Parsing Elliptical Structure[*]

Gregory M. Kobele
Institut für deutsche Sprache und Linguistik
Humboldt Universität zu Berlin

November 28, 2007

1 Introduction

Elliptical sentences are those in which 'a piece has gone missing'. Still, we appear to find these elliptical sentences just as easy to understand as their lengthy brethren.

(1) John wants to play doctor, but Mary doesn't.

(2) John wants to play doctor, but Mary doesn't want to play doctor.

The parsing problem posed by such sentences is that of recovering the meaning of the 'missing piece'. In the case above, we would want the parser to provide us with a representation of 1 from which we would be in a position to compute the meaning of 2 as one of its possible meanings. Whatever the range of possible meanings returned by the parser might be, it must be constrained enough to rule out a reconstruction synonymous with 3.

(3) John wants to play doctor, but Mary doesn't have a fever.

There have been many different proposals in the linguistic literature as to how best to delimit the range of possible meanings. A currently influential

[*]This work was completed while working in the group SIGNES at the laboratoire bordelais de recherche en informatique (LaBRI) as a post-doc employed by INRIA Futurs. My thanks go to all the members of the équipe, and especially Christian Retoré and Sylvain Salvati. This work would have been much worse without Sylvain's insight. I would also like to thank John Hale and Jens Michaelis for comments on an earlier version. As appealing as the alternative would be, the responsibility for any mistakes contained herein rests solely with me.

account has it that the missing piece is syntactically present [4, 8, 19], which reduces the problem of delimiting the range of meanings of elliptical sentences to explaining the conditions under which syntactic structure might be rendered phonologically inert. There are two main approaches to the licensing of deletion. The first, championed recently by Merchant [19], proposes that the phonological content of a phrase may be deleted just in case there is a different, antecedent, phrase, such that the semantic denotations of both phrases are identical.[1] The other option is to take the condition licensing phonological deletion to be a syntactic identity; a subtree may undergo phonological deletion just in case there is another, isomorphic, subtree, which does not undergo phonological deletion. Assuming that syntactic structures are in a functional correspondence with semantic values (as is common), this latter condition is in fact a restricted case of the former.

Many theories of parsing elliptical structures proceed in two stages [4–6, 12, 17, 21, 26].[2] First, a syntactic representation is derived in which ellipsis sites are marked as such, but no representation of their internal structure (or meaning) is given. In order to ascertain whether the sentence is actually well-formed (or whether there is a meaning associated with it), another step is needed, one in which the internal structure (or meaning) of each ellipsis site is reconstructed. In §4, we present a simple algorithm in this vein, and prove its correctness. This two step approach, though correct, suffers from the defect that the recognition problem, when construed as asking the question 'is s meaningful in G', becomes complicated to solve.[3] Our precise presentation of the algorithm makes clear the diagnosis of this problem; the constraints on ellipsis licensing enforced by the first stage are not sufficient to rule out all of the unwanted cases.

In the final section of the paper, §5, we pursue a direct characterization of the well-formed elliptical sentences (in contrast to the two-step approach pursued heretofore). Using context-free tree grammars (CFTGs) [9, 22] under the inside-out (IO) mode of derivation [7] (for which the recognition problem is known to be in LOGCFL [1]), we identify a hierarchy of succes-

[1]This is a slight simplification. The original proposal requires that the existential closures of the respective phrases' denotations be mutually entailing.

[2]Dalrymple et al. [5] and Chung et al. [4] present competence theories, which for present purposes may be identified with extensional descriptions of the parsing algorithm, and not a parsing algorithm in itself. However, their competence theories are couched in this 'two stage' framework, and straightforward implementations would seem to yield algorithms of this type.

[3]This somewhat non-standard formulation of the recognition problem (normally stated as: 'is s assigned a structure in G') is formulated thusly to emphasize that this difficulty is inherent to the two-step approach, and not just the present, syntactic, characterization of it.

sively more complex elliptical dependencies (intuitively similar to the distinction between the 'nested' and 'crossing' dependencies familiar from work on string languages). We show that monadic IO CFTGs are unable to describe the full class of well-formed elliptical structures, and we conjecture that this class is out of the range of IO CFTGs alltogether. Still, (monadic) CFTGs provide a useful approximation of this set, and have several desirable properties, including being closed under intersection with recognizable sets, which means that we can simply incorporate the regular structure of derivations into the statement of the ellipsis licensing condition to obtain a direct, 'one stage' representation of (an approximation of) the grammar. Recent work by Kanazawa [13] has shown how parsing IO CFTGs can be reduced to query evaluation in Datalog, for which there are a wide array of optimizations that can be brought to bear on the problem.

