Introducing *Pattern Lattice Model* as a Form of *Extremely* Usage-Based Model

Kow Kuroda
National Institute of Information and Communications Technology (NICT), Japan

パターン束モデルで“構文”を再考する

(JCLA 11 ワークショップ)

9/28, 2010, 立教大学

二つの主題

- 言語の徹底した (極端な) 用法基盤モデル (Radical) Extremely Usage-Based Model: (R)EUBM) と中途半端な用法基盤モデル (Half-baked Usage-Based Model: HUBM) の対比
  - 黒田 (2007) を参照
- (R)EUBM でのパターン束モデル (Pattern Lattice Model: PLM) の役割
  - 黒田・長谷部 (2009), Kuroda (2009), 吉川 (2010a, b) を参照

発表の概要

- 膨大な事例記憶に基づくヒトの言語処理
  - 徹底した用法基盤モデルの理論的基礎
- スキーマ化とは何か? また何であるべきか? 用法基盤主義を徹底する
  - 徹底した用法基盤主義と中途半端な用法基盤主義の比較
- パターン束モデル (Pattern Lattice Model: PLM) の紹介
  - 徹底した用法基盤主義の効果的な実装法として PLM を紹介
  - パターン束モデルと構文との関係を述べる
- まとめ

膨大な事例記憶に基づく
(ヒトの) 言語処理

完全記憶の仮説とは?

- 完全記憶仮説 (Full Memory Hypothesis) は膨大な事例記憶仮説の特殊形で，具体的には次の仮説:
  - ヒトは自分が知覚したことをすべて覚えていて，新しい入力とそれらの相互作用がヒトの認知活動である
- 注意
  - 覚え (memorizing) と想起=思い出し (remembering/recall) を区別
  - すべてを覚えるが，ほとんどが想起できない状態

膨大な事例記憶の証拠

1. Robert Port の *Rich Phonology* (Port 2005)
2. Jeff Hawkins' *Memory-Prediction Model* (Hawkins and Blakeslee 2004) と *Hierarchical Temporal Memory* (Hawkins and George 2006)
3. *Hyperthymestic Syndrome* (Parker *et al.* 2006, McGaugh 2003)
   - 3 は JCSS27 での黒田 (2010)で扱う

完全記憶仮説の導入の理由

- 膨大な事例記憶に基づく言語処理は，言語の用法基盤モデル (Usage-Based Model: UBM) を基礎づける
- その結果的として得られるのは徹底した用法基盤モデル (Radical (Extremely) Usage-Based Model: R(E)UBM) であって，中途半端な用法基盤モデル (Half-baked Usage-Based Model: HUBM) (e.g., Langacker 1988) ではない
- 論法
  - 膨大な事例記憶に基づく言語処理では，徹底した用法基盤モデルの方が中途半端な用法基盤モデルよりも単純で，それ故に望ましいモデル

スキーマとは何か?
また何であるべきか?
用法基盤主義を徹底する

認知言語学の前提の見直し

- スキーマ化はなぜ起こるのか?

思考実験

- Question 1: 話者が知っている言語表現が (1) ただ一つの時，(2) や (3) というスキーマ化は生じるか?

  (1) *He gave him a new book.*

  (2) NP_1 V NP_2 NP_3

  (3) S V O_1 O_2

- 注意
  - (2), (3) がスキーマ化の産物でないなら，それらが存在する理由は普遍文法ということになる

可能な答え

- Answer 1
  - 知っている事例が (1) のみでも，(おそらくは生得的な) スキーマ化の能力の故に (2) や (3) のスキーマ化が起こる
- Answer 2
  - 知っている事例が (1) のみなら，スキーマ化は起こらない
- Answer 3
  - わからない or 意味のある問題設定ではない

二種類の用法基盤モデル

- 二つの立場
  - 徹底した用法基盤モデルは Answer 2 を採るが
  - 認知言語学で流通している (中途半端な) 用法基盤モデルは Answer 1 (か Answer 3) を採る
- 論点
  - 中途半端な用法基盤主義と徹底した用法基盤主義は，前提を共有せず，同じ説明力をもたない

中途半端な用法基盤モデル

- スキーマ化が起こることの最簡単な説明は次
  - スキーマ化の能力は (生得的な) 一般認知能力の一部
- この説明を採用する中途半端な用法基盤主義では, スキーマ化が起こる理由に関する説明責任が生じる
  - だが, この説明は“スキーマ化はヒトの認知能力の一つである” という認知言語学の理論的前提に埋もれてしまっていて, その妥当性が疑われない

徹底した用法基盤モデル

- スキーマ化が起こらないことの最簡単な説明は次
  - 知っている事例が一個だけなら，スキーマ化は不必要
- 徹底した用法基盤主義でこの説明を採用する場合，スキーマ化が起こる条件の明示化の要請が生じる
  - この論点を以下の議論で掘り下げる

膨大な事例記憶に基づく説明

- 仮説
  - スキーマとは，膨大な事例集合から特定の事例を検索するためのインデクス=検索キーである
- 帰結
  - 事例が一つしかない場合，スキーマは不要

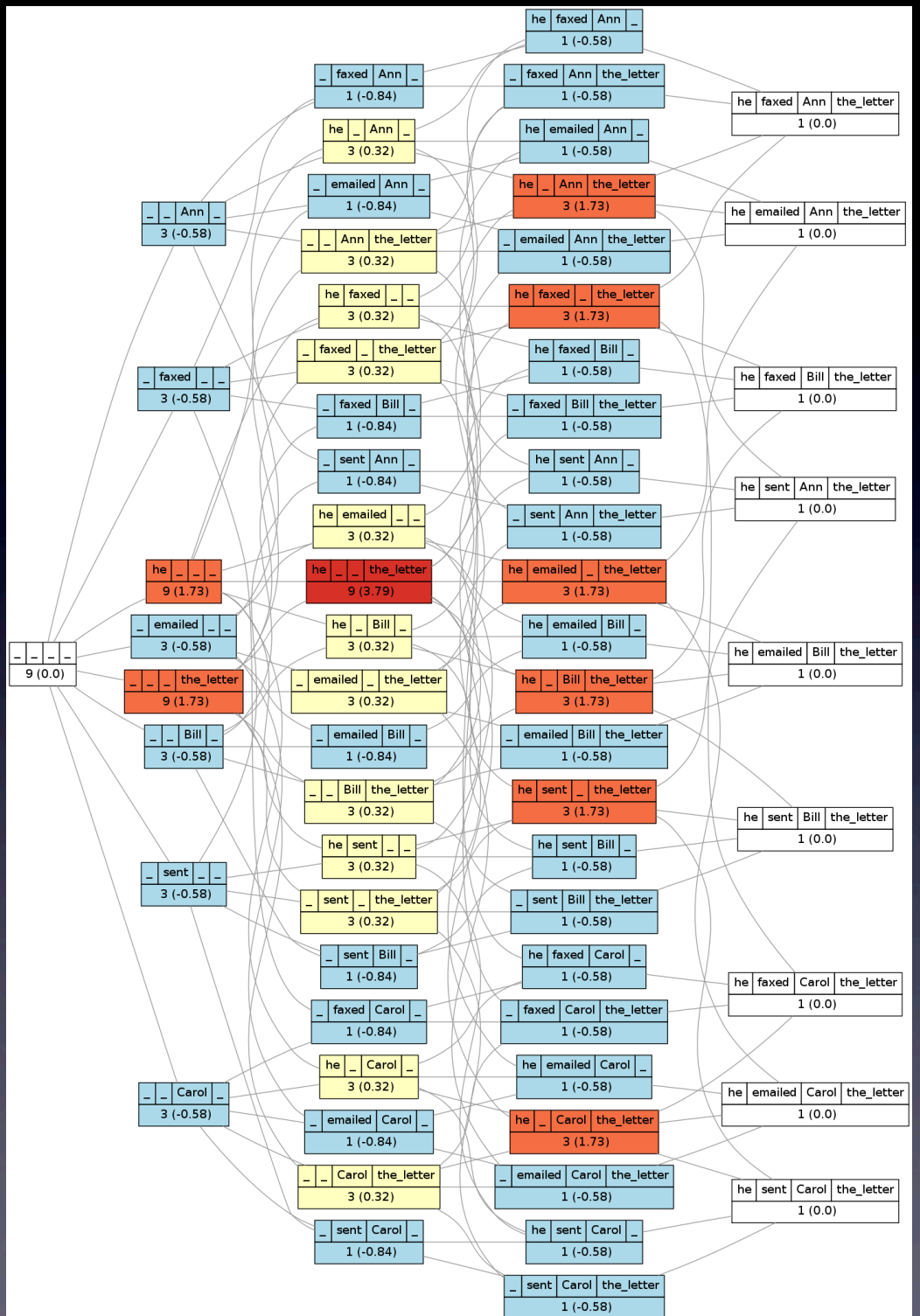
膨大な事例記憶に基づく説明

- 事例数多くなるほどインデクス=検索キーがないと検索が非効率性になる
  - 事例数が少ない場合は総当たり検索でも用が足りる
- ヒトの認知処理の特徴の一つは即応性なので，インデクスがあるのは確実

なぜパターン束モデルか?

