

# Contents

<b>Abstract</b>	<b>7</b>
<b>1 Introduction</b>	<b>9</b>
1.1 The Problem . . . . .	10
1.1.1 Some examples . . . . .	11
1.1.2 Word choice & context variation . . . . .	12
1.2 The goal . . . . .	15
1.2.1 A class-based model . . . . .	16
1.3 Empirical relations . . . . .	17
1.3.1 Representing preferences . . . . .	18
1.3.2 Expressing systematic behavior . . . . .	19
1.3.3 The categorical model . . . . .	21
1.4 Overcoming sparse data . . . . .	22
1.4.1 Systematic grammatical relations . . . . .	23
1.4.2 Forming classes . . . . .	25
1.5 Outline of the text . . . . .	26
<b>2 Language structure</b>	<b>29</b>
2.1 Configuration . . . . .	30
2.1.1 The range of effects . . . . .	30
2.1.2 Characterizing errors . . . . .	32
2.1.3 Degree of constraint . . . . .	34
2.1.4 Syntactic constraints are related to word meaning . . . . .	35
2.2 Structural origins . . . . .	37
2.2.1 Counterexample: An unrestricted language . . . . .	38
2.2.2 Case mechanisms . . . . .	39
2.2.3 The complexity of syntax . . . . .	43
2.2.4 Sentential interference . . . . .	48
2.2.5 Benefits of configuration . . . . .	50
2.3 Expectations . . . . .	51
2.3.1 Consistent configuration . . . . .	52
2.3.2 Associations . . . . .	53

2.3.3	Alternation and classes . . . . .	54
<b>3</b>	<b>The Associative Model</b>	<b>57</b>
3.1	The form of the associative model . . . . .	58
3.1.1	Association . . . . .	58
3.1.2	The Basic Model . . . . .	62
3.1.3	The Class Model . . . . .	66
3.1.4	Class membership and variation . . . . .	69
3.1.5	Implications . . . . .	72
3.2	Simple Acquisition . . . . .	75
3.2.1	Maximum likelihood estimates . . . . .	75
3.2.2	Representational duals: $c(W)$ and $w(C)$ . . . . .	76
3.2.3	$n$ -gram models . . . . .	78
3.3	Category Discovery . . . . .	79
3.3.1	Comparing Distributions . . . . .	79
3.3.2	Identifying class membership from usage data . . . . .	82
<b>4</b>	<b>Smoothing Difficult Data</b>	<b>85</b>
4.1	Sparse data . . . . .	86
4.1.1	Comparing distributions . . . . .	90
4.2	Estimating missing values . . . . .	92
4.2.1	Smoothing . . . . .	92
4.2.2	Non-linguistic smoothing estimates . . . . .	95
4.2.3	Linguistically informed methods . . . . .	98
4.2.4	Classification methods . . . . .	99
4.3	Similarity and Classification . . . . .	102
4.3.1	Behavioral assumptions . . . . .	103
4.3.2	Structural Similarity . . . . .	104
4.4	Organizing the context space . . . . .	107
4.4.1	Grammatical analogy . . . . .	107
4.4.2	Structural context classes . . . . .	108
4.4.3	Bootstrapping Distributional Classes . . . . .	110
<b>5</b>	<b>Quantifying similarity</b>	<b>113</b>
5.1	Sequence measures . . . . .	114
5.1.1	Set measures . . . . .	115
5.1.2	Aligned pairwise comparisons . . . . .	118
5.1.3	Edit Distance . . . . .	121
5.1.4	Linguistic plausibility of comparisons . . . . .	126
5.2	Richer measures . . . . .	126
5.2.1	Locality in the metrics . . . . .	127
5.2.2	String alignments . . . . .	132
5.3	String measure experiments . . . . .	133

## Chapter 2

# Language structure

In order to design an appropriate model of word use, we first need to understand what kind of patterns and structure we should expect in their real use in language. This chapter outlines a very general theory of the configurational properties of language, as imposed by the efficiency of its use as a device for representation and communication. The immediate result, which should not be surprising, is that a small number of configurational constraints types – grammatical relations corresponding to functional combinations – greatly improve the communicative efficiency of sequences of words, in terms of the number of words required and the processing power necessary to decode them. This result implies that we have every reason to expect that a natural language, optimized in part for communicative efficiency, should exhibit exactly the kind of regular structure that a study of correlated co-occurrences is able to find. The effectiveness of the *associative model* and similar methods in the literature is purely contingent on the presence of this regular syntactic structure.

The deeper result is that regularities in semantics should coincide with structural, syntactic regularities. Thus, any procedure which sets out to find certain structural, distributional regularities in the language will also find parallel semantic properties. This implies that a properly structured acquisition of distributional

properties can not only be used to form a syntactic model of a language, but can also be used to bootstrap a study of lexical semantics, and can form a basis for semantic inference procedures.

## 2.1 Configuration

The acquisition of the associative model requires that we discover some non-obvious properties of words from their appearance in corpus. Specifically, the model will capture patterns of context, along with patterns of word use within those contexts, as abstract categorical descriptions. We can later use this descriptive model to accurately predict previously unseen word usage and to infer similarity of meaning across sets of words.

In order to model these patterns, we first need to understand what kinds of patterns we can expect to occur. This will give us an outline of the properties of word use we need the model to represent, and so give rise to the form of the model, as well as hint at methods for acquiring the parameters from actual corpus usage data.