The proofs and definitions presented here are couched in terms of context-free grammars for convenience only. It is straightforward to extend the results presented here to more sophisticated and linguistically informed theories of natural langauge, such as minimalist grammars [25], which also have a regular derivation structure [16]. In particular, although the sets of derivation trees are no longer regular once deletion under identity is added to the CFG or MG frameworks, in terms of the string languages generated, no additional expressive power is gained.

2 Formal Preliminaries

The reader is assumed to be familiar with basic concepts of tree and formal language theory. For completeness we recall a number of them here.

$\mathbb{N}$ is the set $\{0, 1, 2, \ldots\}$ of non-negative integers.

For any set Σ, we denote by Σ^* the set of all finite strings over Σ. For $s \in \Sigma^*$, $|s|$ denotes the length of s, s_i, where $0 \leq i < |s|$, is the i^{th} symbol in s, starting counting at 0, and ϵ is the empty string. A subset $L \subseteq \Sigma^*$ is a (string) language over Σ.

We represent trees using ranked alphabets. A ranked alphabet is a pair $\langle \Sigma, \mathbf{rank} \rangle$, where Σ is a finite set (of node labels), and $\mathbf{rank} : \Sigma \to \mathbb{N}$ is a function assigning to each $\sigma \in \Sigma$ a natural number (the number of children had by a node labelled with σ). We write $\Sigma^{(n)}$ for the subset $\{\sigma \in \Sigma : \mathbf{rank}(\sigma) = n\} \subseteq \Sigma$ of symbols of rank n, and we write $\sigma^{(n)}$ when referring to σ to indicate that $\sigma \in \Sigma^{(n)}$. When no confusion will arise, we identify a ranked alphabet with its carrier set, writing Σ for $\langle \Sigma, \mathbf{rank} \rangle$. If Σ and Δ are disjoint ranked alphabets, then their union $(\Sigma \cup \Delta)$ is the ranked alphabet such that $(\Sigma \cup \Delta)^{(n)} := \Sigma^{(n)} \cup \Delta^{(n)}$.

The set of trees T_Σ over a ranked alphabet Σ is the smallest set T satisfying, for each $n \in \mathbb{N}$, the condition

$$\text{if } \sigma \in \Sigma^{(n)}, \text{ and } t_1, \ldots, t_n \in T, \text{ then } \sigma(t_1, \ldots, t_n) \in T$$

In particular, if $\sigma \in \Sigma^{(0)}$, then $\sigma() \in T$. Instead of $\sigma()$, we will drop the parentheses and write σ. A subset $L \subseteq T_\Sigma$ is a tree language over Σ.

Let $X = \{x_1, x_2, \ldots\}$ be a denumerably infinite set (of variables), and for each $n \in \mathbb{N}$, let $X_n = \{x_1, x_2, \ldots, x_n\}$. (So $X_0 = \emptyset$.) An n-ary context over Σ is an element of $T_\Sigma(X_n)$, which is defined to be the smallest set T such that

1. $X_n \subseteq T$
2. if $\sigma \in \Sigma^{(n)}$, and $t_1, \ldots, t_n \in T$, then $\sigma(t_1, \ldots, t_n) \in T$

In other words, $T_\Sigma(X_n)$ is the closure of the set X_n under the m-ary operation of adding σ^m as the root node, for each $\sigma \in \Sigma$. Note that $T_\Sigma = T_\Sigma(X_0)$. We write C^n to indicate that $C \in T_\Sigma(X_n)$.

An n-ary context C^n can be interpreted as an n-ary function over k-ary contexts in the following way. Given $t_1, \ldots, t_n \in T_\Sigma(X_k)$, we write $C[t_1, \ldots, t_n]$ to indicate the simultaneous substitution of each variable x_i occuring in C with t_i (for $1 \leq i \leq n$), which is defined as per the following.

$$\begin{aligned} x_i[t_1, \ldots, t_n] &= t_i \\ \sigma^{(0)}[t_1, \ldots, t_n] &= \sigma \\ \sigma(y_1, \ldots, y_m)[t_1, \ldots, t_n] &= \sigma(y_1[t_1, \ldots, t_n], \ldots, y_m[t_1, \ldots, t_n]) \end{aligned}$$

If every x_i, $1 \leq i \leq n$, occurs at least once (at most once) in an n-ary context C^n we say that C^n is non-deleting (linear).

