\ &= \left(\textstyle\sum_{a,b,c} Pr(a)Pr(b)Pr(c|a)Pr(d|a,b)\right)\left(\textstyle\sum_{e,f,g} Pr(e)Pr(f|d)Pr(g|d,e)\right) \\ &= \left(\textstyle\sum_a Pr(a)\left(\sum_c Pr(c|a)\left(\sum_b Pr(b)Pr(d|a,b)\right)\right)\right) \cdot \\ &\qquad \left(\textstyle\sum_e Pr(e)\left(\sum_f Pr(f|d)\left(\sum_g Pr(g|d,e)\right)\right)\right) \qquad \cdots (**) \end{aligned}$$

$A, B, \ldots$ が 2 値をとる確率変数であったとき，式 $(*)$ では $2^6-1=63$ 回の加算と $2^6\cdot(7-1)=384$ 回の乗算が必要になる。一方，式 $(**)$ では $(2-1)\cdot 3+(2-1)\cdot 3=6$ 回の加算と $(2\cdot(2-1))^3+(2\cdot(2-1))^3=16$ 回の乗算で済む。

$\{X_i\}$ が $E^+_{X_i}$ と $E^-_{X_i}$ を d-分離する（すなわち $E^-_{X_i} \perp\!\!\!\perp E^+_{X_i} \mid \{X_i\}$）ので式 A.7 は成り立つことが分かる。ここで，

$$\pi_{X_i}(x_i) \stackrel{\text{def}}{=} Pr(x_i, \boldsymbol{e}^+_{X_i}) \tag{A.8}$$

$$\lambda_{X_i}(x_i) \stackrel{\text{def}}{=} Pr(\boldsymbol{e}^-_{X_i} | x_i) \tag{A.9}$$

を導入し，それぞれ X_i の π **確率**，λ **確率**と呼ぶ。すると式 A.7 は次のように書ける[9]：

$$Pr(x_i, \boldsymbol{e}) = \pi_{X_i}(x_i)\lambda_{X_i}(x_i). \tag{A.10}$$

次に，X_i の親集合を $\{U_1, U_2, \ldots, U_M\}$ とおき，X_i の子集合を $\{Y_1, Y_2, \ldots, Y_K\}$ とおく。更に各 $k = 1 \ldots K$ について，X_i を除く Y_k の親集合を $\{Z_{k1}, Z_{k2}, \ldots, Z_{kN}\}$ で表す。このとき，

$E^+_{U_j \backslash X_i} \stackrel{\text{def}}{=}$ リンク $U_j \rightarrow X_i$ に対して U_j 側に含まれる $E^+_{X_i}$ の部分集合,

$E^-_{Y_k \backslash X_i} \stackrel{\text{def}}{=}$ リンク $X_i \rightarrow Y_k$ に対して Y_k 側に含まれる $E^-_{X_i}$ の部分集合

$E^+_{Z_{k\ell} \backslash Y_k} \stackrel{\text{def}}{=}$ リンク $Z_{k\ell} \rightarrow Y_k$ に対して $Z_{k\ell}$ 側に含まれる $E^-_{Y_k \backslash X_i} \setminus E^-_{Y_k}$ の部分集合

図 2.6 を考えると，先程と同様に単結合ネットワークの性質から，$E^+_{U_j \backslash X_i}$, $E^-_{Y_k \backslash X_i}$ $(j = 1 \ldots M,\ k = 1 \ldots K)$ の任意の 2 つは互いに素である。同様に，各 $k = 1 \ldots K$ について $E^-_{Y_k}$, $E^+_{Z_{k\ell} \backslash Y_k}$ $(\ell = 1 \ldots N)$ の任意の 2 つは互いに素である。更に，単結合ネットワークが表す d-分離性より次の条件的独立性が成り立つ。

$$E^+_{U_1 \backslash X_i} \perp\!\!\!\perp E^+_{U_2 \backslash X_i} \perp\!\!\!\perp \cdots \perp\!\!\!\perp E^+_{U_M \backslash X_i} \Big| \emptyset$$

$$E^-_{Y_1 \backslash X_i} \perp\!\!\!\perp E^-_{Y_2 \backslash X_i} \perp\!\!\!\perp \cdots \perp\!\!\!\perp E^-_{Y_K \backslash X_i} \Big| \{X_i\}$$

$$E^+_{Z_{k1} \backslash Y_k} \perp\!\!\!\perp E^+_{Z_{k2} \backslash Y_k} \perp\!\!\!\perp \cdots \perp\!\!\!\perp E^+_{Z_{kN} \backslash Y_k} \Big| \emptyset \quad \text{for each } k = 1 \ldots K$$

これらの条件つき独立性を利用して，条件つき確率 $Pr(x_i|\boldsymbol{e})$ の再帰的計算アルゴリズムが図 2.7 のように導出される。ただし，$\rho_{X_i}(x_i)$ は $Pr(x_i, \boldsymbol{e})$ を，$\pi_{X_i}(x_i)$ は $Pr(x_i, \boldsymbol{e}^+_{X_i})$ を，$\lambda_{X_i}(x_i)$ は $Pr(\boldsymbol{e}^-_{X_i}|x_i)$ を，$\rho_{X_i \backslash Y_k}(x_i)$ は $Pr(x_i, \boldsymbol{e}^+_{X_i \backslash Y_k})$ を，$\lambda_{Y_k \backslash X_i}(x_i)$ は $Pr(\boldsymbol{e}^-_{Y_k \backslash X_i}|x_i)$ をそれぞれ意味する。図 2.6 を見ると分かるように，この再帰的計算は X_i に近いノードから段階的に，かつ放射状に進んでいく。

A.4 Bayesian ネットワークの EM 学習

Bayesian ネットワークにおける最尤推定問題とは，T 回の独立な観測において証拠 $\boldsymbol{E} = \boldsymbol{e}^{(1)}$, $\boldsymbol{E} = \boldsymbol{e}^{(2)}$, ..., $\boldsymbol{E} = \boldsymbol{e}^{(T)}$ が与えられたとき，尤度 $\prod_{t=1}^{T} Pr(\boldsymbol{E} = \boldsymbol{e}^{(t)} \mid \boldsymbol{\theta}_{\text{bn}})$ を最大にするパラメータ $\boldsymbol{\theta}^*_{\text{bn}}$ を見つけることである。t 回目の観測における非証拠変数 $X_i \in (V_{\text{bn}} \setminus E)$ の実現値 x_i が観測できないため，最尤推定法として EM アルゴリズムが考えられる。Bayesian ネットワーク用の EM アルゴリズムでは，Baum-Welch アルゴリズムや Inside-Outside アルゴリズム同様，はじ

