疑似ベイジアンネットを用いた認知モデルのプロトタイピング手法の提案

汎用人工知能研究会

2016-12-15

産業技術総合研究所

人工知能研究センター

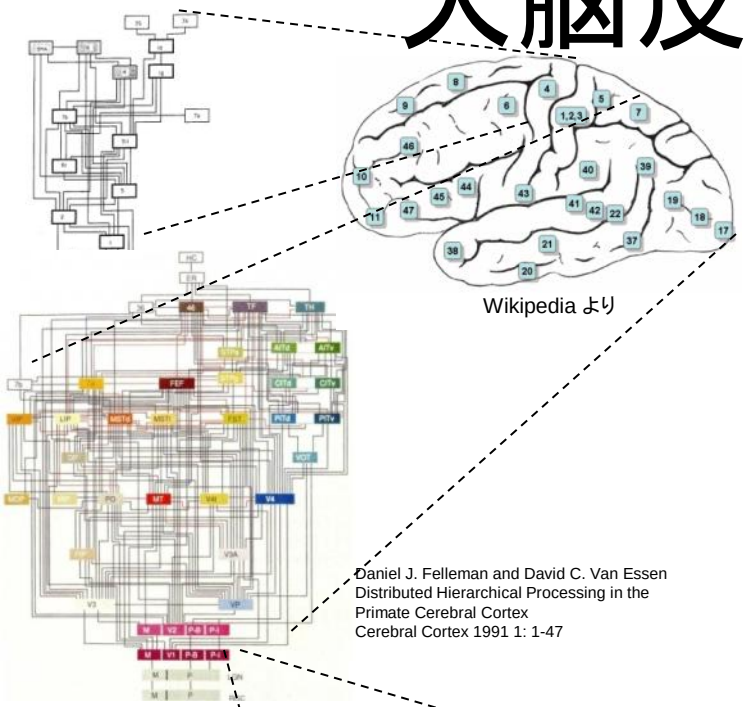
一杉裕志

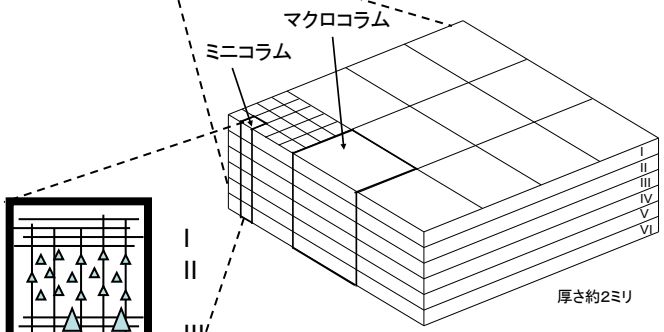
概要

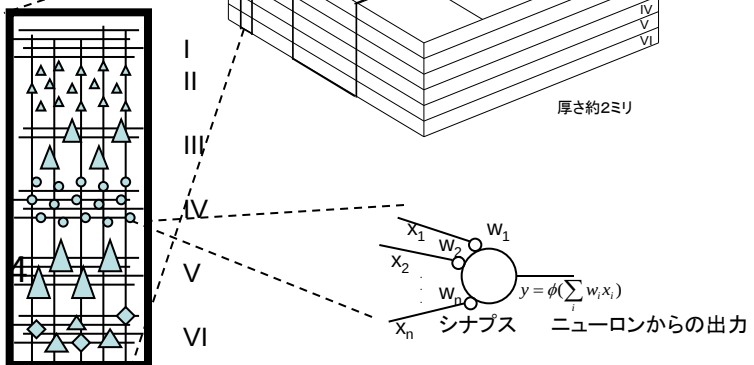
- 大脳皮質ベイジアンネット仮説にもとづく認知モデル実現の意義
- 確率値の0と非0の区別のみを行う「疑似ベイジアンネット」 **QBN** (Quasi Bayesian Networks)
- QBN の条件付確率表に制限を入れた **QBC** (Quasi Bayesian Cognitive circuits)
- QBC を用いた認知モデルのプロトタイピングの事例の紹介

研究の背景

大脳皮質の不思議さ







- 脳の様々な高次機能（認識、意思決定、運動制御、思考、推論、言語理解など）が、**たった50個程度**の領野のネットワークで実現されている。
- **おそらくすべての領野は共通の原理で動いている。**

有望な仮説の1つ：
大脳皮質ベイジアンネットモデル
[Lee and Mumford 2003]

認知モデルと認知アーキテクチャ

- ここでは下記のように定義
  - **認知モデル**: 人間や知能を持った動物の認識、学習、言語理解、運動制御、行動計画などの個別の機能の実現メカニズムに関する仮説を単純化して表現したもの
  - **認知アーキテクチャ**: 部品である個々の認知モデルを1つに結合し、知的に動作するエージェントにするための設計図
- 脳を模倣して認知モデルを結合すれば将来は汎用人工知能に

認知モデル記述の道具

- 計算のモデル
  - チューリングマシン、ラムダ計算、・・・
- ニューラルネットワークモデル
  - パターン認識のモデル：パーセプトロン、Neocognitron
  - 連想記憶のモデル：ホップフィールドネットワーク
  - 時系列学習のモデル：エルマンネット
- 形式論理
  - 述語論理、様相論理
  - 論理型言語 prolog、確率プログラミング言語
- プロダクションシステム
  - ACT-R 、・・・

ベイジアンネットを用いて複雑な認知モデルを実装したい

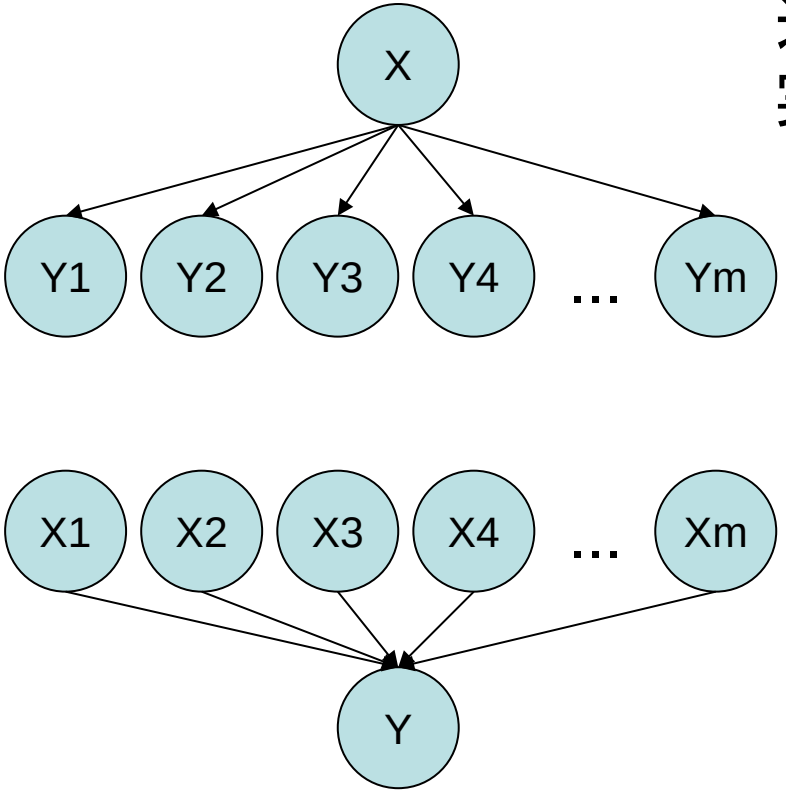
- 大規模機械学習特有の様々な困難:
  - ハイパパラメタの調整
  - 局所解・過適合
  - オーバーフロー・アンダーフロー・ NaN
  - 初期値依存
  - 計算コスト
- モデルが意図したとおりに動かないときに、真の原因の究明に非常に手間がかかる。
- この問題をなんとかしたい。→ プロトタイピング

プロトタイピングの直近の目的

- 制限付きベイジアンネット BESOM で新たに設計した条件付確率表モデルの表現力を確認する。
- 視覚野・言語野に関係する認知モデルの実現可能性をある程度検証する。
  - 認知科学的妥当性、神経科学的妥当性、工学的有用性

「制限付きベイジアンネット」と条件付確率表のモデル

ベイジアンネットの親ノード数の問題



近似推論・学習1ステップの実行に必要な計算量

$O(m)$

複雑な認知モデルを作ろうとするとmは大きくなる。

$O(2^m)$

多くの近似推論・学習アルゴリズムにおいて
親ノードの数に対し
指数関数的な計算量が必要

推論・学習アルゴリズムの例

確率伝搬アルゴリズム
[Pearl 1988]

$$BEL(x) = \alpha\lambda(x)\pi(x)$$

$$\pi(x) = \sum_{u_1,\cdots,u_m} P(x|u_1,\cdots,u_m)\prod_k \pi_X(u_k)$$

$$\lambda(x) = \prod_l \lambda_{Y_l}(x)$$

$$\pi_{Y_l}(x) = \pi(x)\prod_{j\neq l} \lambda_{Y_j}(x)$$

$$\lambda_X(u_k) = \sum_x \lambda(x) \sum_{u_1,\cdots,u_m/u_k} P(x|u_1,\cdots,u_m)\prod_{i\neq k} \pi_X(u_i)$$