### 2.1.1 The range of effects

It should be clear that there are real constraints on the combinations of words in language. One can not simply toss a string of words together and expect to get a meaningful, or well-formed, expression. For example, (2.1) simply cannot be considered an English construction, even if all the symbols are individually acceptable.

eat direct Reluctantly have (fish ? mechanism the strengthen;                   (2.1)

Most would say that (2.1) is unacceptable because it does not meet the obvious formal<sup>1</sup> surface properties that are required of meaningful sentences — it does

---

<sup>1</sup>‘Formal’ here meaning ‘to do with form or shape,’ not necessarily implying any degree of precision or theoretical acceptance.

not match the *syntax* of English — not because of any deeper semantic considerations of failure in meaning. This sequence violates the normal arrangement of words, where ‘normal’ is somehow judged with regard to properties unrelated to specific meaning. If a word refers to something that is or could be participating in a relationship, we call it a NOUN. We know that, typically, in English, nouns appear before, but sometimes also after, a VERB, which identifies the relationship, which is possibly one of action, between those nouns or to the world at large. QUANTIFIERS appear before nouns, telling us to how many and to which items they might refer. These rules of placement and combination, which operate without regard to the deeper meaning of the words, are what is usually referred to as syntax or grammar.

One needn’t look closely into the meaning of words in (2.1) to see that the verbs don’t have nouns as subjects, the determiner is preceded by a noun and followed by a verb, and a host of other obvious form violations have been committed. It fails, by this argument, because there are rules of English grammar that allow certain combinations of word types and disallow others.

Some purely formal theories, such as some versions of transformational grammar, posit a process in which the structure of a sentence is created, and later specific words are introduced into positions by virtue of their category membership (cf. [Chomsky, 1957, ?chomsky:lslt?]). ‘Category’ in this sense is construed rather broadly, corresponding to traditional, high-level grammatical labels such as NOUN, VERB or ADJ. For example, if the first stages generated a structure such as (2.2), it might be filled in with lexical items as in 2.3).

NP    ADV        V    PREP    ADJ    N    (2.2)

I    readily    believe    in    fairy    tales.    (2.3)

This works well enough, up to a point. There is nothing in this theory to restrict us from filling in lexical items which properly belong to the categories, but which

do not combine in meaningful ways. Thus, we can fill in the same grammatically acceptable structure with combinations of words which make less and less sense, as shown in (2.4)–(2.6). The logical result of this sort of replacement are Chomskian “green ideas”-type examples, which are, in that view, grammatical “though nonsensical.” [?chomsky:ss?]

NP	ADV	V	PREP	ADJ	N	
The desk	yesterday	flew	through	rainy	weather.	(2.4)

Sand	metallurgically	accepts	between	pointy	water.	(2.5)
------	-----------------	---------	---------	--------	--------	-------

Cheese	grainily	waives	of	sapphire	maxima.	(2.6)
--------	----------	--------	----	----------	---------	-------

### 2.1.2 Characterizing errors

The kinds of constraints evident in these examples (2.1–2.6) are certainly different in magnitude, but the question should also be considered whether they are different in nature. It is true that we must examine the structural replacement examples much more closely than the ‘word salad’ of (2.1) to realize that something is wrong with what we are reading, that the words on the page don’t come together to make sense. How can we characterize the way these examples fail to meet our expectations about what a well-constructed, intelligible sentence or utterance in a language should be?

We can decipher the later examples to some degree, only because of the relative familiarity of the words used and the roles they are used in. Fortunately, most words have a relatively fixed meaning, and appear most often in only one of these high-level grammatical role categories, making the listener’s job of identifying the intended role of each word easier. When the rules of placement and combination are broken, however, the usual roles of the words come into conflict with the roles they must have in order to function meaningfully in combination. For instance, the words as used in (2.1) have conflicting constraints, and do not have nearly the

flexibility in roles that they need in order to make the sequence an interpretable English sentence.

### **Sources of information**

The information deciding the role each word plays in a construction comes in at least two flavors. One is lexical knowledge – knowledge of the word itself: what roles it usually takes in certain circumstances, what other terms it usually interacts with, and other specific knowledge of the one particular word. The other type is syntactic knowledge – our general knowledge of how words combine, applied to the environment in which the word is used. Information about the possible roles and meanings a word might assume is given to us by the arrangement of its neighbors. General knowledge of word arrangements limits serious problems, such as those of (2.1). More detailed information is also available — because a word must interact with its neighbors, our knowledge of how the neighbors normally operate imposes certain choices on how the word in focus can behave.

To illustrate the differences of these knowledge sources, we can construct an example with a novel word, with which we have no pre-associated lexical constraints. We don't know what this word might mean, but imagine it is used as in:

You should add some *gardle* to the soup. (2.7)

If one assumes that this sentence is interpretable, then we now know many things about the new word, all projected onto it by the remaining words of the construction. “add some X to the soup” tells us generally that *gardle* is a *thing*, a noun, since it is something you add (the lack of a right-hand nominal dependent rules out the possibility that it is a modifier of any sort; as in “some *gardle* turnips”). More specifically, it refers to a kind of thing that is composite and divisible, since it is “*some* *gardle*” rather than ‘a’ or ‘the’. Also, it refers to something that goes

into soup, and so is probably edible. The formal properties of syntactic combination have taken effect here, projecting an interpretation on the word, which seems acceptable because we have no prior knowledge about the word.