[9] Pearl [43] や Russell [51] らの記述では $\pi_i(x_i)$ を $Pr(x_i|e^+_{X_i})$ と解釈する。このとき，式 A.6 の正規化定数 α を $Pr(e^+_{X_i}|e^-_{X_i})$ とすれば，$Pr(x_i|e) = \alpha\pi_{X_i}(x_i)\lambda_{X_i}(x_i)$ が成り立つ。$Pr(x_i|e^+_{X_i})$ は X_i の祖先（原因側）に証拠 $e^+_{X_i}$ が与えられたときの X_i の確率的振舞いを定めるので，確かに証拠 伝搬 の直観には合う。しかし，後に得られる計算式中に正規化定数 β が現れてしまい，本論文で提案する表現形式との対応がとれなくなる。一方，我々は backward-chaining に基づいて $Pr(x_i|e)$ を計算しているだけなので，証拠伝搬の直観に合わなくても問題は生じない。

めに適切な初期パラメータ値 $\boldsymbol{\theta}_{\mathrm{bn}}^{(0)}$ を与え，ノード X_i の親 $\boldsymbol{\Pi}_i$ が実現値 $\boldsymbol{u}$ をとり，X_i が x_i をとる回数の期待値 $\eta(X_i=x_i,\boldsymbol{\Pi}=\boldsymbol{u})$ を式 A.11 で計算し，その期待値を使って式 A.12 でパラメータを更新する。この更新を（対数）尤度が収束するまで繰り返し，収束したらその時のパラメータ値を推定値 $\boldsymbol{\theta}_{\mathrm{bn}}^{*}$ として終了する。

$$\eta^{(m)}(X_i=x_i,\boldsymbol{\Pi}_i=\boldsymbol{u}) := \sum_{t=1}^{T} Pr(X_i=x_i,\boldsymbol{\Pi}_i=\boldsymbol{u}|\boldsymbol{E}=\boldsymbol{e}^{(t)},\boldsymbol{\theta}_{\mathrm{bn}}^{(m)}) \tag{A.11}$$

$$\theta_{X_i}^{(m+1)}(x_i|\boldsymbol{u}) := \frac{\eta^{(m)}(X_i=x_i,\boldsymbol{\Pi}_i=\boldsymbol{u})}{\sum_{x_i'\in\mathcal{D}(X_i)}\eta^{(m)}(X_i=x_i',\boldsymbol{\Pi}_i=\boldsymbol{u})} \tag{A.12}$$

特に，単結合 Bayesian ネットワークにおいては式 2.32 中の $Pr(X_i=x_i,\boldsymbol{\Pi}_i=\boldsymbol{u}|\boldsymbol{E}=\boldsymbol{e}^{(t)},\boldsymbol{\theta}_{\mathrm{bn}}^{(m)})$ を π-λ 計算法に基づいて次のように計算する．再計算しなくて済むように途中の計算結果は保存しておく．

$$\begin{aligned} Pr(X_i=x_i,\boldsymbol{\Pi}_i=\boldsymbol{u}|\boldsymbol{E}=\boldsymbol{e}) &= \alpha Pr(x_i,\boldsymbol{u},\boldsymbol{e}) \\ &= \alpha Pr(\boldsymbol{e}_{X_i}^{-}|x_i)\theta_{X_i}(x_i|\boldsymbol{u})\prod_{j=1}^{M} Pr(u_j|\boldsymbol{e}_{U_j\setminus X_i}^{+}) \\ &= \alpha\lambda_{X_i}(x_i)\theta_{X_i}(x_i|\boldsymbol{u})\prod_{j=1}^{M}\rho_{U_j\setminus X_i}(u_j). \end{aligned} \tag{A.13}$$

パラメータ $\boldsymbol{\theta}_{\mathrm{bn}}^{(m)}$ と添字 $\cdot^{(t)}$ は省略している．また α は正規化定数である．

付 録B　グラフィカルEMアルゴリズムの正当性

本付録では補題 2 と定理 2 の証明を行なう．はじめに幾つかの準備をしておく．

B.1　同値式の展開

以下で考える PRISM プログラム DB は有限支持，排反性，唯一性，非循環支持，独立支持条件を満たすものとする．また，観測データ $\mathcal{G} = \langle G_1, G_2, \ldots, G_T \rangle$ が与えられているものとする．

$DB = F \cup R$ が有限支持，排反性，唯一性，非循環支持，独立支持条件を満たすとしているので，各 $t = 1 \ldots T$ について DB のルール集合 R に対する完備化 $comp(R)$ の下での同値式の連言

$$\begin{aligned} comp(R) \quad \models \quad & \left(G_t \leftrightarrow \tilde{S}^{(t)}_{0,1} \vee \cdots \vee \tilde{S}^{(t)}_{0,m_0} \right) \\ & \wedge \left(\tau^{(t)}_1 \leftrightarrow \tilde{S}^{(t)}_{1,1} \vee \cdots \vee \tilde{S}^{(t)}_{1,m_1} \right) \wedge \cdots \wedge \left(\tau^{(t)}_{K_t} \leftrightarrow \tilde{S}^{(t)}_{K_t,1} \vee \cdots \vee \tilde{S}^{(t)}_{K_t,m_{K_t}} \right) \end{aligned}$$

(式 4.14) が成り立つ．

各同値式 $\tau^{(t)}_k \leftrightarrow \tilde{S}^{(t)}_{k,1} \vee \cdots \vee \tilde{S}^{(t)}_{k,m}$ の右辺 $D^{(t)}_k = (\tilde{S}^{(t)}_{k,1} \vee \cdots \vee \tilde{S}^{(t)}_{k,m_k})$ 中の各選言子 $\tilde{S}^{(t)}_{k,j}$ $(j = 1 \ldots m_k)$ はファクトまたはテーブルアトムの連言であり，従って D_k はファクトまたはテーブルアトムから成る選言標準形 (DNF) の式である．各 $t = 1 \ldots T$ に対して，次の手続き Rewrite(k) を $k = K_t$ から $k = 1$ まで順に行なうことを考える．

1: **procedure** Rewrite(k) **begin**
2:　　次の (a), (b), (c) を可能な限り行なう：
3:　　(a) $\tau^{(t)}_k$ が右辺に出現するような同値式 E を見つける;
4:　　(b) E 中の $\tau^{(t)}_k$ を $D^{(t)}_k$ に置き換える;
5:　　(c) E の右辺が DNF 式にならなくなった場合,
6:　　DNF 式になるように右辺に同値変形を施す;
7:　　同値式 $\tau^{(t)}_k \leftrightarrow D^{(t)}_k$ を取り除く
8: **end**.

