Extremely Usage-based Approach to Syntax

with "Pattern Lattice" Model of Linguistic Knowledge and Performance

Kow KURODA
National Institute of Information and Communication Technologies (NICT), Japan

"Blackship" Lectures at Tokyo University
2010/03/12

Uncomfortable Questions

Uncomfortable Questions

- Why do people (seem to) **find no trouble in understanding noisy, incomplete, and even inconsistent input**?

Uncomfortable Questions

- Why do people (seem to) **find no trouble in understanding noisy, incomplete, and even inconsistent input**?
- Why are **collocations and constructions** so important?

Uncomfortable Questions

- Why do people (seem to) **find no trouble in understanding noisy, incomplete, and even inconsistent input**?
- Why are **collocations and constructions** so important?
- Why is **nonnativeness** of speech and writing **so obvious** to native speakers?

Uncomfortable Questions

- Why do people (seem to) **find no trouble in understanding noisy, incomplete, and even inconsistent input**?
- Why are **collocations and constructions** so important?
- Why is **nonnativeness** of speech and writing **so obvious** to native speakers?
- Why is people's speech **so stereotypical or formulaic**?

Uncomfortable Questions

- Why do people (seem to) **find no trouble in understanding noisy, incomplete, and even inconsistent input**?
- Why are **collocations and constructions** so important?
- Why is **nonnativeness** of speech and writing **so obvious** to native speakers?
- Why is people's speech **so stereotypical or formulaic**?
- Why is the distribution data **so sparse**?

Uncomfortable Questions

- Why do people (seem to) **find no trouble in understanding noisy, incomplete, and even inconsistent input**?
- Why are **collocations and constructions** so important?
- Why is **nonnativeness** of speech and writing **so obvious** to native speakers?
- Why is people's speech **so stereotypical or formulaic**?
- Why is the distribution data **so sparse**?
- Why does **example-based machine translation work**?

Uncomfortable Questions

- Why do people (seem to) **find no trouble in understanding noisy, incomplete, and even inconsistent input**?
- Why are **collocations and constructions** so important?
- Why is **nonnativeness** of speech and writing **so obvious** to native speakers?
- Why is people's speech **so stereotypical or formulaic**?
- Why is the distribution data **so sparse**?
- Why does **example-based machine translation work**?
- Why does **statistics matter** after all?

Burning Question

- These questions seem to boil down to a single question:
  - **How does human linguistic memory work?**
- The present work addresses a specific question (after Port 2007):
  - **What if human has all instances of linguistic expressions stored in vast (implicit) memory** (and virtually no expressions are generated in recognition process)?
- Caveat:
  - This work addresses only questions about comprehension, and will not discuss production.

Strategy

- I know this idea is **crazy**, and completely against the traditional wisdom of (theoretical) linguistics after Chomskian revolution.
- Yet **I take an extreme position** in my theorizing
  - with concerns of:
    - making it easier to draw nontrivial conclusions, and
    - making predictions easier to falsify.
  - and from awareness that **human memories are** (**still**) **far from well understood**.

Remark on Human Memory

- I assume the distinguish between two components/subsystems of human memory, i.e.:
  - **storage** of records and
  - **remembering/recall/retrieval** of stored records.
- There is a striking asymmetry between the two:
  - **Severe limitations on** (**explicit**) **remembering**,
    - especially constraints on working memories (Miller's (1956) magical number 7±2)
  - Virtually **no limits on storage**: this is a suggestion from recent findings in hyperthyemestic syndrome (Parker et al. 2006)

Outline

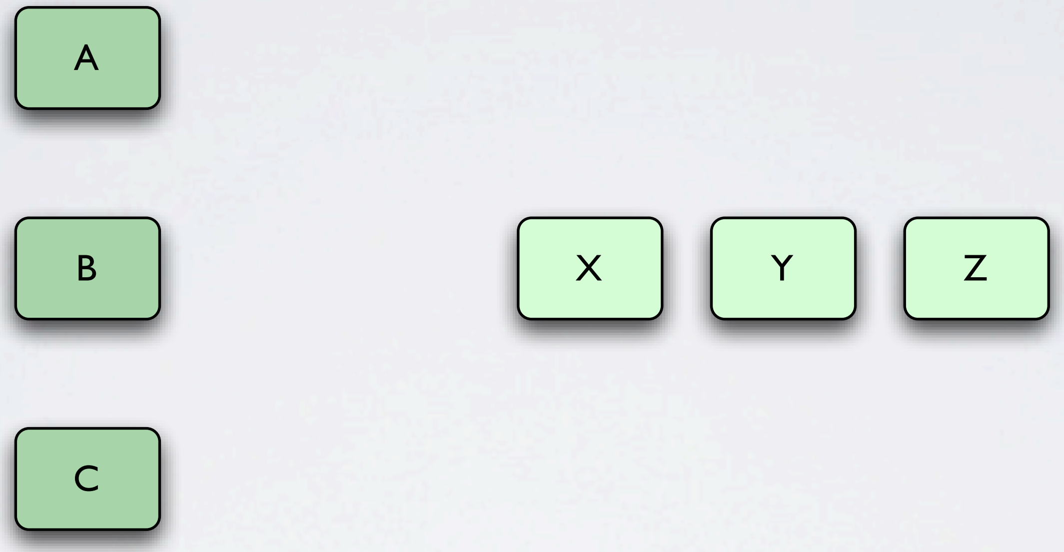
- **Superposition of patterns can implement composition**.
  - **Pattern Lattice** (PL) of a set of expressions defines a **hierarchy of superlxical patterns** used in superposition
- **Simulated Parallel Error-Correction** (SPEC) under pattern lattice provides a better account of "construction" effects (Goldberg 1995, 2006).
- Pattern lattice model (PLM) allows us to conceive of grammar of language as a **management system** rather than as a **generative system**.
- Conclusions

Why not Superposition?

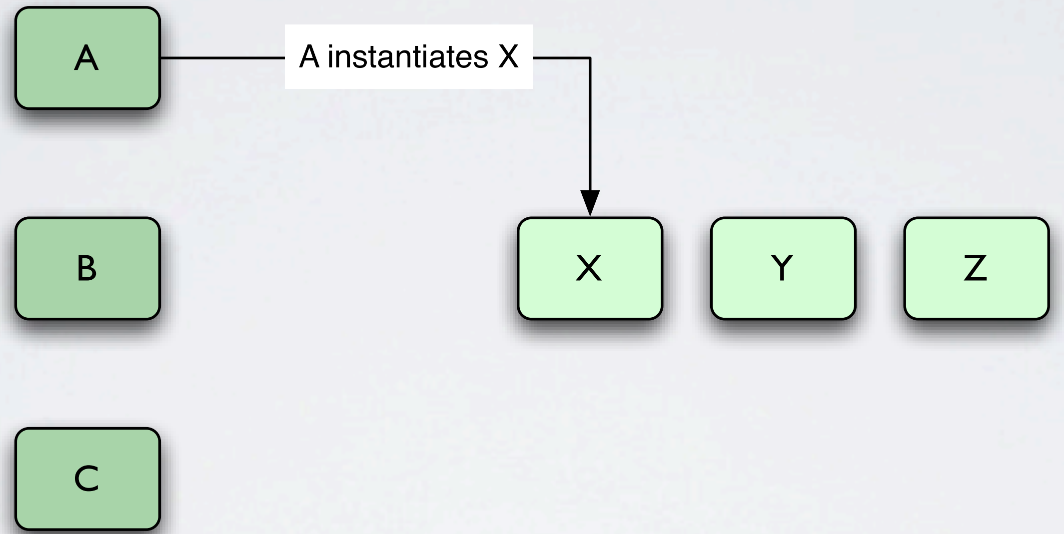
Why Substitution?

- In virtually all linguistic theories, **composition is implemented by substitution**.
- Yet **there is no conceptual necessity for this**: **superposition can do it**, too.
- and it does so with several desirable features.
  - Remark: Conception of composition as substitution **has a long, strong tradition** (e.g. proof theory crucially relies on it (cf. production system (Post 1943)), but **this is a different matter**.