Just as with string languages, sets of trees can be classified according to the kinds of abstract computational resources required to recognize them. A bottom-up tree automaton is a finite-state device which creeps up the tree from children to parent, classifying parents in terms of its classification of their children (the state the machine is in at a given node is that node's 'classification'). A tree is accepted by a bottom-up tree automaton just in case the machine ends up at the root in an accepting state. A tree language is regular iff it is the language accepted by a bottom-up tree automaton. Formally, a bottom-up tree automaton is a quadruple $\mathcal{A} = \langle Q, \Sigma, Q_f, \Delta \rangle$, where

- Q is a finite ranked set (of *states*), where each $q \in Q$ has rank 0

- Σ is a ranked alphabet (the vocabulary over which the trees to be recognized are built)
- $Q_f \subseteq Q$ is the set of *final states*
- $\Delta \subseteq (\bigcup_{n\in\mathbb{N}} \Sigma^{(n)} \times Q^n) \times Q$ is the *transition* relation, where given a transition $\langle \sigma^{(n)}, q_1, \ldots, q_n, q\rangle \in \Delta$ we write

$$\sigma(q_1, \ldots, q_n) \Rightarrow q$$

The move relation ($\rightarrow_{\mathcal{A}} \subseteq T_{\Sigma\cup Q} \times T_{\Sigma\cup Q}$) defined by a bottom-up tree automaton $\mathcal{A}$ is such that $t \rightarrow_{\mathcal{A}} t'$ iff there is a transition $\sigma^{(n)}(q_1, \ldots, q_n) \Rightarrow q \in \Delta$ and a linear non-deleting context $C^1 \in T_{\Sigma\cup Q}(X_1)$ such that $t = C[\sigma^{(n)}(q_1, \ldots, q_n)]$ and $t' = C[q]$. We denote with $\rightarrow^*_{\mathcal{A}}$ the reflexive and transitive closure of $\rightarrow_{\mathcal{A}}$. The set of trees recognized (or accepted) by a bottom-up tree automaton $\mathcal{A}$ is the set

$$L(\mathcal{A}) := \{t \in T_\Sigma : \exists q \in Q_f.\ t \rightarrow^*_{\mathcal{A}} q\}$$

3 Ellipsis in CFGs

We begin by defining context-free grammars, and then extending them with an operation of deletion, intended to model one of the basic operations proposed by linguists to account for ellipsis. We then provide an extensional characterization of well-formed derivations in context-free grammars with deletion, which is intended to model one of the basic restrictions imposed by linguists on the operation of deletion, namely, that the deleted material be recoverable from the context of utterance. We conclude by demonstrating that the tree language determined by this property of derivations is not recognized by any bottom-up tree automaton.

3.1 Context-free grammars

A context-free grammar is a quadruple $G = \langle N, T, P, S\rangle$, where N and T are finite, disjoint sets of *non-terminal* and *terminal* symbols respectively, $S \in N$ is the *start* symbol, and $P \subset N \times (N \cup T)^*$ is a finite set of *productions*, where for $\langle A, v\rangle \in P$ we write $A \rightarrow v$. Without loss of generality, we assume all productions in P are in 'Chomsky normal form' (CNF).[4] A production

[4]Our use of the term here is non-standard, as we are allowing empty productions. It is hoped this will not cause undue difficulty.

ρ is in CNF if it has either of the two forms below, for $a \in T \cup \{\epsilon\}$ and $A, B, D \in N$

$$\begin{gathered} A \to a \\ A \to B\ D \end{gathered}$$

Given a CFG G, we define the set of derivation trees of type A, for $A \in N$, by mutual recursion.

- if $\rho = A \to a \in P$, ρ is a derivation tree of type A
- if $\rho = A \to A_1\ A_2 \in P$, and τ_i is a derivation tree of type A_i, then $\rho(\tau_1, \tau_2)$ is a derivation tree of type A

The set of derivation trees of type A is denoted $D_G(A)$, and is a regular subset of T_P. When G is clear from context, we leave it out and write $D(A)$. Given a derivation tree $\tau \in D_G(A)$, it derives the string $\mathit{yield}(\tau)$, where

$$\begin{aligned} \mathit{yield}(\rho) &:= a \quad \text{where } \rho = A \to a \\ \mathit{yield}(\rho(\tau_1, \tau_2)) &:= \mathit{yield}(\tau_1)\mathit{yield}(\tau_2) \end{aligned}$$

The strings of type A are defined to be

$$L_G(A) := \{\mathit{yield}(\tau) : \tau \in D_G(A)\}$$

The language $L(G) := L_G(S)$ of a context-free grammar G is the set of strings of type S.