- 前提
  - 膨大な事例記憶上の効果的な想起にはインデクスが必要
- 発展的課題
  - どうやって言語の事例記憶のインデクスを定義するか?
- 答え
  - 記憶のタイプを言語記憶に限定した場合，パターン束は，インデクスの有効な記述

パターン束モデルの紹介



パターン束の一例

- 9 個 (文節数=4) の事例集合を入力として Pattern Lattice Builder (PLB) で生成したパターン束
- PLB は http://www.kotonoba.net/rubyfca/pattern で入手可能

例 1

- (1) のパターン束はどうやって生成されるか?

  (1) Ann sent Bill a letter.

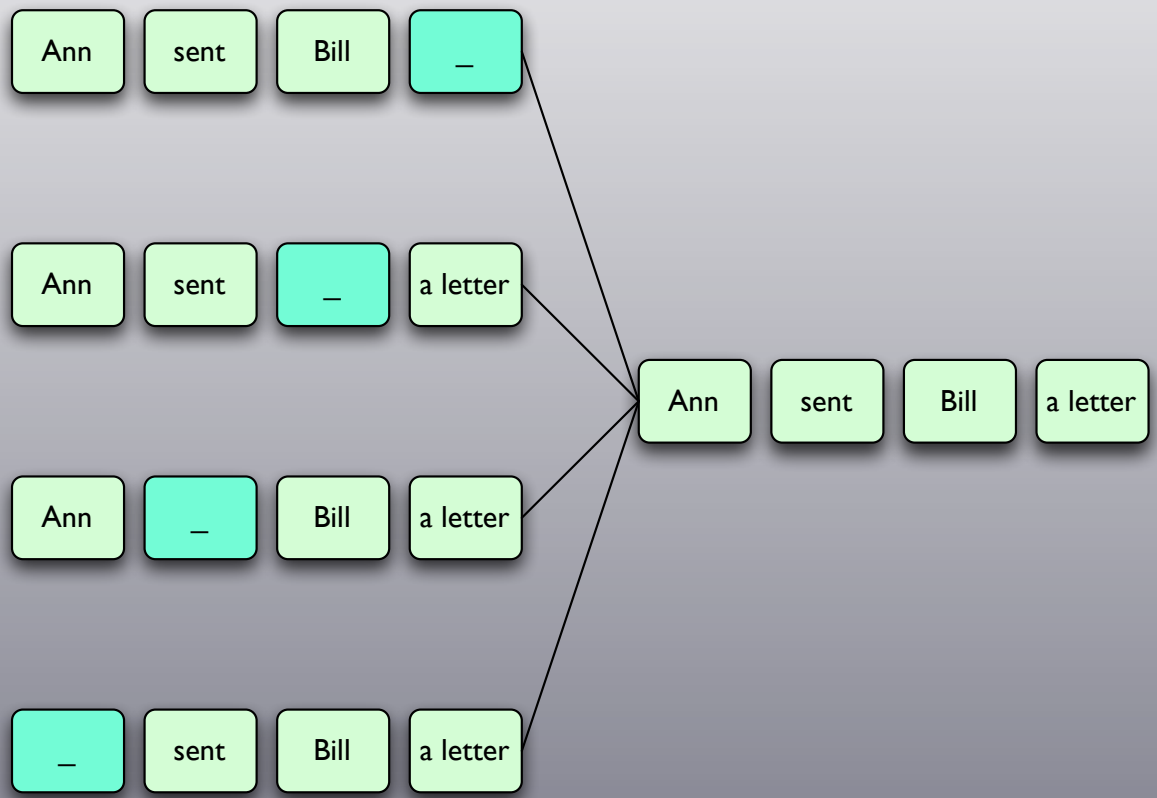
例 I

- (I) が 4 つの文節からなると仮定. これにより次を得る:
  - [Ann, sent, Bill, a letter]
- 留意点:
  - (I) の文節数が4つというのは便宜的な仮定で必然ではない
  - 5つの文節からなる [Ann, sent, Bill, a, letter] を考えてもいい