手続き Rewrite(k) 中の各操作を行なった後に得られる式が $comp(R)$ の下では真になることは明らかである．また Rewrite(k) が有限支持，排反性，唯一性，非循環条件を満たすことは明らかである．更に，命題 7 より Rewrite(k) は独立性条件も保存する．次に，**展開 t-極小支持集合**を定義する．

定義 24（展開 t-極小支持集合） $t = 1 \ldots T$ について $\tau^{(t)}_{DB} = \langle \tau^{(t)}_1, \tau^{(t)}_2, \ldots, \tau^{(t)}_{K_t} \rangle$ とおき，観測ゴール G_t を特別なテーブルアトム $\tau^{(t)}_0$ と見なす．そのとき，$\psi^{(t,k)}_{DB}$ $(k = 0 \ldots K_t)$ を次の 1, 2 いずれかによって定義する：

1. $k' = 0 \ldots K_t$ について $\psi^{(t,K_t)}_{DB}(\tau^{(t)}_{k'}) \stackrel{\text{def}}{=} \tilde{\psi}_{DB}(\tau^{(t)}_{k'})$ と定義する．

2. ある $k = 1 \ldots K_t$ について $\psi_{DB}^{(t,k)}$ が定義されているとき，$\psi_{DB}^{(t,k-1)}$ を $k' = 0 \ldots (k-1)$ について以下のように定義する．

$$\psi_{DB}^{(t,k-1)}(\tau_{k'}^{(t)}) \stackrel{\text{def}}{=} \left\{ \tilde{S} \;\middle|\; \tilde{S} = \tilde{S}', \tilde{S}' \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)}), \tau_k^{(t)} \notin \tilde{S}' \right\} \cup \left\{ \tilde{S} \;\middle|\; \exists \tilde{S}', \tilde{S}'' \left(\begin{array}{l} \tilde{S} = \tilde{S}' \cup \tilde{S}'' \setminus \{\tau_k^{(t)}\}, \\ \tilde{S}' \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)}), \tau_k^{(t)} \in \tilde{S}', \\ \tilde{S}'' \in \psi_{DB}^{(t,k)}(\tau_k^{(t)}) \end{array} \right) \right\} \quad \text{(B.1)}$$

■

上の 2. の場合を手続き Rewrite(k) と一対一に対応づけることができる．また，1. の場合より $t = 1 \ldots T$ について $\tilde{\psi}_{DB}(G_t) = \psi_{DB}^{(t,K_t)}(G_t)$ が成り立つことに注意する．図 B.1 は展開 t-極小支持集合を支持グラフ上で捉えたイメージである．式を簡単にするため，$\tau_{DB}^{(t)}$ の部分集合を定義する．

定義 25（部分テーブルアトム集合） $t = 1 \ldots T$ について $\tau_{DB}^{(t)} = \langle \tau_1^{(t)}, \tau_2^{(t)}, \ldots, \tau_{K_t}^{(t)} \rangle$ とおく．そのとき，$\tau_{DB}^{(t)}$ の部分順序集合 $\tau_{DB}^{(t)}(k) \stackrel{\text{def}}{=} \langle \tau_1^{(t)}, \tau_2^{(t)}, \ldots, \tau_k^{(t)} \rangle$ を定める． ■

次の展開 t-極小支持集合の性質は定義より明らかであるが，便利なので取り上げておく．

命題 9（展開 t-極小支持集合の性質） $t = 1 \ldots T, k = 1 \ldots K_t, k' = 0 \ldots k$ について $\tilde{S} \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)})$ なる $\tilde{S}$ は $\Sigma_{DB} \cup \{\tau_{k'+1}^{(t)}, \ldots, \tau_k^{(t)}\}$ の有限部分集合である．すなわち，$\tilde{S}$ には $\tau_1^{(t)}, \ldots, \tau_{k'}^{(t)}$ および $\tau_{k+1}^{(t)}, \ldots, \tau_{K_t}^{(t)}$ は含まれない． ■

$\psi_{DB}^{(t,0)}$ は同値式書き換え手続き Rewrite(k) を $k = K_t$ から $k = 0$ まで実行して得られた同値

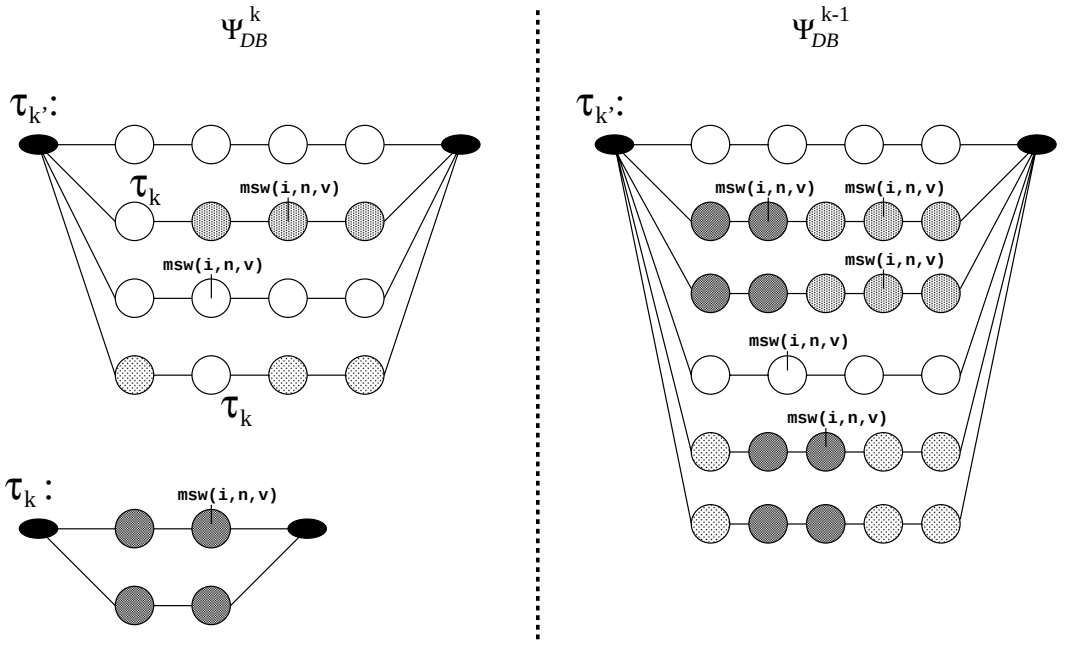

図 B.1: 展開 t-極小支持集合のイメージ．

式の連言に対応するので，次の命題が得られる．

命題 10（展開 t-極小支持集合と極小支持集合の一致） $\psi_{DB}^{(t,0)}(G_t) = \{S_1, S_2, \ldots, S_m\}$ とおいたとき $comp(R) \models G_t \leftrightarrow S_1 \vee S_2 \vee \cdots \vee S_m$ が成り立つ． ■