3.2 Introducing deletion

We take a first step toward modelling ellipsis by introducing context-free grammars with deletion. A CFG with deletion (CFG^E) is a structure of the same type as a CFG, but with a modified notion of a derivation tree. Note that given a CFG $G = \langle N, T, P, S\rangle$, we can thus speak of its *elliptical variant* $G^E = \langle N, T, P, S\rangle$. The definition of being a derivation tree of type A is modified as follows[5]

- if $\rho = A \to a \in P$, then $\rho \in D_{G^E}(A)$
- if $\tau \in D(A)$, then $\textsc{delete}(\tau) \in D_{G^E}(A^E)$
- if $\rho = A \to A_1\ A_2 \in P$, and $\tau_i \in D_{G^E}(A_i) \cup D_{G^E}(A_i^E)$, then $\rho(\tau_1, \tau_2) \in D_{G^E}(A)$

[5]The notation '$\cdot^E$' is intended to be reminiscient of Merchant's 'E-feature' [19].

The language generated by a CFG^E is as before, but with the following case added to the definition of *yield*

$$\mathit{yield}(\text{DELETE}(\tau)) := \epsilon$$

Given $G^E = \langle N, T, P, S\rangle$ a CFG^E, we construct an equivalent CFG $Tr(G^E) = \langle N \cup N^E, T, P \cup P^E, S\rangle$ as follows:

$$\begin{aligned} N^E &:= \{A^E : A \in N\} \\ P^E &:= \{A^E \to \epsilon\} \\ &\quad \cup \{A \to A_1\ A_2^E : A \to A_1\ A_2 \in P\} \\ &\quad \cup \{A \to A_1^E\ A_2 : A \to A_1\ A_2 \in P\} \\ &\quad \cup \{A \to A_1^E\ A_2 : A \to A_1\ A_2 \in P\} \end{aligned}$$

Theorem 1 Given G^E a CFG^E,

$$L(G^E) = L(Tr(G^E))$$

Proof: We show that for any $A \in N$, $L_{G^E}(A) = L_{Tr(G^E)}(A)$. First note that for any $A \in N$, $L_{G^E}(A^E) = \{\epsilon\}$. The same is true of $Tr(G^E)$, by definition (the only rules with A^E on the left hand side are of the form $A^E \to \epsilon$). The proof that $L_{G^E}(A) = L_{Tr(G^E)}(A)$ proceeds by induction on the height of the derivation. We show the inclusion $L_{G^E}(A) \subseteq L_{Tr(G^E)}(A)$, the other direction is similar. For the base case, note that P is included in the productions of $Tr(G^E)$. Now assume that for all derivations $\tau \in D_{G^E}(B)$ of height less than n, there is $\tau' \in D_{Tr(G^E)}(B)$ such that $\mathit{yield}(\tau) = \mathit{yield}(\tau')$. Let $\tau = (A \to A_1\ A_2)(\tau_1, \tau_2) \in D_{G^E}(A)$ of height n. There are four cases, as per whether $\tau_i \in D_{G^E}(A_i)$ or $\tau_i \in D_{G^E}(A_i^E)$. If $\tau_1 \in D_{G^E}(A_1)$ and $\tau_2 \in D_{G^E}(A_2^E)$, let τ_1', τ_2' be their $Tr(G^E)$ equivalents by the inductive hypothesis ($\tau_2' = A_2^E \to \epsilon$). Then $\tau' = (A \to A_1\ A_2^E)(\tau_1', \tau_2')$. The other cases are not interestingly different. $\square$

Letting $\mathcal{L}_{\mathbf{CFG}}$ and $\mathcal{L}_{\mathbf{CFG^E}}$ denote the languages of context-free grammars and their elliptical variants respectively, the following is a corollary of theorem 1.

Corollary 1

$$\mathcal{L}_{\mathbf{CFG^E}} \subseteq \mathcal{L}_{\mathbf{CFG}}$$

3.3 Constraining deletion

As we have implemented it here, deletion is ‘free’; there are no constraints on its application. In order to better approximate the situation in natural language (i.e. to rule out the meaning of sentence 3 as a possible interpretation of sentence 1), we need to be able to rule out those derivations in which subderivations of type A^E appear illegitimately. Assuming the simpler syntactic identity theory of ellipsis licensing outlined in §1, the property of being a derivation of type S in which a subderivation $\textsc{delete}(\tau)$ occurs only if there is a distinct subderivation τ of the same type which is not immediately dominated by $\textsc{delete}$ is not a recognizable one (theorem 2).[6]

Let $\textsc{Con}^n \subset T_\Sigma(X_n)$ be the set of linear non-deleting n-ary contexts over Σ. Note that, given a linear non-deleting context $C \in \textsc{Con}^1$ and a tree $\tau \in T_\Sigma$, there at most one $\tau' \in T_\Sigma$ such that $C[\tau'] = \tau$. This allows us to view the set of linear non-deleting contexts $\textsc{Con}^1$ as specifiying occurances of subtrees in τ. Given a tree $\tau \in T_\Sigma$, the set of linear non-deleting contexts partition naturally according to whether they represent different occurances of the same subtree, τ'.

$$C \approx_\tau C' \text{ iff } \exists \tau'.\ C[\tau'] = C'[\tau'] = \tau$$

We define $\textsc{Con}_\tau$ to contain just those contexts C in $\textsc{Con}^1$ for which there exists a $\tau' \in T_\Sigma$ such that $C[\tau'] = \tau$.

