並列疑似エラー補正法に基づく“破格”な言語表現の (疑似) 解釈

不自然言語処理のための理論的枠組み

黒田 航

京都工芸繊維大学 (非常勤) / 早稲田大学総合研究機構 (客員研究員)

Date 2011-03-10, 言語処理学会第17回年次大会 (豊橋科学技術大学)

“不自然言語処理 (NNLP)” が必要なワケ

- Web 文書に多発する “破格”表現は NLP の鬼門
  - 具体例は後で
- 理由
  - 破格表現はしばしば辞書に存在しない要素を含み，有効な形態素解析結果が得られない
  - 破格表現はしばしば文法的に正しくない表現で，有効な構文解析結果が得られない.

破格表現の例 1/2

- (1) 新表記や新語のあれこれ

  a. **ネ申** (“神” の新表記), **儲** (“信者” の新表記)

  b. **ようつべ** (“YouTube” の新表記), **ガイシュツ** (“既出”の新表記), **マンセー** (“万歳” の新表記), **ふいんき（何故か変換できない）**, **ちょwwwおまwwww** (“ちょっとお前, 何言ってんの?” の異形)

  c. **氏ね** (“死ね” の新表記), **人大杉** (“人多すぎ” の新表記), **空気嫁** (“空気読め” の異表記?), **マスゴミ** (“マスコミ” の異表記?), **腐女子** (“婦女子” の異表記?), **全俺が泣いた** (“全米が泣いた” の 誇張表現/パロディー)

- 詳細は黒田一平 (2010)を参照のこと

破格表現の例 1/2

- (2) うろ覚えや逸脱で生じる拡張用法

  a. 時間を**もてあそぶ**

  b. 熱血**感**, 門外**感**, 衰弱**さ**, 唖然**さ**

- 詳細は黒田・寺崎 (2010) を参照

何が問題か?

- 問題
  - 大規模データに基づくと
    - 新表記と新語/新表記=未知語の境界は非常に曖昧
    - 誤用と新語/新表記=未知語の境界は非常に曖昧
- 論点
  - “不自然言語処理”はこの区別を前提にしない処理であることが必要

本発表の狙い

- **並列疑似エラー補正法** (Parallel Simulated Error Correction: PSEC) (黒田 2009) に基づいて言語表現の (疑似) 解釈を提示し
- “不自然処理” のための理論的枠組みを提示
  - ただし 文節化の最適性の問題を先送りしているため，PSECの不自然言語処理の適用範囲は限定的
- 主張
  - “不自然言語処理” には “絶対記憶” ベース (黒田 2007; Kuroda 2009; 黒田・長谷部 2009) で “類推ベース” (Daelemans and van den Bosch 2005; 佐藤 1997; Skousen 1989) の処理システムの実装が不可欠

並列疑似エラー補正の概要

並列疑似エラー補正とは?

- 入力 X が n 個の部分 $x_1, x_2, \ldots, x_n$ からなる系列だとする．x_i を X の i 番目の文節と呼ぶ
- Step 0 (初期化): i 番目の文節 x_i を変項化した状態 $X'_1, X'_2, \ldots, X'_n$ を仮想的に作り出す
  - この時，X は $\{X'_1, X'_2, \ldots, X'_n\}$ の重ね合わせと定義
- Step 1: X_i' ごとに x_i の値の補完．結果を X_i'' とする．これが疑似エラー補正と呼ぶ処理
- Step 2: Step 1 で X_i'' の疑似エラー補正の (有効な) 候補の数が 0 だった場合，導入する変項の数を一つづつ増やし，段階的に探索範囲を広げる (事例集合の Beam 探索)
  - PSECの探索空間はパターン束 (pattern lattice) (Kuroda 2009; 黒田・長谷部 2009) で記述される
- Step 3: Step 1–Step 2の処理を終了条件を満足するまで繰り返し，その結果である $\{X''_1, X''_2, \ldots, X''_n\}$ を重ね合わせによって統合する

並列疑似エラー補正の図解

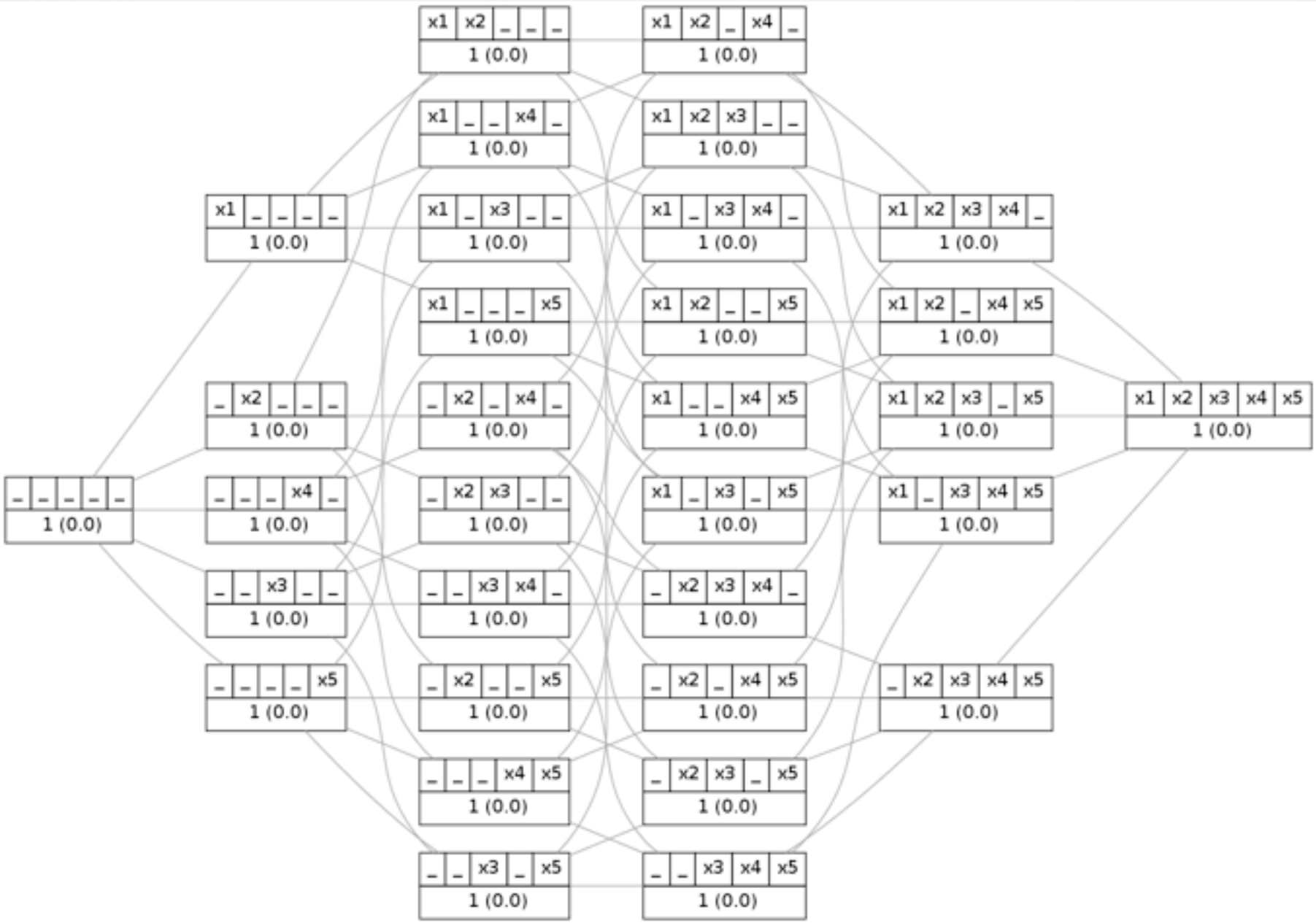
表 1　$X = x_1 \cdot x_2 \cdots x_n$ の PSEC の初期状態 (=1 個の疑似エラーが発生した n 通りの状態)