Step 0: 束の底

Ann sent Bill a letter

Step 1: パターンへの分解 I

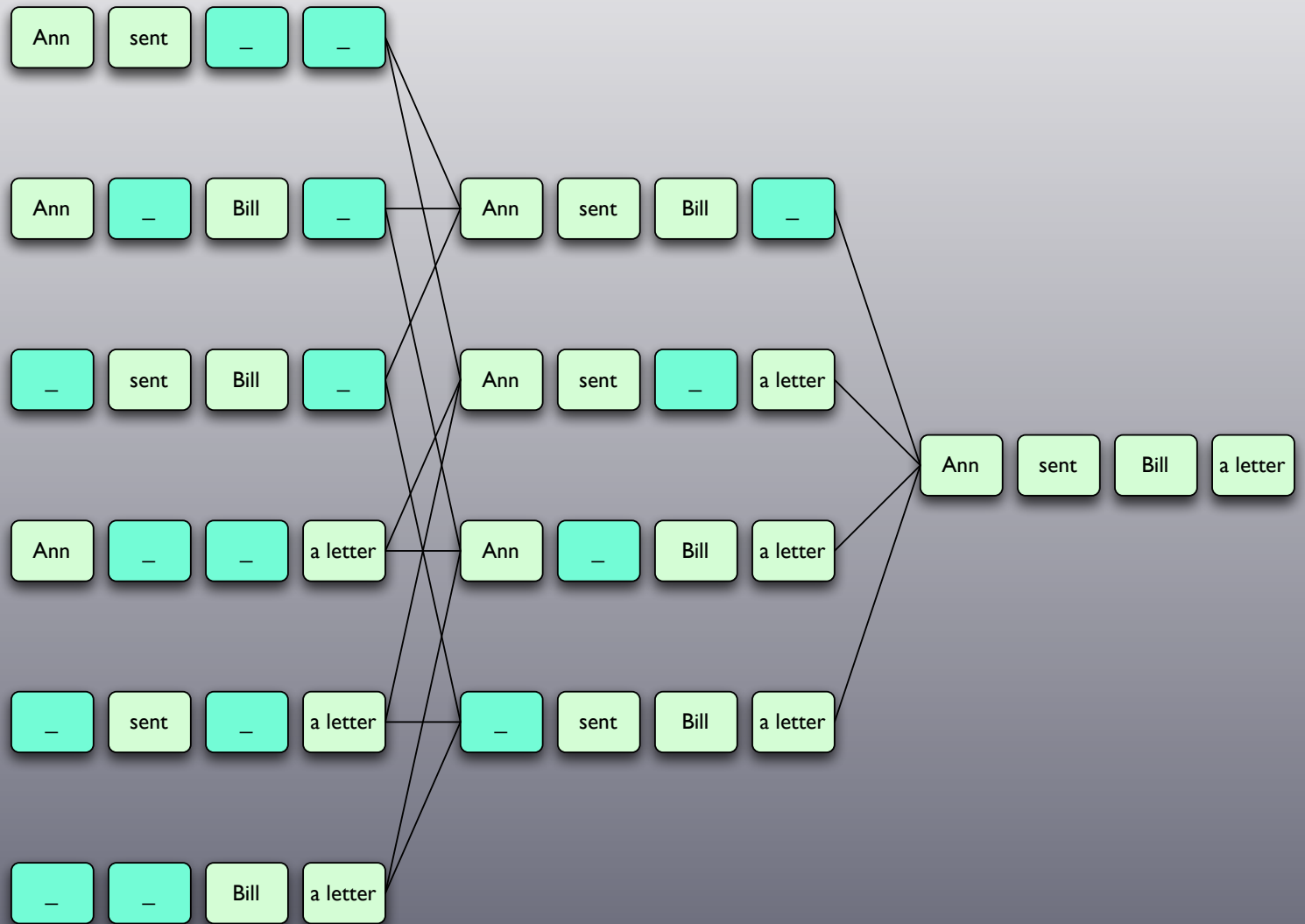


- *k*=4 個の定項の変項化 ("_"への置換) によって *k*=4 個の超語彙的パターン

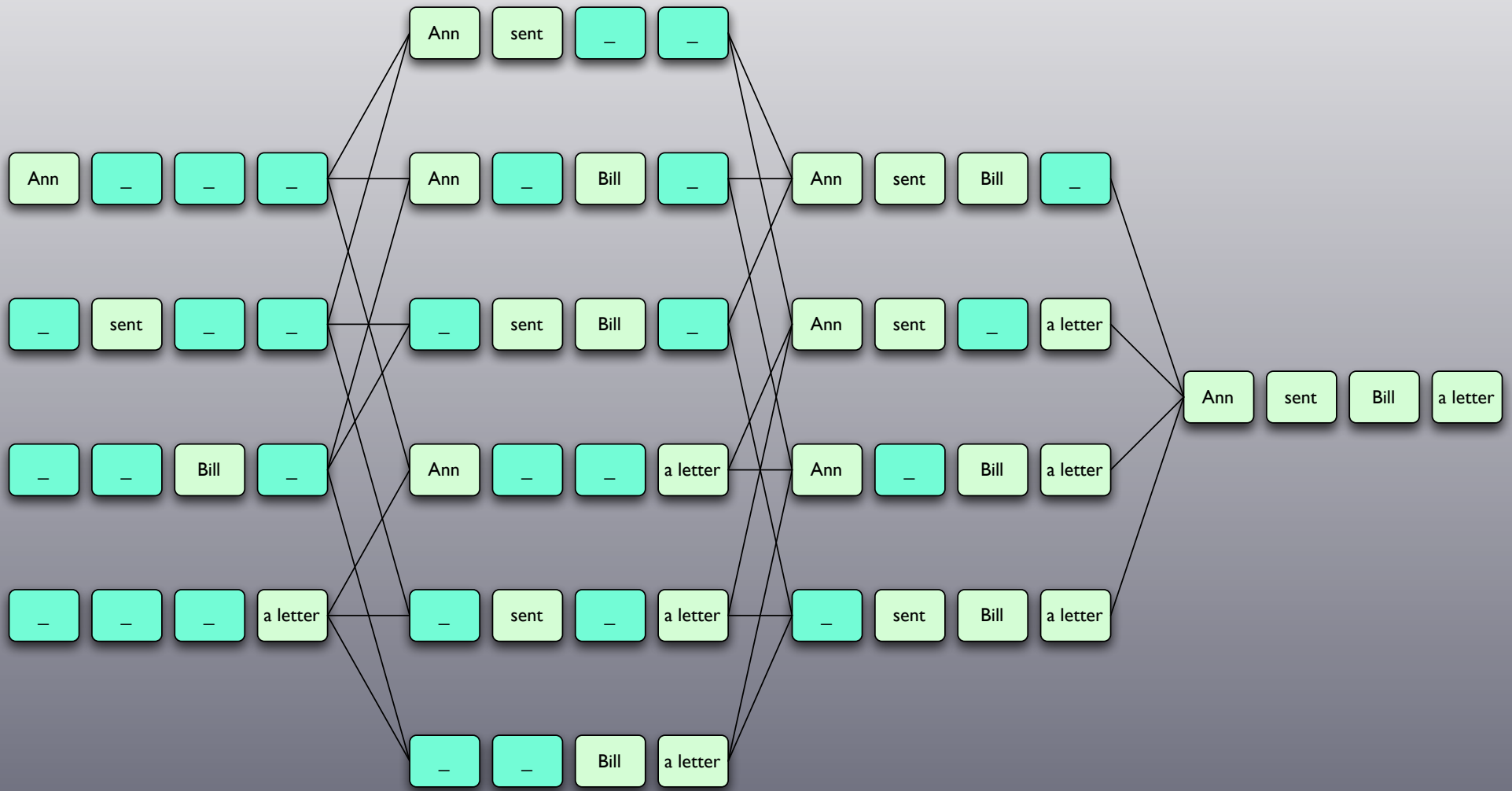
[Ann, sent, Bill, _], [Ann, sent, _, a letter], [Ann, _ Bill, a letter], and [_, sent, Bill, a letter] を生成

- これらは [Ann, sent, Bill, a letter] = (1) の直接の構成要素.