What would happen, though, if we were to take a familiar word and place it in the same construction? Let me use the word *strengthen*, which was so out of place in (2.1). We now have

You should add some strengthen to the soup. (2.8)

which seems totally incongruous – at least as much, if not more than it did in its appearance in (2.1). Our familiar interpretation of *strengthen* as a verb indicating an action clashes with every constraint on meaning that the remainder of the construction projects. No previous knowledge of this word prepares us to make such a leap in altering its familiar interpretation. The general syntax forces a nominal interpretation; the use of ‘some’ gives a composite reading; the verb, ‘add’, tells us the referent is manipulable; and the fact that it is added ‘to the soup’ gives us edibility, all of which conflict directly with the familiar interpretation of ‘strengthen’.

### 2.1.3 Degree of constraint

This example (2.8) sets up a complex matrix of projected constraints, all of which, from the nominal interpretation to the expectation of edibility, are broken by our existing knowledge of the word *strengthen*. All of the expectations that we have associated with the word, and all of the knowledge of its use in normal constructions, clashes with the relations forced upon it by the remainder of the construction.

The earlier *gardle* example (2.7), in contrast, is acceptable simply because the novel word does not carry any constraints along with it. It is merely acting as a carrier for the matrix of constraints projected upon it. We could use this word in place of practically *any* word, in any context, with identical results. While it

may not have a predetermined meaning, we are more or less happy to allow its interpretation according to the constraints imposed by context. Further, those constraints tell us, more or less, what the word must mean.

Example (2.1) violates the primary syntactic constraints of the language in that even the most general level of configuration of the terms does not allow the application of the normal constraints between them. Each word is used in a context so foreign to its normal modes of use that we can not even identify any familiar pattern.

The less incongruous examples (2.4)–(2.6) satisfy the basic agreements of context, but associations implied by the configuration fail to hold at a finer level of detail. In example (2.7), the implications of the configuration are quite acceptable, even though the word *gardle* contributes no information to the meaning. The final example, (2.8), contains a very different sort of error, one which the configuration imposes a coherent set of constraints, but that set clashes in every way with the familiar use of the offending word.

#### **2.1.4 Syntactic constraints are related to word meaning**

These examples demonstrate clearly that words place constraints on the words and constructions appearing jointly with them. *Any* configuration of words creates such constraints, whether they can be combined into a sensible, unified whole or not. If one is to call the construction part of the language, however, the words associated under that configuration must not carry conflicting information. I take this as a basic observation about the structure of language.

Some of that information is traditionally called ‘syntactic’, such as the category membership (N, V, etc.), while some is more toward the ‘semantic’ end of the spectrum — for instance, we know that cheese doesn’t do *anything* through any kind of maxima. However, we find that these aren’t truly separable properties of lexical items, but differ only in the granularity of the properties they describe.

One does not find, in any language, high-level syntactic categories that are assigned to lexical items irrespective of their meaning. Verbs (words assigned to the VERB category) for example, denote actions and relations. They never denote fixed objects one might find on one's desk. The information one has about a word's meaning, whether on the gross level (e.g. as is a member of VERB) or at finer grain (e.g. the non-volitional nature of cheese), determines the types of structures it can sensibly appear with.

Some would say that the knowledge that *gardle*, for instance, is a noun, is purely syntactic knowledge, and not related in any sense to what the word *means*. I would argue, though, that even knowledge of just the part of speech, as here, vastly limits the range of possible meanings, and as such contributes very real information to the meaning of the word. Since its interpretation is restricted to some part of the semantic space, it would seem that this is truly knowledge, however imprecise, of the *meaning* of the word.

In this informational constraint view of language, effects that often are identified separately as syntactic or semantic are inextricably tied together — the configurations in which a word is used are dictated, in large part, by what that word means. The following sections demonstrate that the coincidence of these separate effects results in a far greater efficiency of representation than would be possible if configuration were independent of meaning.

There is nothing, I believe, in the *functional* concept of language as a communicative and representational device that limits the ordering and structure of combinations. The conclusion supported here, however, is that it is the *practical, operational* concerns of instantiating and using a language that lead to configurational constraints on word use.

## 2.2 Structural origins

We have seen that there are definite limits on the manner in which words may combine in language. Is there anything in the concept of *language* itself that inherently limits the ways in which symbols might be put together to create complex relationships? In that these limitations are easily violated — it is simple to form a group of words, tossed together like those of (2.1), for which there is no available interpretation — it is easy enough to see that such limits exist. What is the structure of well-formed strings such that they form a useful part of the language?

One problem quickly encountered pursuing these questions is that we all have strong preconceptions of what ‘language’ is. For the present discussion, I take a rather loose view that language is primarily a representational system, using linear sequences of symbols to refer to mental constructs, which can exist in familiar predicate-argument relationships. Prototypical representations of these complex, multi-relational mental constructions can be found in modern cognitive theories (cf. [Sowa, 1984]), linguistics (cf. [?montague?]), and formal logic theory (cf. [?frege?, ?russel?]), stretching all the way back to the predicative mentalist logics of ancient philosophy (cf. [?aristotle:?]).

‘Language,’ as the term used here, is a system for relating linear strings of symbols to these highly structured, non-linear mental constructions. Whether the symbols refer to some set of basic primitives or could refer recursively to other constructs and relationships is not in itself material.

While a language system with unconstrained configuration of words is conceptually feasible, it is not without implications. Some simple structural limitations greatly increase the functionality and economy of the system. Some of these economies, such as limited memory capacity and sensory bandwidth, are inherently constrained by our neurological construction. We can begin by taking these as a practical starting point.