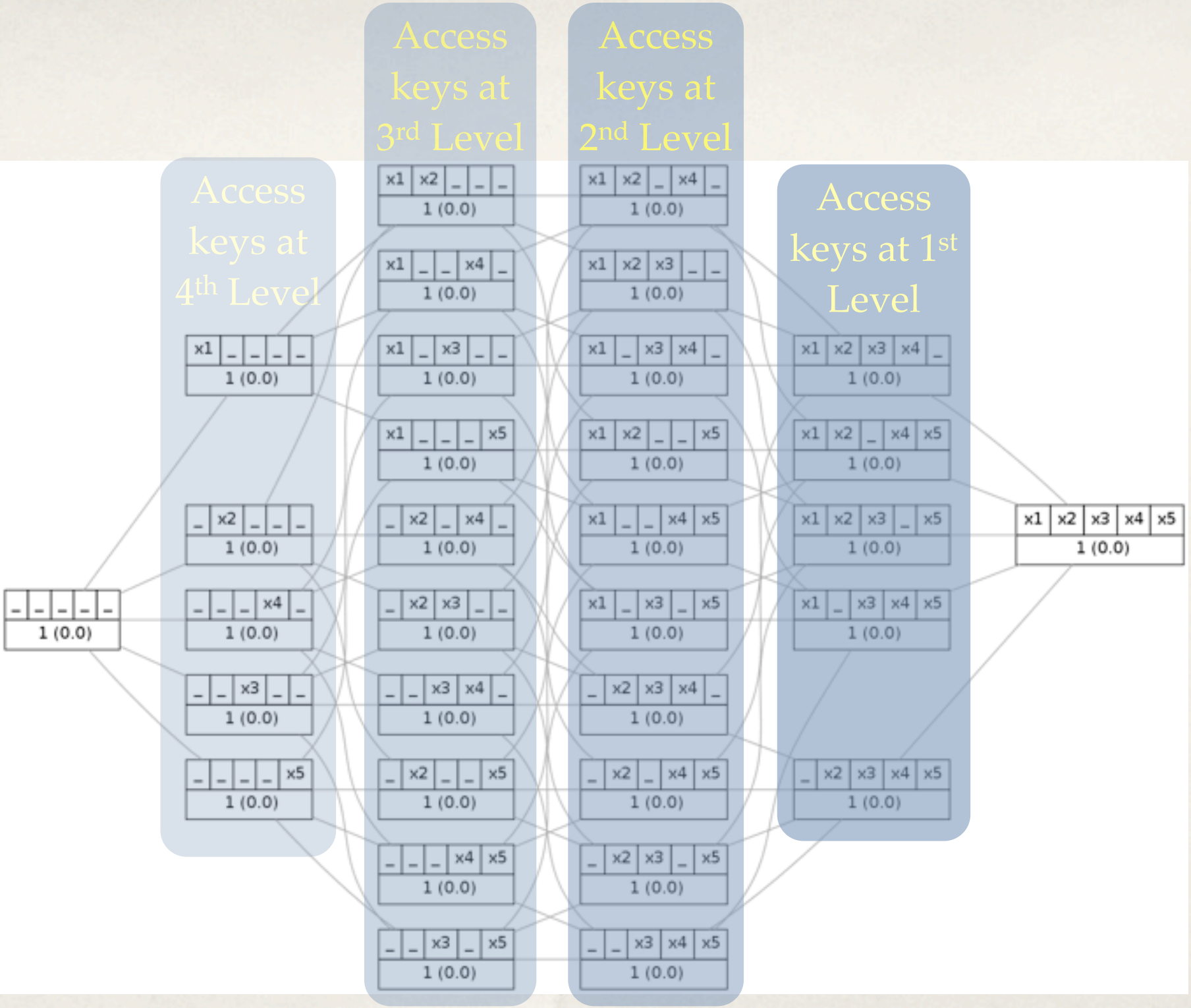
X	x_1	x_2	$\cdots$	x_i	$\cdots$	x_n	変項化
X'_1	Δ_1	x_2	$\cdots$	x_i	$\cdots$	x_n	$x_1 \Rightarrow \Delta_1$
X'_2	x_1	Δ_2	$\cdots$	x_i	$\cdots$	x_n	$x_2 \Rightarrow \Delta_2$
X'_i	x_1	x_2	$\cdots$	Δ_i	$\cdots$	x_n	$x_i \Rightarrow \Delta_i$
X'_n	x_1	x_2	$\cdots$	x_i	$\cdots$	Δ_n	$x_n \Rightarrow \Delta_n$

表 2　$X = x_1 \cdot x_2 \cdots x_n$ の疑似エラー Δ_i の補正後の状態 (x''_i は特定の語句)

X	x_1	x_2	$\cdots$	x_i	$\cdots$	x_n	補正
X''_1	x''_1	x_2	$\cdots$	x_i	$\cdots$	x_n	$\Delta_1 \Rightarrow x''_1$
X''_2	x_1	x''_2	$\cdots$	x_i	$\cdots$	x_n	$\Delta_2 \Rightarrow x''_2$
X''_i	x_1	x_2	$\cdots$	x''_i	$\cdots$	x_n	$\Delta_i \Rightarrow x''_i$
X''_n	x_1	x_2	$\cdots$	x_i	$\cdots$	x''_n	$\Delta_n \Rightarrow x''_n$