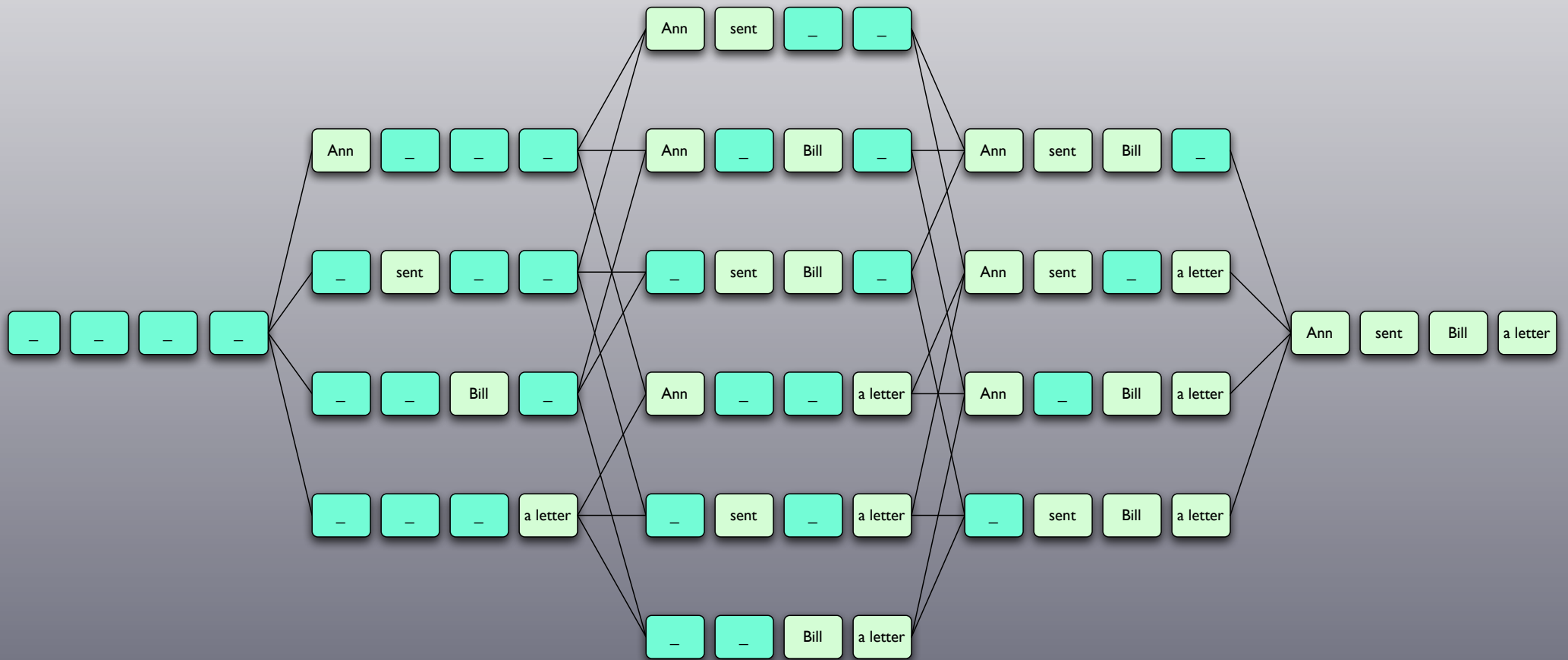
Step 2: パターンへの分解 2



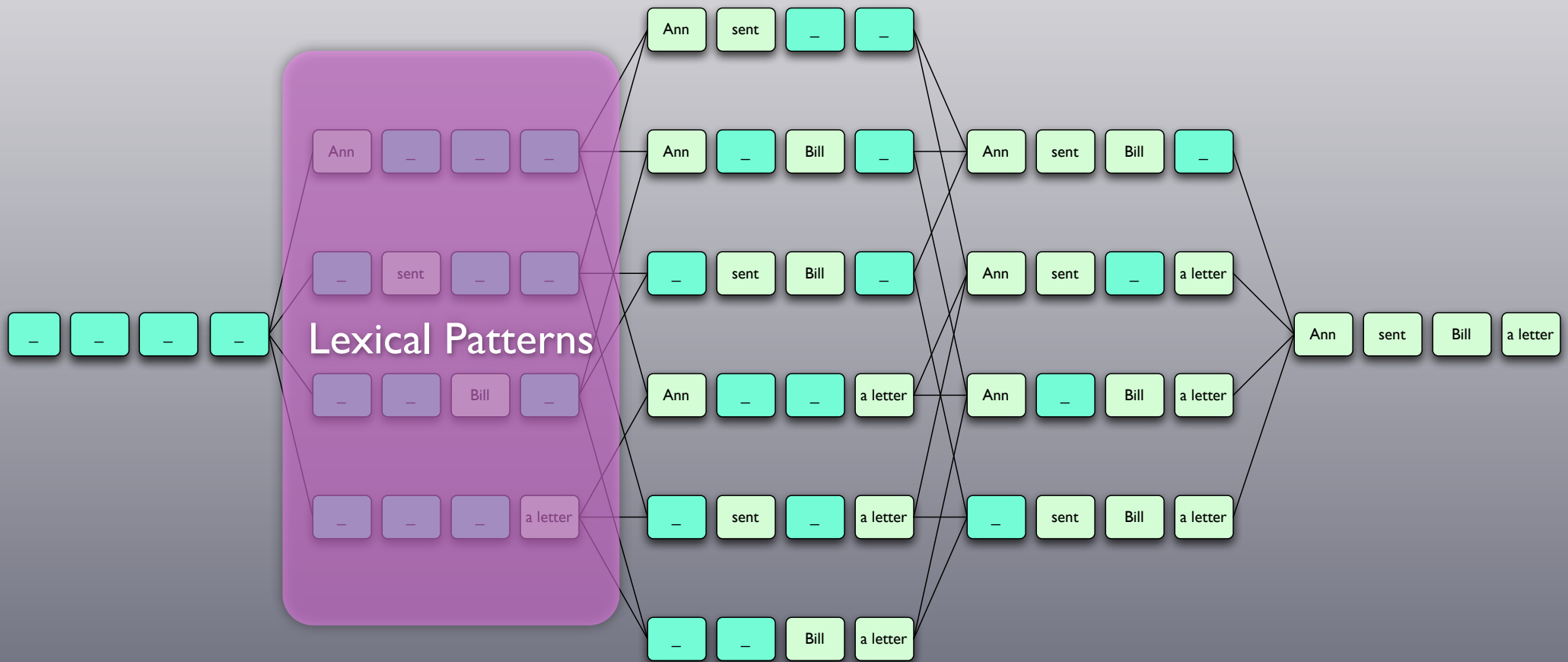
Step 3: パターンへの分解 3



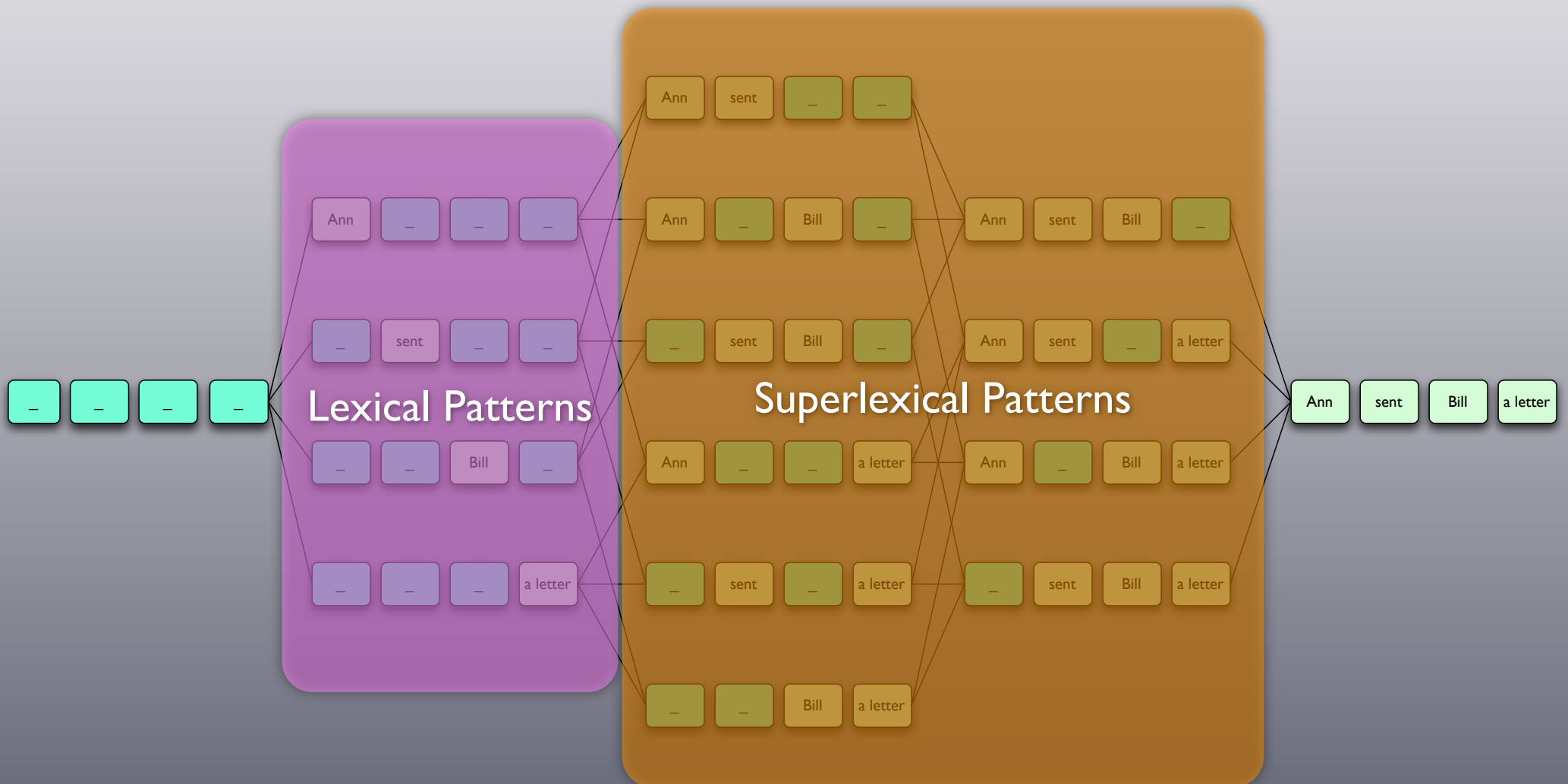
Step 4: (1) のパターン束



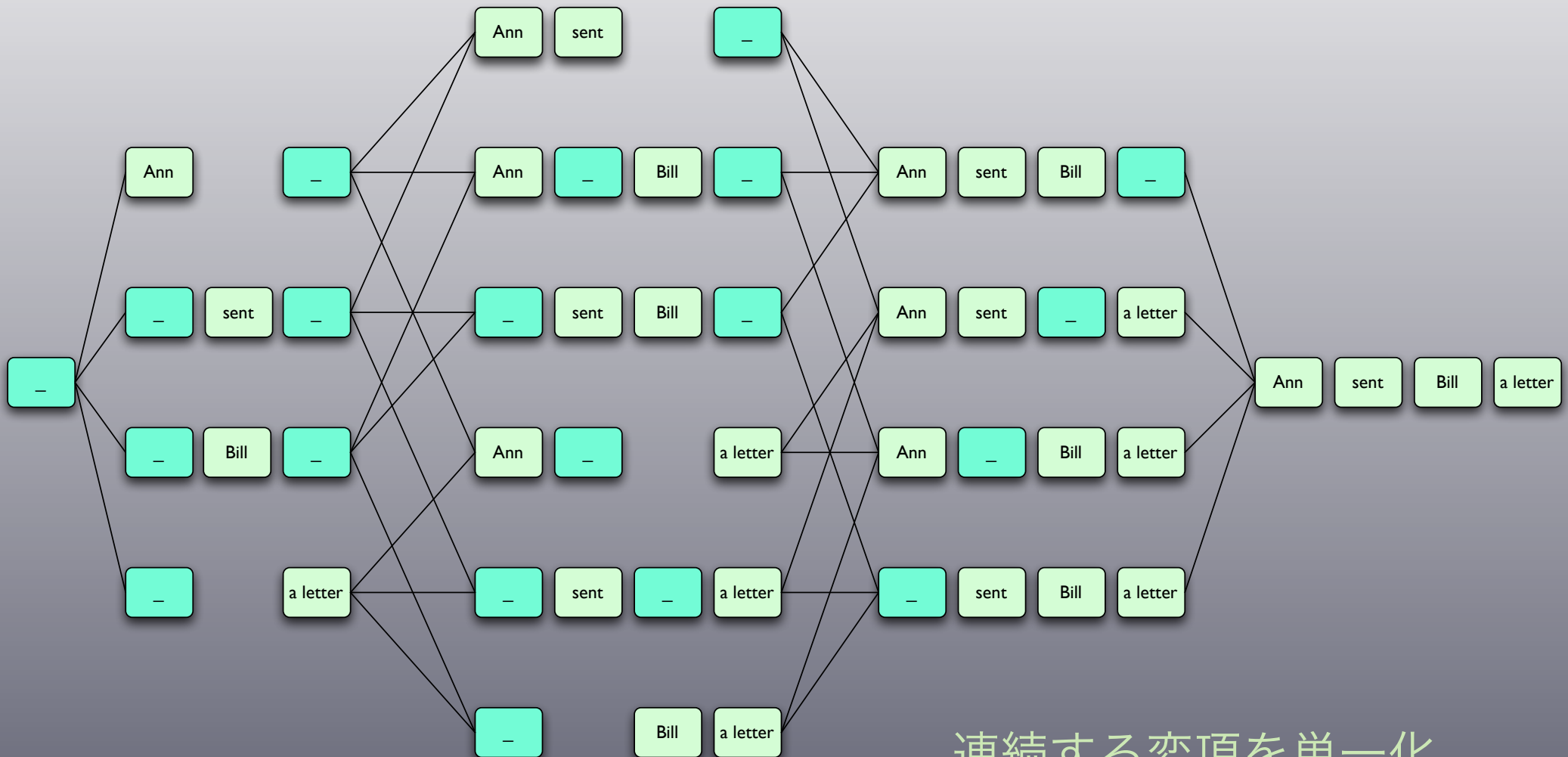
Step 4: (1) のパターン束



Step 4: (1) のパターン束



(I) のパターン束 (簡略版)