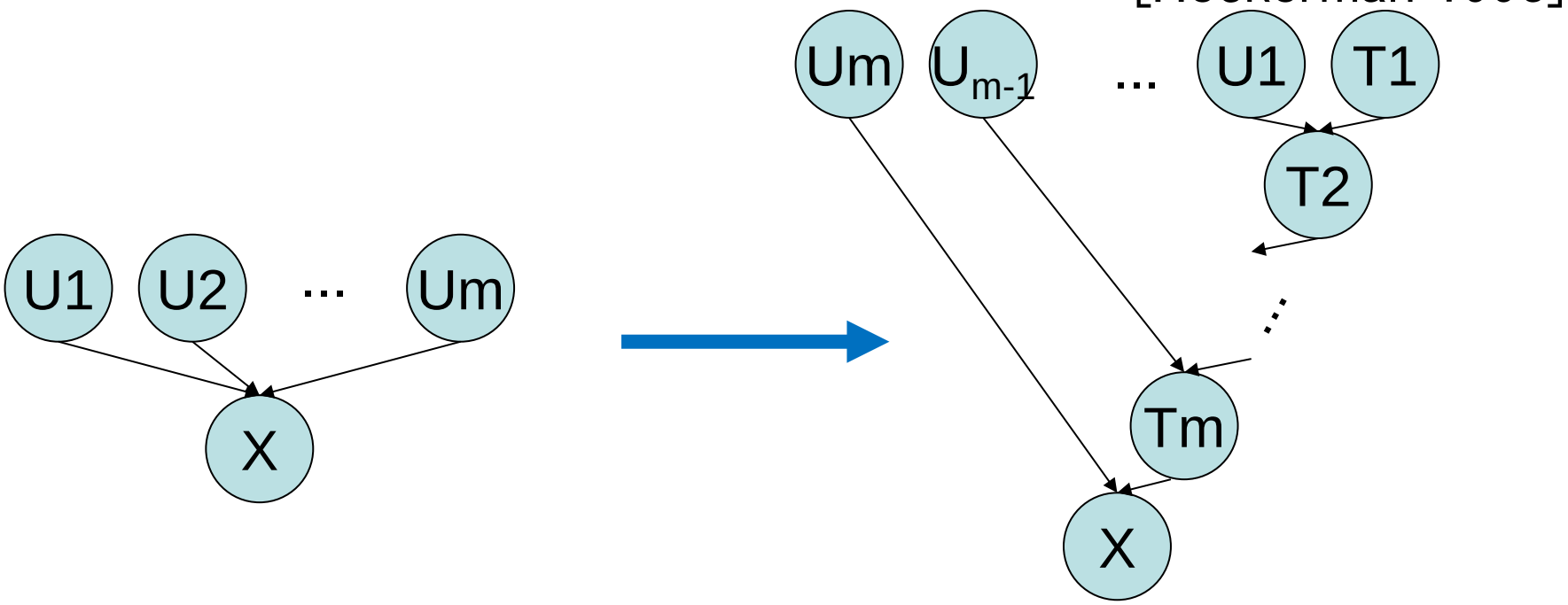
隠れ変数があるときの勾配法
[Binder et al. 1997]

$$\Delta w = \Sigma_{x,pa(x)} \frac{P_\theta(x,pa(x)|e)}{P_\theta(x|pa(x))} \frac{\partial P_\theta(x|pa(x))}{\partial w}$$

どちらも親ノードの数 m に対して
1ステップあたりの計算量のオーダーが O(2^m)

解決策：定数個の親ノードを持つ等価なネットワークに変換

[Heckerman 1993]



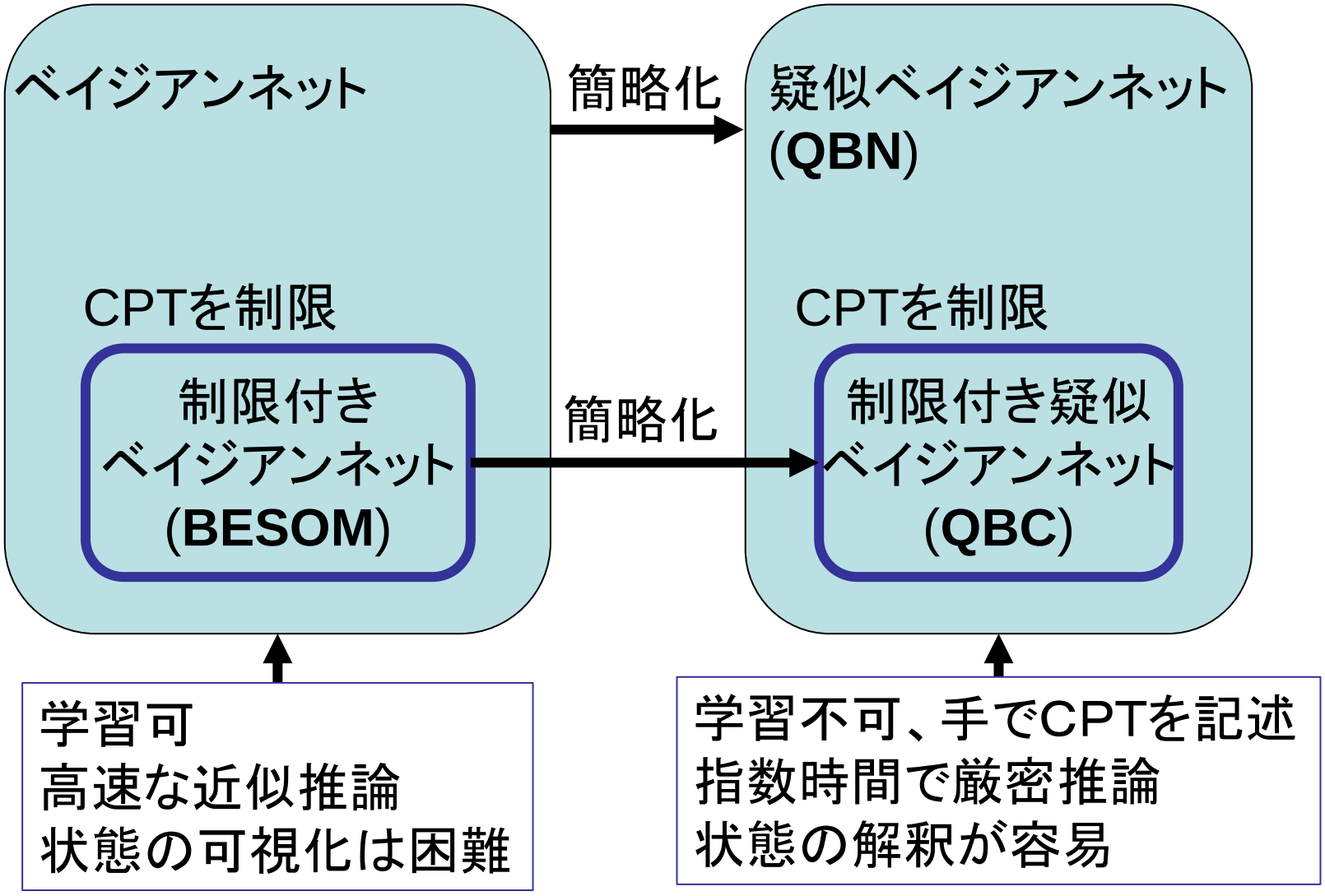
親ノードの数 m 個

親ノードの数は高々2個で深さが O(m)

このような変換が可能な条件付確率表モデルであれば、
推論・学習の1ステップの実行が O(2^m) から **O(m) に高速化**

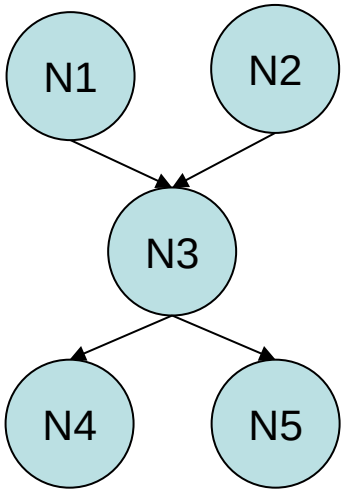
- 親ノードが m 個あるベイジアンネットを O(m) で動作させるためには条件付確率表に制限を入れる必要がある。
- **表現力の高さと効率の良さの両立ができるかどうかが問題。**
- 現在、3種類のノードの導入を検討中。
- 「疑似ベイジアンネット」を使った視覚野・言語野の認知モデルのプロトタイピングで表現力を確認中。

QBN と QBC


ベイジアンネット
簡略化
疑似ベイジアンネット
(QBN)
CPTを制限
制限付き
ベイジアンネット
(BESOM)
簡略化
CPTを制限
制限付き疑似
ベイジアンネット
(QBC)
学習可
高速な近似推論
状態の可視化は困難
学習不可、手でCPTを記述
指数時間で厳密推論
状態の解釈が容易