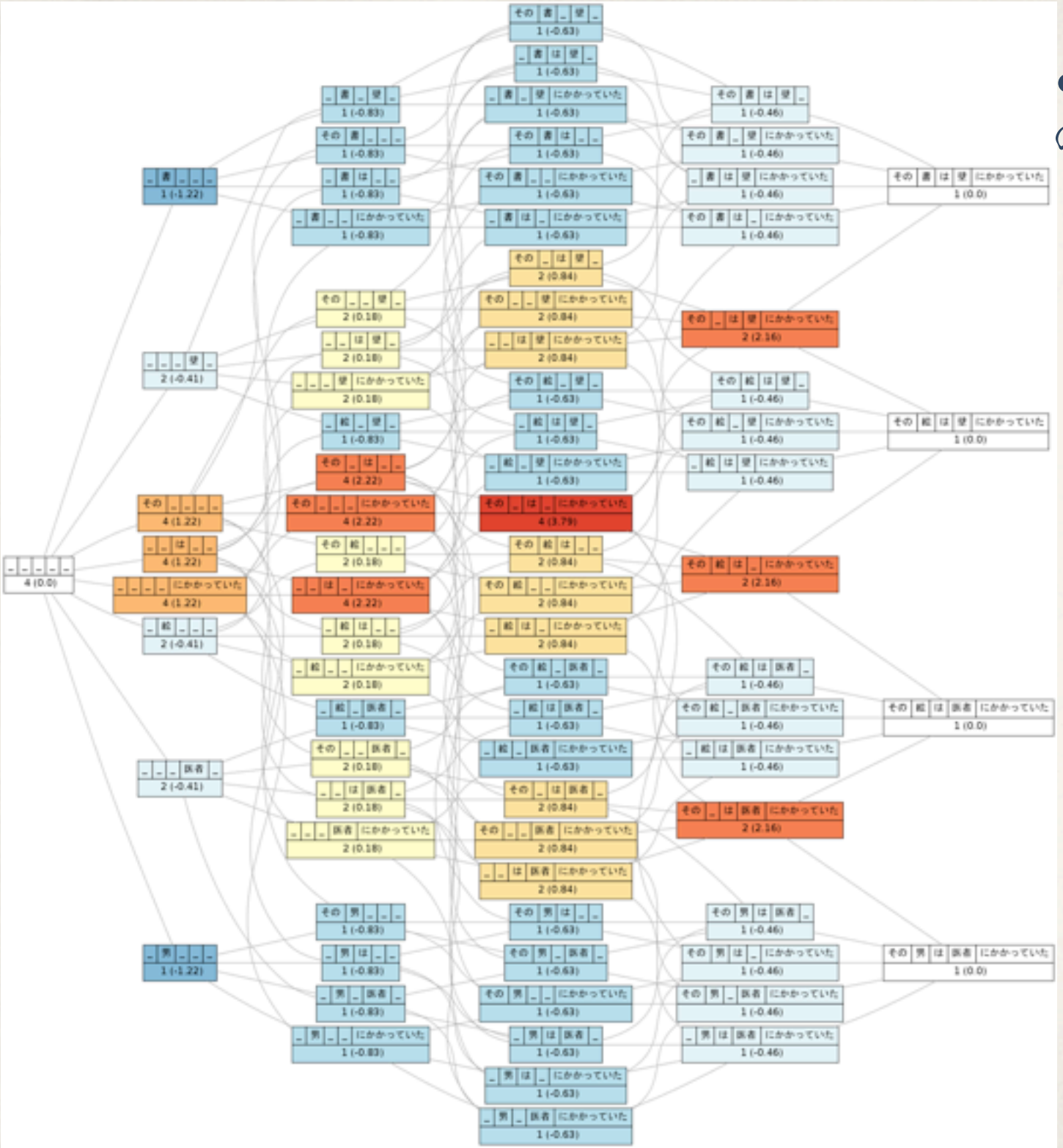
- $X = x_1 \cdots x_i \cdots x_n$ とする
- 表1 は Δ_i は x_i の変項化で，Δ_i を含む疑似エラー状態 X' (全部で n 個) を表わす
- 表2 は疑似エラーが補正された状態 X'' (全部で n 個) を表わす
- X'' を重ね合わせた状態を入力 Xの意味表示と同一視
- やっていることは意味表示の Fourier 変換?


x1 x2 _ _ _
1 (0.0)
x1 x2 _ x4 _
1 (0.0)
x1 _ _ x4 _
1 (0.0)
x1 x2 x3 _ _
1 (0.0)
x1 _ _ _ _
1 (0.0)
x1 _ x3 _ _
1 (0.0)
x1 _ x3 x4 _
1 (0.0)
x1 x2 x3 x4 _
1 (0.0)
x1 _ _ _ x5
1 (0.0)
x1 x2 _ _ x5
1 (0.0)
x1 x2 _ x4 x5
1 (0.0)
_ x2 _ _ _
1 (0.0)
_ x2 _ x4 _
1 (0.0)
x1 _ _ x4 x5
1 (0.0)
x1 x2 x3 _ x5
1 (0.0)
x1 x2 x3 x4 x5
1 (0.0)
_ _ _ _ _
1 (0.0)
_ _ _ x4 _
1 (0.0)
_ x2 x3 _ _
1 (0.0)
x1 _ x3 _ x5
1 (0.0)
x1 _ x3 x4 x5
1 (0.0)
_ _ x3 _ _
1 (0.0)
_ _ x3 x4 _
1 (0.0)
_ x2 x3 x4 _
1 (0.0)
_ _ _ _ x5
1 (0.0)
_ x2 _ _ x5
1 (0.0)
_ x2 _ x4 x5
1 (0.0)
_ x2 x3 x4 x5
1 (0.0)
_ _ _ x4 x5
1 (0.0)
_ x2 x3 _ x5
1 (0.0)
_ _ x3 _ x5
1 (0.0)
_ _ x3 x4 x5
1 (0.0)