例 2

- (1) の他に (2) と (3) があるとする:

  (1) [Ann, sent, Bill, a letter]

  (2) [Ann, faxed, Bill, a letter]

  (3) [Carol, sent, Bill, a letter]

Example 2

- (1) の他に (2) と (3) があるとする:

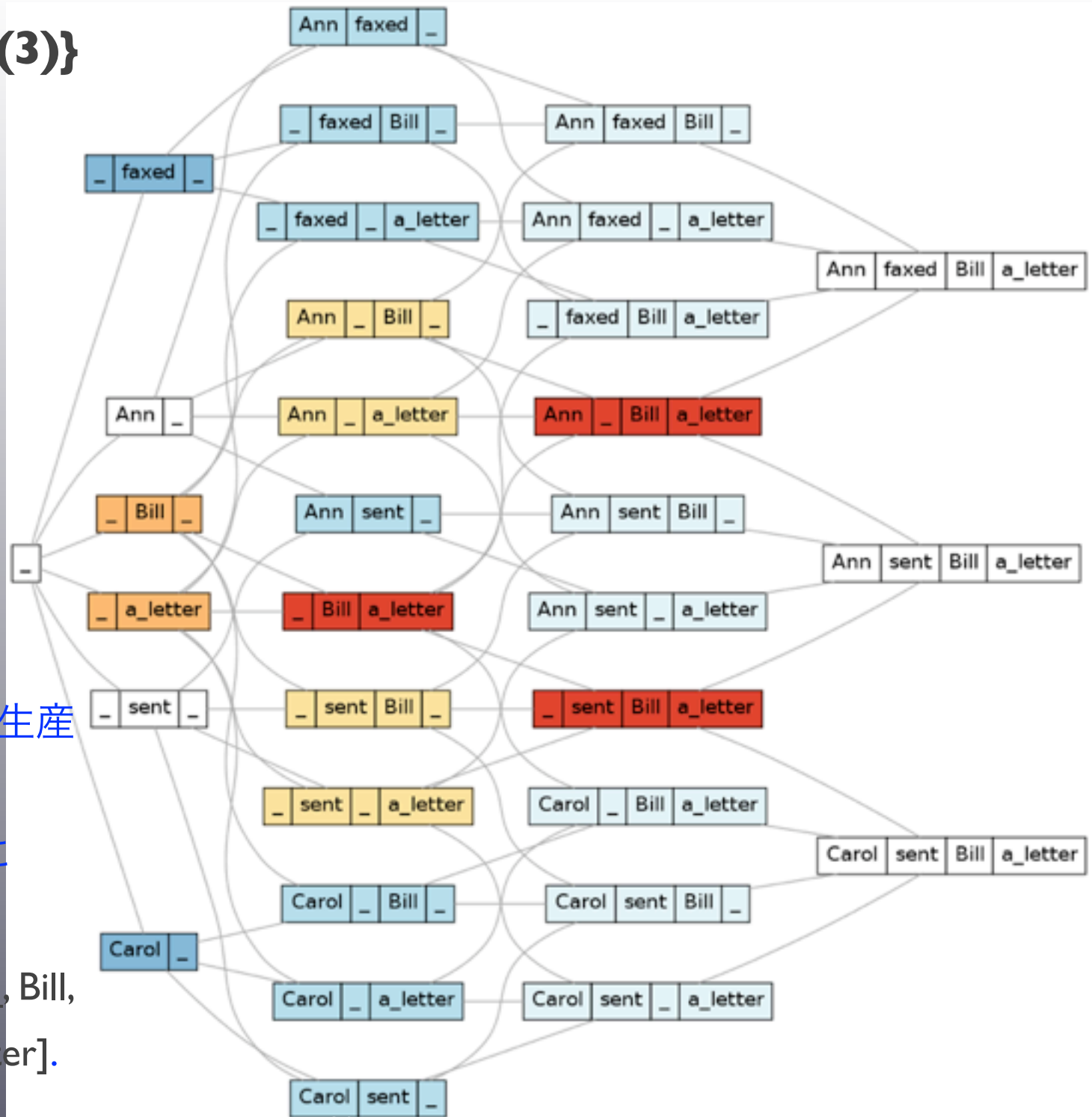
  (1) [Ann, sent, Bill, a letter]

  (2) [Ann, faxed, Bill, a letter]

  (3) [Carol, sent, Bill, a letter]

- Note:
  - Goldberg (1995)では (2) が ditransitive construction で認可される実例

事例集合 {(1), (2), (3)} のパターン束



- 簡易版で表示
- 色温度がランクごとの生産性 (z-score) を表わす
- このPLでは (1) が (2) にもっとも類似した事例
- 仲介スキーマは [Ann, _, Bill, a letter] か [_, _, Bill, a letter].

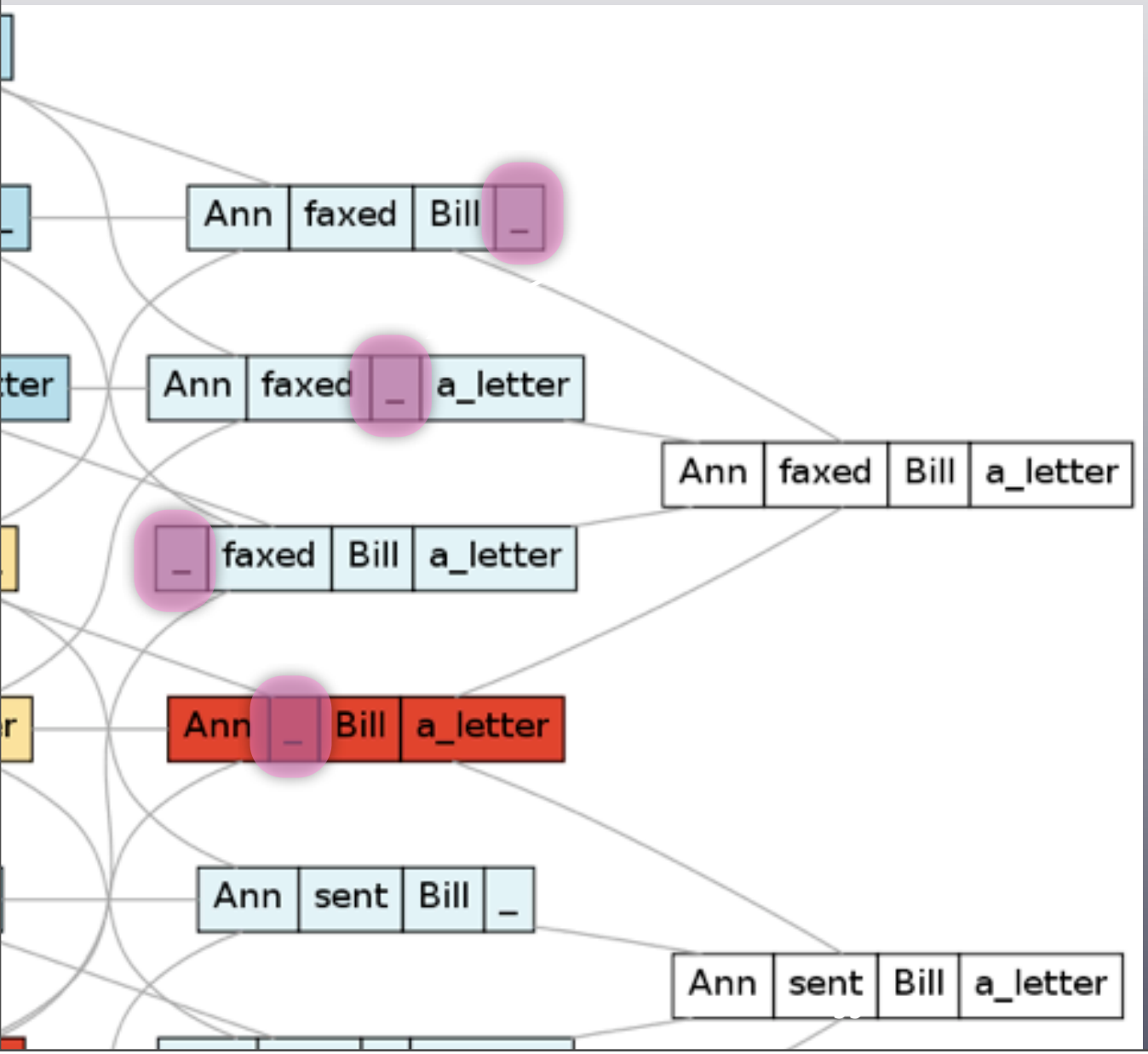
パターン束モデルと意味解釈

- 発展的課題
  - PLM では意味解釈はどうなるのか?
- 関連する問題
  - 構成的意味と非構成的意味の関係はどうなるのか?