手続き Rewrite(k) と $\psi_{DB}^{(t,k)}$ の定義が対応づけられ，Rewrite(k) が独立支持条件を保存することから，テーブルアトム（観測ゴール $G_t = \tau_0^{(t)}$ を含む）は次のように計算できる．

命題 11（展開 t-極小支持集合に基づくテーブルアトムの確率計算） $t = 1 \ldots T$, $k = 1 \ldots K_t$, $k' = 0 \ldots k$, $\tilde{S} \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)})$ に対し，$\boldsymbol{\theta} \in \Theta$ について次の式が成り立つ：

$$P_F(\tilde{\boldsymbol{S}}=\mathbf{1}|\boldsymbol{\theta}) = \prod_{i \in I, v \in V_i} \theta_{i,v}^{\sigma_{i,v}(\tilde{S})} \prod_{\tau \in \tilde{S} \cap \{\tau_{k'+1}^{(t)}, \ldots, \tau_k^{(t)}\}} P_{DB}(\tau=1|\boldsymbol{\theta}). \tag{B.2}$$

特に $k = k'$ のとき，命題 9 より $\tilde{S} \in \psi_{DB}^{(t,k)}(\tau_k^{(t)})$ に対し，

$$P_F(\tilde{\boldsymbol{S}}=\mathbf{1}|\boldsymbol{\theta}) = \prod_{i \in I, v \in V_i} \theta_{i,v}^{\sigma_{i,v}(\tilde{S})} \tag{B.3}$$

が成り立つ ($t = 1 \ldots T, k = 1 \ldots K_t$)．そして $t = 1 \ldots T, k = 1 \ldots K_t, k' = 0 \ldots k$ および $\boldsymbol{\theta} \in \Theta$ について次の式が成り立つ：

$$P_{DB}(\tau_{k'}^{(t)} = 1|\boldsymbol{\theta}) = \sum_{\tilde{S} \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)})} P_F(\tilde{\boldsymbol{S}}=\mathbf{1}|\boldsymbol{\theta}). \tag{B.4}$$

■

B.2 手続き Learn-gEM の正当性

本節では図 4.7 の手続き Learn-gEM が Learn-PRISM-Naive と等価であることを示す．そのためには最終的な配列変数の値 $\mathcal{P}[t, \tau_k^{(t)}]$, $\eta[i, v]$ に関して，パラメータ $\boldsymbol{\theta}$ の下での

1. 観測ゴール確率の等価性: $\mathcal{P}[t, G_t] = P_{DB}(G_t = 1|\boldsymbol{\theta}) = \sum_{S \in \psi_{DB}(G_t)} P_F(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})$
2. 期待値計算の等価性: $\eta[i, v] = \eta(i, v|\boldsymbol{\theta}) = \sum_{S \in \psi_{DB}(G_t)} P_F(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(S)$

の 2 つを示せばよい．

B.2.1 観測ゴール確率の等価性

展開 t-極小支持集合に基づく観測ゴール G_t ($t = 1 \ldots T$) の確率計算に関して次が成り立つ．

命題 12（展開 t-極小支持集合における観測ゴールの確率の関係） 各 $t = 1 \ldots T$ について，展開 t-極小支持集合の集合 $\psi_{DB}^{(t,1)}$, $\psi_{DB}^{(t,2)}$, …, $\psi_{DB}^{(t,K_t)}$ を考えたとき，次が $k = 1 \ldots K_t$ について成り立つ：

$$\sum_{\tilde{S} \in \psi_{DB}^{(t,k)}(G_t)} P_{DB}(\tilde{\boldsymbol{S}}=\mathbf{1}|\boldsymbol{\theta}) = \sum_{\tilde{S} \in \psi_{DB}^{(t,k-1)}(G_t)} P_{DB}(\tilde{\boldsymbol{S}}=\mathbf{1}|\boldsymbol{\theta}). \tag{B.5}$$

■

（証明） 命題 11 の $k'=0$ の場合を考えると，$t=1\ldots T, k=0\ldots K_t, \tilde{S}' \in \psi_{DB}^{(t,k)}(G_t)$ について，

$$P_{DB}(\tilde{\boldsymbol{S}}'=\mathbf{1}|\boldsymbol{\theta}) = \prod_{i\in I, v\in V_i} \theta_{i,v}^{\sigma_{i,v}(\tilde{S})} \prod_{\tau\in\tilde{S}'\cap\tau_{DB}^{(t)}(k)} P_{DB}(\tau=1|\boldsymbol{\theta}).$$

が成り立つ．$\tilde{S}' \in \psi_{DB}^{(t,k)}(G)$ について $\tau_k \in \tilde{S}'$ と $\tau_k \notin \tilde{S}'$ それぞれの場合を考える．

- まず，$\tau_k \notin \tilde{S}'$ のとき，$\tau_{DB}^{(t)}(k)\cap\tilde{S}' = \tau_{DB}^{(t)}(k-1)\cap\tilde{S}'$ が成り立つ．

$$\prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}')} \prod_{\tau\in\tau_{DB}^{(t)}(k)\cap\tilde{S}'} P_{DB}(\tau=1|\boldsymbol{\theta}) = \prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}')} \prod_{\tau\in\tau_{DB}^{(t)}(k-1)\cap\tilde{S}'} P_{DB}(\tau=1|\boldsymbol{\theta}) \qquad \text{(B.6)}$$

  である．また，$\tau_k \notin \tilde{S}'$ と $\psi_{DB}^{(t,k-1)}$ の定義の仕方から $\tilde{S}' \in \psi_{DB}^{(t,k-1)}(G_t)$ が成り立つ．