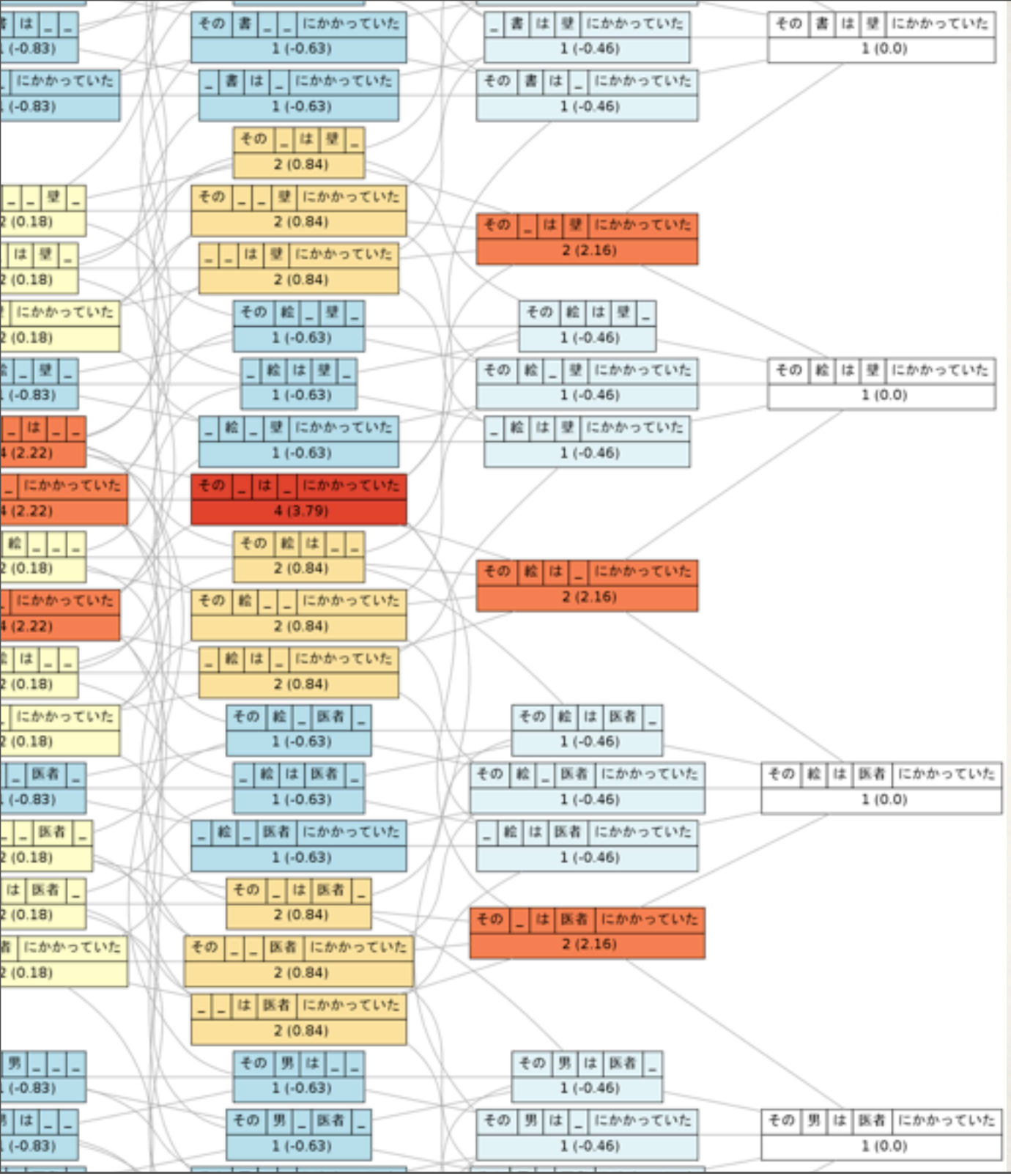

Access keys at 4th Level
Access keys at 3rd Level
Access keys at 2nd Level
Access keys at 1st Level
_ _ _ _ _ 1 (0.0)
x1 _ _ _ _ 1 (0.0)
_ x2 _ _ _ 1 (0.0)
_ _ _ x4 _ 1 (0.0)
_ _ x3 _ _ 1 (0.0)
_ _ _ _ x5 1 (0.0)
x1 x2 _ _ _ 1 (0.0)
x1 _ _ x4 _ 1 (0.0)
x1 _ x3 _ _ 1 (0.0)
x1 _ _ _ x5 1 (0.0)
_ x2 _ x4 _ 1 (0.0)
_ x2 x3 _ _ 1 (0.0)
_ _ x3 x4 _ 1 (0.0)
_ x2 _ _ x5 1 (0.0)
_ _ _ x4 x5 1 (0.0)
_ _ x3 _ x5 1 (0.0)
x1 x2 _ x4 _ 1 (0.0)
x1 x2 x3 _ _ 1 (0.0)
x1 _ x3 x4 _ 1 (0.0)
x1 x2 _ _ x5 1 (0.0)
x1 _ _ x4 x5 1 (0.0)
x1 _ x3 _ x5 1 (0.0)
_ x2 x3 x4 _ 1 (0.0)
_ x2 _ x4 x5 1 (0.0)
_ x2 x3 _ x5 1 (0.0)
_ _ x3 x4 x5 1 (0.0)
x1 x2 x3 x4 _ 1 (0.0)
x1 x2 _ x4 x5 1 (0.0)
x1 x2 x3 _ x5 1 (0.0)
x1 _ x3 x4 x5 1 (0.0)
_ x2 x3 x4 x5 1 (0.0)
x1 x2 x3 x4 x5 1 (0.0)