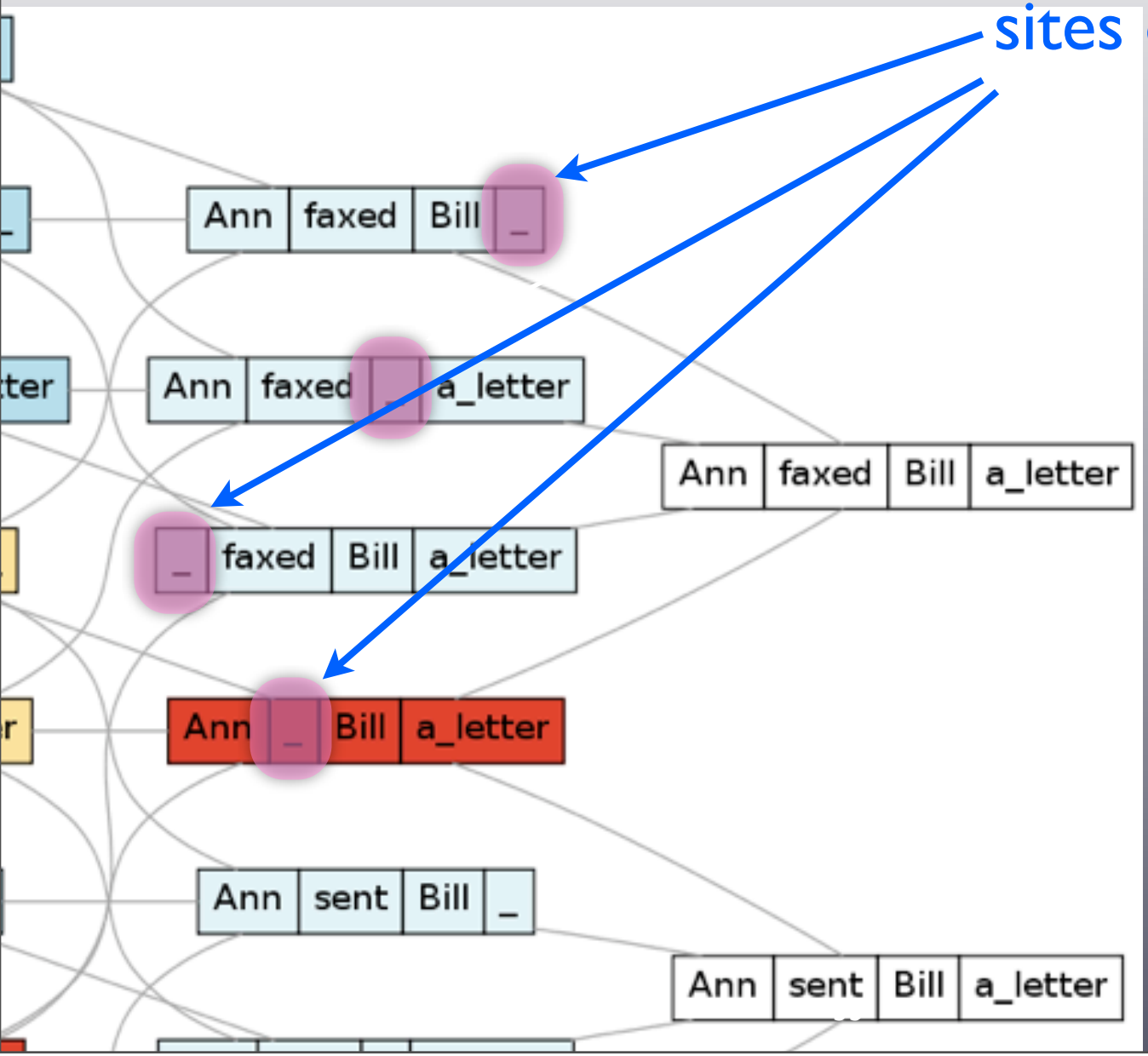
並列疑似エラー補正

- 並列疑似エラー補正 (Parallel Simulated Error Correction: PSEC) は，事例 e = [$u_1, u_1, ..., u_n$] の解釈で，PL(e) を使うアルゴリズム
- 膨大な事例記憶に基づく言語処理の基本原理:
  - e_i の意味解釈は e_i に最類似した事例 e_j の意味の転用によって実現される
- 具体的な動作:
  - PL(E) が全事例 $E = \{e_1, ..., e_n\}$ 間の類似度を表現しているので，PL(E) を使って e_j を探す

UPSEC の動作例


Ann faxed Bill _
ter
Ann faxed _ a_letter
Ann faxed Bill a_letter
_ faxed Bill a_letter
r
Ann _ Bill a_letter
Ann sent Bill _
Ann sent Bill a_letter

UPSEC の動作例


sites of simulated errors
Ann faxed Bill _
Ann faxed _ a_letter
Ann faxed Bill a_letter
_ faxed Bill a_letter
Ann _ Bill a_letter
Ann sent Bill _
Ann sent Bill a_letter

UPSEC の利点

- PL(E) 上の Parallel Simulated Error Correction では
  - 超語彙的パターン (e.g., [Ann, _, Bill, a letter]) の意味は，変項 "_" に生じる期待値を求めれば決まる
  - "より実例に近い情報源にある情報が優先される" という原則から "構文" の効果が説明でき，
  - かつ構成的意味解釈と非構成的意味解釈を分け隔てなく説明できる
- 構成的意味と非構成的意味の違いは優先順位の違いでしかない

PLM と CG の関係

- 構文と構文的意味の存在は PL(E) とその上での PSEC を想定すると，完全に予測可能
- つまり
  - 構文文法 (Construction Grammar) は PLM から派生し
  - 構文の存在は理論的に要請される必要なし

CG の評価 1/2

- 規準 1
  - CG が妥当な理論なら，任意の言語の構文のタイプ $C_1, C_2, ..., C_n$ をすべて，おのおのの事例をすべてを列挙できる
- 規準1は明らかに満足されていないし，される見こみもない
- 規準 2
  - 上が無理でも，少なくともある言語の特定の構文 C_i の事例のすべての事例を列挙できる
- 困ったことに，規準2も満足されていない

CG の評価 2/2

- 評価
  - 経験科学の評価規準では，規準 I が満足されていない“理論” が妥当な理論だと考えるのは無理
  - 少なくとも規準 I が満足されない限り，“CG は人文系の研究者の暇つぶし” と言われても反論の余地なし
- もっと一般的に言うと
  - 本当に記述力と予測力のある (=規準Iを満たす) 言語理論の構築をもっと真剣にやらないと，言語学は関連分野から相手にされないですよ

PLM の場合は?

- 現時点で PLM は規準1を満たす完成度にはないが, 規準2を満足する完成度には達している
  - 本WSの長谷部氏の発表はその実演
- PLM は規準1を満たす理論になる見こみがある
  - 十分な量の言語資料と十分な精度の解析ツール (e.g., Supertagger) が揃えば, それは決して不可能ではない

(現状の) パターン束モデルの難点

- 言語処理モデルとしての難点
  - 完全記憶の仮定という非常に強い仮定に基づいているので，非現実的な面がある
- 統語理論としての難点
  - 現状では構文間の関係 (e.g., 構文交替) がうまく捉えられない
- 構文理論の対抗理論としての難点?
  - 構文の非構文の区別が連続的 (ただし良いか悪いかは別)

まとめ

本発表で示したこと

- 膨大な事例記憶に基づく言語処理と互換性があるのは中途半端な用法基盤モデルではなく，徹底した用法基盤モデルである
- 膨大な事例記憶に基づく言語処理で生じる効率的な事例検索がパターン束モデル (PLM) で実装される
- 構文文法の理論的要請は PLM の理論的帰結である
  - PLM があれば構文文法は不要である