Now we can define the property of ‘deletion under identity with an overt antecedent’, which holds of a tree $\tau \in T_{\Sigma \cup \{\textsc{delete}\}}$ iff both of the following conditions are satisfied.

1. For every $C \in \textsc{Con}_\tau$, if the parent of x_1 in $C[x_1]$ is $\textsc{delete}$, then there is an equivalent (w.r.t. $\approx_\tau$) $C' \in \textsc{Con}_\tau$ such that $C \neq C'$.

2. For every $C \in \textsc{Con}_\tau$, there is some equivalent $C' \in \textsc{Con}_\tau$ such that the parent of x_1 in $C'[x_1]$ is *not* $\textsc{delete}$.

Condition 1 requires that deletion be under identity, and condition 2 that it be recoverable from the context of speech (e.g. ruling out the case where one ellipsis site serves as the antecedent for another, which reciprocates). Let $\textsc{DelId} \subset T_{\Sigma \cup \{\textsc{delete}\}}$ be the set of trees over $\Sigma \cup \{\textsc{delete}\}$ meeting conditions 1 and 2.

Theorem 2 $\textsc{DelId}$ is not recognizable.

[6]Moving to the semantic licensing theory only complicates matters.

Proof: By contradiction. Let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$, and let $\mathcal{A}$ be a regular tree automaton with n states, accepting exactly those trees which meet conditions 1 and 2. Let $t \in T_\Sigma$ be the tree below

$$\overbrace{g(\ldots(g}^{(n+1)\text{ times}}(a))\ldots)$$

The tree $f(t, \textsc{delete}(t))$ meets 1 and 2, and so is by definition accepted by $\mathcal{A}$. By familiar arguments, there is some strictly larger $t' \in T_\Sigma$ such that $\mathcal{A}$ is in the same state after processing t' as t. Therefore, $f(t, \textsc{delete}(t')) \in L(\mathcal{A})$, however it violates condition 1, contradicting our assumption that $L(\mathcal{A}) =$ DelId. □

4 Parsing CFG^Es

The practical interest of theorem 2 is that is tells us that conventional CFG parsing technology does not suffice to parse CFG^Es (the derivation tree sets of CFGs are all recognizable), thus providing formal justification for the two-step processes proposed previously. In this section we provide an abstract characterization of these two-step parsing proposals. The first step determines a set of underspecified trees, which represent where an ellipsis site might be, and the type of the constituent elided, but not the identity of elided constituents. This set, being regular, can be obtained using standard CFG parsing algorithms (indeed, it is simply the set of parses of s in the translated grammar, $Tr(G^E)$, as shown in theorem 3). The next step decodes each underspecified tree obtained in the first step into a set of parse trees in the CFG^E grammar. For each underspecified tree, there are a finite (and exponentially bounded) number of fully specified trees that it encodes. This ambiguity seems to represent a genuine feature of natural language; sentence 4 can be read as synonymous with either of 5 or 6. From our perspective, this is due to the fact that there are two possible antecedent derivations of the required type, and the grammar permits both.

(4) John wants to be considered hip, but he isn't.

(5) John wants to be considered hip, but he isn't hip.

(6) John wants to be considered hip, but he isn't considered hip.

As stated, the description of the parsing process proceeds in two stages. We begin with a string s, and a grammar G^E. In the first stage, we specify a set of trees $L(\mathcal{A}_s)$ by means of a regular tree automaton (which can be thought of as specifying the chart constructed by a chart parser). In the second step we specify, for each $t \in L(\mathcal{A}_s)$, a set $L(\mathcal{A}_t)$ of derivation trees of type S. The set $\textsc{Parse}(s, G^E) = \{\tau \in D_{G^E}(S) : \mathit{yield}(\tau) = s\}$ is the set of all possible parses of s in G^E, and is related to $\mathcal{A}_s$ and $\mathcal{A}_t$ in the following manner (theorem 4):