PSEC in Action

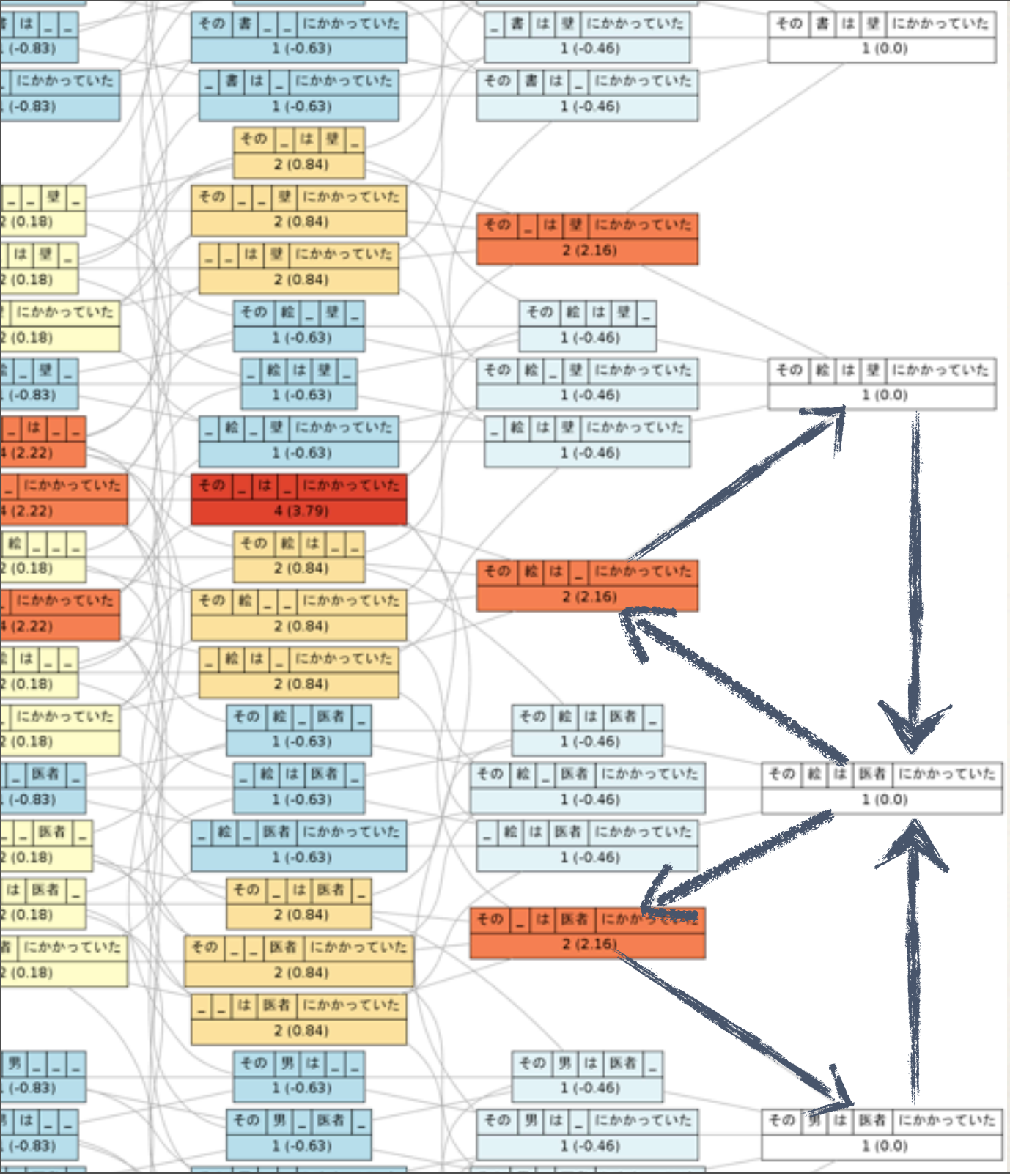
- (5)
  a. (その, 絵, は, 壁, にかかっていた)
  b. (その, 男, は, 医者, にかかっていた)
- (6)
  a. (その, 絵, は, 医者, にかかっていた)
  b. (その, 男, は, 壁, にかかっていた)
- (5′)
  a. (その, 書, は, 壁, にかかっていた)
  b. (その, コート, は, 壁, にかかっていた)
  c. (その, 女, は, 医者, にかかっていた)
  d. (その, 子供, は, 医者, にかかっていた)
- 重要な点
  - (5′a, b) は超語彙的パターン “その_は壁にかかっていた” で仲介された (5a) の “友人”
  - (5′c, d) は超語彙的パターン “その_医者にかかっていた” で仲介された (5b) の “友人”



- { (5a), (5b), (6a), (5'a) } のパターン束
  - RubyPLB で作成



- { (5a), (5b), (6a), (5'a) } のパターン束
  - RubyPLB で作成
- (6a) の解釈はパターン:
  - (その, 絵, は, _, にかかっていた)
  - (その, _, は, 医者, にかかっていた)
- で参照される具体事例 (i.e., (5a) と (5'a)) の解釈から影響される
  - 一般に超語彙的パターン *P* の事例として参照される類似事例の数が多いほど *P* バイアスは強い



- { (5a), (5b), (6a), (5'a) } のパターン束
  - RubyPLB で作成
- (6a) の解釈はパターン:
  - (その, 絵, は, _, にかかっていた)
  - (その, _, は, 医者, にかかっていた)
- で参照される具体事例 (i.e., (5a) と (5'a)) の解釈から影響される
  - 一般に超語彙的パターン P の事例として参照される類似事例の数が多いほど P バイアスは強い

PSECの特徴

- PSEC は入力の “解釈” ≈ “意味表示” を明示的に与えない
- 従って，PSEC が与えるのは “解釈” というより文意の類義性のクラス，あるいは“疑似” 解釈
  - これが良いことか悪いことか，目的による
- 私の立場
  - 本当にコトバの意味が何であるかを知ることができるかは，よくわからない

PSECの長所と短所

- 長所
  - 入力が文法的に適格でなくてよい
    - 入力は断片でよいのでMoving Window 方式で意味解析が実行できる
  - 処理がアナロジー基盤 (佐藤 1997; Skousen 1989) なので “辞書” や “文法” が必要最低限
  - 構文効果 (Goldberg 1995) やブレンド効果 (Fauconnier 1997) を自然に記述する
- 短所
  - 絶対記憶ベースなので計算資源を猛烈に喰う
    - NLP研究者の皆さんが計算アルゴリムを工夫してくれたら，もっと現実的に動くかも
  - 文節のための次善知識が必要
    - 最適な文節を自動認識できない限りは PSECは実用的ではない

並列疑似エラー補正の応用例

応用例

- 誤用の補正
  - PSECの本来の用途だが，論文では言及していない
- 明白な未知語の意味の推定
  - 本発表では割愛
- パロディー認識 (土屋 2010, 2011)
  - パロディー認識は類義性認識の特殊な場合
    - 同義性認識 (乾 2007) は類義性認識の特殊な場合