### 2.2.1 Counterexample: An unrestricted language

Let us start with a simple example, representing a simple symmetric two argument relation as a string of words in a language. Let  $Z$  be the relation, while  $X$  and  $Y$  are the arguments<sup>2</sup>. In predicate logic we might write the relation we want to express linguistically as  $Z(X, Y)$ .  $Z$  is a predicate which takes two arguments,  $\lambda xy, Z(x, y)$ . Further,  $Z$  is symmetric in its interpretation, so that  $Z(X, Y) \equiv Z(Y, X)$ , and either will do. Let us introduce three words corresponding in meaning to these three entities:

$$\begin{aligned} \llbracket a \rrbracket &\rightarrow X \\ \llbracket b \rrbracket &\rightarrow Y \\ \llbracket c \rrbracket &\rightarrow Z \end{aligned} \tag{2.9}$$

We want to be able to convey the relationship  $Z(X, Y)$  using these symbols  $a$ ,  $b$ ,  $c$ . To begin to see where some structural constraints might arise, let us start by looking at the extreme alternative: a language with no syntactic properties at all. Imagine there exists a language  $\mathcal{L}_0$  in which symbol order and configuration play no part at all in determining the functionality of the language, where I will define *functionality* as being the representational and communicative aspects of language, and which we can loosely refer to as the ‘meaning’ of statements.

Because we have no constraints on configuration, if  $a$ ,  $b$ ,  $c$  are symbols in  $\mathcal{L}_0$  we can state by assumption that the following sentences will have precisely identical interpretation, namely  $Z(X, Y)$ :

$$\begin{aligned} \llbracket a \ b \ c \rrbracket \\ \llbracket a \ c \ b \rrbracket &\Rightarrow Z(X, Y) \\ \llbracket b \ a \ c \rrbracket \end{aligned} \tag{2.10}$$

To ground this, say, for instance, that  $X$  and  $Y$  are two objects in a ‘blocks world,’ two blocks, say, (but more generally any objects, physical or not – e.g.

---

<sup>2</sup>I will use capital italic letters,  $X$ ,  $Y$ , etc. for conceptual entities, and small letters,  $a$ ,  $b$ ,  $c$ , for words in the language

ideas, a chair, the observable property of blueness, etc.) and  $Z$  is some symmetric relation that can exist between two things, such as being near. We could then paraphrase any of the expressions in (2.10) as “the things referred to by  $a$  and  $b$  (i.e.  $X$  and  $Y$ ) are near each other.” The non-configurational assumption allows us to re-order any string and claim that it must retain its interpretation. In this language, the composition of the meaning of the symbols,  $a$ ,  $b$ ,  $c$ , into the meaning of the relationship  $Z(X, Y)$  is accomplished solely through the semantic features of the symbols. Ordering plays no part.

One of the more difficult problems of using a linear language is the representation of ordered, complex, nonlinear relationships. Concerning symmetric relations such as ‘near’, this language  $\mathcal{L}_0$  has adequate expression. But, if the relationship is not symmetric, it fails to represent the desired meaning. Say that  $c$  is a symbol for ‘on top of’ instead of ‘near’. To be able to represent the situation adequately, we must be able to identify which of  $a$  or  $b$  is on top. The immediate result is that this is not possible in  $\mathcal{L}_0$  with the three symbols  $a$ ,  $b$ , and  $c$ . Since the expressions (2.10) are, by assumption, equivalent, it is not possible to have two separate representations for ‘ $a$  on top of  $b$ ’ and ‘ $b$  on top of  $a$ ’.

One could of course resolve this difficulty by imposing an ordering constraint on the symbols, using, for instance, the first object as the one on top, and the other underneath. In declaring that in  $\mathcal{L}_0$ , interpretation is not dependent on configuration, we have eliminated this possible solution.

### 2.2.2 Case mechanisms

What we are missing so far in interpreting this language is a mechanism for identifying the *case* of the terms – generally, their correspondence to the roles of the relation (cf. [?filemore:case?]). For instance, imagine the meaning of  $c$  to be a logical predicate something like  $\lambda z_1 z_2 Z(z_1, z_2)$ . If  $Z$  is non-symmetric, we need a mechanism for separating and identifying the two argument variables,  $z_1, z_2$ .

In the case scenario, we make some fixed assignment between the two variables and a finite set of possible *cases*. The possible argument fillers, *a* and *b* in this example, are then also required to carry case, such that a listener may assign them properly to the roles of the predicate.

The filler/role relationship between *c* and *Z*, being the representation of the relation itself, is not typically identified as a 'case' *per se*, but is differentiated as the verb or predicate. The verb or predicate term is often accorded special significance in structuring the relationship given in the clause (cf. [?vanvalen:RRG?]). This can be justified in that typically, the verb is the term with the greatest number of predictable, yet unspecified relational components (cf. [?montague?]). That is, the verb specifies known variable positions; other words, such as object terms, while they project constraints about what structures they may participate in, are typically less definite about the relationships they will participate in. Since we are already positing that the listener knows the referents for each of these symbols, we can consider the verb to carry a special type of 'predicate case,' which is simply its role as the predicate term.

There are a number of mechanisms by which we can label or assign case to the argument terms in the language, so that they compose properly into the semantic relationship we desire to communicate. However, almost all of them ultimately involve using configuration as a mechanism for making this assignment. Here we review a few mechanisms commonly employed in natural languages.

