Free Indexation: Combinatorial Analysis and A Compositional Algorithm*

Sandiway Fong
545 Technology Square, Rm. NE43-810, MIT Artificial Intelligence Laboratory, Cambridge MA 02139
Internet: sandiway@ai.mit.edu

Abstract
The principle known as 'free indexation' plays an important role in the determination of the referential properties of noun phrases in the principle-and-parameters language framework. First, by investigating the combinatorics of free indexation, we show that the problem of enumerating all possible indexings requires exponential time. Secondly, we exhibit a provably optimal free indexation algorithm.

1 Introduction
In the principles-and-parameters model of language, the principle known as 'free indexation' plays an important part in the process of determining the referential properties of elements such as anaphors and pronouns. This paper addresses two issues. (1) We investigate the combinatorics of free indexation. By relating the problem to the n-set partitioning problem, we show that free indexation must produce an exponential number of referentially distinct phrase structures given a structure with n (independent) noun phrases. (2) We introduce an algorithm for free indexation that is defined compositionally on phrase structures. We show how the compositional nature of the algorithm makes it possible to incrementally interleave the computation of free indexation with phrase structure construction. Additionally, we prove the algorithm to be an 'optimal' procedure for free indexation. More precisely, by relating the compositional structure of the formulation to the combinatorial analysis, we show that the algorithm enumerates precisely all possible indexings, without duplicates.

2 Free Indexation
Consider the ambiguous sentence:
(1) John believes Bill will identify him

In (1), the pronounal "him" can be interpreted as being coreferential with "John", or with some other person not named in (1), but not with "Bill". We can represent these various cases by assigning indices to all noun phrases in a sentence together with the interpretation that two noun phrases are coreferential if and only if they are coindexed, that is, if they have the same index. Hence the following indexings represent the three coreference options for pronounal "him":²

(2) a. John₁ believes Bill₂ will identify him₁
    b. John₁ believes Bill₂ will identify him₃
    c. *John₁ believes Bill₂ will identify him₂

In the principles-and-parameters framework (Chomsky [3]), once indices have been assigned, general principles that state constraints on the locality of reference of pronouns and names (e.g. "John" and "Bill") will conspire to rule out the impossible interpretation (2c) while, at the same time, allow the other two (valid) interpretations. The process of assigning indices to noun phrases is known as "free indexation," which has the following general form:

(4) Assign indices freely to all noun phrases.

In such theories, free indexation accounts for the fact that we have coreferential ambiguities in language. Other principles interact so as to limit the

²Note that the indexing mechanism used above is too simplistic a framework to handle binding examples involving inclusion of reference such as:

(3) a. We₁ think that I₁ will win
    b. We₁ think that I₁ will win
    c. *We₁ like myself₁
    d. John told Bill that they should leave

Richer schemes that address some of these problems, for example, by representing indices as sets of numbers, have been proposed. See Lasnik [9] for a discussion on the limitations of, and alternatives to, simple indexation. Also, Higginbotham [7] has argued against coindexation (a symmetric relation), and in favour of directed links between elements (linking theory). In general, there will be twice as many possible 'linkings' as indexings for a given structure. However, note that the asymptotic results of Section 3 obtained for free indexation will also hold for linking theory.
number of indexings generated by free indexation to those that are semantically well-formed.

In theory, since the indices are drawn from the set of natural numbers, there exists an infinite number of possible indexings for any sentence. However, we are only interested in those indexings that are distinct with respect to semantic interpretation. Since the interpretation of indices is concerned only with the equality (and inequality) of indices, there are only a finite number of semantically different indexings. For example, "John\textsubscript{1} likes Mary\textsubscript{2}" and "John\textsubscript{23} likes Mary\textsubscript{4}" are considered to be equivalent indexings. Note that the definition in (4) implies that "John believes Bill will identify him\textsuperscript{2}" has two other indexings (in addition to those in (2)):

\begin{enumerate}
\item *John\textsubscript{1} believes Bill\textsubscript{1} will identify him\textsubscript{1}
\item *John\textsubscript{1} believes Bill\textsubscript{1} will identify him\textsubscript{2}
\end{enumerate}