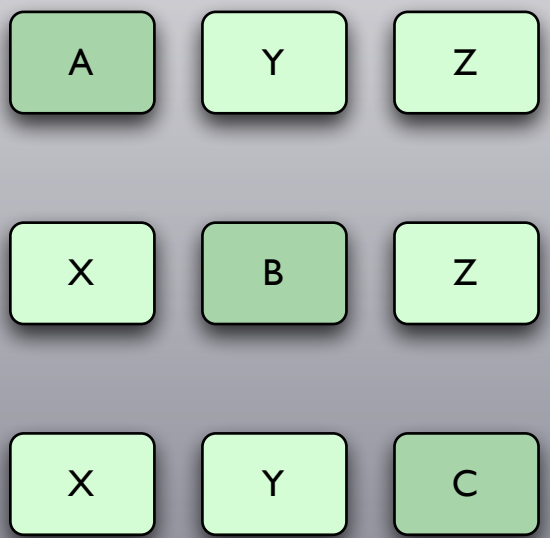
Thank you
for your Attention

パターンの重ね合わせによる合成

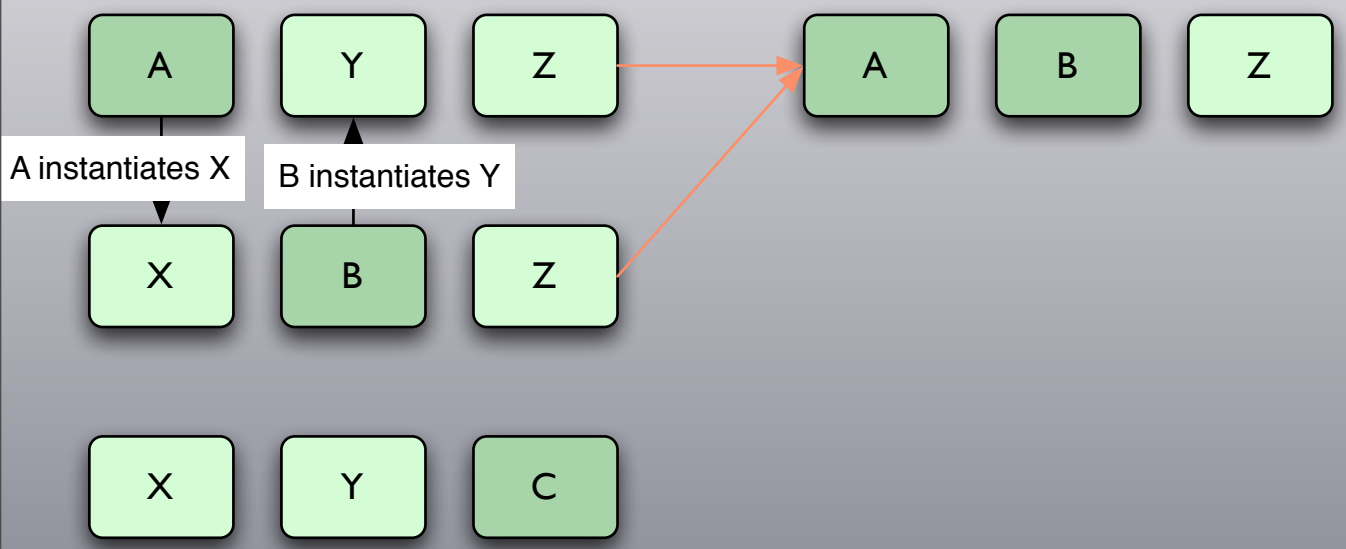
パターンの重ね合わせ

- C は次の条件を満足する時，その時に限り $A = a_1 \cdot a_2 \cdots a_m$ と $B = b_1 \cdot b_2 \cdots b_n$ の重ね合わせである:
  1. A と B が同じ数の文節をもつ ($m = n$: equi-cardinality)
  2. $a_i = b_i$ か instance-of(a_i, b_i) か instance-of(b_i, a_i) がどのi についても成立
- 注意
  - II が成立するなら，上書き修正なしの重ね合わせ
  - II が不成立なら，上書き修正ありの重ね合わせ = Blending