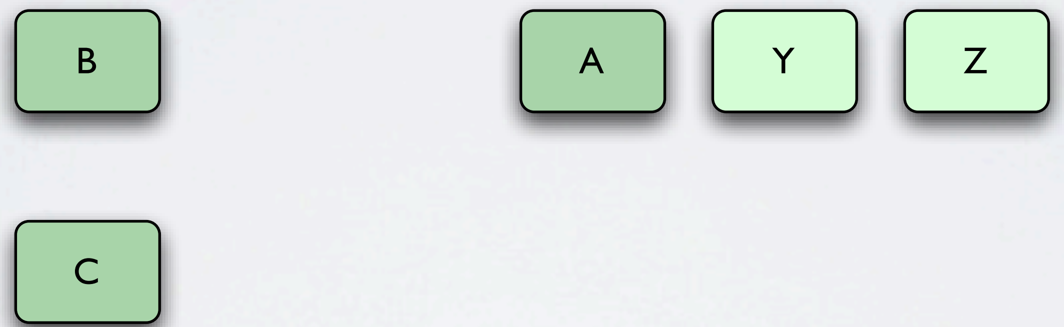
SUBSTITUTION



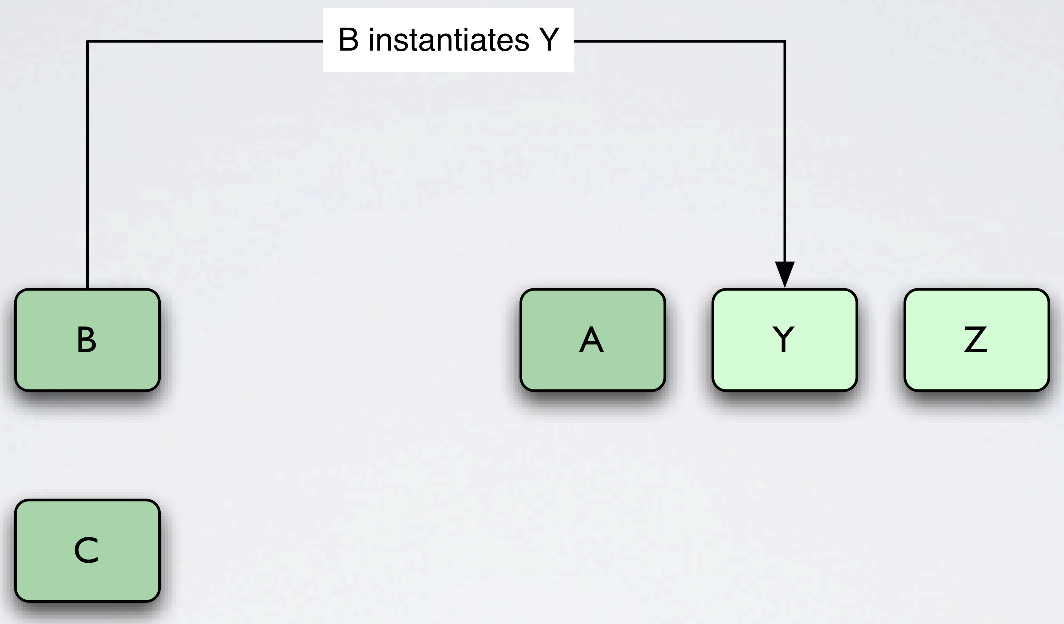
SUBSTITUTION



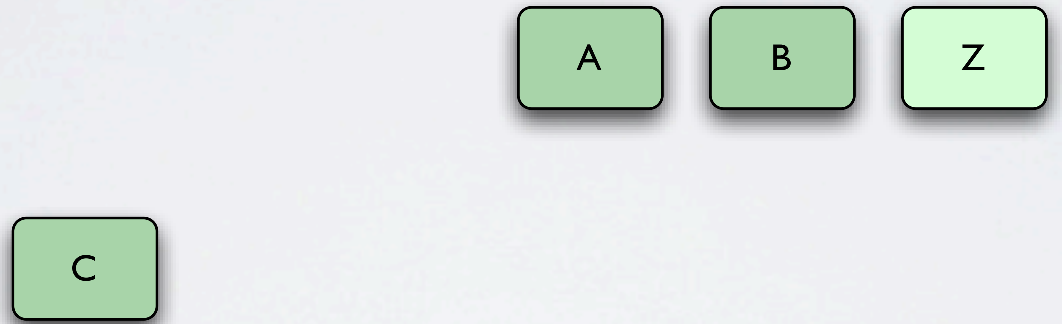
SUBSTITUTION



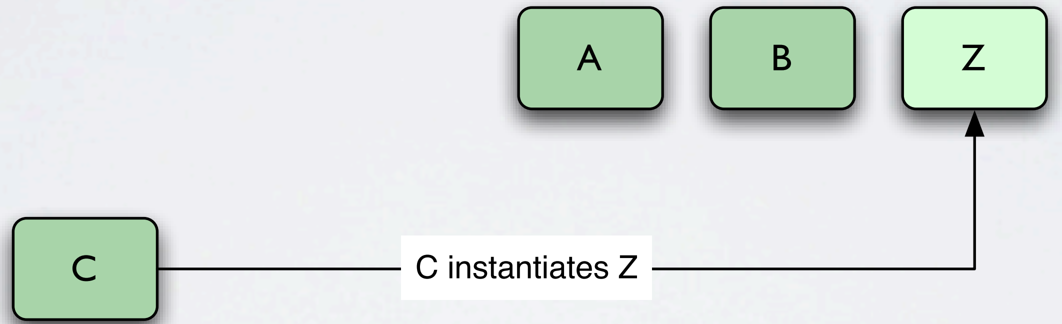
SUBSTITUTION



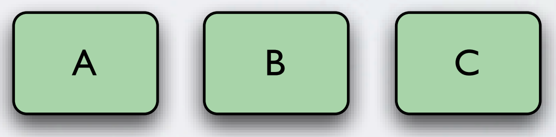
SUBSTITUTION



SUBSTITUTION



SUBSTITUTION (Result)



Desirable Properties

- Representation of items (to be inserted) can be **context-free** and therefore **redundancy-free**.
- Host structures can be defined **freely but systematically** if they are defined by something called "grammar."
  - This can answer the problem of human creativity.
- In essence, substitutional model **guarantees economy and generalization**
  - but only in terms of description load, and if computational time is counted as a resource, the conclusion should be different.

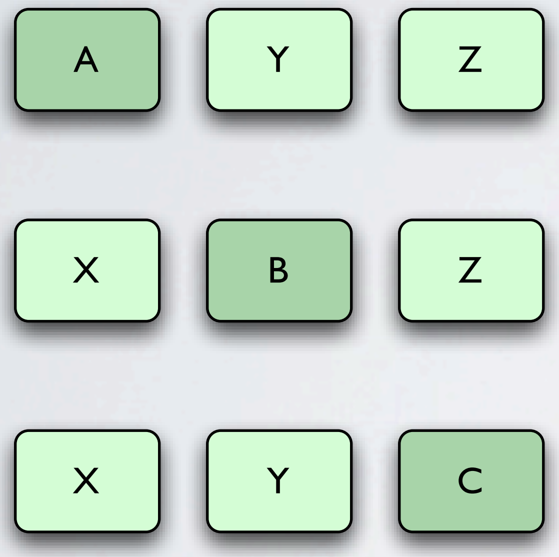
Superposition

- C is superposition of $A = a_1 \cdot a_2 \cdots a_m$ and $B = b_1 \cdot b_2 \cdots b_n$ iff:
  I. A and B have the same number of segments ($m = n$: equi-cardinality)
  II. either $\mathbf{a_i = b_i}$ or **instance-of(a_i, b_i)** or **instance-of(b_i, a_i)** holds for every i.
    - If II holds, it is **superposition without specification overrides**.
    - If II is violated, it is **superposition with specification overrides**.
  - Note: *a* is an instance of *b* iff *a* is an underspecification of *b*.
- It is trivial to implement superposition using feature structure.

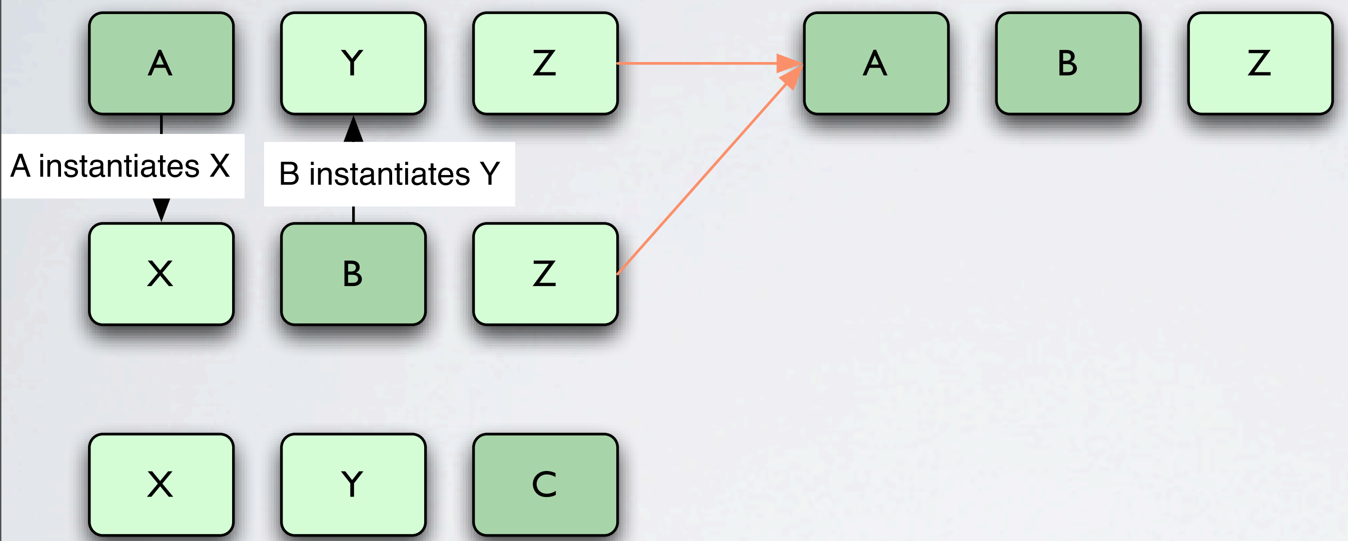
Relevant Terminology

- Superposition without overrides is (a special case of) **unification**.
- Superposition with overrides is (a special case of) **blending** in the sense of Fauconnier & Turner (1996, *et seq*.)
    - The latter case can **deal with inconsistencies** (e.g., conflicts in feature specification) between source structures, which is not allowed in the former.