In some versions of the theory, indices are only freely assigned to those noun phrases that have not been coindexed through a rule of movement (Move-a). (see Chomsky [3] (pg.331)). For example, in "Who\textsubscript{1} did John see \{NP\textsubscript{t}\}?", the rule of movement effectively stipulates that "Who" and its trace noun phrase must be coreferential. In particular, this implies that free indexation must not assign different indices to "who" and its trace element. For the purposes of free indexation, we can essentially 'collapse' these two noun phrases, and treat them as if they were only one. Hence, this structure contains only two independent noun phrases.

3 The Combinatorics of Free Indexation

In this section, we show that free indexation generates an exponential number of indexings in the number of independent noun phrases in a phrase structure. We achieve this result by observing that the problem of free indexation can be expressed in terms of a well-known combinatorial partitioning problem.

Consider the general problem of partitioning a set of \( n \) elements into \( m \) non-empty (disjoint) subsets. For example, a set of four elements \{\(w, x, y, z\}\} can be partitioned into two subsets in the following seven ways:

\[
\begin{align*}
\{w, x, y\} \cup \{z\} & \quad \{w, x\} \cup \{y, z\} \\
\{w, x, z\} \cup \{y\} & \quad \{w, y\} \cup \{x, z\} \\
\{w, y, z\} \cup \{x\} & \quad \{w, z\} \cup \{x, y\} \\
\{x, y, z\} \cup \{w\} &
\end{align*}
\]

The number of partitions obtained thus is usually represented using the notation \( \binom{n}{m} \) (Knuth [8]). In general, the number of ways of partitioning \( n \) elements into \( m \) sets is given by the following formula. (See Purdom & Brown [10] for a discussion of (8).)

\[
\binom{n+1}{m+1} = \binom{n}{m} + (m+1)\binom{n}{m+1}
\]

for \( n, m \geq 0 \)

The number of ways of partitioning \( n \) elements into zero sets, \( \binom{n}{0} \), is defined to be zero for \( n > 0 \) and one when \( n = 0 \). Similarly, \( \binom{0}{m} \), the number of ways of partitioning zero elements into \( m \) sets is zero for \( m > 0 \) and one when \( m = 0 \).

We observe that the problem of free indexation may be expressed as the problem of assigning 1, 2, ..., \( n \) distinct indices to \( n \) noun phrases where \( n \) is the number of noun phrases in a sentence. Now, the general problem of assigning \( m \) distinct indices to \( n \) noun phrases is isomorphic to the problem of partitioning \( n \) elements into \( m \) non-empty disjoint subsets. The correspondence here is that each partitioned subset represents a set of noun phrases with the same index. Hence, the number of indexings for a sentence with \( n \) noun phrases is:

\[
\sum_{m=1}^{n} \binom{n}{m}
\]

(The quantity in (7) is commonly known as Bell's Exponential Number \( B_n \); see Berge [2].) The recurrence relation in (6) has the following solution (Abramowitz [1]):

\[
\binom{n}{m} = \frac{1}{m!} \sum_{k=0}^{m} (-1)^{m-k} \binom{m}{k} k^n
\]

Using (8), we can obtain a finite summation form for the number of indexings:

\[
B_n = \sum_{m=1}^{n} \sum_{k=0}^{m} \frac{(-1)^{m-k}}{(m-k)k!} k^n
\]
It can also be shown (Graham [6]) that $B_n$ is asymptotically equal to (10):

\[
m_n \sim \frac{m_n - n - \frac{1}{2}}{\sqrt{\ln n}}
\]

where the quantity $m_n$ is given by:

\[
m_n \ln m_n = n - \frac{1}{2}
\]