**Morphological case** Morphological case marking, the mechanism which the general case theory is unfortunately named for, involves a systematic alteration of the base lexical item which identifies the case in which it is to be used. This change can take the form of affixation, infixation, vowel shifting, or other structural morphological change. In our example of  $Z(z_1, z_2)$ , we might have two affixes, one of which  $*_1$  marks its item as being the first argument,  $z_1$ , the other,  $*_2$ , which marks the argument belonging to the second position,  $z_2$ . We would then write

any of the forms (2.11) to express the proper relationship.

$$\begin{array}{ll}
 a_1 & b_2 & c & \text{morphological case for argument assignment} \\
 a_1 & c & b_2 & \\
 b_2 & c & a_1 & \text{(2.11)}
 \end{array}$$

**Functional markers** Similarly, case can be identified externally to the lexical item through the use of prepositions (e.g. English) or post-positional markers (e.g. Japanese). In this strategy, case is identified by associating an external marker with the base item, where the sole purpose of the marker is to identify the case of the fillers so that they may be assigned to the proper functional role in the predication. This is quite similar to an affix mechanism, except that the marker is not considered an integral part of the lexical item, but nevertheless appears adjacent to it as a marker (although other components, e.g. modifiers or quantifiers, may be allowed to interpose).

If we use a prepositional scheme, we might have a marker item  $m_1$  which identifies the following item as the first argument  $z_1$ , and  $m_2$  which identifies the second as  $z_2$ . We can then write any of the forms (2.12) and ensure the proper interpretation.

$$\begin{array}{ll}
 m_1 & a & m_2 & b & c & \text{prepositional marking for argument assignment} \\
 m_1 & a & c & m_2 & b & \\
 m_2 & b & c & m_1 & a & \text{(2.12)}
 \end{array}$$

**Configuration as a case identifier** Case can also be identified purely positionally, through the order and configuration of the set of words that represent the predication. In the simplest situation, it might be that the first argument filler to appear is assigned the first role, the second to the second, and so on for as many



and certainly one that relies on the configuration of the elements involved.

### 2.2.3 The complexity of syntax

We can compare the complexity of the language under various argument mapping or case marking schemes in terms of the number of symbols needed for a given level of expression, and evaluate these representations in terms of their economy of storage. Say that we wish to be able to represent a set  $N$  of objects and a set  $R$  of binary relations, using the terms necessary to represent the objects and predicate elements, and only the minimum necessary to ensure the interpretation of the argument mapping. By evaluating the number of symbols a language user needs to represent the possible relations in the language, we estimate the memory load required to be a speaker of that language.

In the case of the completely non-configurational language above,  $\mathcal{L}_0$ , we need only

$$|\mathcal{L}_0| = |N| + |R| \tag{2.14}$$

total symbols. The language user in this case needs to store (remember) only one term for each object and each relation that could be mentioned. This base estimate is really only a reference point, since the best one can do with this language is talk about unordered, symmetric relations.

This works only for the symmetric relations; solutions for the asymmetric case must still be examined. Before going on to the configuration dependent case marking schemes outlined above, let us explore some other options.

#### Extending the vocabulary

One mechanism for representing the asymmetric predicates without requiring the use of configurational mechanisms is to extend the vocabulary of object terms such that they also represent the possible predicates. In the asymmetric relation 'on top of', we could, perhaps, posit another word,  $a_c$ , which means "a when a

is on top of something.” We now need a second word for every possible object that could be on top of another. And, if we want to generalize this solution to all other asymmetric binary relations, we need to have an extra form of each object for each relation. I should note that this extension requires the new terms to be structurally unrelated to the existing ones, since we’ve already shown that a structural shift to create a new term, such as affixation, is really a configurational mechanism in disguise.

Using the alternate-object approach requires a set of new object terms  $N^+$  whose size is the size of  $N \times R$ , which gives us  $|N^+| = |N||R|$ . A side effect of encoding the relation within the altered form of the object is that the explicit relation term becomes redundant in the expression. We have effectively Skolemized the binary relation, encapsulating the relation and one term into a new relational term which only requires one argument. If  $a_c$  means ‘ $a$  when  $a$  is in relation  $c$  with  $X$ ’, then the expression  $a_c X$  is equivalent to  $a c X$ , and so we can eliminate  $|R|$  redundant terms from the language. The functional denotations of these symbols can be written as the following lambda expressions:

$$\begin{aligned} \llbracket \mathbf{a} \rrbracket &\equiv X \\ \llbracket \mathbf{c} \rrbracket &\equiv \lambda z_1. \lambda z_2. (Z(z_1, z_2) \vee Z(z_2, z_1)) \\ \llbracket \mathbf{a}_c \rrbracket &\equiv \lambda z_2. Z(X, z_2) \end{aligned} \tag{2.15}$$

With the additional terms required for the Skolemized objects, and the original set of unmarked object terms, the size of this new non-configurational form of the language  $\mathcal{L}_1$  is much larger than the original asymmetric case.

$$|\mathcal{L}_1| = |R||N| + |N| \tag{2.16}$$

If we then attempt to extend beyond binary relations, to include more than two arguments, we must expand our vocabulary still further. For each argument we wish to add to the relations, we need another  $|N||R|$  terms to express it with unique symbols. For  $k$ -place relations, then, we need a larger number of symbols:

$$|\mathcal{L}_k| = k|N||R| \tag{2.17}$$

In order to express non-symmetric relations, the language user here needs to store a vastly greater number of terms for each object, on the order of the square of the base estimate.