パロディー認識

- (14) の四つのコトワザのパロディー

  a. 学問に王道なし

  b. 死人に口なし

  c. (X の) 看板に偽りなし

  d. 触らぬ神に祟りなし

(14a)のパロディー

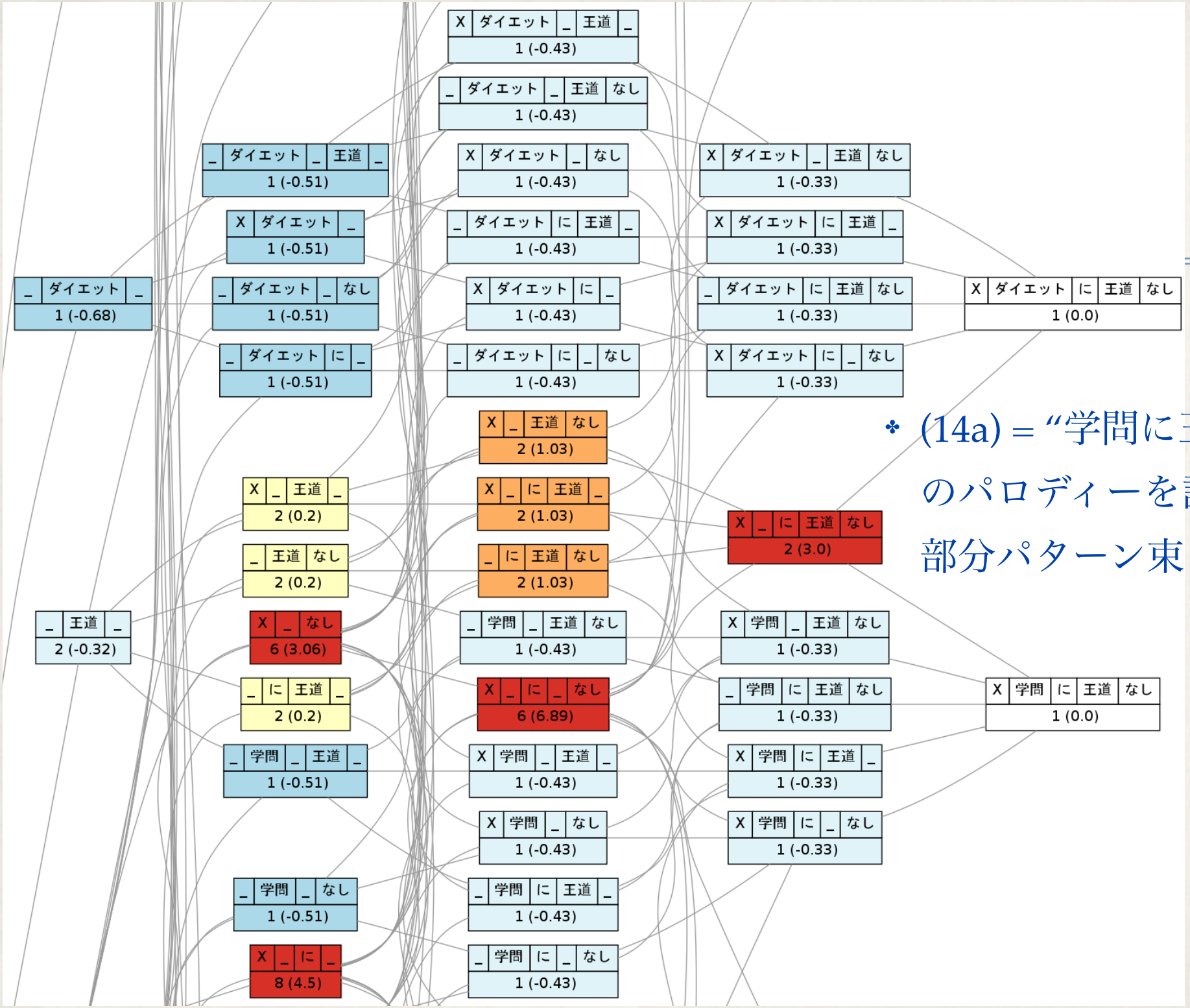
- (15)

  a. {i. ダイエット; ii. 外国語学習; iii. 相場; vi. 婚活; v. 脆弱性対策; vi. 物件選び} に王道なし

  b. 学問に {i. 近道; ii. 横道; iii. 国境} なし

  c. 学問に抜け道あり

- 原典表現との対応づけを実現するパターン
  - (15a) は (_, に, 王道, なし)
  - (15b) は (学問, に, _, なし) か (学問, に, _道, なし)
  - (15c) は (学問, に, _ , _)



- (14a) = “学問に王道なし” のパロディーを記述する部分パターン束

(14b)のパロディー

- (16)