$$\textsc{Parse}(s, G^E) = \bigcup_{t \in L(\mathcal{A}_s)} L(\mathcal{A}_t)$$

4.1 Constructing $\mathcal{A}_s$

Here we describe the result of the first stage of the parsing process, the set of underspecified trees $L(\mathcal{A}_s)$. We define $\mathcal{A}_s$ indirectly as the machine recognizing $L(\mathcal{A}_s^{Tr}) \cap L(\mathcal{A}^E)$ (as $\mathcal{A}_s^{Tr}$ and $\mathcal{A}^E$ are bottom-up tree automata, $\mathcal{A}_s$ is constructively determined from their presentation). Whereas $\mathcal{A}_s^{Tr}$ will recognize any parse of s with possible ellipsis sites underspecified, $\mathcal{A}^E$ will accept only those underspecified trees in which the postulated ellipsis sites are licensed (by the presence of an antecedent which doesn't contain the ellipsis site so postulated). In other words, $\mathcal{A}^E$ recognizes only trees with licensed ellipsis sites, irrespective of their yield, and $\mathcal{A}_s^{Tr}$ recognizes only trees with yield s, irrespective of their ellipsis sites.

Given a CFG^E $G^E = \langle N, T, P, S\rangle$, and a string $s \in T^*$, we construct the bottom-up tree automaton $\mathcal{A}_s^{Tr} = \langle Q, \Sigma, Q_f, \Delta\rangle$, which we will present in terms of the CKY parsing algorithm for simplicity.

- A state $q = \langle i, j, A\rangle$ is a triple, where $0 \leq i, j \leq |s|$, and $A \in N$

$$Q := \{\langle i, j, A\rangle : 0 \leq i \leq j \leq |s| \wedge A \in N\}$$

- A state is final iff it of the form $\langle 0, |s|, S\rangle$

$$Q_f := \{\langle 0, |s|, S\rangle\}$$

- The output vocabulary Σ consists of the transitions P, as well as the elements of the set $\{x_A : A \in N\}$

- The transition relation Δ is the smallest set containing the transitions

  - For every $A \in N$, and every $0 \leq i \leq |s|$,

$$x_A \Rightarrow \langle i, i, A\rangle$$

– for every production $\rho = A \to \epsilon \in P$, and every $0 \le i \le |s|$

$$\rho \Rightarrow \langle i, i, A\rangle$$

– for every production $\rho = A \to a \in P$, with $a = s_i$

$$\rho \Rightarrow \langle i, i+1, A\rangle$$

– for every production $\rho = A \to B\ D \in P$, and every pair of states $q_B = \langle i, j, B\rangle$ and $q_D = \langle j, k, D\rangle$,

$$\rho(q_B, q_D) \Rightarrow \langle i, k, A\rangle$$

The automaton so constructed will recognize trees of the form in figure 1 given the obvious grammar on the input "Mary will not."

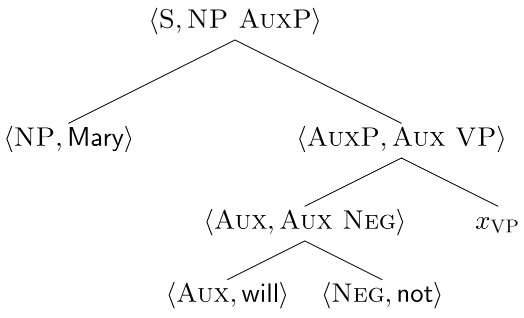


Figure 1: A tree recognized by $\mathcal{A}_s^{Tr}$ but not by $\mathcal{A}^E$

The trees in $L(\mathcal{A}_s)$ are in fact isomorphic to the parses of s in $Tr(G^E)$.

Theorem 3 For $G^E \in \mathbf{CFG^E}$, and $s \in T^*$,

$$L(\mathcal{A}_s^{Tr}) \cong \textsc{Parse}(s, Tr(G^E))$$

Proof: It is straightforward to give a homomorphism $h^{E\to Tr}$ mapping trees in $L(\mathcal{A}_s^{Tr})$ to those in $\textsc{Parse}(s, Tr(G^E))$, and in the other direction, the two state transducer $h^{Tr\to E}$ with the following productions suffices, where $A_i' = A_i^E$ if $q_i = q^E$, and A_i otherwise:

$$\begin{aligned}(A \to a) &\Rightarrow q(A \to a)\\ (x_A) &\Rightarrow q^E(A^E \to \epsilon)\\ (A \to A_1\ A_2)(q_1(\tau_1), q_2(\tau_2)) &\Rightarrow q((A \to A_1'\ A_2')(\tau_1, \tau_2))\end{aligned}$$