### Generalizing marked objects

We can generalize this alternate-object approach to a more economic scheme, that requires only twice the original number of terms. In the binary case, if we assign an arbitrary conceptual ordering of the roles for all the binary relations, then we need only have one special form for each object for each role, but not for each relation. The object given in special form can now be taken as the first of the ordered roles in the relation, while the normal form object fills the remaining role. This model of role assignment requires added processing on the part of the interpreter of this language, since each object term must now be assigned to the correct relational role, whereas the symmetric relations needed no assignment, and the Skolemized language encapsulated the assignment so that no further processing was needed. This scheme can be expanded to  $k$ -place relations by having  $k$  families of special forms corresponding to the  $k$  possible relational positions. With  $k$  terms for each object, the size of this language  $\mathcal{L}_{obj}$  is given in eq. (2.18). This is a linear function of our base estimate for the asymmetric case, and perhaps not too great a tax on the mental capacity of the user. So by substituting a small amount of processing, we can reduce the size complexity of the language by a factor of  $O(|N|)$ .

$$|\mathcal{L}_{obj}| = k|N| + |R| \tag{2.18}$$

We can see how this works through an example. Eq. 2.19 gives the interpretation of a group of object-denoting symbols, where the meaning of each is particular to the argument position it will play when composed with a predicate term. The example sentence (2.20) shows how these combine, irrespective of their relative positions or configuration, into the predication.

$$\begin{aligned}
[[a_1]] &\rightarrow X \text{ as argument 1} \\
[[b_2]] &\rightarrow Y \text{ as argument 2} \\
[[c_3]] &\rightarrow Z \text{ as argument 3} \\
[[f]] &\rightarrow \lambda x_1.\lambda x_2.\lambda x_3.F(x_1, x_2, x_3)
\end{aligned} \tag{2.19}$$

$$[[b_2 \ a_1 \ f \ c_3]] \rightarrow F(X, Y, Z) \tag{2.20}$$

### Comparing case-marking systems

Let us compare these systems to the configuration-based case marking systems outlined above, which are modelled after the processes appearing in actual languages. One factor we have not discussed yet is the question of how many case markers or case forms are really needed. One solution, which is very expensive in terms of the number of forms needed, is parallel to the mechanism used in  $\mathcal{L}_k$ . For each predicate, we need a special case marker (either of the functional/prepositional or the morphological shift sort) for each argument position of the predicated relationship. If we count each marker needed to identify the object as a separate term, then this language,  $\mathcal{L}_m$ , needs as many altered forms or marker terms for the objects as there are possible positions in predicates, as given in (2.21).

$$|\mathcal{L}_{m1}| = |N||R| + |N| \tag{2.21}$$

A more economical scheme could be derived, parallel to  $\mathcal{L}_{obj}$ , in which the same special case forms are used across all predicates. Only as many special forms are needed as there are separate argument positions in the predicates. Under this scheme, we would use one prepositional marker or one type of morphological shift for each differentiated argument position, regardless of the predicate used. Since we do not have to generate a term for each predicate, we only need as many

extra items as possible arguments, resulting in the smallest language size yet (2.22).

$$|\mathcal{L}_{case}| = |N| + |R| + k \quad (2.22)$$

This case marked language,  $\mathcal{L}_{case}$ , is very similar to  $\mathcal{L}_{obj}$ , in which each term was multiplied into  $k$  forms for the argument positions. However, here in  $\mathcal{L}_{case}$ , we do not need to generate  $k|N|$  new terms, but only need to introduce *a total* of  $k$  new terms to indicate argument positions. By simply juxtaposing the markers with the object terms, we can identify them as assigned to the proper place in the interpreted relation. This of course requires that the interpretive process is able to compose these adjacent pairs into the proper type of intermediate structure, which is a more complex type of processing than was required by the  $\mathcal{L}_{obj}$  or especially the  $\mathcal{L}_0$  language.

$$\begin{aligned} \llbracket \mathbf{a} \ m_1 \rrbracket &\rightarrow X \text{ as argument 1} \\ \llbracket \mathbf{b} \ m_2 \rrbracket &\rightarrow Y \text{ as argument 2} \\ \llbracket \mathbf{c} \ m_3 \rrbracket &\rightarrow Z \text{ as argument 3} \end{aligned} \quad (2.23)$$

$$\llbracket \mathbf{f} \ \mathbf{b} \ m_2 \ \mathbf{a} \ m_1 \ \mathbf{c} \ m_3 \rrbracket \rightarrow F(X, Y, Z) \quad (2.24)$$

### Complete configuration

In the end, if we rely completely on configuration, we needn't add any new terms at all. By making the interpretation of each predicate dependent on the order of its argument terms, we can use the same object terms, with no modification or marking, regardless of which argument position they are intended for. In the simplest mapping, we could have the first object to appear be used as the first argument, the second object as the second argument, etc. This language,  $\mathcal{L}_{conf}$ , has the same vocabulary size as the original asymmetric language  $\mathcal{L}_0$ .

$$|\mathcal{L}_{conf}| = |N| + |R| \quad (2.25)$$

Interpreting examples of  $\mathcal{L}_{conf}$  might require only simple processing, assigning the objects to argument slots in the order they appear, if something like an operator prefix notation is used. Or perhaps an interpretive semantics similar to the lambda calculus could be used to map arguments for each term (cf. ??).

$$\llbracket \mathbf{f\ b\ c\ a} \rrbracket \rightarrow F(Y, Z, X) \quad (2.26)$$

Comparing the sizes in succession (eq. 2.27) is quite dramatic, showing that an enormous degree of vocabulary storage efficiency is gained through using configurational mechanisms.