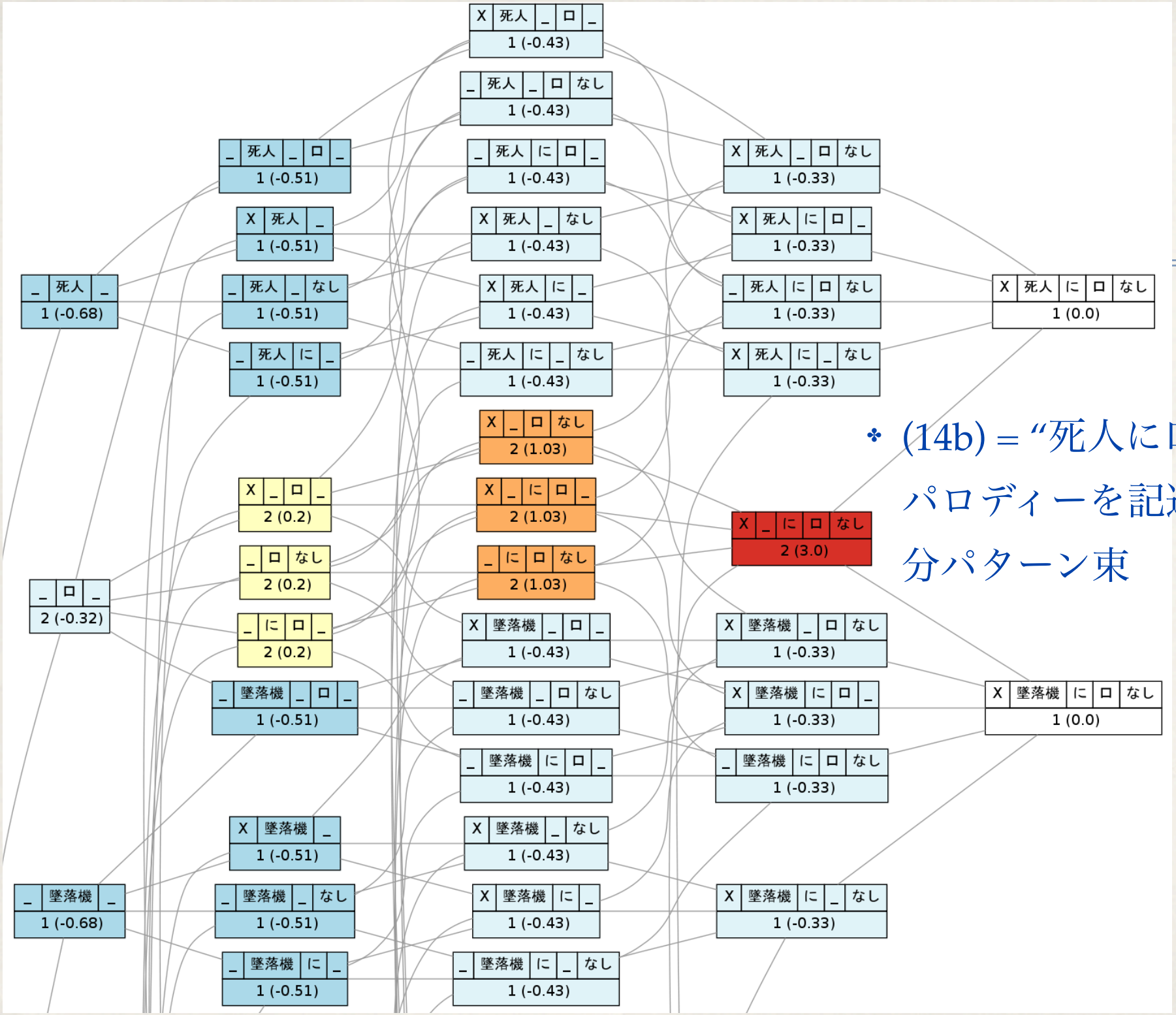
  a. {i. 主任; ii. 模型; iii. 墜落機} に口なし

  b. 死者に口なし

- 原典との対応づけを実現するパターン

  - (16a, b) はいずれも (_, に, 口, なし) ☜ 1st level

    - (_, 死人, に, 口, _) ☜ 2nd levelの事例は見つからず



- (14b) = “死人に口なし”のパロディーを記述する部分パターン束

(14c)のパロディー

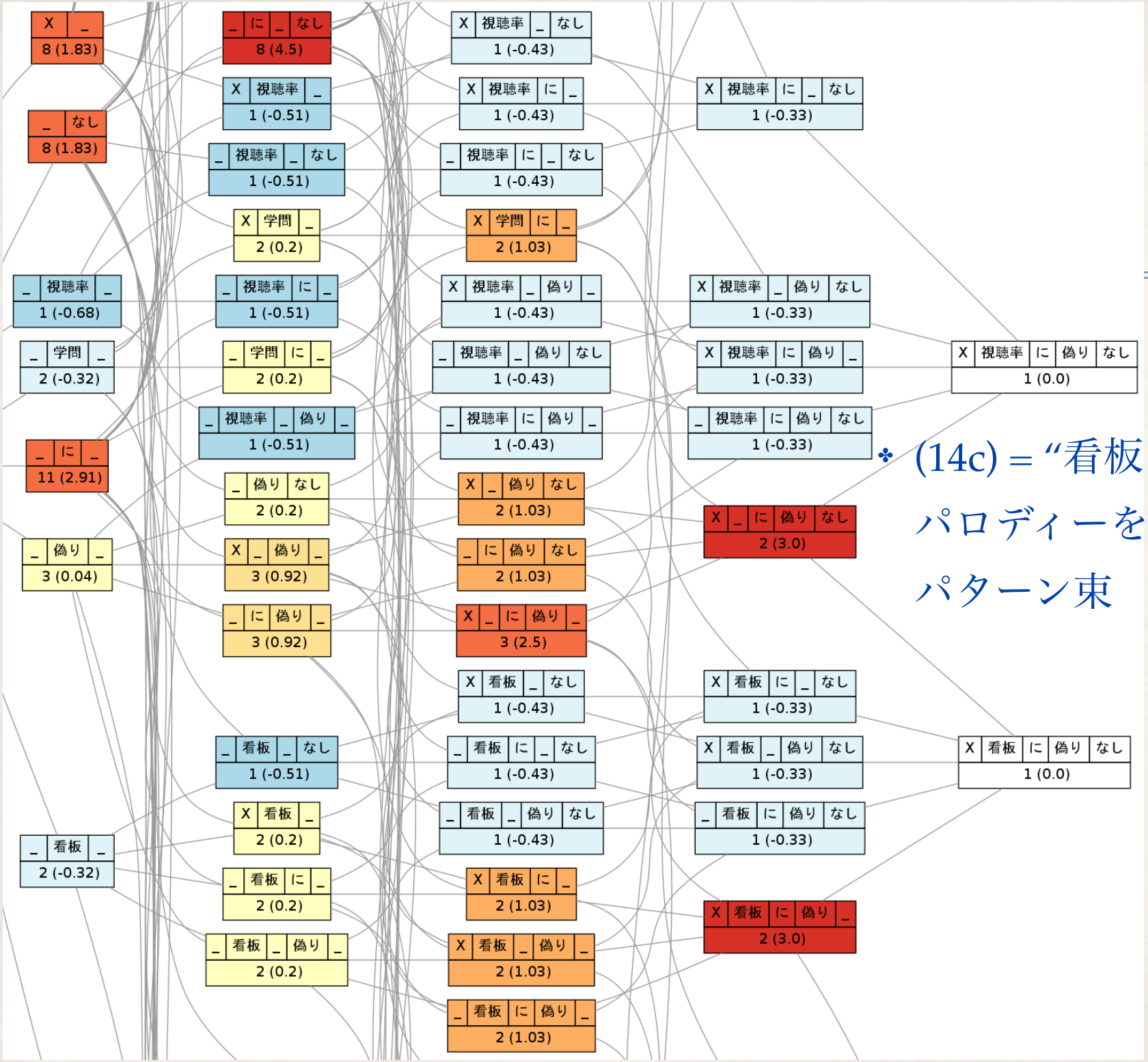
- (17)

  a. {i. 評判; ii. 模型視聴率; iii. キャッチコピー} に偽りなし

  b. 看板に偽りあり

- 原典との対応づけを実現するパターン

  - (17a) は (_, に, 偽り, なし) ☜ 1st level
  - (17b) は (看板, に, 偽り, _) ☜ 1st level



- (14c) = “看板に偽りなし”のパロディーを記述する部分パターン束

(14d)のパロディー

- (18)