That is, (10) is both an upper and lower bound on the number of indexings. More concretely, to provide some idea of how fast the number of possible indexings increases with the number of noun phrases in a phrase structure, the following table exhibits the values of (9) for the first dozen values of $n$:

| NPs | Indexings | NPs | Indexings |
|-----|-----------|-----|-----------|
| 1   | 1         | 7   | 877       |
| 2   | 2         | 8   | 4140      |
| 3   | 5         | 9   | 21147     |
| 4   | 15        | 10  | 678570    |
| 5   | 52        | 11  | 4123597   |
| 6   | 203       | 12  | 4123597   |

4 A Compositional Algorithm

In this section, we will define a compositional algorithm for free indexation that provably enumerates all and only all the possible indexings predicted by the analysis of the previous section.

The PO-PARSER is a parser based on a principles-and-parameters framework with a uniquely flexible architecture ([5]). In this parser, linguistic principles such as free indexation may be applied either incrementally as bottom-up phrase structure construction proceeds, or as a separate operation after the complete phrase structure for a sentence is recovered. The PO-PARSER was designed primarily as a tool for exploring how to organize linguistic principles for efficient processing. This freedom in principle application allows one to experiment with a wide variety of parser configurations.

Perhaps the most obvious algorithm for free indexation is, first, to simply collect all noun phrases occurring in a sentence into a list. Then, it is easy to obtain all the possible indexing combinations by taking each element in the list in turn, and optionally coindexing it with each element following it in the list. This simple scheme produces each possible indexing without any duplicates and works well in the case where free indexation applies after structure building has been completed.

The problem with the above scheme is that it is not flexible enough to deal with the case when free indexing is to be interleaved with phrase structure construction. Conceivably, one could repeatedly apply the algorithm to avoid missing possible indexings. However, this is very inefficient, that is, it involves much duplication of effort. Moreover, it may be necessary to introduce extra machinery to keep track of each assignment of indices in order to avoid the problem of producing duplicate indexings. Another alternative is to simply delay the operation until all noun phrases in the sentence have been parsed. (This is basically the same arrangement as in the non-interleaved case.) Unfortunately, this effectively blocks the interleaved application of other principles that are logically dependent on free indexation to assign indices. For example, this means that principles that deal with locality restrictions on the binding of anaphors and pronouns cannot be interleaved with structure building (despite the fact that these particular parser operations can be effectively interleaved).

An algorithm for free indexation that is defined compositionally on phrase structures can be effectively interleaved. That is, free indexing should be defined so that the indexings for a phrase is some function of the indexings of its sub-constituents. Then, coindexings can be computed incrementally for all individual phrases as they are built. Of course, a compositional algorithm can also be used in the non-interleaved case.

Basically, the algorithm works by maintaining a set of indices at each sub-phrase of a parse tree. Each index set for a phrase represents the range of indices present in that phrase. For example, "Who did John see?" has the phrase structure and index sets shown in Figure 1.

There are two separate tasks to be performed whenever two (or more) phrases combine to form a larger phrase. First, we must account for the possibility that elements in one phrase could be coindexed (cross-indexed) with elements from the other phrase. This is accomplished by allowing indices from one set to be (optionally) merged with distinct indices from the other set. For example, the phrases "[NP John]" and "[VP likes him]" have index sets \{i\} and \{j\}, respectively. Free indexation must allow for the possibilities that "John" and "him" could be coindexed or maintain distinct indices. Cross-indexing accounts for this by optionally merging indices i and j. Hence, we obtain:

\[
\text{(12) a. John, likes him, } i \text{ merged with } j
\]

For expository reasons, we consider only pure indices. The actual algorithm keeps track of additional information, such as agreement features like person, number and gender, associated with each index. For example, irrespective of configuration, "Mary" and "him" can never have the same index.
b. Johni likes himj,  i not merged with j

Secondly, we must find the index set of the aggregate phrase. This is just the set union of the index sets of its sub-phrases after cross-indexation. In the example, "John likes him", (12a) and (12b) have index sets \{i\} and \{i, j\}.