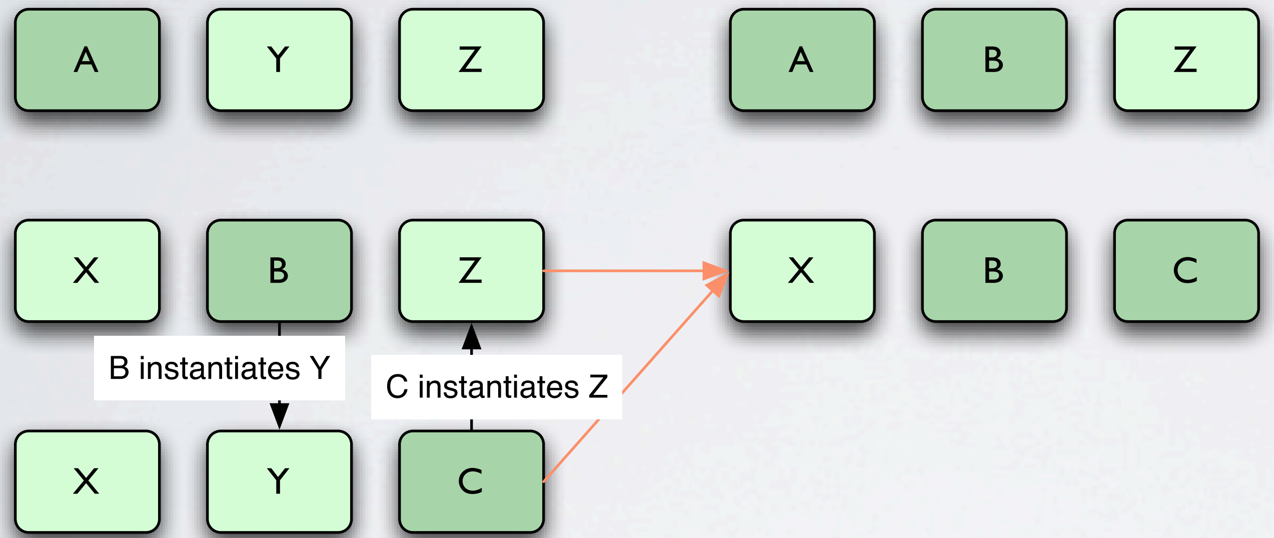
SUPERPOSITION



SUPERPOSITION

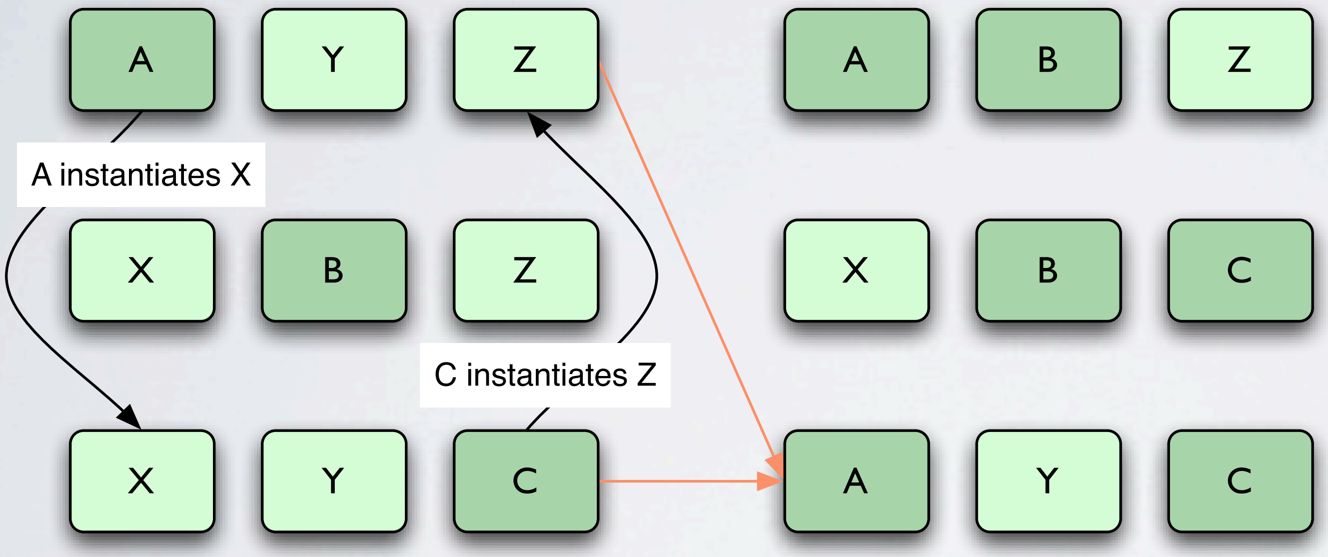


SUPERPOSITION

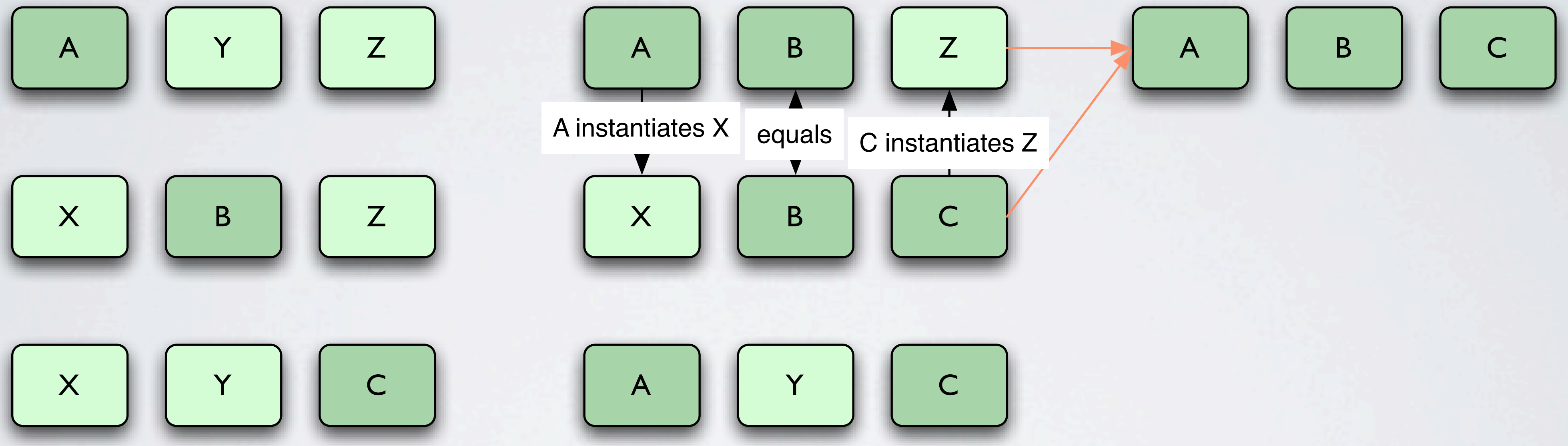


Friday, March 12, 2010

SUPERPOSITION

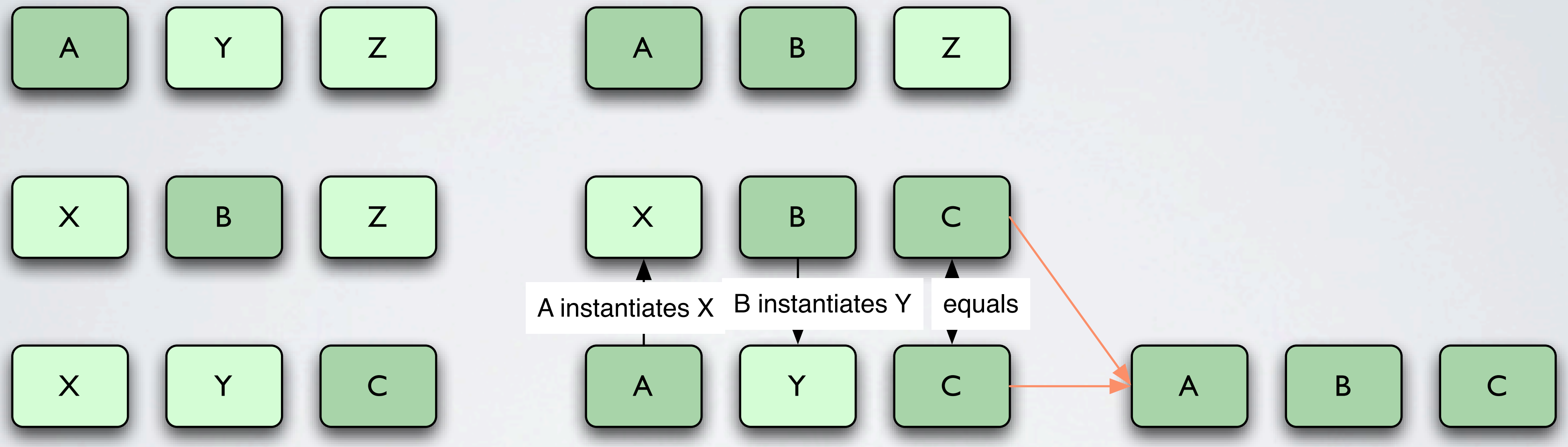


SUPERPOSITION (Result 1)

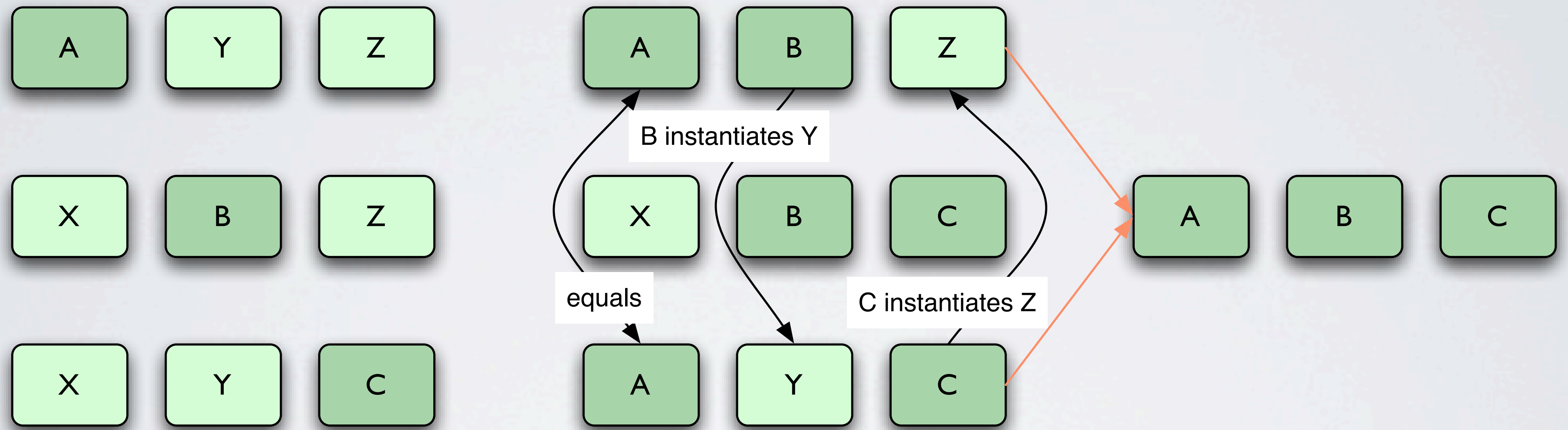


Friday, March 12, 2010

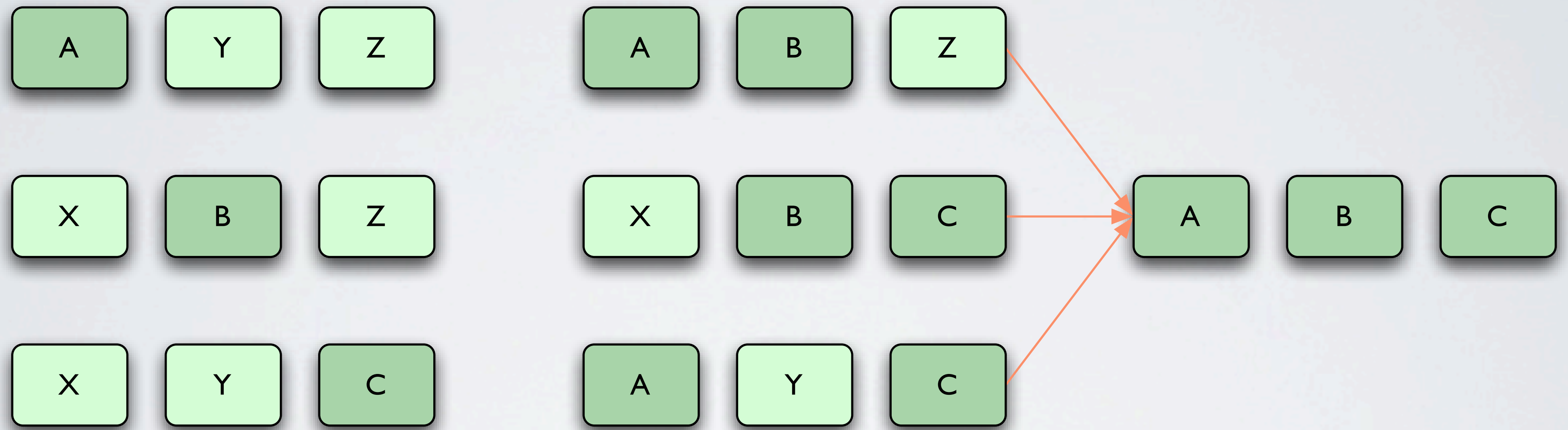
SUPERPOSITION (Result 2)



SUPERPOSITION (Result 3)



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.

Role of Redundancies

- **No superposition is possible if there are no redundancies in item representations.**
- Implications:
  - Phrase structure analysis under the **principle of proper analysis** is a roundabout to superposition.
  - Theoretically, superposition over a set of phrase structures is possible (e.g., Sadock's *Autolexical Syntax* (1991)), but it usually gets more complicated than superposition of (flat) patterns.

Desirable Properties

- Superposition **does not require proper analysis**,
  - and **phrase structure analysis**, either.
- Yet superposition
  - allows **composition without host structures**,
  - allows **composition under overlaps** over elements with redundancies,
  - and solves "bracketing paradox" (Spencer 1988) automatically

Examples of Overlap 1/2

- In morphology: ***generative grammarian*** (bracketing paradox Spencer 1988)
  - superposition of [$_{u1}$ **generative**][$_{u2}$ **grammar**] and [$_{u2}$ **grammar**][$_{u3}$ **-ian**]
- In syntax: ***an easy book to read*** (discontinuous constituent in McCawley 1988)
  - superposition of [u_1 **an**][u_2 **easy**][u_3 **book**] and [u_2 **easy**][u_3 ...][u_4 **to**] [u_5 **read**] with overlaps at u_2 and u_3.
- Remark:
  - **overlap is involved in most cases in which syntactic movement is necessary.**

Examples of Overlap 2/2

- 浮世絵師 (mundanity painter at Edo era) is superposition of [u_1 浮世][u_2 絵] (mundanity pictures at Edo era) and [u_2 絵][u_3 師] with overlap at u_2
- 投影像 (projective image) is superposition of [u_1 投][u_2 影] (projection) and [u_2 影][u_3 像] (image of shadow) with overlap at u_2.
- Remark:
  - **overlapping seems to be more frequent in head-final languages.**

Comparison

	Substitution	**Superposition**
item encoding and generativity	context-free	context-sensitive
memory load	minimum	maximum
size of lexicon	minimum	maximum
overlaps	can't handle	can handle
relation of semantics to syntax	extrinsic	intrinsic

Summary of Part I

- Superposition can implement composition properly.
  - Simply, composition **need not** be implemented by substitution.
  - Superpositional model of composition **can be computational** if superposition is properly defined as a formal operation.
- Superposition is desirable if we target overlapping phenomena.
  - Overlapping is far from well understood but is ubiquitous, and is likely to **have been overlooked due to linguist's (naïve) belief in proper analysis**.

Where do Patterns Come from?

Pattern Lattice Model in a nutshell

New Burning Question

- You may ask:
  - Alright, I understood that patterns have desirable properties, but **where do patterns come from** after all?
  - **Aren't they generated by grammar or something like that?**
- My answers:
  - Structure called "pattern lattice" (PL) over a set of expressions works as a generator of patterns.
  - This makes the challenge by the second question unsuccessful.