疑似ベイジアンネットQBNの定義例

P(N1,N2,N3,N4,N5)
=P(N4|N3)P(N5|N3)P(N3|N1,N2)P(N1)P(N2)



N1	P(N1)
False	non-zero
True	non-zero

N2	P(N2)
False	non-zero
True	non-zero

N3	N1	N2	P(N3|N1,N2)
False	False	False	non-zero
True	True	False	non-zero
True	False	True	non-zero
True	True	True	non-zero
これら以外			zero

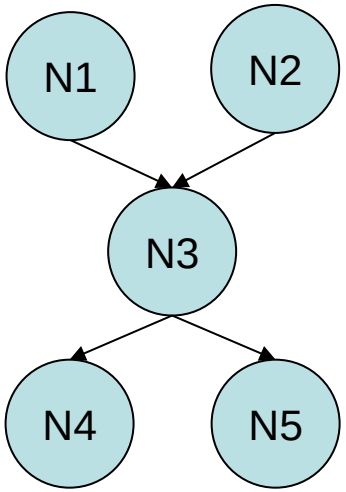
N4	N3	P(N4|N3)
False	False	non-zero
True	True	non-zero
これら以外		zero

N5	N3	P(N5|N3)
False	False	non-zero
True	True	non-zero
これら以外		zero

・変数は多値でも構わない。
・変数の値に任意の記号を用いることができる。

疑似ベイジアンネットの解

$$P(N1,N2,N3,N4,N5) = P(N4|N3)P(N5|N3)P(N3|N1,N2)P(N1)P(N2)$$



N1	P(N1)
False	non-zero
True	non-zero

N2	P(N2)
False	non-zero
True	non-zero

N3	N1	N2	P(N3|N1,N2)
False	False	False	non-zero
True	True	False	non-zero
True	False	True	non-zero
True	True	True	non-zero
これら以外			zero

N4	N3	P(N4|N3)
False	False	non-zero
True	True	non-zero
これら以外		zero

N5	N3	P(N5|N3)
False	False	non-zero
True	True	non-zero
これら以外		zero

すべての解（同時確率が非0になる値の組）

result 1
N5 : False
N4 : False
N3 : False
N2 : False
N1 : False

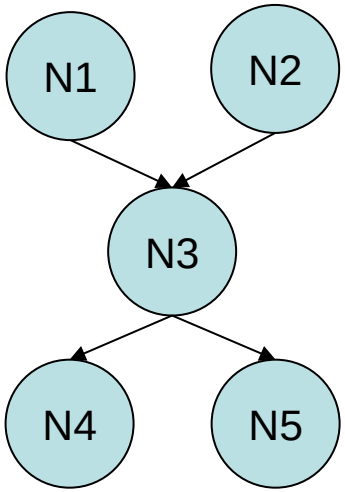
result 2
N5 : True
N4 : True
N3 : True
N2 : True
N1 : False

result 3
N5 : True
N4 : True
N3 : True
N2 : False
N1 : True

result 4
N5 : True
N4 : True
N3 : True
N2 : True
N1 : True

解が満たすべき条件

P(N1,N2,N3,N4,N5)
=P(N4|N3)P(N5|N3)P(N3|N1,N2)P(N1)P(N2)



N1	P(N1)
False	non-zero
True	non-zero

N2	P(N2)
False	non-zero
True	non-zero

同時確率が非0

N3	N1	N2	P(N3|N1,N2)
False	False	False	non-zero
True	True	False	non-zero
True	False	True	non-zero
True	True	True	non-zero
これら以外			zero

すべての条件付確率が非0

N4	N3	P(N4|N3)
False	False	non-zero
True	True	non-zero
これら以外		zero

N5	N3	P(N5|N3)
False	False	non-zero
True	True	non-zero
これら以外		zero

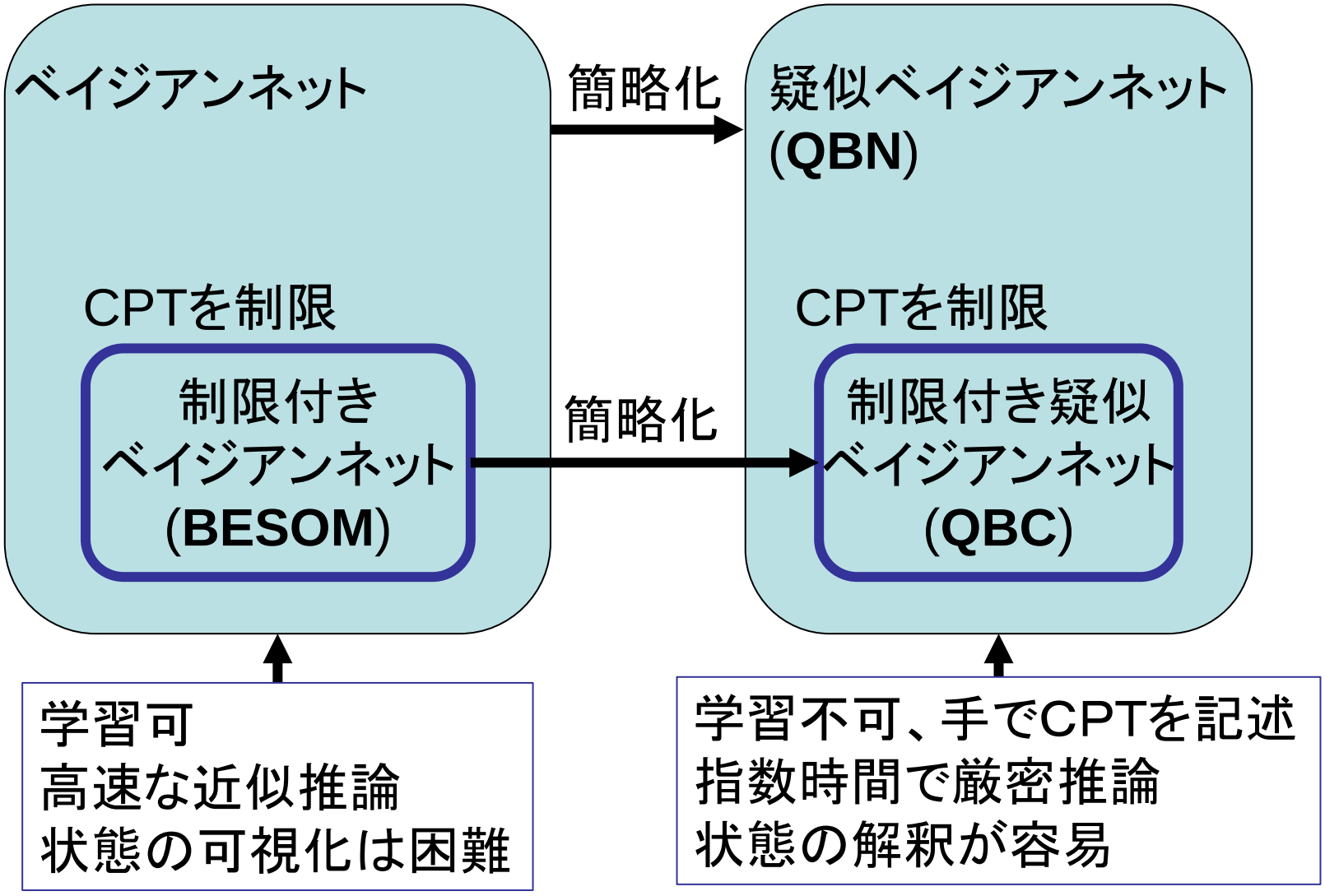
result 1
N5 : False
N4 : False
N3 : False
N2 : False
N1 : False

result 2
N5 : True
N4 : True
N3 : True
N2 : True
N1 : False

result 3
N5 : True
N4 : True
N3 : True
N2 : False
N1 : True

result 4
N5 : True
N4 : True
N3 : True
N2 : True
N1 : True

QBN と QBC


ベイジアンネット
簡略化
疑似ベイジアンネット
(QBN)
CPTを制限
制限付き
ベイジアンネット
(BESOM)
簡略化
CPTを制限
制限付き疑似
ベイジアンネット
(QBC)
学習可
高速な近似推論
状態の可視化は困難
学習不可、手でCPTを記述
指数時間で厳密推論
状態の解釈が容易

QBCとは

- QBN の条件付確率表を制限したものがQBC
- 使用する3種類のノード:
  - ORノード
  - ゲートノード
  - 排他ノード

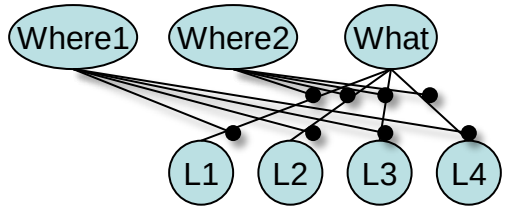
$$P(X=0|u_1,\dots,u_m)=\Pi_{k=1}^{m}(1-w_k)^{u_k}$$
$$P(X=1|u_1,\dots,u_m)=1-\Pi_{k=1}^{m}(1-w_k)^{u_k}$$

$$P(X=0|c_1,\dots,c_m,u)=1-(1-w_u)^{\neg u}\Pi_{k=1}^{m}(1-w_k)^{c_k}$$
$$P(X=1|c_1,\dots,c_m,u)=(1-w_u)^{\neg u}\Pi_{k=1}^{m}(1-w_k)^{c_k}$$