More precisely, let \( I_P \) be the set of all indices associated with the Binding Theory-relevant elements in phrase \( P \). Assume, without loss of generality, that phrase structures are binary branching. Consider a phrase \( P = [P_1 X Y] \) with immediate constituents \( X \) and \( Y \). Then:

1. Cross Indexing: Let \( I_X \) represent those elements of \( I_X \) which are not also members of \( I_Y \), that is, \( (I_X - I_Y) \). Similarly, let \( I_Y \) be \( (I_Y - I_X) \).

   (a) If either \( I_X \) or \( I_Y \) are empty sets, then done.

   (b) Let \( z \) and \( y \) be members of \( I_X \) and \( I_Y \), respectively.

   (c) Either merge indices \( z \) and \( y \) or do nothing.

   (d) Repeat from step (1a) with \( I_X = I_X - \{z\} \) in place of \( I_X \). Replace \( I_Y \) with \( I_Y - \{y\} \) if \( z \) and \( y \) have been merged.

2. Index Set Propagation: \( I_P = I_X \cup I_Y \).

The nondeterminism in step (1c) of cross-indexing will generate all and only all (i.e. without duplicates) the possible indexings. We will show this in two parts. First, we will argue that

---

\( I_X \) and \( I_Y \) are defined purely for notational convenience. That is, the algorithm directly operates on the elements of \( I_X \) and \( I_Y \). The above algorithm cannot generate duplicate indexings. That is, the algorithm only generates distinct indexings with respect to the interpretation of indices. As shown in the previous section, the combinatorics of free-indexing indicates that there are only \( \binom{n}{2} \) possible indexings. Next, we will demonstrate that the algorithm generates exactly that number of indexings. If the algorithm satisfies both of these conditions, then we have proved that it generates all the possible indexings exactly once.

1. Consider the definition of cross-indexing. \( I_X \) represents those indices in \( X \) that do not appear in \( Y \). (Similarly for \( I_Y \).) Also, whenever two indices are merged in step (1b), they are 'removed' from \( I_X \) and \( I_Y \) before the next iteration. Thus, in each iteration, \( z \) and \( y \) from step (1b) are 'new' indices that have not been merged with each other in a previous iteration. By induction on tree structures, it is easy to see that two distinct indices cannot be merged with each other more than once. Hence, the algorithm cannot generate duplicate indexings.

2. We now demonstrate why the algorithm generates exactly the correct number of indexings by means of a simple example. Without loss of generality, consider the right-branching phrase scheme shown in Figure 2.

Now consider the decision tree shown in Figure 3 for computing the possible indexings of the right-branching tree in a bottom-up fashion.

Each node in the tree represents the index set of the combined phrase depending on whether the noun phrase at the same level is cross-
indexed or not. For example, \{i\} and \{i,j\} on the level corresponding to \(NP_j\) are the two possible index sets for the phrase \(P_{ij}\). The path from the root to an index set contains arcs indicating what choices (either to coin-
dex or to leave free) must have been made in order to build that index set. Next, let us just consider the cardinality of the index sets in the decision tree, and expand the tree one more level (for \(NP_k\)) as shown in Figure 4.

Informally speaking, observe that each decision tree node of cardinality \(i\) 'generates' \(i\) child nodes of cardinality \(i\) plus one child node of cardinality \(i+1\). Thus, at any given level, if the number of nodes of cardinality \(m\) is \(c_m\), and the number of nodes of cardinality \(m-1\) is \(c_{m-1}\), then at the next level down, there will be \(mc_m + c_{m-1}\) nodes of cardinality \(m\). Let \(c(n,m)\) denote the number of nodes at level \(n\) with cardinality \(m\). Let the top level of the decision tree be level 1. Then:

\[
(13) \quad c(n+1,m+1) = c(n,m)+(m+1)c(n,m+1)
\]

Observe that this recurrence relation has the same form as equation (6). Hence the algorithm generates exactly the same number of indexings as demanded by combinatorial analysis.