Definition of Pattern Lattice (PL)

- A pattern lattice is a complete lattice over a sequence of units with a fixed number n (i.e., patterns of length n) under the instance-of relation.
- An expression (including pattern) $E = e_1 \cdot e_2 \cdots e_m$ (e_i denotes the ith segment of E) is an instance of pattern $P = p_1 \cdot p_2 \cdots p_n$ (p_i denotes the ith segment of P) if and only if:

A. E and P **have the same number of segments** (i.e., $m=n$).

B. either **$e_i = p_i$** or **instance-of(e_i, p_i)** holds for every i.

Example 1

- Take (1) for example:

(1) **Ann sent Bill a letter.**

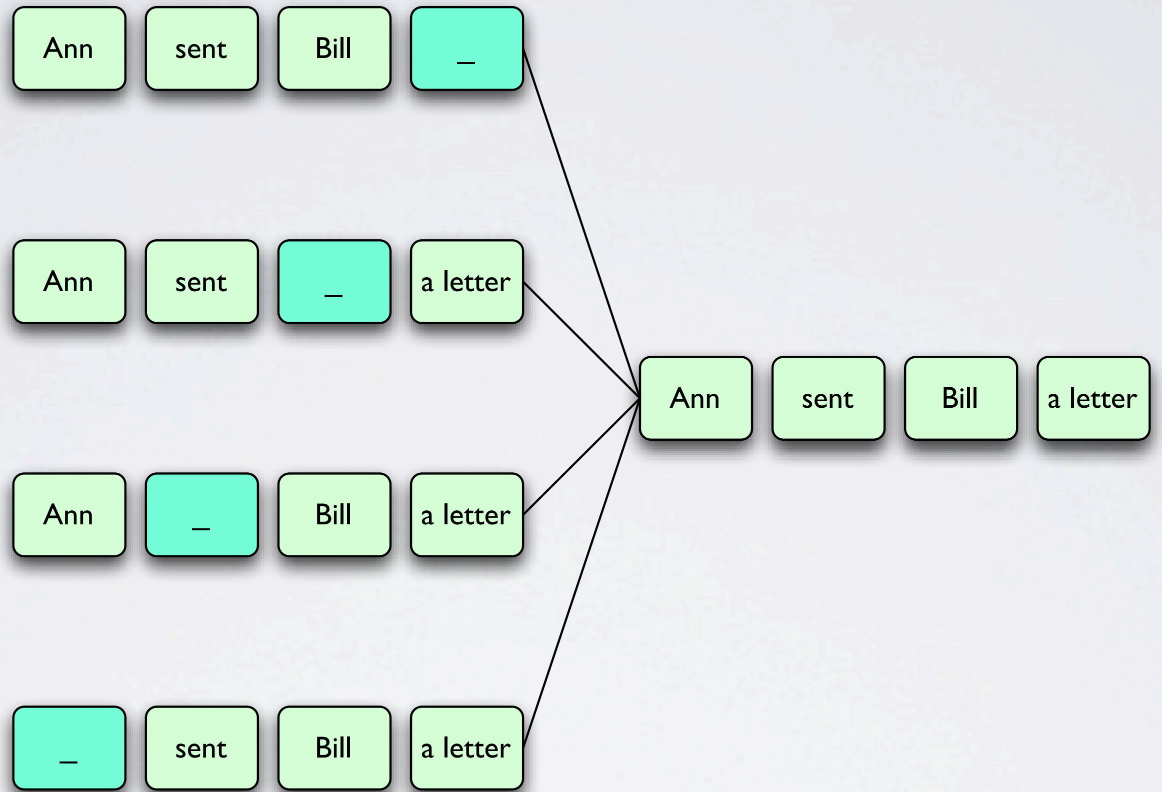
Example 1

- Suppose (1) has 4 segments.
  - [**Ann**, **sent**, **Bill**, **a letter**]
- Remark:
  - I just assume that this segmentation with four segments is (nearly) optimal.
  - Its optimality is not justified intrinsically in the PLM. It needs to be justified extrinsically either by relying on unsupervised classification/learning methods or more radically stochastic methods like Monte Carlo simulation.

At the Bottom

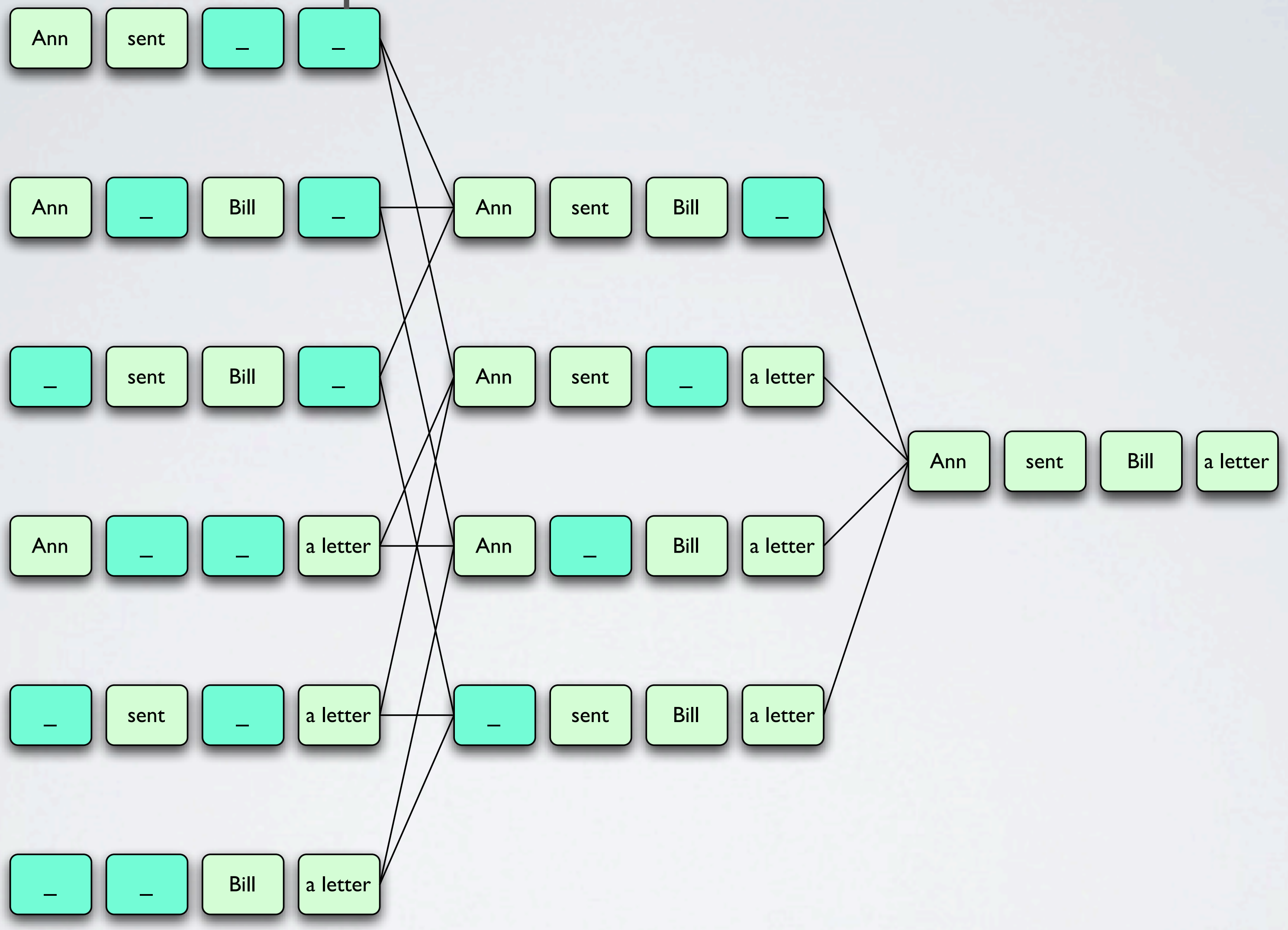
Ann | sent | Bill | a letter

Decomposition into Patterns



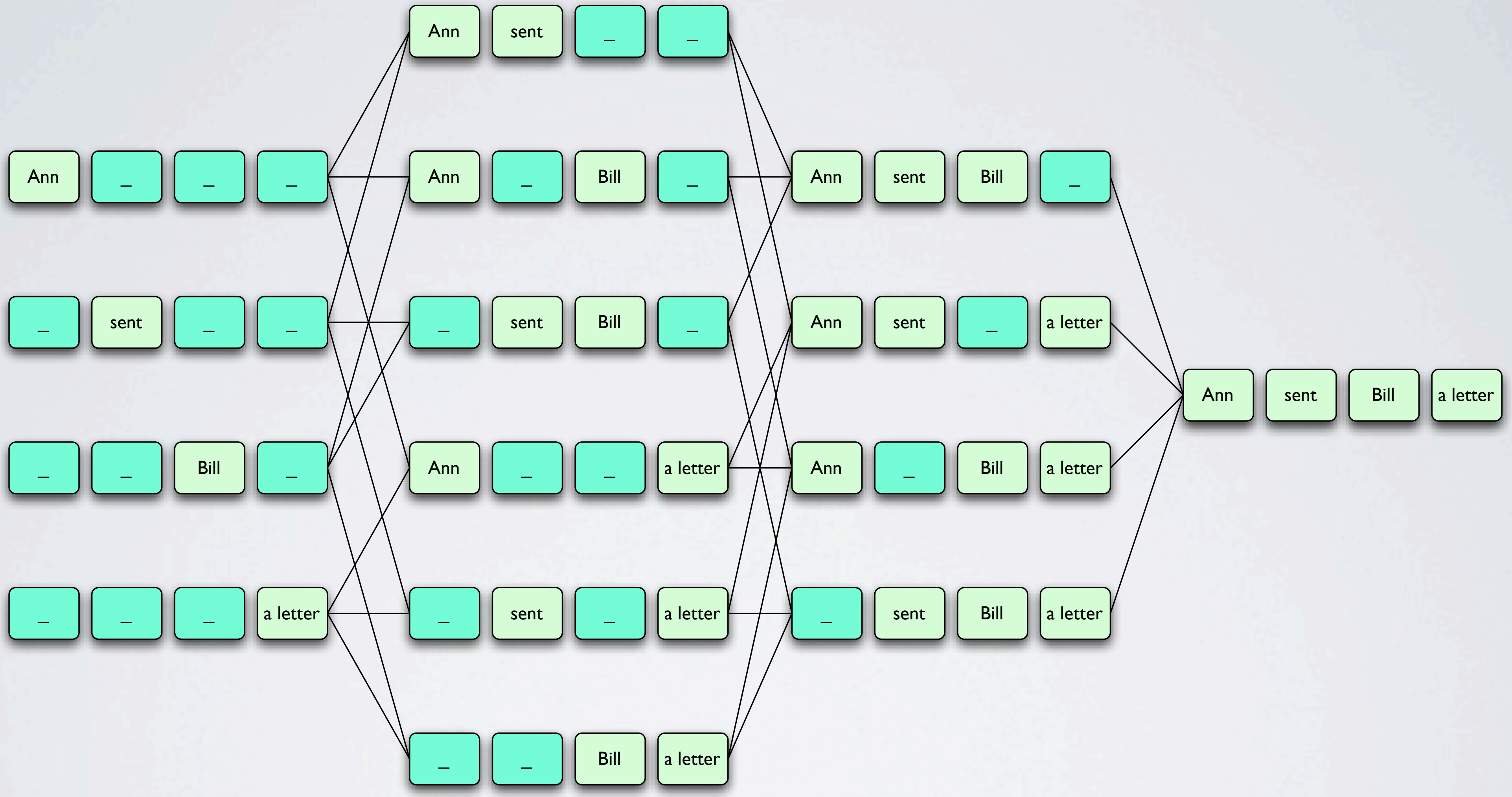
- Decomposition introduces variables denoted by "_". In general, expression *E* of size *n* has *m* **immediate components** when it has *m* constants in it ($m \leq n$).
- [Ann, sent, Bill, _], [Ann, sent, _, a letter], [Ann, _ Bill, a letter], and [_, sent, Bill, a letter] are immediate components of [Ann, sent, Bill, a letter] = (1).