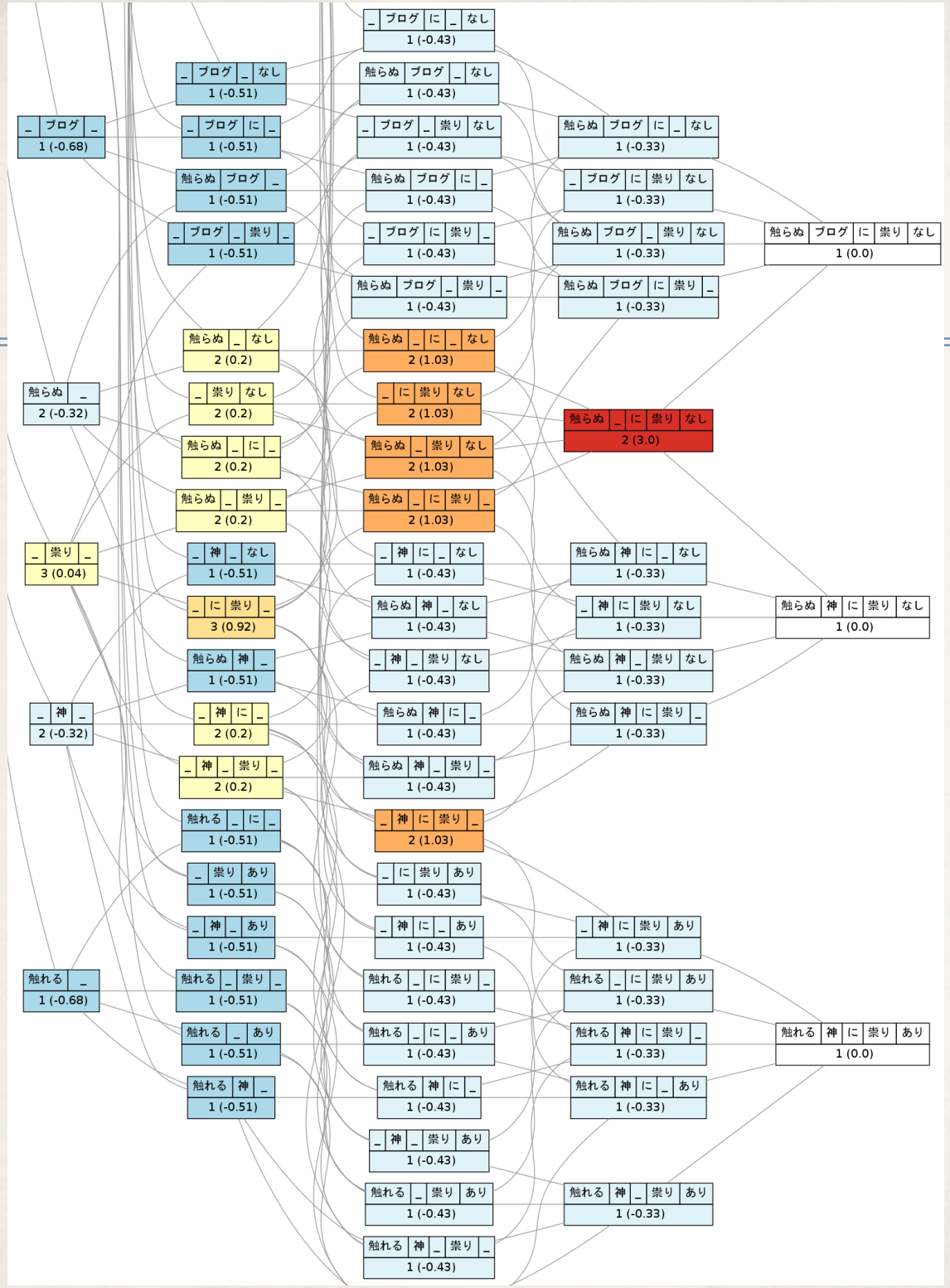
  a. 触らぬ {i. ブログ; ii. 姑; iii. クレーマー} に祟りなし

  b. 下らぬ株に祟りなし

  c. 触れる神に祟りあり

- 原典との対応づけを実現するパターン

  - (18a) は (触らぬ, _, に, 祟り, なし) ☜1st level
  - (18b) は (_, _, に, 祟り, なし) ☜2nd level
  - (18c) は (_, 神, に, 祟り , _) ☜2nd level



(14d) = “触らぬ神に祟りなし” のパロディーを記述する部分パターン束

パロディー生成の意味類型

- 原典 O のパロディー P への編集を対 $e = (o, p)$ で表わす
  - o は原典中の語句で p はパロディー中の語句 (e.g., e = (学問, ダイエット))
- (19)
  a. o と p が同義語 (e.g., (14b) で e = (死人, 死者))
  b. p が o の下位語 (e.g., (14a) で e= (学問, 外国語学習); (14d) で e = (神, クレーマー))
  c. p と o が同類語 (=兄弟語) (e.g., (14b) で e = (看板, 評判), (学問, ダイエット))
  d. p が o の対義語 (e.g., (14b) で e = (王道, 近道); (14c) で e = (なし, あり))
  e. p が o の語呂合わせ (e.g., (14b) で e = (死人, 主任), e= (口, グッチ))

まとめ

論点

- 事実
  - ヒトは驚くべき速度と精度でパロディーを処理できる
- これが意味すること
  - ヒトは事例ベースの言語処理をしている
  - ヒトは類推ベースの言語処理をしている
- パターン束はそのような処理の初歩的なモデル化を提供する

PSECの長所と短所

- 長所
  - 入力が文法的に適格でなくてよい
    - 入力は断片でよいのでMoving Window 方式で意味解析が実行できる
  - 処理がアナロジー基盤 (佐藤 1997; Skousen 1989) なので “辞書” や “文法” が必要最低限
  - 構文効果 (Goldberg 1995) やブレンド効果 (Fauconnier 1997) を自然に記述する
- 短所
  - 絶対記憶ベースなので計算資源を猛烈に喰う
    - NLP研究者の皆さんが計算アルゴリムを工夫してくれたら，もっと現実的に動くかも
  - 文節のための次善知識が必要
    - 最適な文節を自動的に認識することができない限りは PSEC は実用的ではない

References

- Walter Daelemans and Antal van den Bosch (2005). *Memory-based Natu- ral Language Processing*. Cambridge Unversity Press.
- Gilles R. Fauconnier (1997). *Mappings in Thought and Language*. Cambridge University Press.
- 乾 健太郎 (2007). 自然言語処理と言い換え. 日本語学 26(11): 50–59.
- 黒田 航 (2009). パターンのラティス下での疑似並列エラー修復に基 づく文意の構築. In 日本認知科学会第 26 回大会発表論文集, pp. 236–237.
- 黒田 航 and 長谷部 陽一郎 (2009). Pattern Lattice を使った (ヒトの) 言語 知識と処理のモデル化. In 言語処理学会第 15 回大会発表論文集, pp. 670–673.
- 佐藤 理史 (1997). アナロジーによる機械翻訳. 共立出版.
- Royal Skousen (1989). *Analogical Modeling of Language*. Kluwer Aca- demic Publisher.
- 土屋 智行 (2010).言語の創造性の基盤としての定型表現: 慣用句およびことわざの拡張用法の調査 In 認知科学会第25回会大会発表論文集.
- 土屋 智行 (2011). 定型から逸脱した言語表現の分析. In 第 17 回言語処 理学会大会発表論文集.

Thank you for Your Attention

(and Patience)