5 Conclusions

This paper has shown that free indexation produces an exponential number of indexings per phrase structure. This implies that all algorithms that compute free indexation, that is, assign indi-
ces, must also take at least exponential time. In this section, we will discuss whether it is possible for a principle-based parser to avoid the combinatorial 'blow-up' predicted by analysis.

First, let us consider the question whether the 'full power' of the free indexing mechanism is necessary for natural languages. Alternatively, would it be possible to 'shortcut' the enumeration procedure, that is, to get away with producing fewer than \(B_n\) indexings? After all, it is not obvious that a sentence with a valid interpretation can be constructed for every possible indexing. However, it turns out (at least for small values of \(n\); see Figures 5 and 6 below) that language makes use of every combination predicted by analysis. This implies, that all parsers must be capable of producing every indexing, or else miss valid interpret-
tations for some sentences.

There are \(B_3 = 5\) possible indexings for three noun phrases. Figure 5 contains example sen-
tences for each possible indexing.\(^9\) Similarly, there are fifteen possible indexings for four noun phrases. The corresponding examples are shown in Figure 6.

Although it may be the case that a parser must be capable of producing every possible indexing, it does not necessarily follow that a parser must enumerate every indexing when parsing a particular sentence. In fact, for many cases, it is possible to avoid exhaustively exploring the search space of possibilities predicted by combinatorial analysis. To do this, basically we must know, a priori, what classes of indexings are impossible for a given sentence. By factoring in knowledge about restrictions on the locality of reference of the items to be indexed (i.e. binding principles), it is possible to explore the space of indexings in a controlled fash-
ion. For example, although free indexation implies that there are five indexings for "John thought [s Tom forgave himself ] ", we can make use of the fact that "himself" must be coindexed with an element within the subordinate clause to avoid gen-

\(^{9}\)To make the boundary cases match, just define \(c(0,0)\) to be 1, and let \(c(0,m) = 0\) and \(c(n,0) = 0\) for \(m > 0\) and \(n > 0\), respectively.

\(^{9}\)PRO is an empty (non-overt) noun phrase element.
erating indexings in which "Tom" and "himself" are not coindexed.\footnote{This leaves only two remaining indexings: (1) where "John" is coindexed with "Tom" and "himself", and (2) where "John" has a separate index. Similarly, if we make use of the fact that "Tom" cannot be coindexed with "John", we can pare the list of indexings down to just one (the second case).}

References

[1] M. Abramowitz & I.A. Stegun, *Handbook of Mathematical Functions*. 1965. Dover.

[2] Berge, C., *Principles of Combinatorics*. 1971. Academic Press.

[3] Chomsky, N.A., *Lectures on Government and Binding: The Pisa Lectures*. 1981. Foris Publications.

[4] Chomsky, N.A., *Some Concepts and Consequences of the Theory of Government and Binding*. 1982. MIT Press.

[5] Fong, S. & R.C. Berwick, "The Computational Implementation of Principle-Based Parsers," *International Workshop on Parsing Technologies*. Carnegie Mellon University. 1989.

[6] Graham, R.L., D.E. Knuth, & O. Patashnik, *Concrete Mathematics: A Foundation for Computer Science*. 1989. Addison-Wesley.

[7] Higginbotham, J., "Logical Form, Binding, and Nominals," *Linguistic Inquiry*. Summer 1983. Volume 14, Number 3.

[8] Knuth, D.E., *The Art of Computer Programming: Volume 1 / Fundamental Algorithms*. 2nd Edition. 1973. Addison-Wesley.

[9] Lasnik, II. & J. Uriagereka, *A Course in GB Syntax: Lectures on Binding and Empty Categories*. 1988. M.I.T. Press.

[10] Purdom, P.W., Jr. & C.A. Brown, *The Analysis of Algorithms*. 1985. CBS Publishing.