Decomposition into Patterns



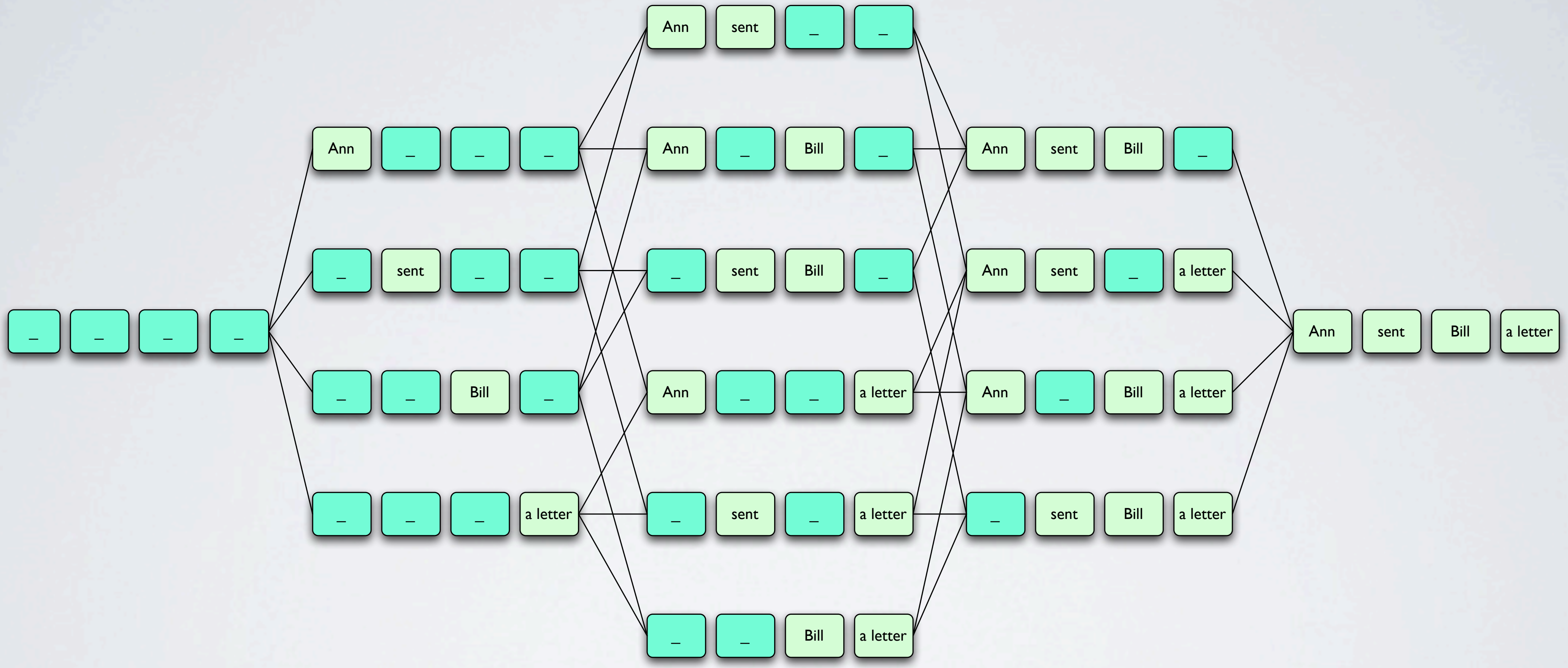
[Ann, sent, _, _], [Ann, _, Bill, __], and [_, sent, Bill, _] are **immediate components** of [Ann, sent, Bill, _], and so on.

Decomposition into Patterns



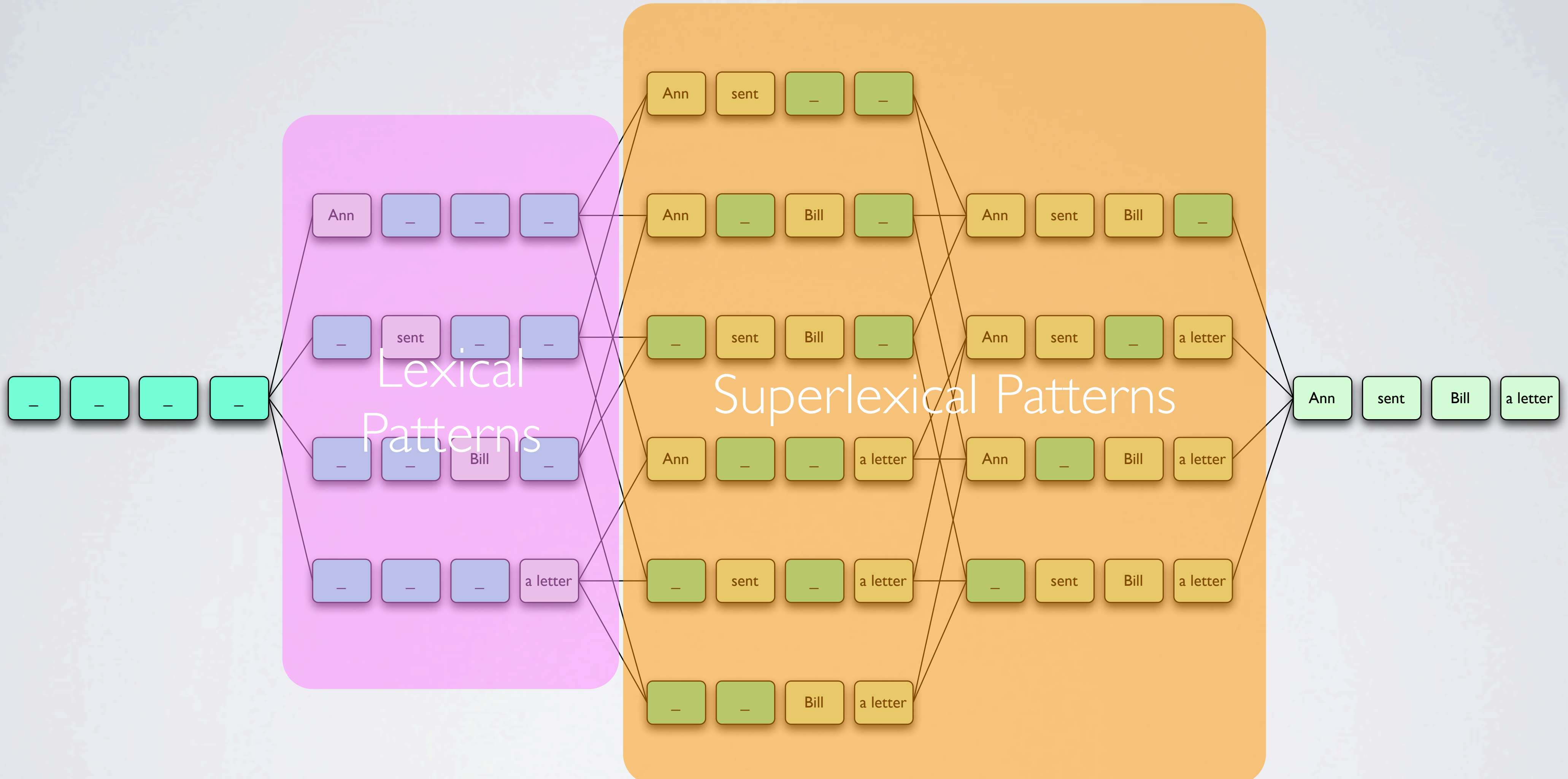
[Ann, _, _, _] and [_, sent, _, _] are **immediate components** of [Ann, sent, _, _], and so on.

Pattern Lattice Built for (1)



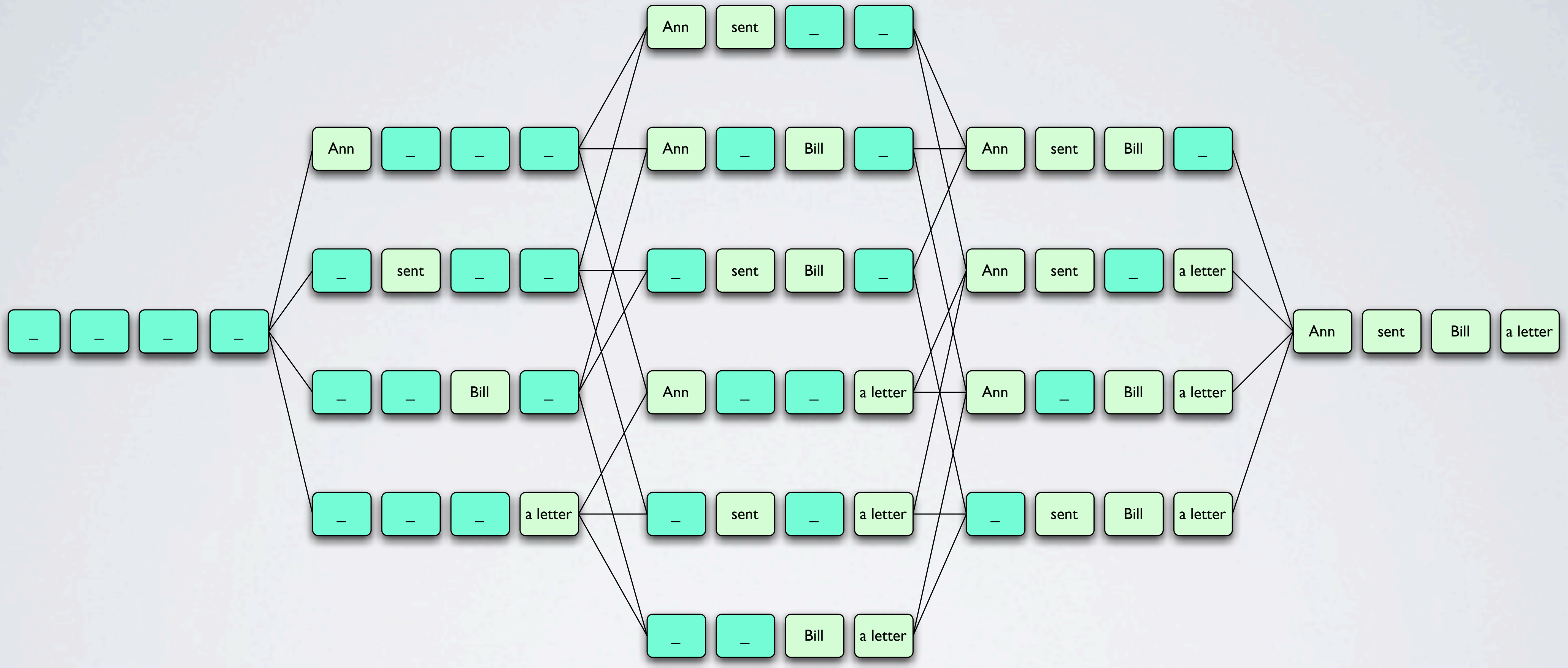
[_, _, _, _] is **the only immediate component** of [Ann, _, _, _], [_, sent, _, _], [_, _, Bill, _], and [_, _, _, a letter].