It is easy to verify that $h^{E\to Tr}$ is the inverse of $h^{Tr\to E}$. Constructing a tree automaton $\mathcal{A}_{Tr(G^E)}$ for the set $\textsc{Parse}(s, Tr(G^E))$ in the style of the CKY algorithm yields an automaton nearly isomorphic with $\mathcal{A}_s^{Tr}$: the states and productions of $\mathcal{A}_{Tr(G^E)}$ reflect the fact that the nodes of the trees recognized by $\mathcal{A}_{Tr(G^E)}$ are labelled by productions like $A^E \to \epsilon$ instead of x_A and by e.g. $A \to A_1\ A_2^E$ instead of $A \to A_1\ A_2$. As the maps between trees are symbol-to-symbol, verification that for $t \in L(\mathcal{A}_{Tr(G^E)})$, $h^{Tr\to E}(t) \in L(\mathcal{A}_s^{Tr})$ and vice versa can proceed by induction on the heights of the respective trees. $\square$

As shown by theorem 3, the trees accepted by $\mathcal{A}_s^{Tr}$ include those which violate conditions 1 and 2. The automaton $\mathcal{A}^E$ verifies that postulated ellipsis sites have antecedents (condition 1) which are not themselves deleted (condition 2). Furthermore, anticipating the actions of the $\mathcal{A}_t$, $\mathcal{A}^E$ prohibits postulated ellipsis sites taking as their licensing antecedents subtrees which contain them. Given G^E, we define $\mathcal{A}^E = \langle Q, \Sigma, Q_f, \Delta\rangle$, where

- A state $q = \langle A, \alpha, \alpha^E\rangle$ is a triple, where $A \in N$, and $\alpha, \alpha^E \subseteq N$ are the sets of types of antecedents made available by the scanned subtree, and the sets of types of antecedents required by the scanned subtree, respectively.

$$Q := \{\langle A, \alpha, \alpha^E\rangle : A \in N \wedge \alpha, \alpha^E \subseteq N\}$$

- A state is final iff it of the form $\langle S, \sigma, \emptyset\rangle$, for $\sigma \subseteq N$.

$$Q_f := \{\langle S, \sigma, \emptyset\rangle : \sigma \subseteq N\}$$

- The output vocabulary Σ consists of the transitions P, as well as the elements of the set $\{x_A : A \in N\}$

- The transition relation Δ is the smallest set containing the transitions

  – For every $A \in N$,
  $$x_A \Rightarrow \langle A, \emptyset, \{A\}\rangle$$

  – for every production $\rho = A \to a \in P$
  $$\rho \Rightarrow \langle A, \{A\}, \emptyset\rangle$$

  – for every production $\rho = A \to B\ D \in P$, and every pair of states $q_B = \langle B, \beta, \beta^E\rangle$ and $q_D = \langle D, \delta, \delta^E\rangle$,
  $$\rho(q_B, q_D) \Rightarrow \langle A, (\beta \cup \delta \cup \{A\}), ((\beta^E - \delta) \cup (\delta^E - \beta))\rangle$$

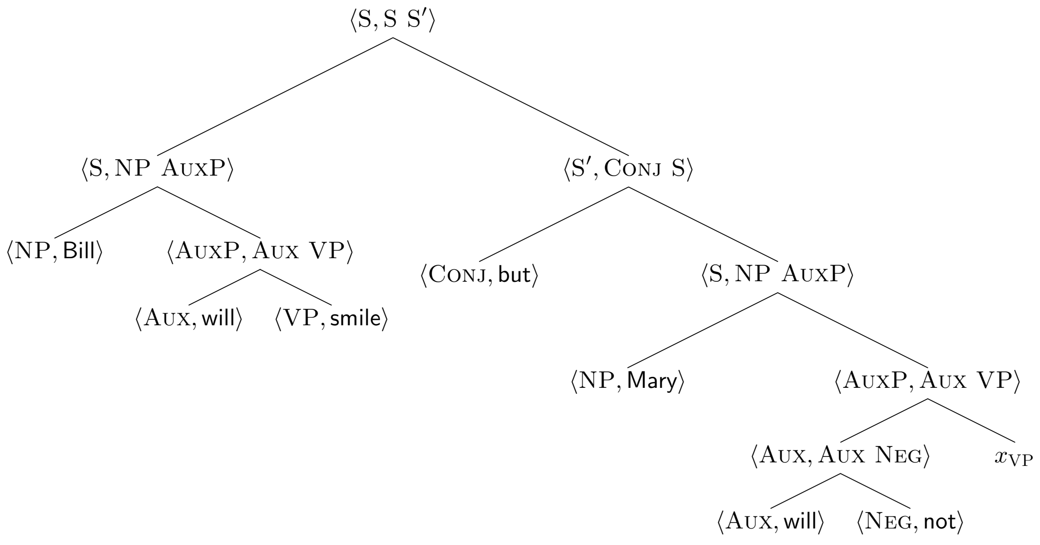


Figure 2: A tree recognized both by $\mathcal{A}_s^{Tr}$ and $\mathcal{A}^E$

The automaton so constructed will rule out trees as in figure 1, in which the postulated ellipsis site does not have an antecedent. Note that it will recognize trees of the form in figure 2. In the tree in figure 2, the automaton will be in state q after recognizing the left daughter of the root, and in state r after recognizing the right daughter, where

$$q = \langle \text{S}, \{\text{NP}, \text{Aux}, \text{VP}, \text{AuxP}\}, \emptyset \rangle$$
$$r = \langle \text{S}', \{\text{NP}, \text{Aux}, \text{AuxP}, \text{Conj}, \text{S}'\}, \{\text{VP}\} \rangle$$