- 一方 $\tau_k \in \tilde{S}'$ のとき，

$$\begin{aligned}
&\prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}')} \prod_{\tau\in\tau_{DB}^{(t)}(k)\cap\tilde{S}'} P_{DB}(\tau=1|\boldsymbol{\theta}) \\
&= \prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}')} P_{DB}(\tau_k^{(t)}=1|\boldsymbol{\theta}) \prod_{\tau\in\tau_{DB}^{(t)}(k-1)\cap\tilde{S}'} P_{DB}(\tau=1|\boldsymbol{\theta}) \qquad \left(\text{独立支持条件より}\right) \\
&= \left(\prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}')}\right)\left(\sum_{\tilde{S}''\in\psi_{DB}^{(t,k)}(\tau_k^{(t)})}\prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}'')} P_{DB}(\tau_k=1|\boldsymbol{\theta})\right)\cdot \\
&\qquad \left(\prod_{\tau\in\tau_{DB}^{(t)}(k-1)\cap\tilde{S}'} P_{DB}(\tau=1|\boldsymbol{\theta})\right) \qquad \left(\text{式 B.4, B.3 より}\right) \\
&= \sum_{\tilde{S}''\in\psi_{DB}^{(t,k)}(\tau_k^{(t)})}\left(\prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}')}\prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}'')}\prod_{\tau\in\tau_{DB}^{(t)}(k-1)\cap\tilde{S}'} P_{DB}(\tau=1|\boldsymbol{\theta})\right) \\
&= \sum_{\tilde{S}''\in\psi_{DB}^{(t,k)}(\tau_k^{(t)})}\left(\prod_{i,v}\theta_{i,v}^{\sigma_{i,v}(\tilde{S}'\cup\tilde{S}'')}\prod_{\tau\in\tau_{DB}^{(t)}(k-1)\cap\tilde{S}'} P_{DB}(\tau=1|\boldsymbol{\theta})\right) \qquad \text{(B.7)}
\end{aligned}$$

  が成り立つ．ここで $\tilde{S}=\tilde{S}'\cup\tilde{S}''\setminus\{\tau_k\}\cdots(*)$ とおくと $\psi_{DB}^{(t,k)}$ の定義の仕方より $\tilde{S}\in\psi_{DB}^{(t,k-1)}$ が成り立つ．また，定義から $\sigma_{i,v}(\tilde{S})=\sigma_{i,v}(\tilde{S}'\cup\tilde{S}'')$ が成り立つ．命題 9 の $k=k'$ の場合より $\tilde{S}''$ にテーブルアトムは含まれず，これと $(*)$ の $\tilde{S}$ の作り方から $\tau_{DB}^{(t)}(k-1)\cap\tilde{S}=\tau_{DB}^{(t)}(k-1)\cap\tilde{S}'$ が成り立つ．

ここで下を定める．定義 24 より $\psi_{DB}^{(t,k-1)}(G_t)=\Psi_0^{(t,k)}\cup\Psi_1^{(t,k)}$ が成り立つ．

$$\begin{aligned}
\Psi_0^{(t,k)} &= \left\{\tilde{S} \;\middle|\; \tilde{S}=\tilde{S}', \tilde{S}'\in\psi_{DB}^{(t,k)}(G_t), \tau_k^{(t)}\in\tilde{S}'\right\} \\
\Psi_1^{(t,k)} &= \left\{\tilde{S} \;\middle|\; \exists\tilde{S}', \tilde{S}''\left(\begin{array}{l}\tilde{S}=\tilde{S}'\cup\tilde{S}''\setminus\{\tau_k^{(t)}\},\\ \tilde{S}'\in\psi_{DB}^{(t,k)}(G_t), \tau_k^{(t)}\in\tilde{S}',\\ \tilde{S}''\in\psi_{DB}^{(t,k)}(\tau_k)\end{array}\right)\right\}
\end{aligned}$$

式 B.5 の左辺を次のように同値変形する．途中で式 B.6 および式 B.7 を使っている．

$$
\begin{aligned}
&\sum_{\tilde{S}' \in \psi_{DB}^{(t,k)}(G_t)} P_{DB}(\tilde{\boldsymbol{S}}' = \mathbf{1}|\boldsymbol{\theta}) \\
&= \sum_{\substack{\tilde{S}' :\ \tilde{S}' \in \psi_{DB}^{(t,k)}(G_t) \\ \tau_k^{(t)} \notin \tilde{S}'}} P_{DB}(\tilde{\boldsymbol{S}}' = \mathbf{1}|\boldsymbol{\theta}) + \sum_{\substack{\tilde{S}' :\ \tilde{S}' \in \psi_{DB}^{(t,k)}(G_t) \\ \tau_k^{(t)} \in \tilde{S}'}} P_{DB}(\tilde{\boldsymbol{S}}' = \mathbf{1}|\boldsymbol{\theta}) \\
&= \sum_{\tilde{S} \in \Psi_0^{(t,k)}} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) \\
&\qquad + \sum_{\substack{\tilde{S}' :\ \tilde{S}' \in \psi_{DB}^{(t,k)}(G_t) \\ \tau_k^{(t)} \in \tilde{S}'}} \sum_{S'' \in \psi_{DB}^{(t,k)}(\tau_k^{(t)})} \prod_{i,v} \boldsymbol{\theta}_{i,v}^{\sigma_{i,v}(\tilde{S}' \cup \tilde{S}'')} \prod_{\tau \in \tau_{DB}^{(t)}(k-1) \cap \tilde{S}'} P_{DB}(\tau = 1|\boldsymbol{\theta}) \\
&= \sum_{\tilde{S} \in \Psi_0^{(t,k)}} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) + \sum_{\tilde{S} \in \Psi_1^{(t,k)}} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) \\
&\qquad \left(\begin{array}{l} \text{第 2 項の変形において } \sigma_{i,v}(\tilde{S}' \cup \tilde{S}'' \setminus \{\tau_k^{(t)}\}) = \sigma_{i,v}(\tilde{S}' \cup \tilde{S}'') \text{ および} \\ \tau_{DB}^{(t)}(k-1) \cap \left(\tilde{S}' \cup \tilde{S}'' \setminus \{\tau_k^{(t)}\}\right) = \tau_{DB}^{(t)}(k-1) \cap \tilde{S}' \text{ が成り立つ点に注意} \end{array} \right) \\
&= \sum_{\tilde{S} \in \Psi_0^{(t,k)} \cup \Psi_1^{(t,k)}} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) \\
&= \sum_{\tilde{S} \in \psi_{DB}^{(t,k-1)}(G_t)} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) \qquad \left(\text{定義 24 より}\right)
\end{aligned}
$$

以上より命題は成り立つ． □

命題 12 をテーブルアトムの確率計算に一般化することができる．証明は命題 12 と同様である．

命題 13（展開 t-極小支持集合におけるテーブルアトムの確率の関係） 各 $t = 1 \ldots T$ について，展開 t-極小支持集合の集合 $\psi_{DB}^{(t,1)}, \psi_{DB}^{(t,2)}, \ldots, \psi_{DB}^{(t,K_t)}$ を考えたとき，次が $k = 1 \ldots K_t$, $k' = 0 \ldots k-1$ について成り立つ：

$$
\sum_{\tilde{S} \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)})} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) = \sum_{\tilde{S} \in \psi_{DB}^{(t,k-1)}(\tau_{k'}^{(t)})} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}). \tag{B.8}
$$