Pattern Lattice Built for (1)



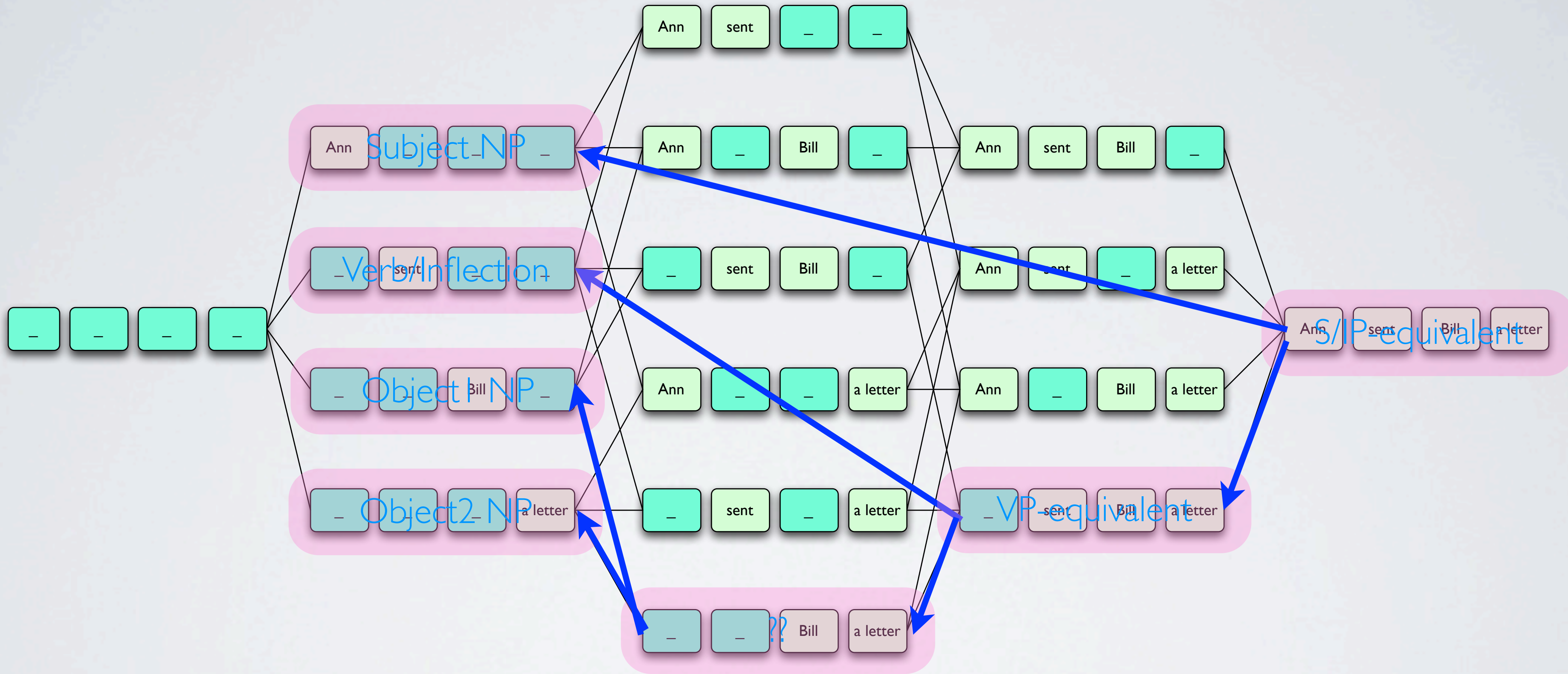
[_, _, _, _] is **the only immediate component** of [Ann, _, _, _], [_, sent, _, _], [_, _, Bill, _], and [_, _, _, a letter].

Constituency



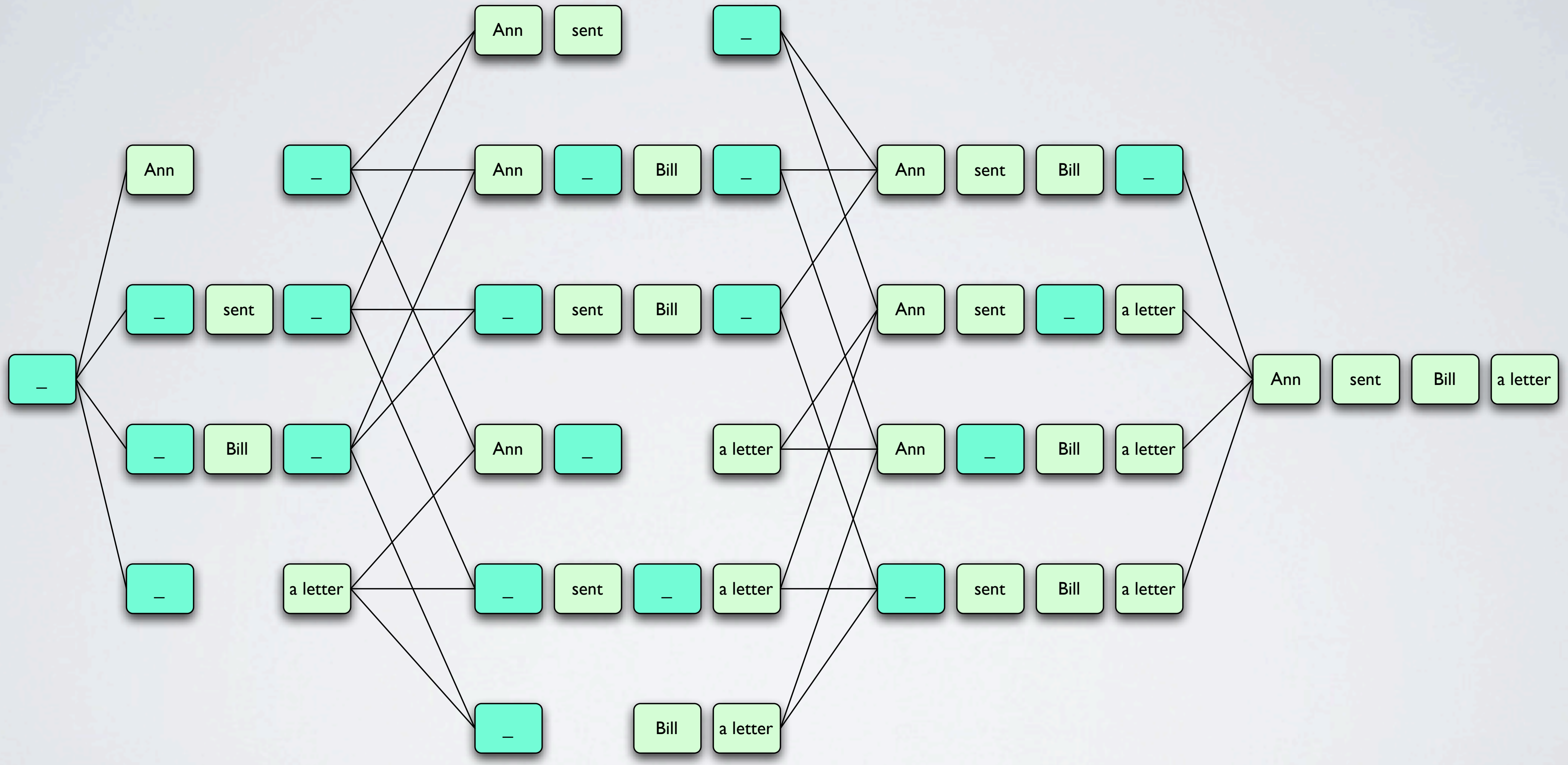
Equivalents of constituents **are implicitly specified**: they are simply **patterns that contain only continuous constants**.

Constituency



Equivalents of constituents **are implicitly specified**: they are simply **patterns that contain only continuous constants.**

In Simplified form



variable sequences are simplified.

Semantics under a Patter Lattice

with “Simulated Parallel Error Correction” Model of Meaning Construction

Yet Another Burning Question

- You may further ask
  - Alright, but **how does semantics works under Pattern Lattice Model?**
  - More specifically, what guarantees the compositional nature of semantic interpretation?
- My answers:
  - We have an algorithm called **Simulated Parallel Error Correction** (SPEC) that gives the "right" interpretation of a given input.
  - SPEC is an extremely exemplar-based algorithm that can handle both the compositional and noncompositional aspects of interpretation.

SPEC in a Nutshell

- SPEC (see my paper for details) is a mechanism for semantic interpretation of a given expression *E* that **works in the same way as example-based machine translation works** (EBMT: 佐藤 1997; Sato & Nagao 1993).
- Basic correspondences:
  - Input expressions in SPEC correspond to expressions of source language in EBMT.
  - Superlexical (usually, sentential) semantics in SPEC correspond to expressions of target language (i.e., translations) in EMBT.
- In a sense, EBMT is a version of SPEC in that it equates semantic representations with expressions observed at surface.

Example 2

- Suppose we have other two examples (2) and (3) with 4 segments:

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

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

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

Example 2

- Suppose we have other two examples (2) and (3) with 4 segments:

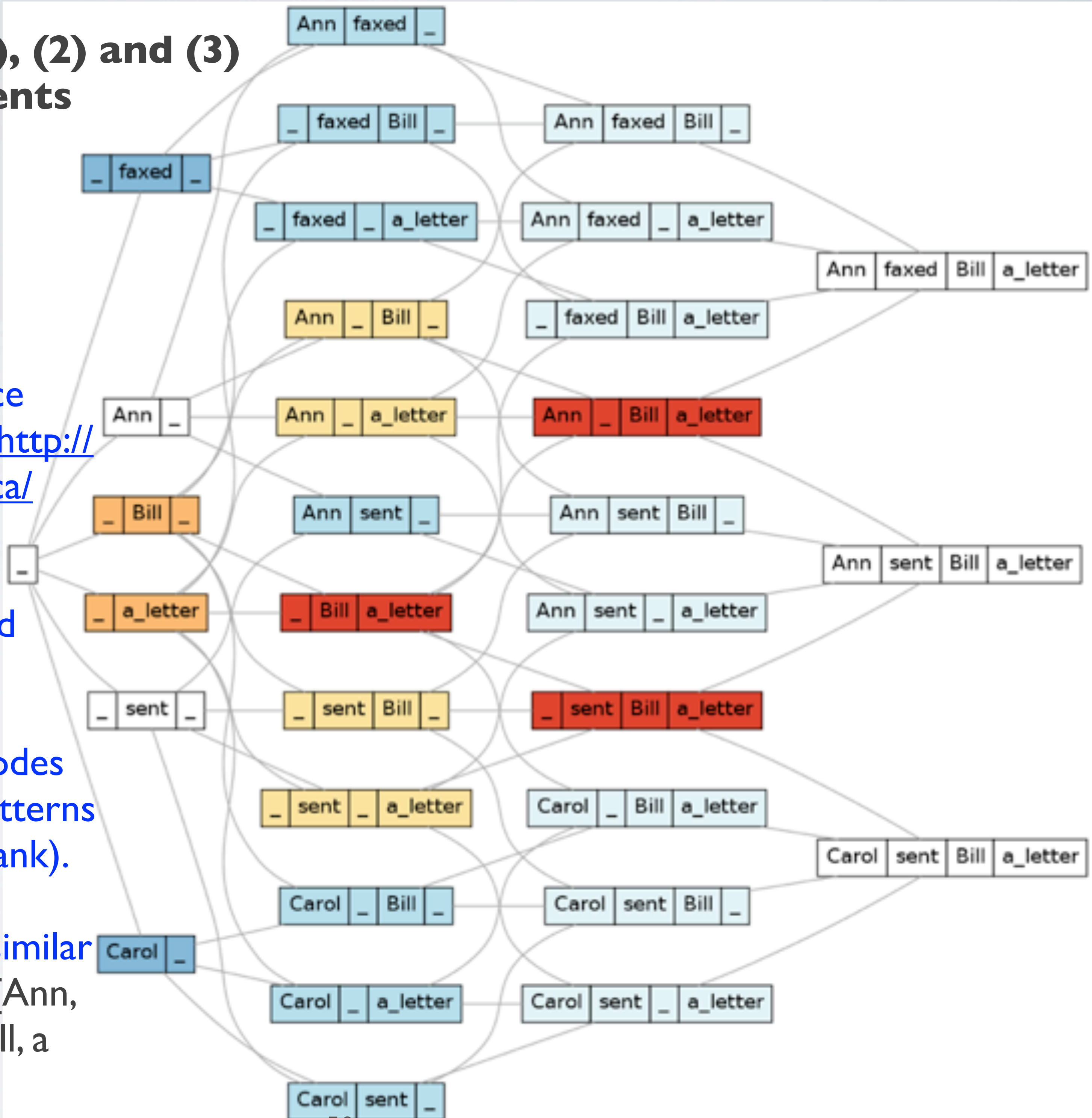
  (1) [Ann, sent, Bill, a letter]

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

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

- Note:
  - Goldberg (1995) treated (2) as an example of Ditransitive Construction.