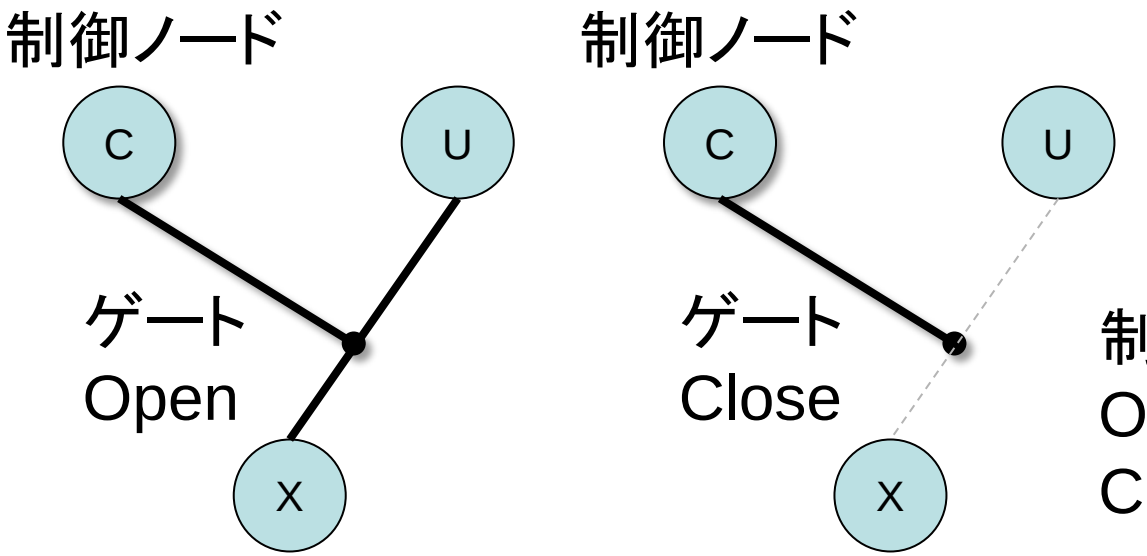
$P(R_X=0|X_1,\dots,X_m)=1$ (X_k のうち2つ以上が1の場合)
$P(R_X=1|X_1,\dots,X_m)=1$ (X_kのうち高々1つが1の場合)

- QBCのネットワークを図に書くときは、独自の記法を使用

例



ゲートノードの性質

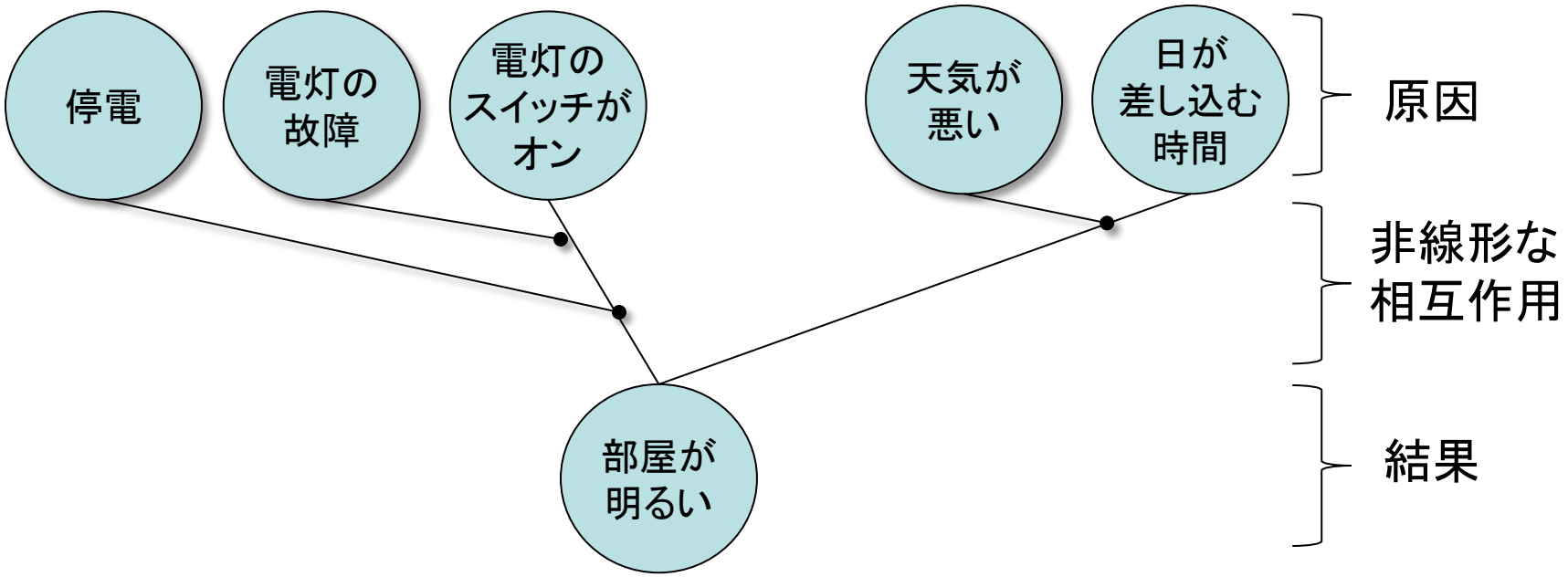


制御信号 C の値が
Open なら U と X が結合、
Close なら切断

- **論理回路のような感覚**で生成モデルを設計できる。
- **あくまでベイジアンネット**なので、
  - 出力（結果）から信号源（原因）の推定もできる。
  - データからゲートの制御方法を学習できる。
  - 因果関係の強さは0と1の間の値をとれる。

注：QBC ではできない

簡単なQBCの例



推論の例:

・夜、電灯のスイッチがオンなのに暗い。
　→ 停電または電灯の故障。

・停電なのに明るい。
　→ 日が差し込む時間で天気が良い。

自然界によくある
因果関係が
簡潔に表現可能

QBC の実装

Java ソースコード中の DSL（domain-specific language）で QBCを定義

```
List<TableNode> tableNodeList = new ArrayList<>();
List<GateMatrix> gateMatrixes = new ArrayList<>();
tableNodeList.add(tableNode(What, table(
    row(V1,A,A,A,A),
    row(V2,B,B,B,B),
    row(V3,C,C,C,C)
    )));
tableNodeList.add(tableNode(Where1, table(
    row(V1,I,O,I,O),
    row(V2,O,I,O,I)
    )));
tableNodeList.add(tableNode(Where2, table(
    row(V1,I,I,O,O),
    row(V2,O,O,I,I)
    )));
tableNodeList.add(tableNode(L1, table(
    row(A),row(B),row(C))));
tableNodeList.add(tableNode(L2, table(
    row(A),row(B),row(C))));
tableNodeList.add(tableNode(L3, table(
    row(A),row(B),row(C))));
tableNodeList.add(tableNode(L4, table(
    row(A),row(B),row(C))));
gateMatrixes.add(new GateMatrix(
    list(Where1,Where2), list(What), list(L1,L2,L3,L4)
    ));
```

QBNに変換

```
netData = { {
  {L4, What->L4, },
  {A, V1, },
  {B, V2, },
  {C, V3, },
  {0, 0, },
 }, {
  {What, },
  {V1, },
  {V2, },
  {V3, },
  {0, },
 }, {
  {L1, What->L1, },
  {A, V1, },
  {B, V2, },
  {C, V3, },
  {0, 0, },
 }, {
  {L3, What->L3, },
  {A, V1, },
  {B, V2, },
  {C, V3, },
  {0, 0, },
 }, {
  {Where2, },
  {V1, },
  {V2, },
  {0, },
 }, {
  {Where1, },
  {V1, },
  {V2, },
  {0, },
 }, {
```

```
{L2, What->L2, },
  {A, V1, },
  {B, V2, },
  {C, V3, },
  {0, 0, },
 }, {
  {What->L1, Where1, Where2, What, },
  {0, V1, __, __, },
  {0, __, V1, __, },
  {V1, V2, V2, V1, },
  {V2, V2, V2, V2, },
  {V3, V2, V2, V3, },
 }, {
  {What->L2, Where1, Where2, What, },
  {0, V2, __, __, },
  {0, __, V1, __, },
  {V1, V1, V2, V1, },
  {V2, V1, V2, V2, },
  {V3, V1, V2, V3, },
 }, {
  {What->L3, Where1, Where2, What, },
  {0, V1, __, __, },
  {0, __, V2, __, },
  {V1, V2, V1, V1, },
  {V2, V2, V1, V2, },
  {V3, V2, V1, V3, },
 }, {
  {What->L4, Where1, Where2, What, },
  {0, V2, __, __, },
  {0, __, V2, __, },
  {V1, V1, V1, V1, },
  {V2, V1, V1, V2, },
  {V3, V1, V1, V3, },
 }, };
```

```
Where1, Where0, What, L3, L2, L1, L0,
V0, V0, A, 0, 0, 0, A,
V0, V0, A, 0, 0, 0, A,
V0, V1, A, 0, 0, A, 0,
V0, V1, A, 0, 0, A, 0,
V0, V0, B, 0, 0, 0, B,
V0, V0, B, 0, 0, 0, B,
V0, V1, B, 0, 0, B, 0,
V0, V1, B, 0, 0, B, 0,
V0, V0, C, 0, 0, 0, C,
V0, V0, C, 0, 0, 0, C,
V0, V1, C, 0, 0, C, 0,
V0, V1, C, 0, 0, C, 0,
V1, V0, A, 0, A, 0, 0,
V1, V0, A, 0, A, 0, 0,
V1, V1, A, A, 0, 0, 0,
V1, V1, A, A, 0, 0, 0,
V1, V0, B, 0, B, 0, 0,
V1, V0, B, 0, B, 0, 0,
V1, V1, B, B, 0, 0, 0,
V1, V1, B, B, 0, 0, 0,
V1, V0, C, 0, C, 0, 0,
V1, V0, C, 0, C, 0, 0,
V1, V1, C, C, 0, 0, 0,
V1, V1, C, C, 0, 0, 0,
24 solutions.
```