■

次の補題によって Learn-gEM と Learn-PRISM-Naive における観測ゴール確率の等価性が保証される．

補題 3（t-極小支持集合に基づくゴール確率と極小支持集合に基づくゴール確率の関係） 各 $t = 1 \ldots T$ について任意の $\boldsymbol{\theta} \in \Theta$ の下で

$$
\sum_{\tilde{S} \in \tilde{\psi}_{DB}(G_t)} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) = \sum_{S \in \psi_{DB}(G_t)} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) \tag{B.9}
$$

が成り立つ． ■

（証明） $\tilde{\psi}_{DB}(G_t) = \psi_{DB}^{(t,K_t)}(G_t)$ かつ $\psi_{DB}(G_t) = \psi_{DB}^{(t,0)}(G_t)$ であることと，命題 12 より成り立つ． □

B.2.2 期待値計算の等価性

手続き GET-EXPECTATIONS における配列変数 $\mathcal{P}, \mathcal{Q}, \mathcal{R}, \eta$ の値の変化を観察すると次の命題が成り立つことが分かる．

命題 14（手続き GET-EXPECTATIONS の計算結果） 手続き GET-EXPECTATIONS の終了時には $t = 1 \ldots T$, $k = 0 \ldots K_t$ および $i \in I$, $v \in V_i$ について以下が成り立つ．$G_t = \tau_0^{(t)}$ である点に注意する．

$$\eta[i,v] = \sum_{t=1}^{T} \frac{1}{\mathcal{P}[t,G_t]} \sum_{k=0}^{K_t} \mathcal{Q}[t,\tau_k^{(t)}] \sum_{\tilde{S} \in \tilde{\psi}_{DB}(\tau_k^{(t)})} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) \sigma_{i,v}(\tilde{S}) \tag{B.10}$$

$$\mathcal{Q}[t,\tau_k^{(t)}] = \begin{cases} \displaystyle\sum_{k'=0}^{k-1} \mathcal{Q}[t,\tau_{k'}^{(t)}] \sum_{\tilde{S} \in \tilde{\psi}_{DB}(\tau_{k'}^{(t)}),\, \tau_k^{(t)} \in \tilde{S}} P_{DB}(\tilde{\boldsymbol{S}}_{\setminus \tau_k^{(t)}} = \mathbf{1}|\boldsymbol{\theta}) & \text{if } k \geq 1 \\ 1 & \text{if } k = 0 \end{cases} \tag{B.11}$$

ただし，$\tilde{S}$ に対して $\tilde{\boldsymbol{S}}_{\setminus \tau_k^{(t)}}$ は $\tilde{S} \setminus \{\tau_k^{(t)}\}$ の要素を並べた確率ベクトルである．また，$\tau_0^{(t)} = G_t$ であることに注意する． ■

$\mathcal{P}[t,G_t]$ には $P_{DB}(G_t = 1|\boldsymbol{\theta})$ が格納されていることは補題 3 が保証している．次に，後の証明に用いる 2 つの簡単化を示す．後者は図 B.1 より直観的に分かる．

命題 15（簡単化 1） $t = 1 \ldots T, k = 1 \ldots K_t$ なる t, k を考える．そのとき任意の $k'' = (k+1) \ldots K_t$, $k' = 0 \ldots k-1$ について次が成り立つ：

$$\sum_{\tilde{S}:\ \tilde{S} \in \psi_{DB}^{(t,k'')}(\tau_{k'}^{(t)}),\ \tau_k^{(t)} \in \tilde{S}} P_{DB}(\tilde{\boldsymbol{S}}_{\setminus \tau_k^{(t)}} = \mathbf{1}|\boldsymbol{\theta}) = \sum_{\tilde{S}:\ \tilde{S} \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)}),\ \tau_k^{(t)} \in \tilde{S}} P_{DB}(\tilde{\boldsymbol{S}}_{\setminus \tau_k^{(t)}} = \mathbf{1}|\boldsymbol{\theta}). \tag{B.12}$$

■

（証明） 命題 13 の特別な形として $j = (k+1) \ldots K_t$ について

$$\sum_{\tilde{S}:\ \tilde{S} \in \psi_{DB}^{(t,j)}(\tau_{k'}^{(t)}),\ \tau_k^{(t)} \in \tilde{S}} P_{DB}(\tilde{\boldsymbol{S}}_{\setminus \tau_k^{(t)}} = \mathbf{1}|\boldsymbol{\theta}) = \sum_{\tilde{S}:\ \tilde{S} \in \psi_{DB}^{(t,j-1)}(\tau_{k'}^{(t)}),\ \tau_k^{(t)} \in \tilde{S}} P_{DB}(\tilde{\boldsymbol{S}}_{\setminus \tau_k^{(t)}} = \mathbf{1}|\boldsymbol{\theta}).$$

が成り立つ．これを $j = k''$ から $j = k+1$ まで適用すれば式 B.12 は成り立つ． □

命題 16（簡単化 2） $t = 1 \ldots T$, $1 \leq k' < k \leq K_t$ について以下が成り立つ．

$$\begin{aligned} \sum_{\tilde{S} \in \psi_{DB}^{(t,k-1)}(\tau_{k'}^{(t)})} P_{DB}(\boldsymbol{S} = \mathbf{1}|\boldsymbol{\theta}) \sigma_{i,v}(\tilde{S}) &= \sum_{\tilde{S}' \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)})} P_{DB}(\boldsymbol{S}' = \mathbf{1}|\boldsymbol{\theta}) \sigma_{i,v}(\tilde{S}') \\ &\quad + \sum_{\tilde{S}':\ \tilde{S}' \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)}), \tau_k^{(t)} \in \tilde{S}'} P_{DB}(\tilde{\boldsymbol{S}}'_{\setminus \tau_k^{(t)}} = \mathbf{1}|\boldsymbol{\theta}) \cdot \\ &\qquad \sum_{\tilde{S}'':\ \tilde{S}'' \in \psi_{DB}^{(t,k)}(\tau_k^{(t)})} P_{DB}(\tilde{\boldsymbol{S}}'' = \mathbf{1}|\boldsymbol{\theta}) \sigma_{i,v}(\tilde{S}'') \end{aligned} \tag{B.13}$$