Pattern lattice for (1), (2) and (3) with 4 segments



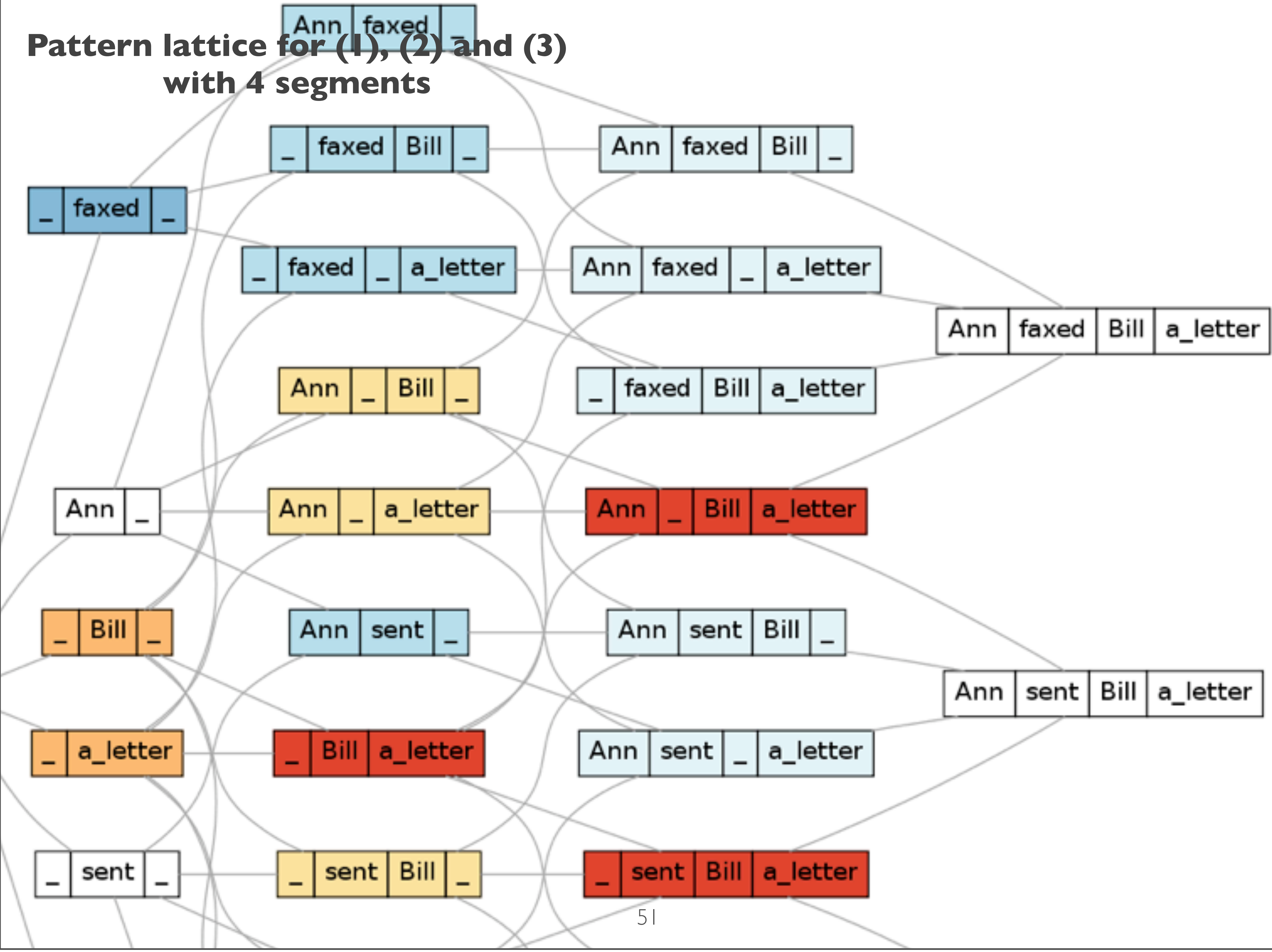
- Built using Pattern Lattice Builder (PLB) available at http://www.kotonoba.net/rubyfca/pattern

- Represented in simplified form.

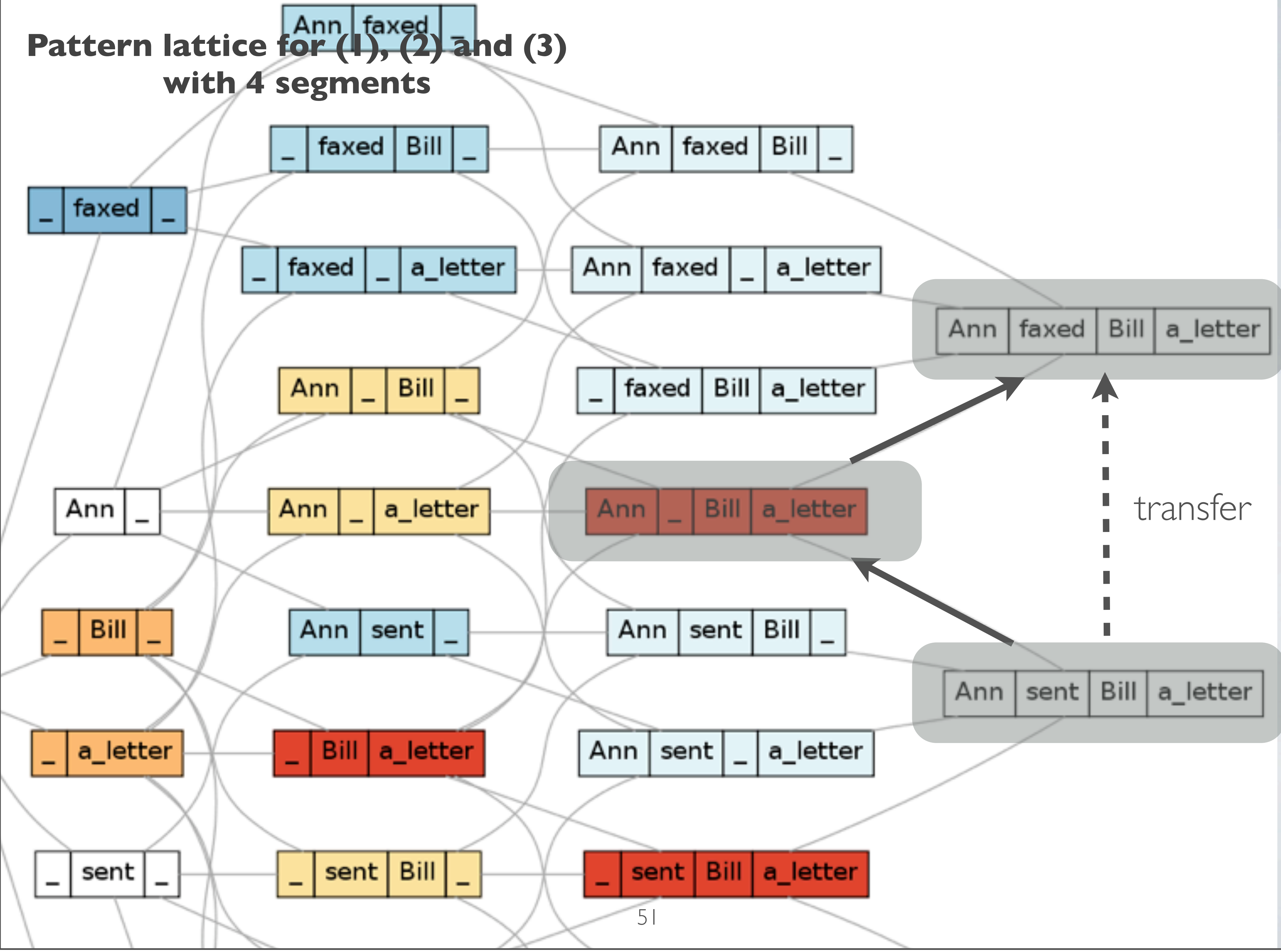
- Color temperature encodes relative productivity of patterns (in terms of z-score for rank).

- (1) is one of (2)'s most similar instances due to pattern [Ann, _, Bill, a letter] or [_, _, Bill, a letter].