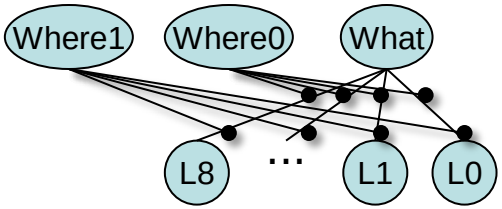
SATソルバ等を利用した全解探索

QBCを使う利点と欠点

- 利点：認知モデル設計の本質に専念できる。
  - 「ネットワーク全体を最適化する」というベイジアンネットの特性は残している。
  - 局所解・過適合等の機械学習特有の問題は起きない。
  - 決定論的に動作。
  - 変数の値に意味のある文字列が使えるので可読性が高い。
- 欠点：学習不可、最悪指数時間

視覚野の腹側経路･背側経路のプロトタイプ

背側経路とグリッド細胞



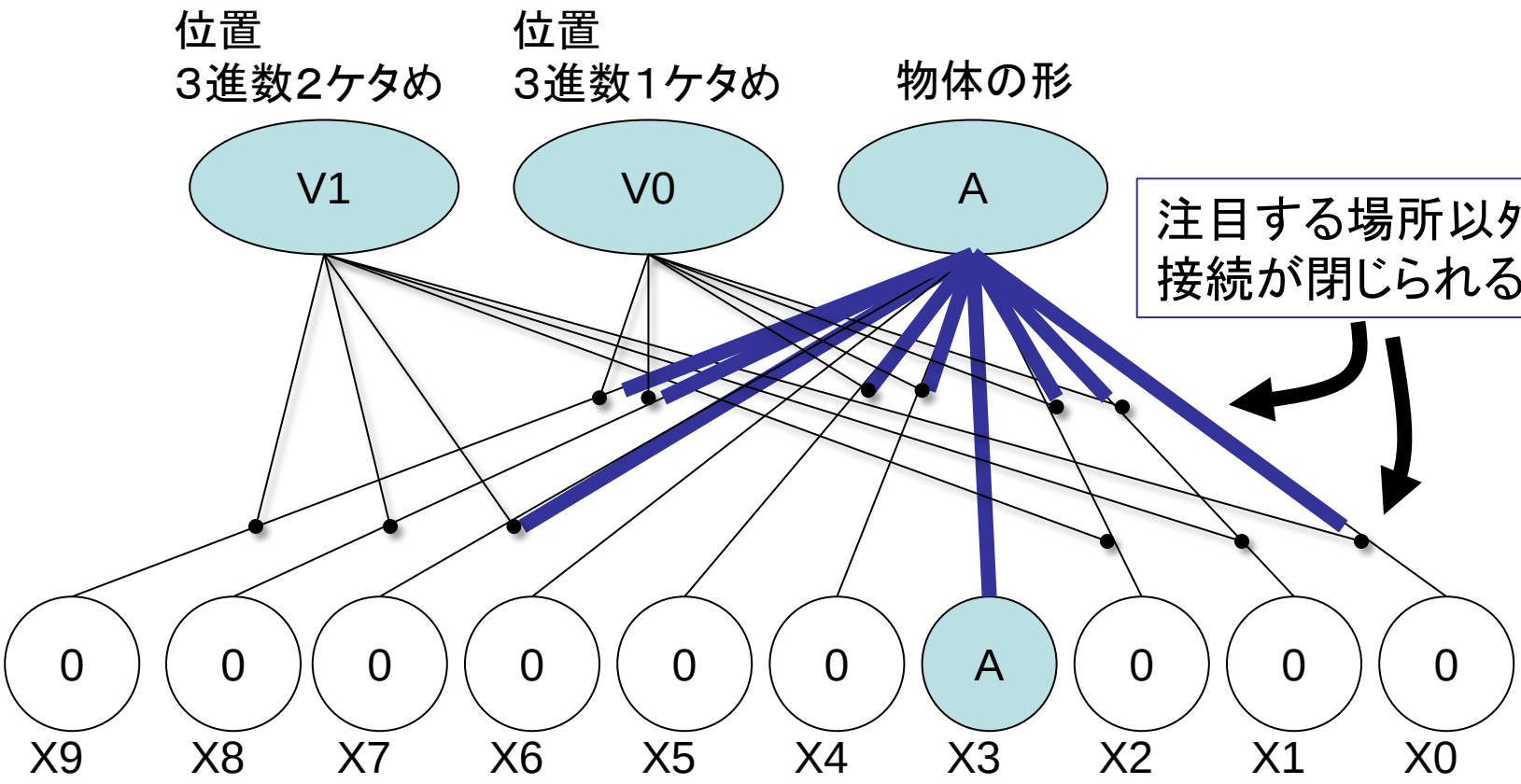

```
net.tableNodeList.add(tableNode(Where1, table(
        row(V0,I,I,I,I,I,O,O,O),
        row(V1,I,I,I,O,O,O,I,I,I),
        row(V2,O,O,O,I,I,I,I,I,I)
        )));
net.tableNodeList.add(tableNode(Where0, table(
        row(V0,I,I,O,I,I,O,I,I,O),
        row(V1,I,O,I,I,O,I,I,O,I),
        row(V2,O,I,I,O,I,I,O,I,I)
        )));
net.tableNodeList.add(tableNode(What, table(
        row(A,A,A,A,A,A,A,A,A,A),
        row(B,B,B,B,B,B,B,B,B,B)
        )));
net.tableNodeList.add(tableNode(L8, table(row(A),row(B))));
net.tableNodeList.add(tableNode(L7, table(row(A),row(B))));
net.tableNodeList.add(tableNode(L6, table(row(A),row(B))));
net.tableNodeList.add(tableNode(L5, table(row(A),row(B))));
net.tableNodeList.add(tableNode(L4, table(row(A),row(B))));
net.tableNodeList.add(tableNode(L3, table(row(A),row(B))));
net.tableNodeList.add(tableNode(L2, table(row(A),row(B))));
net.tableNodeList.add(tableNode(L1, table(row(A),row(B))));
net.tableNodeList.add(tableNode(L0, table(row(A),row(B))));
net.gateMatrixes.add(new GateMatrix(
        list(Where1,Where0), list(What),
        list(L8,L7,L6,L5,L4,L3,L2,L1,L0)
        ));
```

すべての解

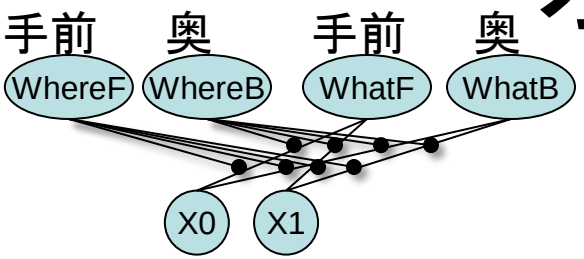
```
Where1, Where0, What, L8, L7, L6, L5, L4, L3, L2, L1, L0,
V0, V0, A, 0, 0, 0, 0, 0, 0, 0, 0, A,
V0, V0, B, 0, 0, 0, 0, 0, 0, 0, 0, B,
V0, V1, A, 0, 0, 0, 0, 0, 0, 0, A, 0,
V0, V1, B, 0, 0, 0, 0, 0, 0, 0, B, 0,
V0, V2, A, 0, 0, 0, 0, 0, 0, A, 0, 0,
V0, V2, B, 0, 0, 0, 0, 0, 0, B, 0, 0,
V1, V0, A, 0, 0, 0, 0, 0, A, 0, 0, 0,
V1, V0, B, 0, 0, 0, 0, 0, B, 0, 0, 0,
V1, V1, A, 0, 0, 0, 0, A, 0, 0, 0, 0,
V1, V1, B, 0, 0, 0, 0, B, 0, 0, 0, 0,
V1, V2, A, 0, 0, 0, A, 0, 0, 0, 0, 0,
V1, V2, B, 0, 0, 0, B, 0, 0, 0, 0, 0,
V2, V0, A, 0, 0, A, 0, 0, 0, 0, 0, 0,
V2, V0, B, 0, 0, B, 0, 0, 0, 0, 0, 0,
V2, V1, A, 0, A, 0, 0, 0, 0, 0, 0, 0,
V2, V1, B, 0, B, 0, 0, 0, 0, 0, 0, 0,
V2, V2, A, A, 0, 0, 0, 0, 0, 0, 0, 0,
V2, V2, B, B, 0, 0, 0, 0, 0, 0, 0, 0,
```