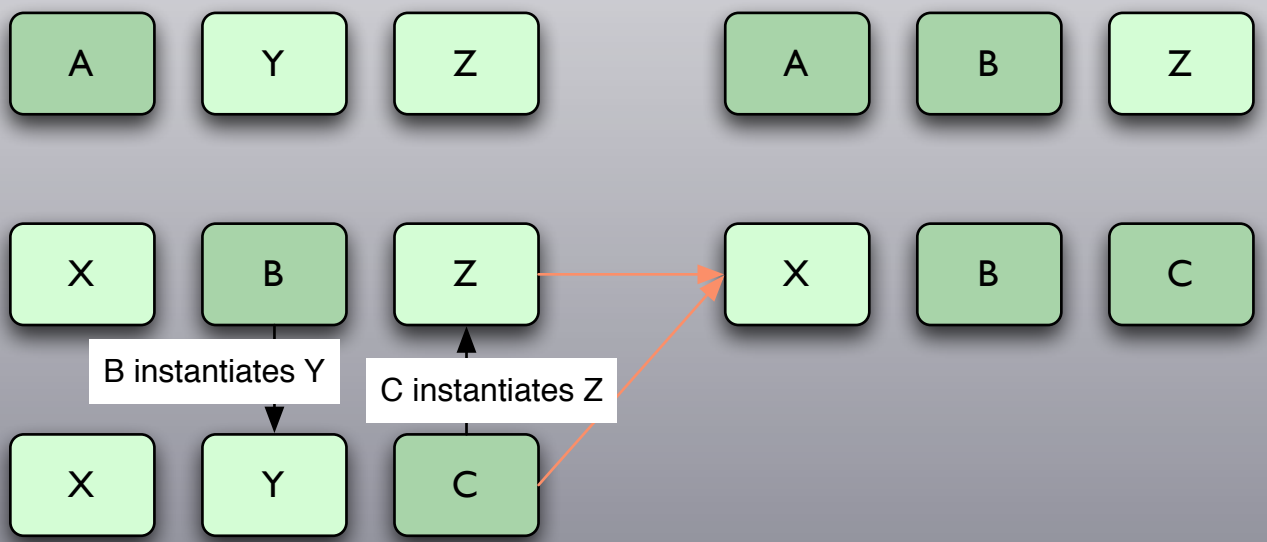
SUPERPOSITION



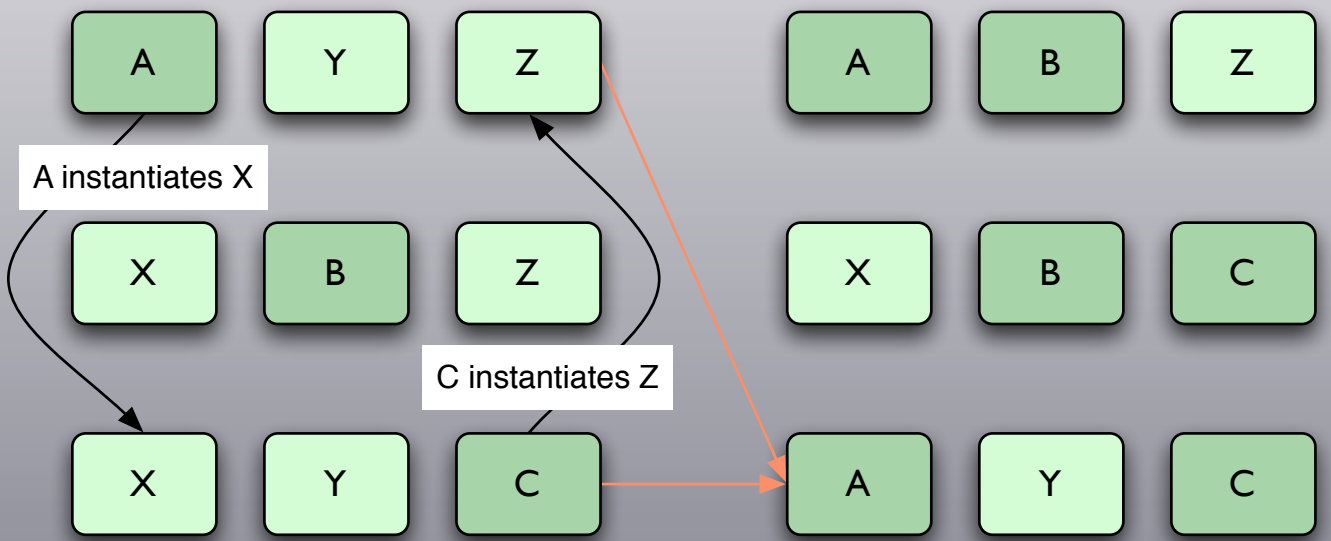
SUPERPOSITION



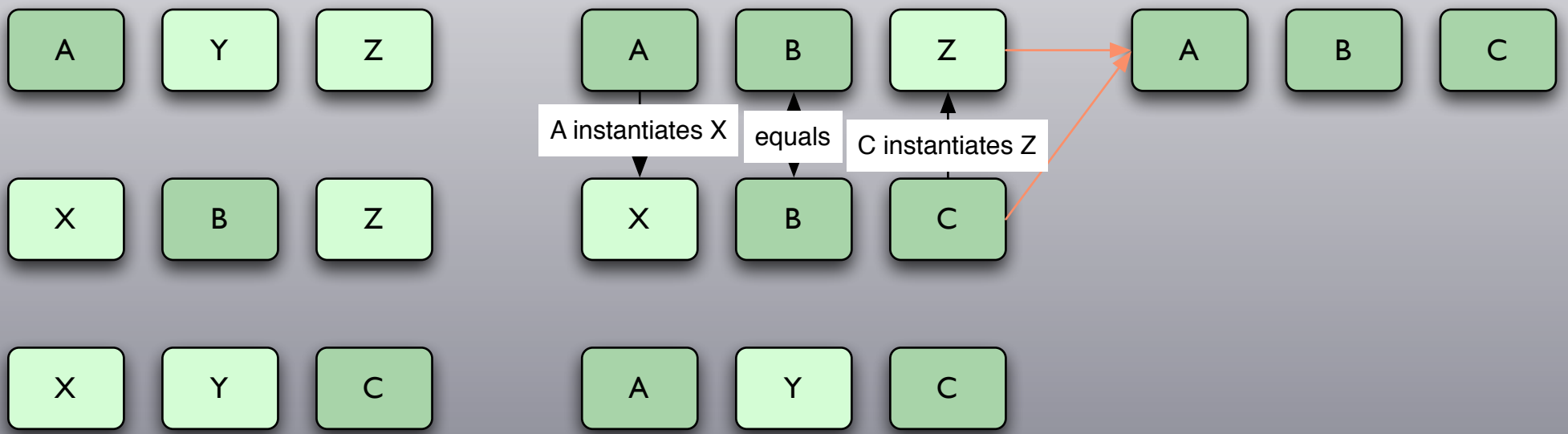
SUPERPOSITION



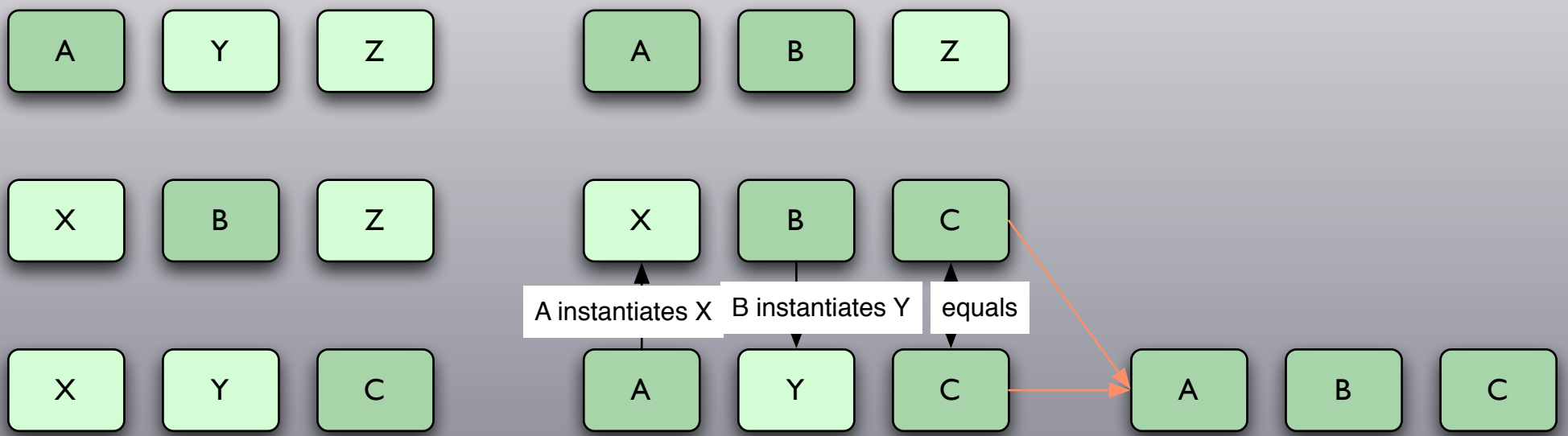
SUPERPOSITION



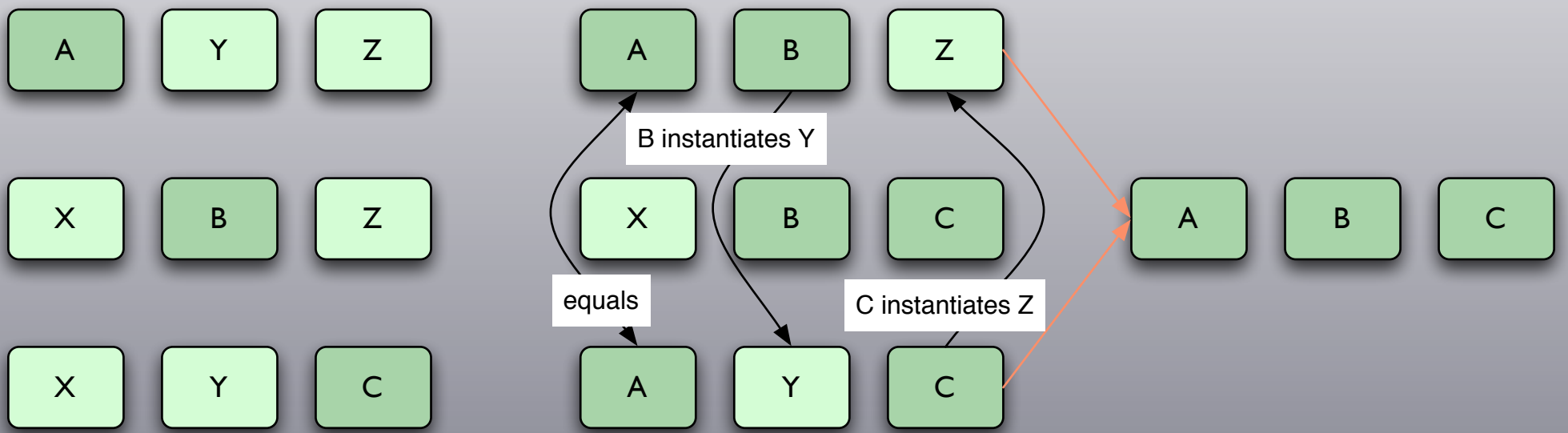
SUPERPOSITION (ResuLt1)



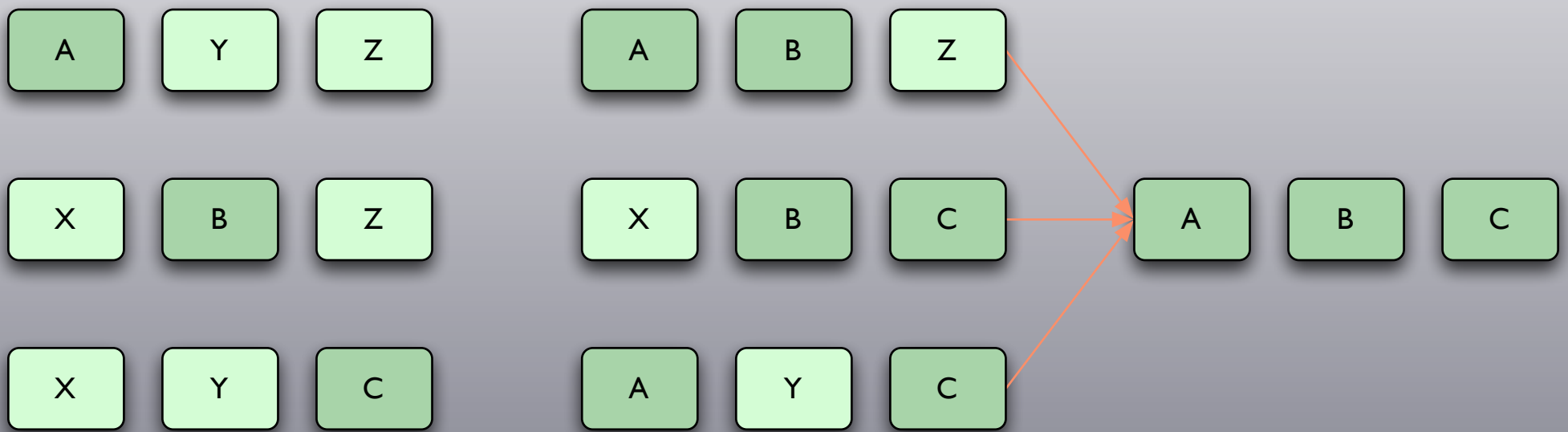
SUPERPOSITION (ResuLt2)



SUPERPOSITION (ResuLt3)



SUPERPOSITION (RESULTS)



Note:

- Results 1, 2 and 3 are not mutually exclusive, and there is no reason to choose one of them.

- In other words, uniqueness of sources is not guaranteed in superposition.