Pattern lattice for (1), (2) and (3) with 4 segments

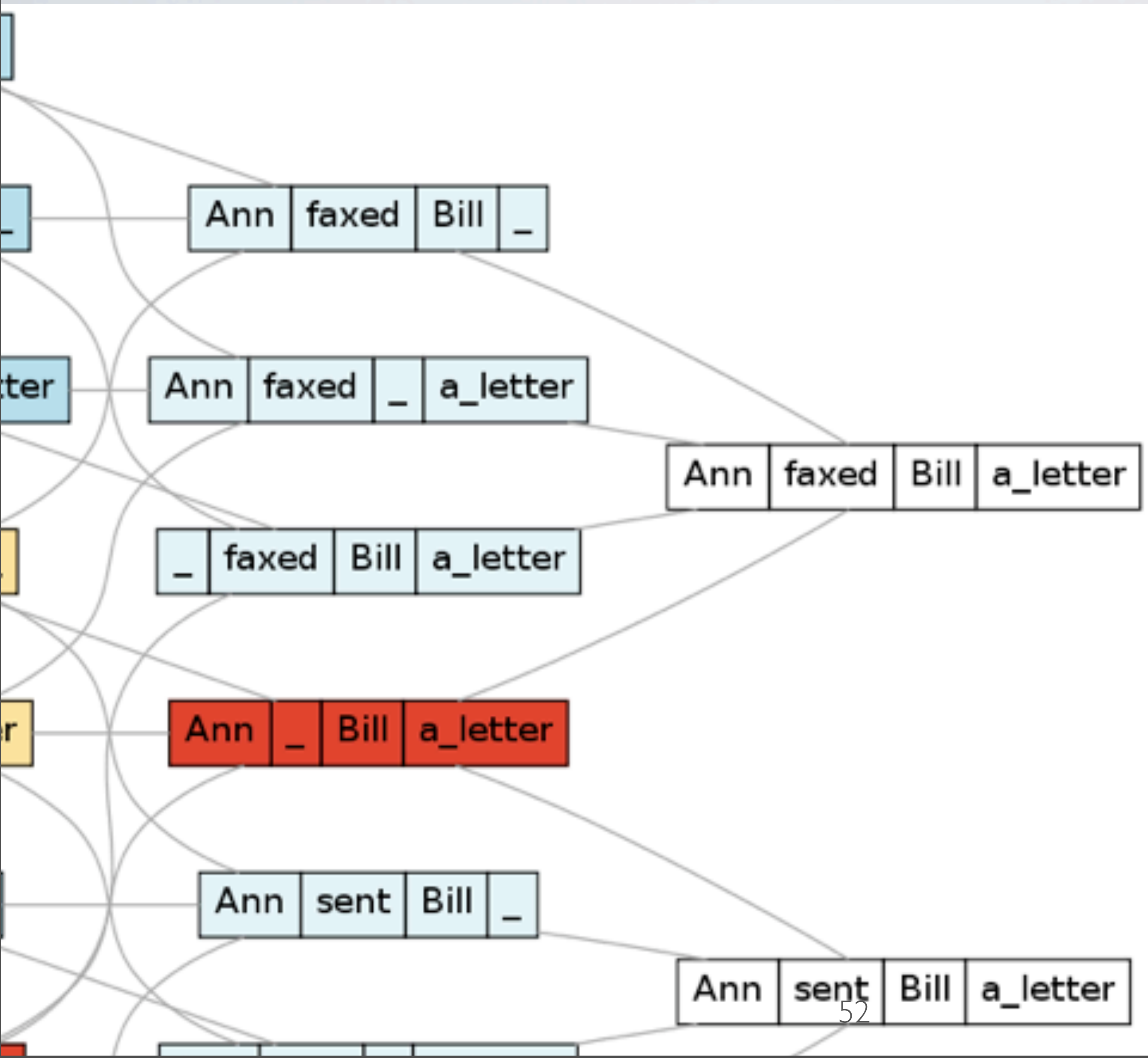


Pattern lattice for (1), (2) and (3) with 4 segments



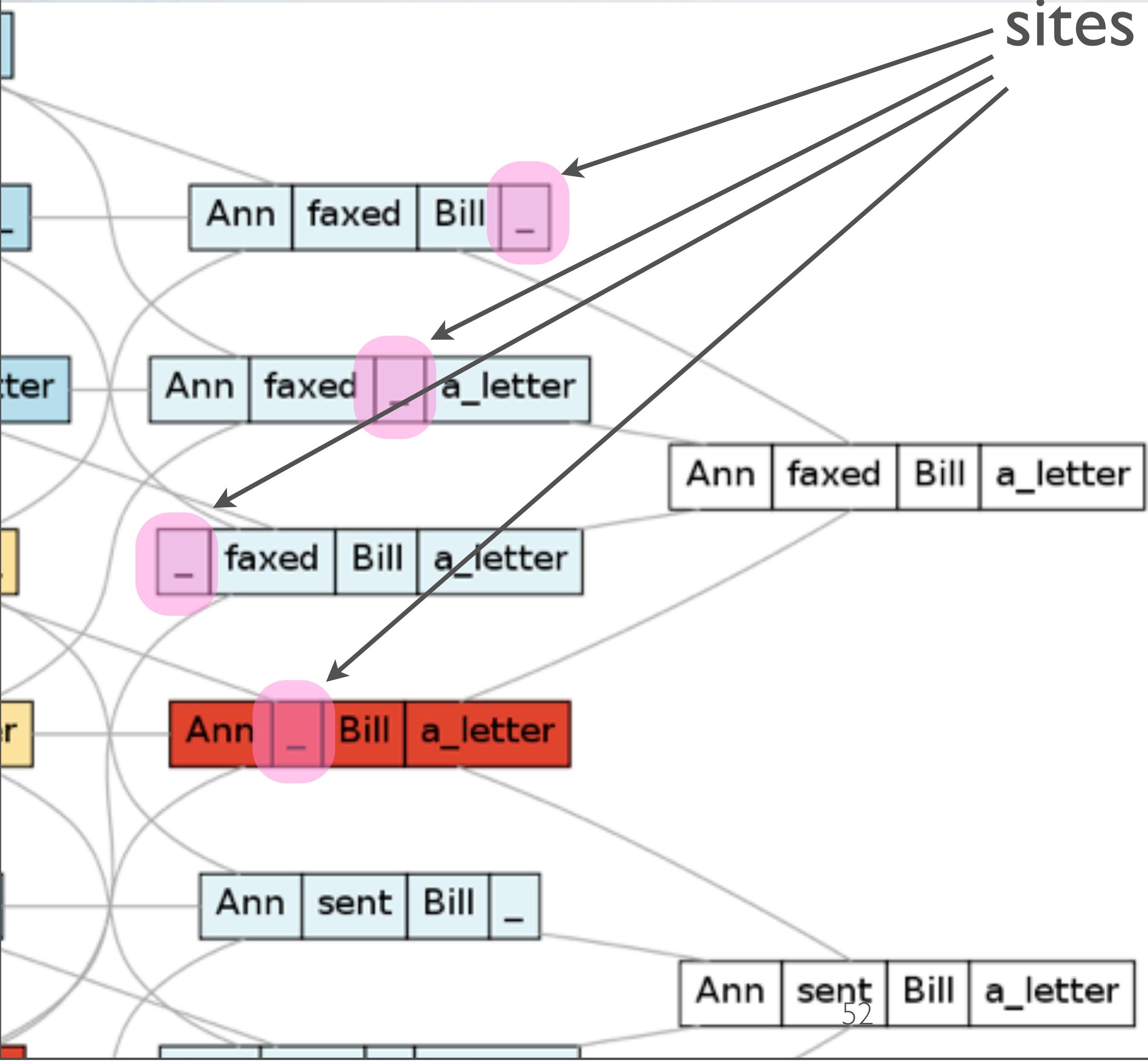
Friday, March 12, 2010

How SPEC Works



- Introduced variables are regarded as simulated errors that need correction.
- Each simulated error is corrected by equating it with the most likely constant based on the distributions, identified due to pattern completion.
- Different superlexical patterns have different instantiation distributions! This is why some patterns have strong bias for particular variable, and others do not.

How SPEC Works



sites of simulated errors

- Introduced variables are regarded as simulated errors that need correction.
- Each simulated error is corrected by equating it with the most likely constant based on the distributions, identified due to pattern completion.
- Different superlexical patterns have different instantiation distributions! This is why some patterns have strong bias for particular variable, and others do not.

Noncompositionality

- Under SPEC under PL, **superlexical semantics**, i.e., (expected) semantics of superlexical patterns, **is always preferred over lexical semantics**, i.e., semantics of lexical patterns.
    - Reason: Semantics of lexical patterns is accessed only when superlexical semantics at lower ranks turned out to be informative enough.
    - Roughly, SPEC equates the semantics of an expression *E* as an **expected semantics** over a set of instances similar enough to *E*.
- Claim: this gives the most straightforward account of why **collocations and constructions with noncompositional semantics are more important** in semantic interpretation.

Summary of Part II

- **No grammar is necessary for construction of a pattern lattice**.
    - All we need is a mechanism for segmentation and variable-introduction.
    - Note, however, that segmentation can be realized stochastically (using Monte Carlo method) or through unsupervised learning (using Hierarchical Bayes (Mochihashi et al. 2009)).
- SPEC under PL provides a straightforward account for effects of "constructions" without stipulating constructions *per se*.
    - **Basic tenets of Construction Grammar** (Fillmore 1988; Goldberg 1995) **are theoretical consequences of SPEC under PL**: they need not be stipulated as a doctrine.

DISCUSSION

What Can We Expect?

- A radically memory-based model of language is expected to
  - explain the importance of **collocations** (Sinclair 1991) or **multiword units/ expressions** (Sag et al. 2002), and
  - better explain the effects of **constructions** (Fillmore 1988; Goldberg 1995, 2006) with no stipulation for constructions *per se*.
- and it is also expected to
  - explain the **formulaicity** of language (Wray 2002), and
  - explain the mysterious **survival of lower-frequency items** (my personal point of view)

What is Grammar after all?

- PLM implements **a radically memory-based model** (RMBM) in which **virtually no instances are generated**.
    - A "new" expression *E* is recognized as superposition of patterns, p_1, p_2, ..., p_n that usually have only partial matches on *E*.
- In a RMBM, grammar is best understood as a **management system** rather than a **generative system**.
    - In a vast memory system, **all instances need to be indexed for effective retrieval**: PL does this.
    - Trade-off between rapid and flexible enough responses and redundancies.

Good News and Bad News

- Good news
  - Radically memory-based models explain better (at least in terms of descriptive adequacy), and will provide implementations that perform better (at least in terms of precision).
  - Lay foundations for **usage-based model** (Bybee 2001, Langacker 1988, Tomasello 2003) and **example-based machine translation** (佐藤 1997; Sato & Nagao 1990).
- Bad news
  - Language is not guaranteed to be as **systematic** as linguists want it to be.
  - Memory-based models are **computationally expensive**, and **harder to implement** (at least for realistic performance).

Conclusions

- This talk
  - presented an alternative to traditional, **substitutional** model of linguistic syntax and semantics.
  - proposed superposition-based model called **pattern lattice model** (PLM) which is **both compositional and computational** and argues for a **radically memory-based view** of language in which grammar of language is conceived of as **memory-management system** rather than a generative system.
  - showed that PLM under SPEC provides a natural account for "constructional meanings" without postulating constructions *per se*.

Acknowledgements

- The following people helped me in preparing this talk:
  - Yoichiro HASEBE (Doshisha University)
    - for implementation of Pattern Lattice Builder available at **http://www.kotonoba.net/rubyfca/pattern/**
  - Yasunari HARADA (Waseda University) and Masaya IDEGUTI
    - for insightful comments and suggestions
- This work was partly supported in part by Grant-in-Aid for Scientific Research (B) (19330156) (PI: Masanori Nakagawa, Tokyo Institute of Technology) from the Japan Society for the Promotion of Science

References

- Bybee, J. L. (2001). *Phonology and Language Use*. Cambridge University Press.
- Fauconnier, G. R. & Turner, M. (1996). Blending as a central process of grammar. In Goldberg, A. D. (Ed.), *Conceptual Structure, Discourse, and Language*. CSLI.
- Fillmore, C. J. (1988). The mechanisms of Construction Grammar. In *BLS* 14: 35-55.
- Goldberg, A. D. (1995). *Constructions: A Construction Grammar Approach to Argument Structure*. University of Chicago Press.
- Goldberg, A. E. (2006). *Constructions at Work: The Nature of Generalizations in Language*. Oxford University.
- Langacker, R. W. (1987). *Foundations of Cognitive Grammar, Vol. 1*. Stanford University Press.
- Langacker, R. W. (1988). A usage-based model. In Rudzka-Ostyn, B. (Ed.), *Topics in Cognitive Linguistics*, pp. 127-161. John Benjamins.
- McCawley, J. D. (1988). *The Syntactic Phenomena in English, Vol. 2*. University of Chicago Press.
- Miller, G. A. (1956). The magical number seven, plus or minus two. *The Psychological Review* 63 (2): 81-97.
- Mochihashi, D., T. Yamada, and N. Ueda (2009). Bayesian unsupervised word segmentation with nested Pitman-Yor language modeling. In Proc. of the Joint Conf. of the 47th Annual Meeting of the ACL and the 4th Inter. Joint Conf. on NLP of the AFNLP,. pp. 100-108.
- Parker, E. S., Cahill, L. & McGaugh, J. L. (2006). A case of unusual autobiographical remembering. *Neurocase* 12(1): 35-49.
- Port, R. (2007). How are words stored in memory? Beyond phones and phonemes. *New Ideas in Psychology* 25 (2): 143-170.
- Post, E. (1943). Formal reductions of the combinatorial decision problem. *Amer. J. of Mathematics* 65(2): 197-215.
- Sadock, J. (1991). *Autolexical Syntax*. University of Chicago Press.
- Sag, I., Baldwin, T., Bond, F., Copestake, A. & Flinckinger, D. (2002). Multiword expressions: A pain in the neck for NLP. In *Proc. of the 3rd Inter. Conf. on Intelligent Text Processing and Computational Linguistics*, pp. 1-15.
- 佐藤 理史 (1997) アナロジーによる機械翻訳. 共立出版.
- Sato, S. & M. Nagao (1990). Toward memory-based translation. In *Proc. of The 3rd Inter. Conf. on Computational Linguistics*, pp. 247-252.
- Spencer, A. (1988). Bracketing paradoxes and the English lexicon. *Language* 64: 663-682.
- Tomasello, M. (2003). *Constructing a Language: A Usage-based Theory of Language Acquisition*. Harvard University Press.
- Wray, A. (2002). *Formulaic Language and the Lexicon*. Cambridge University Press.

Thank You for Your Attention