■

（証明） 式を簡単にするため，ある $t = 1 \ldots T$ に固定し，添字 $\cdot^{(t)}$ や $\cdot_t$ は省略する．また，$\psi_{DB}^{(t,k)}$ を単に ψ^k と書く．ψ^k の定義より，左辺は次のように変形される．

$$\sum_{\tilde{S} \in \psi^{k-1}(\tau_{k'})} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1}|\boldsymbol{\theta}) \sigma_{i,v}(\tilde{S}) = \sum_{\tilde{S}':\ \tilde{S}' \in \psi^k(\tau_{k'}),\ \tau_k \in \tilde{S}'} P_{DB}(\tilde{\boldsymbol{S}}' = \mathbf{1}|\boldsymbol{\theta}) \sigma_{i,v}(\tilde{S}')$$

$$+ \sum_{\substack{\tilde{S}:\ \tilde{S}=\tilde{S}'\cup\tilde{S}''\setminus\{\tau_k\}\\ \tilde{S}'\in\psi^k(\tau_{k'}),\ \tau_k\notin\tilde{S}'\\ \tilde{S}''\in\psi^k(\tau_k)}} P_{DB}(\tilde{\boldsymbol{S}}=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}) \qquad \text{(B.14)}$$

この式の右辺第 2 項を次のように変形する．$\tilde{S}$ を $\tilde{S}'\setminus\{\tau_k\}$ と $\tilde{S}''$ の 2 つに分けて考える．

$$\begin{aligned}
\text{(第 2 項)} &= \sum_{\substack{(\tilde{S}',\tilde{S}''):\ \tilde{S}'\in\psi^k(\tau_{k'}),\ \tau_k\notin\tilde{S}'\\ \tilde{S}''\in\psi^k(\tau_k)}} P_{DB}(\tilde{\boldsymbol{S}}'_{\setminus\tau_k}=\mathbf{1}|\boldsymbol{\theta})P_{DB}(\tilde{\boldsymbol{S}}''=\mathbf{1}|\boldsymbol{\theta})\underbrace{\sigma_{i,v}(\tilde{S}'\cup\tilde{S}''\setminus\{\tau_k\})}_{=\sigma_{i,v}(\tilde{S}')+\sigma_{i,v}(\tilde{S}'')}\\
&= \sum_{\substack{(\tilde{S}',\tilde{S}''):\ \tilde{S}'\in\psi^k(\tau_{k'}),\ \tau_k\notin\tilde{S}'\\ \tilde{S}''\in\psi^k(\tau_k)}} P_{DB}(\tilde{\boldsymbol{S}}'_{\setminus\tau_k}=\mathbf{1}|\boldsymbol{\theta})P_{DB}(\tilde{\boldsymbol{S}}''=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}')\\
&\quad + \sum_{\substack{(\tilde{S}',\tilde{S}''):\ \tilde{S}'\in\psi^k(\tau_{k'}),\ \tau_k\notin\tilde{S}'\\ \tilde{S}''\in\psi^k(\tau_k)}} P_{DB}(\tilde{\boldsymbol{S}}'_{\setminus\tau_k}=\mathbf{1}|\boldsymbol{\theta})P_{DB}(\tilde{\boldsymbol{S}}''=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}'')\\
&= \sum_{\tilde{S}':\ \tilde{S}'\in\psi^k(\tau_{k'}),\ \tau_k\notin\tilde{S}'} P_{DB}(\tilde{\boldsymbol{S}}'_{\setminus\tau_k}=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}')\overbrace{\sum_{\tilde{S}''\in\psi^k(\tau_k)} P_{DB}(\tilde{\boldsymbol{S}}''=\mathbf{1}|\boldsymbol{\theta})}^{P_{DB}(\tau_k=1|\boldsymbol{\theta})}\\
&\quad + \sum_{\tilde{S}':\ S'\in\psi^k(\tau_{k'}),\ \tau_k\notin\tilde{S}'} P_{DB}(\tilde{\boldsymbol{S}}'_{\setminus\tau_k}=\mathbf{1}|\boldsymbol{\theta})\sum_{\tilde{S}''\in\psi^k(\tau_k)} P_{DB}(\tilde{\boldsymbol{S}}''=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}'')\\
&= \sum_{\tilde{S}':\ \tilde{S}'\in\psi^k(\tau_{k'}),\ \tau_k\notin\tilde{S}'} P_{DB}(\tilde{\boldsymbol{S}}'=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}')\\
&\quad + \sum_{\tilde{S}':\ \tilde{S}'\in\psi^k(\tau_{k'}),\ \tau_k\notin\tilde{S}'} P_{DB}(\tilde{\boldsymbol{S}}'_{\setminus\tau_k}=\mathbf{1}|\boldsymbol{\theta})\sum_{\tilde{S}''\in\psi^k(\tau_k)} P_{DB}(\tilde{\boldsymbol{S}}''=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}'')
\end{aligned} \qquad \text{(B.15)}$$

式 B.14 の右辺第 1 項と式 B.15 の右辺第 1 項の和は $\sum_{\tilde{S}'\in\psi^k(\tau_{k'})} P_{DB}(\tilde{\boldsymbol{S}}'=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}')$ に等しい．従って式 B.13 は成り立つ． □

次の補題によって Learn-gEM と Learn-PRISM-Naive における期待値計算の等価性が保証される．

補題 4（期待値計算の等価性） $i\in I,\ v\in V_i$ について

$$\eta[i,v]=\sum_{t=1}^{T}\frac{1}{P_{DB}(G_t=1|\boldsymbol{\theta})}\sum_{S\in\psi_{DB}(G_t)} P_{DB}(\boldsymbol{S}=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(S) \qquad \text{(B.16)}$$

が成り立つ． ■

（証明） はじめに，式 B.10 の右辺の一部を

$$\eta[t,i,v]\stackrel{\text{def}}{=}\sum_{k=0}^{K_t}\mathcal{Q}[t,\tau_k^{(t)}]\sum_{\tilde{S}\in\tilde{\psi}_{DB}(\tau_k^{(t)})} P_{DB}(\tilde{\boldsymbol{S}}=\mathbf{1}|\boldsymbol{\theta})\sigma_{i,v}(\tilde{S}) \qquad \text{(B.17)}$$

とおく．ここで式を簡単にする為に，任意の $t=1\ldots T,\ i\in I,\ v\in V_i$ を固定する．手続き Get-Expectations における $\mathcal{Q}$ の計算結果である式 B.11 を式 B.12 により簡単化する．すると，

$$\mathcal{Q}[t,\tau_k^{(t)}]=\sum_{k'=0}^{k-1}\mathcal{Q}[t,\tau_{k'}^{(t)}]\sum_{\tilde{S}\in\psi_{DB}^{(t,k)}(\tau_{k'}^{(t)}),\tau_k\in\tilde{S}} P_{DB}(\tilde{\boldsymbol{S}}_{\setminus\tau_k}=\mathbf{1}|\boldsymbol{\theta}) \qquad \text{(B.18)}$$