Where0 のユニットは
一次元格子点上に受容野を持つ。

ネットワークの状態の例



オクルージョン



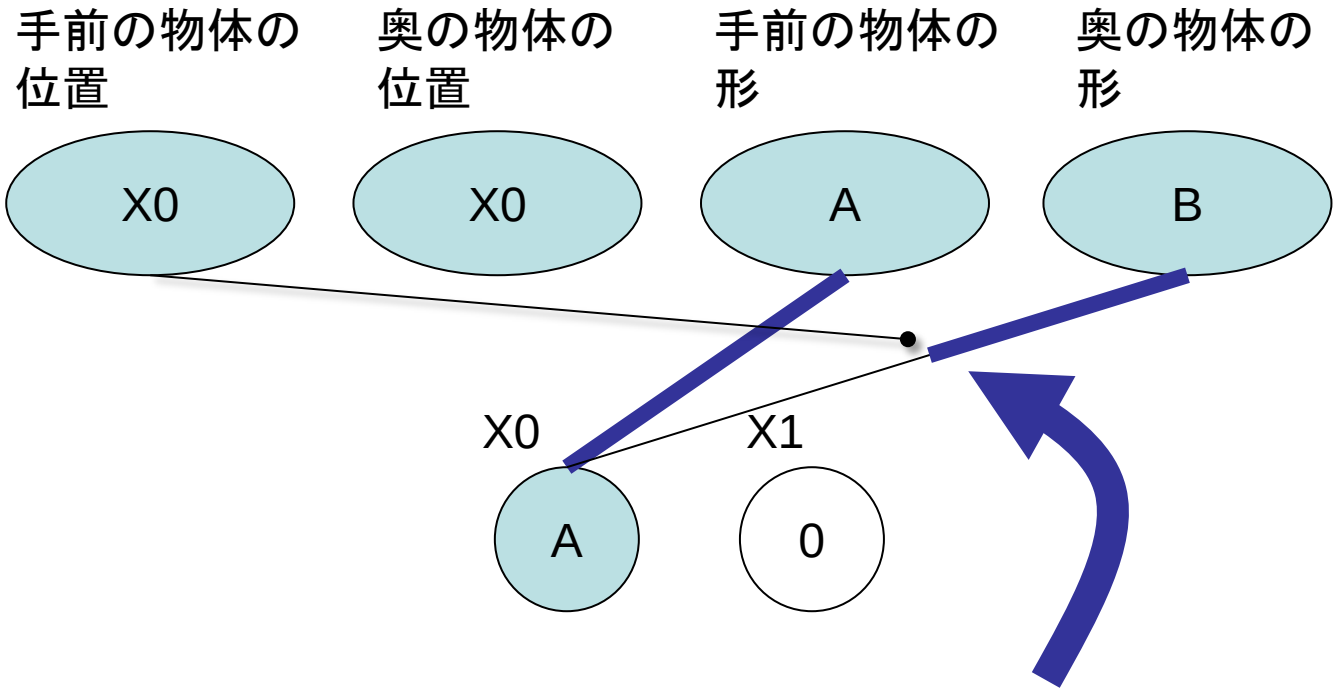
2つの物体の位置が重なっているとき、
手前のものだけが見える。

```
net.tableNodeList.add(tableNode(WhereF, table(
        // WhatF->X0, WhatB->X0, WhatF->X1, WhatB->X1
        row(X0,O,I,I,O),
        row(X1,I,O,O,I)
        )));
 net.tableNodeList.add(tableNode(WhereB, table(
        // WhatF->X0, WhatB->X0, WhatF->X1, WhatB->X1
        row(X0,O,O,O,I),
        row(X1,O,I,O,O)
        )));
 net.tableNodeList.add(tableNode(WhatF, table(
        row(A,A,A),
        row(B,B,B),
        row(O,O,O)
        )));
 net.tableNodeList.add(tableNode(WhatB, table(
        row(A,A,A),
        row(B,B,B),
        row(O,O,O)
        )));
 net.tableNodeList.add(tableNode(X0, table(row(A),row(B))));
 net.tableNodeList.add(tableNode(X1, table(row(A),row(B))));
 net.gateMatrixes.add(new GateMatrix(
        list(WhereF,WhereB), list(WhatF,WhatB), list(X0,X1) ));
```

すべての解

WhereF, WhereB, WhatF, WhatB, X0, X1,
X0, X0, 0, 0, 0, 0,
X0, X0, 0, A, 0, 0,
X0, X0, 0, B, 0, 0,
X0, X0, A, 0, A, 0,
X0, X0, A, A, A, 0,
X0, X0, A, B, A, 0,
X0, X0, B, 0, B, 0,
X0, X0, B, A, B, 0,
X0, X0, B, B, B, 0,
X0, X1, 0, 0, 0, 0,
X0, X1, 0, A, 0, A,
X0, X1, 0, B, 0, B,
X0, X1, A, 0, A, 0,
X0, X1, A, A, A, A,
X0, X1, A, B, A, B,
X0, X1, B, 0, B, 0,
X0, X1, B, A, B, A,
X0, X1, B, B, B, B,
X1, X0, 0, 0, 0, 0,
X1, X0, 0, A, A, 0,
X1, X0, 0, B, B, 0,
X1, X0, A, 0, 0, A,
X1, X0, A, A, A, A,
X1, X0, A, B, B, A,
X1, X0, B, 0, 0, B,
X1, X0, B, A, A, B,
X1, X0, B, B, B, B,
X1, X1, 0, 0, 0, 0,
X1, X1, 0, A, 0, 0,
X1, X1, 0, B, 0, 0,
X1, X1, A, 0, 0, A,
X1, X1, A, A, 0, A,
X1, X1, A, B, 0, A,
X1, X1, B, 0, 0, B,
X1, X1, B, A, 0, B,
X1, X1, B, B, 0, B,