$$\begin{aligned} |\mathcal{L}_1| &= |R||N| + |N| \\ |\mathcal{L}_k| &= k|N||R| \\ |\mathcal{L}_m| &= k|R| + |N| \\ |\mathcal{L}_{obj}| &= k|N| + |R| \\ |\mathcal{L}_{case}| &= |N| + |R| + k \\ |\mathcal{L}_{conf}| &= |N| + |R| \end{aligned} \quad (2.27)$$

#### 2.2.4 Sentential interference

Focussing for a moment on the less configurational solutions, it is clear that there are deeper problems than relative argument assignment. Even if we assume the efficient case marking morphology ( $\mathcal{L}_{case}$ ) or distinct case forms ( $\mathcal{L}_{obj}$ ), many problems remain for the asyntactic languages. A particularly difficult question among them is the problem of delimiting relations. So far, we have dealt only with cases having one predicate along with the objects participating in it. What happens when a speaker begins to mention more than one relationship? Some mechanism is needed so that the various relationships are kept separate in interpretation,

and that the components and specifiers of one relationship do not get mixed up in those of the other. This division, although somewhat ill-defined, is treated as the clause or sentence level of syntactic analysis.

One obvious way to define a sentence in  $\mathcal{L}_{case}$  would be to have objects participate in the closest relation for which the role is not filled. While this mechanism does not enforce any strict arrangement on the objects *within* a clause, it is still a configurational constraint, in that it will not allow arbitrary ordering of the terms involved. Intermixing the expressions for two separate relations would confuse the connection between a relational term and its arguments. Swapping two objects with the same case marking will also confuse the interpretation of the two objects.

We will not immediately be able to support multiple relations under non-configuration using either the distinct case  $\mathcal{L}_{obj}$  or the case-marked object terms  $\mathcal{L}_{case}$ . Since neither of these forms carry information about which relation they are to take part in, the possibility of confusion always exists when more than one relation is used. However, we could use  $\mathcal{L}_k$ . Because the Skolemized versions of the object terms carry the identity of their relation with them, multiple relations using this mechanism will not become confused. In (2.28), multiple relationships  $A(A_1, \dots)$ ,  $B(B_1, \dots)$ , and  $C(C_1, \dots)$  can be represented using the symbols in any order, since special object forms are used to indicate participation in each predicate type. When reference needs to be made to two relations *of the same type*, however, it will no longer be able to differentiate between the objects of the two. And still, this solution carries the enormous size burden of requiring a separate form of each object term for each relation that is expressed within the language, compared to only a form for each case type, as in  $\mathcal{L}_{obj}$ , or even fewer for the case marked language  $\mathcal{L}_{case}$ .

$$a_p \ b_p \ c_p \ a_1 \ b_1 \ c_1 \ a_2 \ b_2 \ c_2 \ \dots \tag{2.28}$$

By adding something as simple as a relational delimiter between each clause

or relationship expression, we can solve this problem for the otherwise non-configurational languages. As long as the delimiters can be used to isolate the symbolic representation of each relationship, we can associated the correct objects with each predicate symbol, as in (2.29).

$$a_p \ x_1 \ y_2 \mid x_1 \ b_p \ y_2 \mid y_2 \ x_1 \ c_p \mid \dots \quad (2.29)$$

This simple type of configurational constraint also reduces the amount of storage and coordination necessary to process the language. Let us use just the storage required to process one relation as a measure of processing complexity. Looking at the relations made up of triples of symbols  $(a_p, x_1, y_2)$  in example 2.29, we see that the case-marked language with relational delimiter requires only as much symbol storage to process the sentence as there may be symbols in a single relation. When a delimiter is reached, all the symbols since the last delimiter are known to comprise an entire relation, and no further input storage is needed.

In the skolemized case (2.28), however, an arbitrary number of symbols and relations may intervene between the symbols of any given relation  $(a_1, b_1, c_1)$  in this case). Even though they may have a large memory for events and relationships, human language users obviously have limited memory for strings of symbols (cf. [?miller:magic-number?]), and therefore this sort of configuration *must* be part of the interpretation process. Simple constraints such as the relation delimiters have an enormous effect on the efficiency and complexity of language processing.

### 2.2.5 Benefits of configuration

This exploration of the limits of a non-configuration language may seem a pointless reduction of real languages, *all of which* carry configurational constraints. It serves, though, to highlight the direct connection between configuration, representational capacity, and language size. Using even simple configuration mechanisms for interpreting language results in a more compact possible mental representation.

What's been shown is that there is good reason to believe that syntax will be used in real language situations where storage and processing efficiency are real considerations. This argument may be unnecessary, given that grammar has been recognized and systematically studied since at least the 2<sup>nd</sup> century BC (cf. [Panini]), but reducing the problem to this level provides a basis on which to ground our expectation of real language structure.

The use of ordered syntactic constraints on representation have obvious advantages of storage and processing efficiency, for the goals of representation and interpretation. These constraints allow the semantic relations between symbols to be displayed in their configuration, rather than requiring that the individual relationships between symbols be associated with the symbols themselves, and all the possible relationships known by the language user. A rich variety of possible relationships is permitted using far fewer distinct symbols than any of the non-configurational alternatives.

## 2.3 Expectations

Assuming that real language uses configurational mechanisms such as these for meaning representation, we can begin to infer some of the properties of usage we should see in actual language use.