が得られる．ここで式を簡単にするために

$$Q_k \stackrel{\text{def}}{=} \mathcal{Q}[t, \tau_k^{(t)}] \quad \text{for each } k = 0 \ldots K_t \tag{B.19}$$

$$P_{k'}^k \stackrel{\text{def}}{=} \sum_{\tilde{S}:\, \tilde{S} \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)}), \tau_k^{(t)} \in \tilde{S}} P_{DB}(\tilde{\boldsymbol{S}}_{\setminus \tau_k} = \mathbf{1} | \boldsymbol{\theta}) \quad \text{for } 0 \leq k' \leq k \leq K_t \tag{B.20}$$

$$E_{k'}^k \stackrel{\text{def}}{=} \sum_{\tilde{S} \in \psi_{DB}^{(t,k)}(\tau_{k'}^{(t)})} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1} | \boldsymbol{\theta}) \sigma_{i,v}(\tilde{S}) \quad \text{for } 0 \leq k' \leq k \leq K_t \tag{B.21}$$

とおく．t, i, v を固定している点に注意する．上の記号を使うと上で示した式 B.17, B.18, B.13 はそれぞれ下の式に置き換えられる．

$$\eta[t, i, v] = \sum_{k=0}^{K_t} Q_k E_k^{K_t} \tag{B.22}$$

$$Q_k = \sum_{k'=0}^{k-1} P_{k'}^k Q_{k'} \quad \text{for } k = 1 \ldots K_t \tag{B.23}$$

$$E_{k'}^{k-1} = E_{k'}^k + P_{k'}^k E_k^k \quad \text{for } 0 \leq k' < k \leq K_t \tag{B.24}$$

ここで $U^k \stackrel{\text{def}}{=} \sum_{k'=0}^k Q_{k'} E_{k'}^k$ とおく．すると以下の変形が $k = 0 \ldots K_t$ について成り立つ．

$$\begin{aligned}
U^k &= \sum_{k'=0}^{k} Q_{k'} E_{k'}^k \\
&= \sum_{k'=0}^{k-1} Q_{k'} E_{k'}^k + Q_k E_k^k \\
&= \sum_{k'=0}^{k-1} Q_{k'} E_{k'}^k + \left(\sum_{k'=0}^{k-1} P_{k'}^k Q_{k'} \right) E_k^k \qquad \left(\text{式 B.23 より}\right) \\
&= \sum_{k'=0}^{k-1} E_{k'}^k Q_{k'} + \sum_{k'=0}^{k-1} P_{k'}^k E_k^k Q_{k'} \\
&= \sum_{k'=0}^{k-1} \left(E_{k'}^k + P_{k'}^k E_k^k \right) Q_{k'} \\
&= \sum_{k'=0}^{k-1} E_{k'}^{k-1} Q_{k'} \qquad \left(\text{式 B.24 より}\right) \\
&= U^{k-1}
\end{aligned}$$

すなわち $k = 0 \ldots K_t$ について $U^k = U^{k-1}$ が成り立つ．$\mathcal{Q}$ の計算結果（式 B.11）と命題 10 よりそれぞれ

$$\begin{aligned}
Q_0 &= \mathcal{Q}[t, \tau_0^{(t)}] = \mathcal{Q}[t, G_t] = 1 \\
E_0^0 &= \sum_{\tilde{S} \in \psi_{DB}^{(t,0)}(\tau_0^{(t)})} P_{DB}(\tilde{\boldsymbol{S}} = \mathbf{1} | \boldsymbol{\theta}) \sigma_{i,v}(\tilde{S}) = \sum_{S \in \psi_{DB}(G_t)} P_{DB}(\boldsymbol{S} = \mathbf{1} | \boldsymbol{\theta}) \sigma_{i,v}(S)
\end{aligned}$$

が成り立ち，従って $U^0 = \sum_{S \in \psi_{DB}(G_t)} P_{DB}(\boldsymbol{S} = \mathbf{1} | \boldsymbol{\theta}) \sigma_{i,v}(S)$ である．一方，式 B.22 より $U^{K_t} = \eta[t, i, v]$ が成り立つ．よって，

$$\eta[t, i, v] = \sum_{S \in \psi_{DB}(G_t)} P_{DB}(\boldsymbol{S} = \mathbf{1} | \boldsymbol{\theta}) \sigma_{i,v}(S) \tag{B.25}$$

が成り立つ．一方，式 B.10 と $\mathcal{P}[t, G_t]$ に $P_{DB}(G_t = 1|\boldsymbol{\theta})$ が格納されていること（補題 3 より）から

$$\eta[i, v] = \sum_{t=1}^{T} \frac{1}{P_{DB}(G_t = 1|\boldsymbol{\theta})} \sum_{S \in \psi_{DB}(G_t)} P_{DB}(\boldsymbol{S} = \mathbf{1}|\boldsymbol{\theta}) \sigma_{i,v}(S) \tag{B.26}$$

が成り立つ．よって補題は示された． □

補題 2 と定理 2 を再び掲げ，それぞれ証明する．以上の準備の下では自明である．

補題 2（手続き LEARN-PRISM-NAIVE とグラフィカル EM アルゴリズムの等価性）有限支持，排反性，唯一性条件，非循環支持，独立支持条件を満たす PRISM プログラム DB と観測データ $\mathcal{G}$ を考える．そして，2 つの手続き LEARN-PRISM-NAIVE と LEARN-GEM に同じ初期値 $\boldsymbol{\theta}^{(0)}$ を与え，収束基準を同じにすると，両者が返す収束値 $\boldsymbol{\theta}^*$ は一致する． ■

（証明） 手続き LEARN-GEM, LEARN-PRISM-NAIVE の内容と補題 3, 補題 4 より明らか． □

定理 2（グラフィカル EM アルゴリズムの正当性）有限支持，排反性，唯一性条件，非循環支持，独立支持条件を満たす PRISM プログラム DB と観測データ $\mathcal{G}$ を考える．そのとき，手続き LEARN-GEM は尤度 $\lambda_{DB}(\mathcal{G}|\boldsymbol{\theta})$ を局所的に最大にする $\boldsymbol{\theta} \in \Theta$ を出力する． ■

（証明） 定理 1 と補題 2 より明らか． □