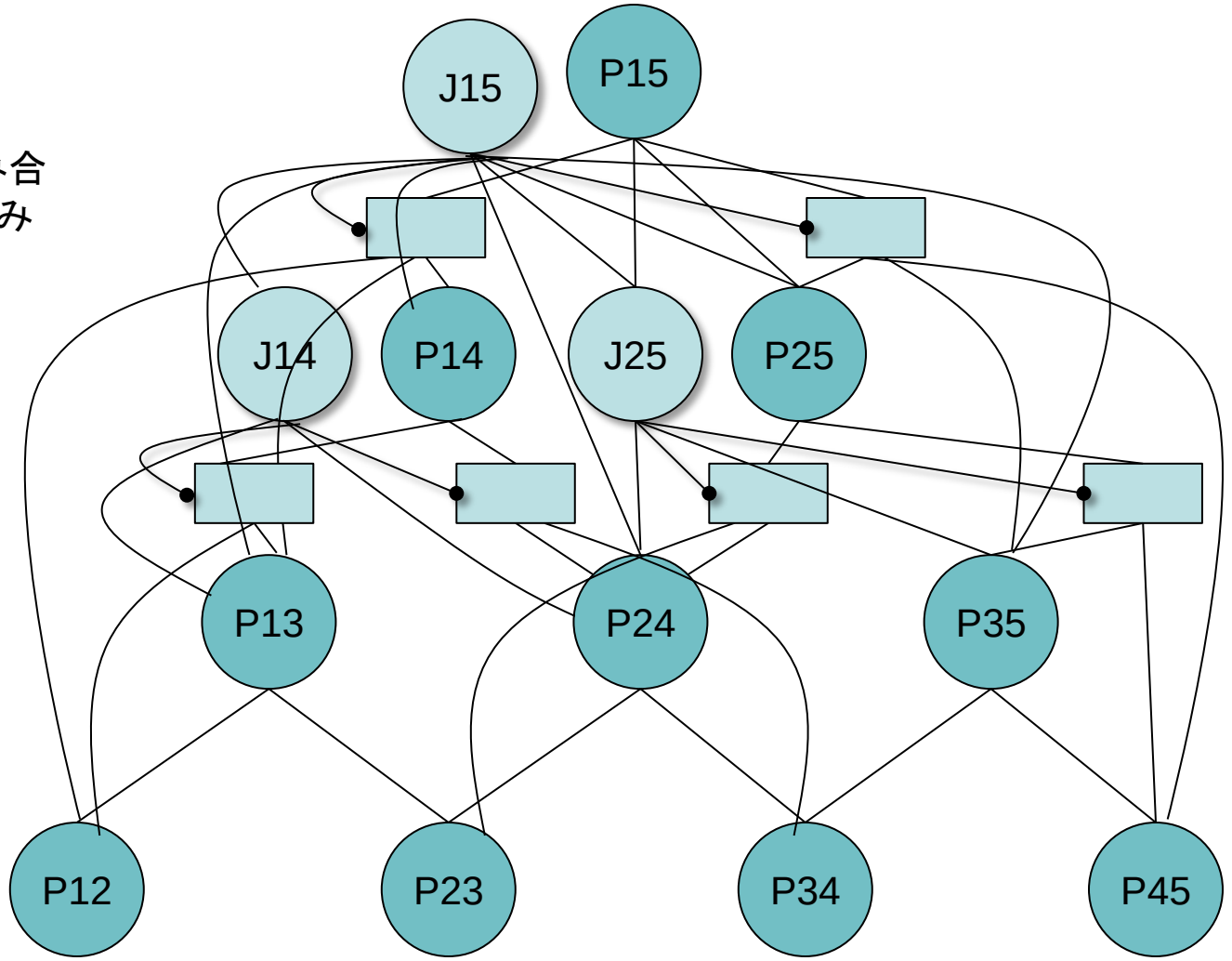
ネットワークの状態の例



言語野の認知モデル構築に向けた要素技術のプロトタイプ

構文解析をする回路のプロトタイプ

J15 は、
区間
[1,2]と[2,5]、
[1,3]と[3,5]、
[1,4]と[4,5]の
いずれかの組み合わせのゲートのみを開ける。



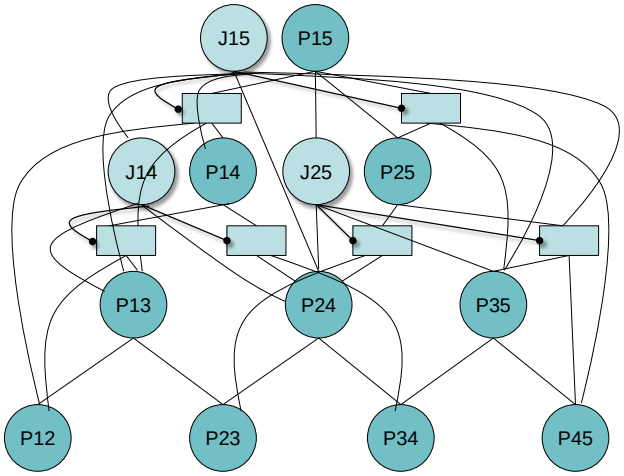
構文解析器の QBC 定義

QBC定義の一部（全体で約160行）

```
net.tableNodeList.add(tableNode(P13, table(
        row(N1,N11,N12),
        row(N2,N21,N22),
        row(N11,N111,N112),
        row(N12,N121,N122),
        row(N21,N211,N212),
        row(N22,N221,N222),
        row(I,__,__))));
net.tableNodeList.add(tableNode(J15, table(
        //childNames: [P13, J14, P14, J25, P25,
P35, P24, P15->P12, P15->P13, P15->P14, P15->P25,
P15->P35, P15->P45]
        row(J2, // 区間 [1,2],[2,5]
            I,I,I,
            or(J3,J4),__,__,
            __,
            O,I,I,
            O,I,I),
        row(J3, // 区間 [1,3],[3,5]
            __,I,I,
            I,I,__,
            I,
            I,O,I,
            I,O,I),
        row(J4, // 区間 [1,4],[4,5]
            __,or(J2,J3),__,
            I,I,I,
            __,
            I,I,O,
            I,I,O)
        )));
```

文法

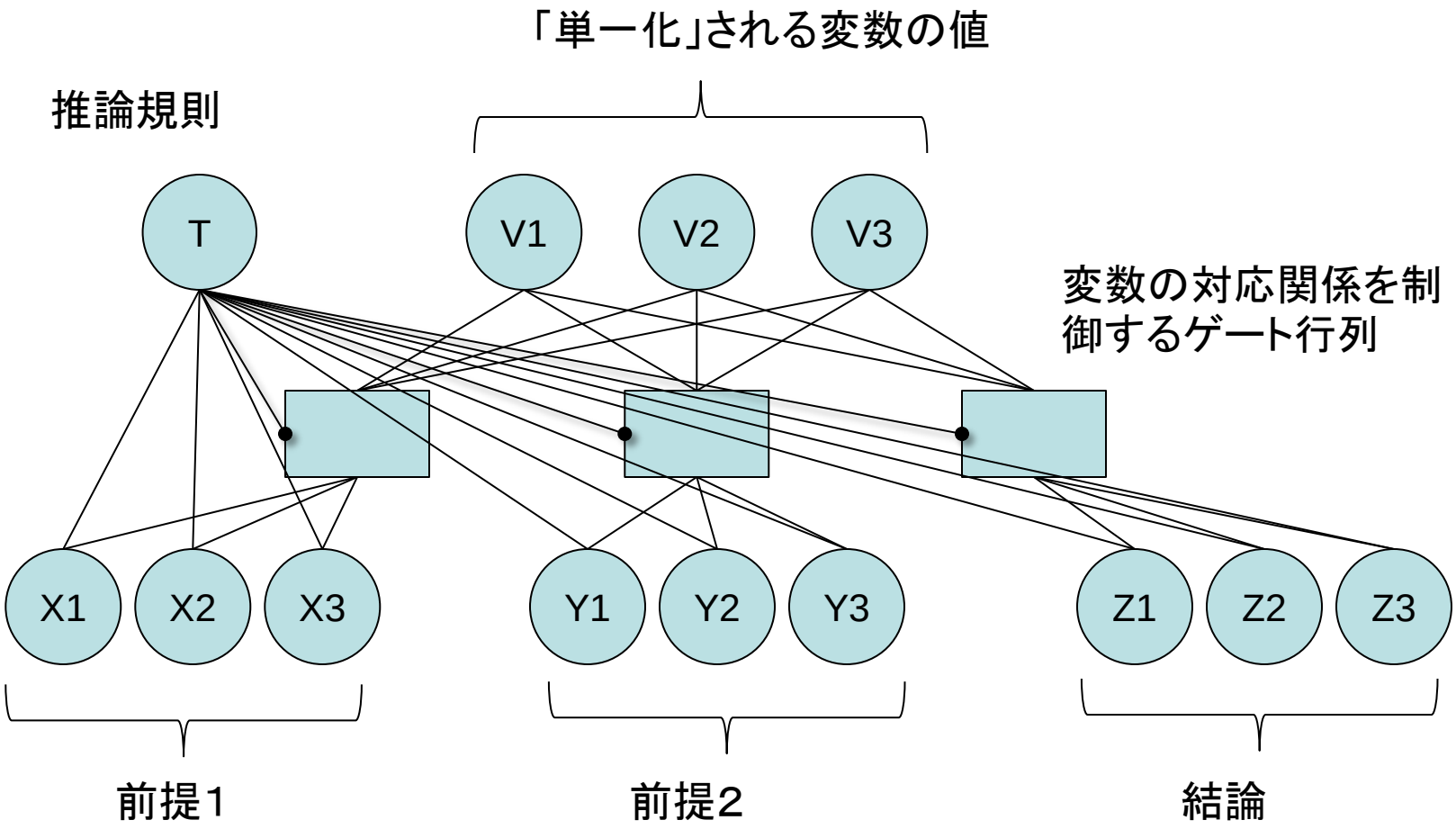
S -> N1 N2
N1 -> N11 N12
N2 -> N21 N22
N11 -> N111 N112
N12 -> N121 N122
N21 -> N211 N212
N22 -> N221 N222



すべての解

P15, P14, P25, P13, P24, P35, P12, P23, P34, P45, J15, J14, J25,
S, 1, N2, 1, 1, N22, N1, N21, N221, N222, J2, 1, J3,
S, 1, N2, 1, N21, 1, N1, N211, N212, N22, J2, 1, J4,
S, 1, 1, N1, 1, N2, N11, N12, N21, N22, J3, 1, 1,
S, N1, 1, 1, N12, 1, N11, N121, N122, N2, J4, J2, 1,
S, N1, 1, N11, 1, 1, N111, N112, N12, N2, J4, J3, 1,
5 solutions.