There is no reason to expect that only one of these configurational solutions will be used. We might expect the same information to be represented by different mechanisms under different circumstances, or by multiple mechanisms at the same time. For instance, (2.30) shows a simple directed relationship, in which the objects, 'he' and 'me', are special case forms of the type outlined in  $\mathcal{L}_{obj}$ . But, we see also that the same information is carried by the ordering of the terms, as in  $\mathcal{L}_{case}$ , since we can construct a similar relationship without case terms (2.31). And we see that the special case forms do not operate independently of configuration; we cannot re-order the case-marked terms arbitrarily (2.32).

$$\text{He hit me} \rightarrow \textit{hit}(\textit{he}, \textit{me}) \quad (2.30)$$

$$\text{Joe hit Bob} \rightarrow \textit{hit}(\textit{joe}, \textit{bob}) \quad (2.31)$$

$$* \text{ Me hit he} \rightarrow \textit{/hit}(\textit{he}, \textit{me}) \quad (2.32)$$

Along with efficiency of representation, it is not a stretch easily that real languages have also optimized over time for communicative efficiency, in terms of simplicity of processing for both speaker and listener, and especially for robustness under noisy conditions. In order to be robust, we might expect to see evidence for the representational structure of a sequence of words encoded in multiple, redundant ways. This is exactly what we do see in real language.

### 2.3.1 Consistent configuration

These configurational mechanisms, as we've shown them, are used to specify the relationships between the meanings of the words, in order that the listener can assemble the meaning of the whole. From this, it follows easily that the configurations in which the word participates all project configurational constraints compatible with the word, and have interpretations compatible with the word. Multiple occurrences of the word will all have lexical environments consistent with interpreting that word.

#### Associative behavior

This is the *associative behavior* of words which we will focus on. That words will have a consistent projection onto syntax implies that there are certain configurations they will be associated with, and only particular words and word classes will occur in those configurations. Because these lexical environments for the word are relatively fixed, dependent on the properties of the word itself, we should be able to identify them by observing actual usage over corpora.

This is the assumption behind the use of the usage distribution of a word as a proxy description for that word's linguistic properties (cf. [Periera *et al.*, 1993, Schütze, 1993a] and others). Since the projection of the word into the space of syntactic usage is governed by its intrinsic lexical properties, then that projection, the distribution of usage, can be used as a representation of those properties.

This is also the basic assumption behind  $n$ -gram word occurrence models. These capture the distribution and probability of the co-occurrence sets for a word, with the implication that the unobserved future behavior of the word will follow the same patterns. Later, when given a co-occurrence pattern or context, the model can be used to accurately predict which words will occur in that context.

### 2.3.2 Associations

Hindle and Rooth [?hindle+rooth:structural?] show this in a model of prepositional phrase attachment, in which co-occurrence statistics are used to decide whether a prepositional phrase modifies a preceding noun or verb. The decision model uses appearance data “as evidence of an underlying relationship, but makes no attempt to determine what sort of relationship is involved.” This result is somewhat surprising, because it implies that we can observe these the presence of these properties even *without requiring a semantic model* of the nature underlying relationships.

It does not require a specific theory of semantic properties and their interactions to note which terms are used in conjunction, and to discover which configurations never appear. It also implies that semantic properties can be attributed without having a detailed model of syntax, and without requiring a parsing theory which decodes these configurations in sequences into semantic relations.

Having a semantic theory certainly does not impede the method, though. Pustejovsky et al. [?pusto:cl1993?], for example, use data of a term's appearance within a syntactic relationship to support an assignment to it semantic properties which

license that relationship under the theory. They also propose that appearances of a term within a structured collection of syntactic configurations is sufficient evidence to posit for the word a detailed semantic structure licensing those configurations.

### 2.3.3 Alternation and classes

If a particular predicative relation is always expressed using the same configuration of terms, one can observe, *by studying the configuration alone*, which groups of symbols are allowed in the relation, and identify their respective roles. This is the basic assumption underlying any attempt at distributional linguistic analysis, whether it is based on numeric probability or symbolic grammar induction techniques.

Harris suggested [?harris:papers:23?] that “morphemes can be grouped into classes in such a way that members of a class have rather similar sets of co-occurents, and each class in turn occurs with specific other classes to make a sentence structure.” We extend that here to say that if a word is observed to be used within a particular set of syntactic configurations, then it shares internal properties with other words that appear within the same set. Words that alternate in similar syntactic positions then, are participating in similar relationships, and have similar intrinsic qualities that license that appearance.

#### Grouping words

We can use those implied similarities to group words into collections with similar or identical properties. This kind of clustering has been demonstrated by a number of studies (e.g. [Brown *et al.*, 1992, Periera *et al.*, 1993] and others). The kind of *distributional smoothing* demonstrated in [Dagan *et al.*, 1993] relies on this type of group behavior. When words are shown to participate in similar syntactic behaviors, they are grouped into a class. Following that, the class can be used

as a proxy for all its members, such that behavior seen for one member can be extrapolated to another word, for which it may never have been observed.

This type of reasoning uses the intrinsic properties of words, as inferred from their syntactic behavior, as the implied basis for further predictions. This is fundamentally different from the  $n$ -gram models, which use only the direct behavioral evidence.

### **Summary**

It is this association between the configuration and the meaning representation of words that we have highlighted in this chapter, and shown to have a reasonable basis in the efficient use of language. It has been shown that this connection provides a sufficient and suitable foundations for models of word use. The further chapters will outline a general model for representing these associations between words and contexts, and for representing groups or classes with similar associations.

The ideal modelling process, in which we can collect corpus usage and identify unique associations for every word or possible context, is however untenable due to the sparse co-occurrence frequencies found in actual usage. We will present some specific techniques for acquiring a class model even with such uncooperative data.