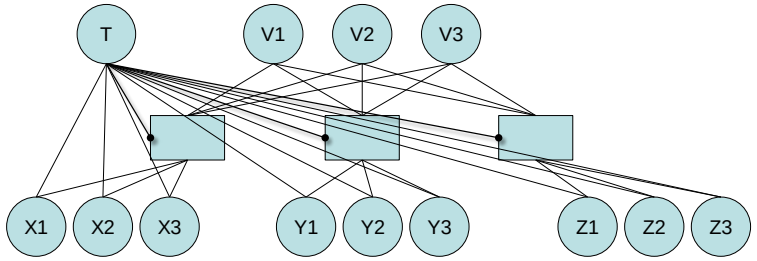
ユニフィケーションを使って推論規則の適用を実行する回路



推論規則適用のQBC定義

```
QBCnet net = new QBCnet();
Object var = or(A,B,C);
net.tableNodeList.add(tableNode(T, table(
      // childNames: [X1, X2, X3, Y1, Y2, Y3, Z1, Z2, Z3,
      // V1->X1, V2->X1, V3->X1, V1->X2, V2->X2, V3->X2, V1->X3, V2->X3, V3->X3,
      // V1->Y1, V2->Y1, V3->Y1, V1->Y2, V2->Y2, V3->Y2, V1->Y3, V2->Y3, V3->Y3,
      // V1->Z1, V2->Z1, V3->Z1, V1->Z2, V2->Z2, V3->Z2, V1->Z3, V2->Z3, V3->Z3]
      row(T1, // V1, V1->V2  ==>  V2
            var,N,N, var,THEN,var, var,N,N,
            O,I,I, I,I,I, I,I,I,
            O,I,I, I,I,I, I,O,I,
            I,O,I, I,I,I, I,I,I),
      row(T2, // not V2, V1->V2  ==>  not V1
            N,NOT,var, var,THEN,var, N,NOT,var,
            I,I,I, I,I,I, I,O,I,
            O,I,I, I,I,I, I,O,I,
            I,I,I, I,I,I, O,I,I),
      row(T3, // V1 and V2, not V1  ==>  V2
            var,AND,var, N,NOT,var, var,N,N,
            O,I,I, I,I,I, I,O,I,
            I,I,I, I,I,I, O,I,I,
            I,O,I, I,I,I, I,I,I),
      row(T4, // V1->V2, V2->V3  ==>  V1->V3
            var,THEN,var, var,THEN,var, var,THEN,var,
            O,I,I, I,I,I, I,O,I,
            I,O,I, I,I,I, I,I,O,
            O,I,I, I,I,I, I,I,O)
      )));
List<TableRow> vars9Table = table(
      row(A, A,A,A, A,A,A, A,A,A),
      row(B, B,B,B, B,B,B, B,B,B),
      row(C, C,C,C, C,C,C, C,C,C),
      row(N, N,N,N ,N,N,N, N,N,N));
List<TableRow> varsTable = table(
      row(A), row(B), row(C), row(N));
List<TableRow> opsTable = table(
      row(AND), row(OR), row(NOT), row(THEN), row(N));
net.tableNodeList.add(tableNode(V1, vars9Table));
net.tableNodeList.add(tableNode(V2, vars9Table));
net.tableNodeList.add(tableNode(V3, vars9Table));
net.tableNodeList.add(tableNode(X1, varsTable));
net.tableNodeList.add(tableNode(X2, opsTable));
net.tableNodeList.add(tableNode(X3, varsTable));
net.tableNodeList.add(tableNode(Y1, varsTable));
net.tableNodeList.add(tableNode(Y2, opsTable));
net.tableNodeList.add(tableNode(Y3, varsTable));
net.tableNodeList.add(tableNode(Z1, varsTable));
net.tableNodeList.add(tableNode(Z2, opsTable));
net.tableNodeList.add(tableNode(Z3, varsTable));
net.gateMatrixList.add(gateMatrix(
      list(), list(T),
      list(X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3)
      ));
net.gateMatrixList.add(gateMatrix(
      list(T), list(V1,V2,V3),
      list(X1,X2,X3,Y1,Y2,Y3,Z1,Z2,Z3)
      ));
return net;
```

推論規則

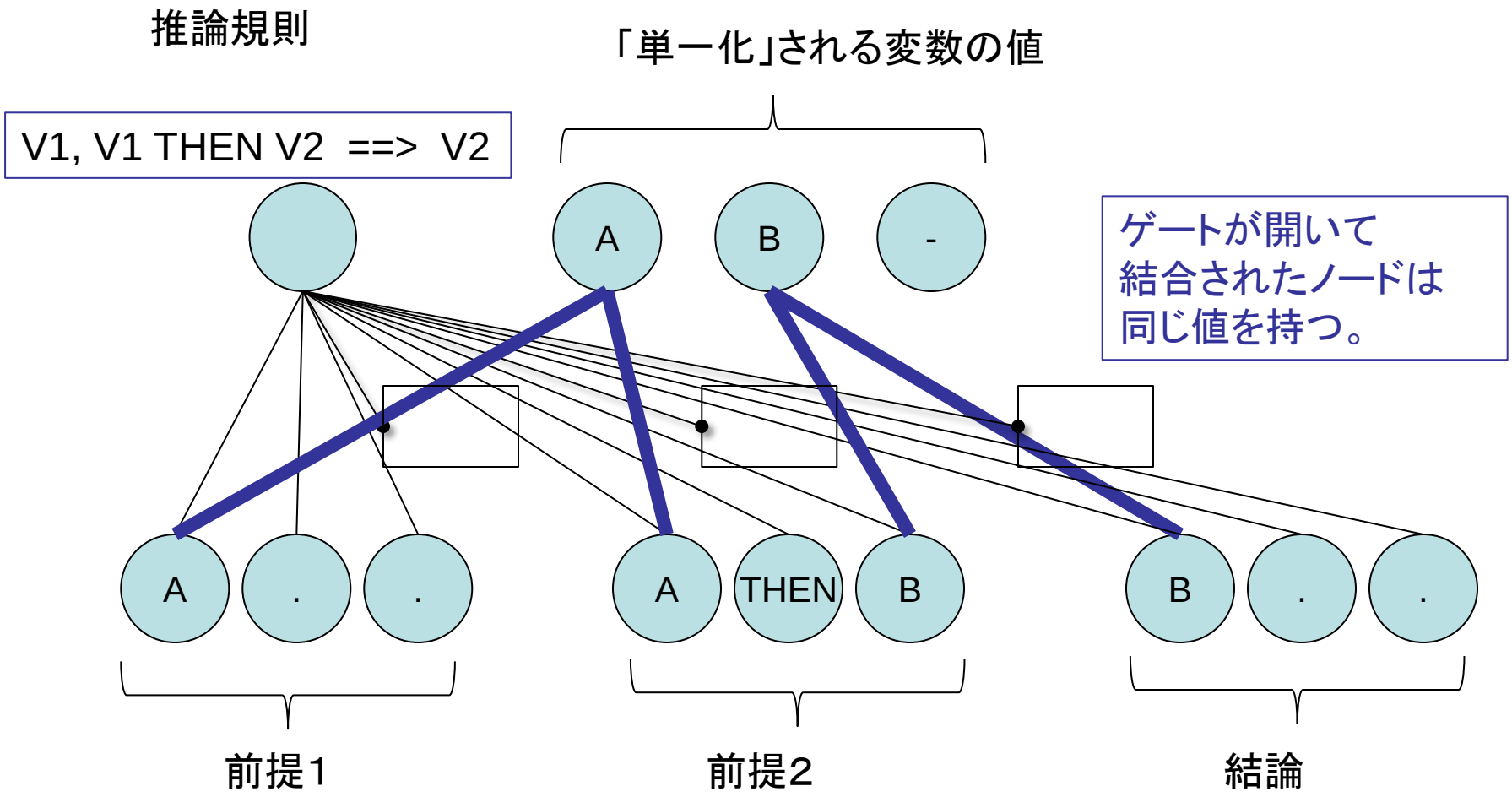


V1, V1 THEN V2 ==> V2
NOT V2, V1 THEN V2 ==> NOT V1
V1 AND V2, NOT V1 ==> V2
V1 THEN V2, V2 THEN V3 ==> V1 THEN V3

すべての解

T, V1, V2, V3, X1, X2, X3, Y1, Y2, Y3, Z1, Z2, Z3,
T1, A, A, A, A, ., ., A, THEN, A, A, ., .,
T1, B, A, A, B, ., ., B, THEN, A, A, ., .,
T1, C, A, A, C, ., ., C, THEN, A, A, ., .,
T1, A, A, B, A, ., ., A, THEN, A, A, ., .,
T1, B, A, B, B, ., ., B, THEN, A, A, ., .,
T1, C, A, B, C, ., ., C, THEN, A, A, ., .,
...
T4, A, C, A, A, THEN, C, C, THEN, A, A, THEN, A,
T4, B, C, A, B, THEN, C, C, THEN, A, B, THEN, A,
T4, C, C, A, C, THEN, C, C, THEN, A, C, THEN, A,
T4, A, C, B, A, THEN, C, C, THEN, B, A, THEN, B,
T4, B, C, B, B, THEN, C, C, THEN, B, B, THEN, B,
T4, C, C, B, C, THEN, C, C, THEN, B, C, THEN, B,
T4, A, C, C, A, THEN, C, C, THEN, C, A, THEN, C,
T4, B, C, C, B, THEN, C, C, THEN, C, B, THEN, C,
T4, C, C, C, C, THEN, C, C, THEN, C, C, THEN, C,
162 solutions.

推論規則を1つ選択したときのネットワークの状態の例



まとめと今後

- QBCは制限付きベイジアンネットを使った認知モデル設計のための便利な道具
  - 大規模機械学習の様々な困難を回避し、モデル設計に集中できる。
- 視覚野・言語野に関する認知モデルのプロトタイプを実装中。
- 設計した認知モデルが「真の制限付きベイジアンネット」で本当に動くかどうかは今後検証。