4.2 Constructing $\mathcal{A}_t$

The trees accepted by $\mathcal{A}_s$ are possible parses of the input string s in G^E, in that they indicate all the non-elliptical structure of s, as well as where ellipsis under identity with an overt antecedent has taken place, but they do not represent the internal structure of these ellipsis sites. Given a tree $t \in L(\mathcal{A}_s)$, the automaton $\mathcal{A}_t$ accepts all and only those derivations of type S which t can be instantiated by specifying the internal structure of its ellipsis sites. Given t, let e be an enumeration of occurances of variables in t. We construct $\mathcal{A}_t = \langle Q, \Sigma, Q_f, \Delta \rangle$ as follows:

- A state $q = \langle A, l, v \rangle$ is a triple, where $A \in N$, l is an address of a node in t, and v is an $|e|$-ary sequence consisting of occurances of subtrees in t

and a special symbol, $\perp$. The intuition behind v is that the i^{th} variable occurance e_i is instantiated as the subtree occuring at v_i, regardless of how many times it is copied as part of the instantiation of other variables.

$$Q := \{\langle A, l, v\rangle : A \in N, l \in \textsc{Con}_t, v \in (\textsc{Con}_t \cup \{\perp\})^{|e|}\}$$

- $\Sigma = P$
- A state $q \in Q$ is final iff $l = x_1$ the context for t itself

$$Q_f := \{\langle A, x_1, v\rangle : A \in N, v \in (\textsc{Con}_t \cup \{\perp\})^{|e|}\}$$

- The transition relation Δ is the smallest set containing the transitions
  - For $C \in \mathbf{Con}_t$ such that $C[A \to a] = t$
    $$(A \to a) \Rightarrow \langle A, C, \perp^{|e|}\rangle$$
  - For $v_i = \textbf{if } v'_i = \perp \textbf{ then } v''_i \textbf{ else if } (v''_i = \perp \vee v'_i = v''_i) \textbf{ then } v'_i$, and for some t_1, t_2, $C[(A \to A_1\ A_2)(x_1, t_2)] = C_1$ and $C[(A \to A_1\ A_2)(t_1, x_1)] = C_2$
    $$(A \to A_1\ A_2)(\langle A_1, C_1, v'\rangle, \langle A_2, C_2, v''\rangle) \Rightarrow \langle A, C, v\rangle$$
  - For $C' = e_i$, $v'_i = C$, and either $v_i = C$ or $v_i = \perp$
    $$\textsc{delete}(\langle A, C, v\rangle) \Rightarrow \langle A, C', v'\rangle$$

The transition for the case of DELETE can be thought of in terms of 'jumps'; the automaton starts out thinking it is recognizing a particular subtree of t, but then, upon encountering a node labelled DELETE, realizes that it was in fact recognizing an instance of one of the x_A in t. It then 'jumps' (by changing C to C') to the location of the x_A that it has decided it was in fact recognizing, and continues.

Theorem 4 For any $G^E \in CFG^E$ and $s \in T^*$,

$$\textsc{Parse}(s, G^E) = \bigcup_{t \in L(\mathcal{A}_s)} L(\mathcal{A}_t)$$

Proof: For the inclusion in the left-to-right direction, note that, for every $\tau \in \textsc{Parse}(s, G^E)$, the result of replacing each subtree rooted in a node labeled DELETE by the appropriate x_A is a tree $t \in L(\mathcal{A}_s)$. Therefore, $\tau \in L(\mathcal{A}_t)$. For the other direction, let $t \in L(\mathcal{A}_s)$, and $\tau \in L(\mathcal{A}_t)$. By the correctness of the CKY algorithm, *yield*$(\tau) = s$. Furthermore, each subtree rooted in a node labeled DELETE in τ has an overt identical antecedent, and therefore $\tau \in \textsc{Parse}(s, G^E)$. □

4.3 Discussion

Although correct, the two-step approach presented here relies on the second, instantion, step, to rule out 'erroneously accepted' underspecified trees. In particular, while the construction of $\mathcal{A}_s$ (correctly!) rules out cases in which a variable is licensed by a subtree which contains it (which would yield infinite regress, as much discussed in the linguistics literature [2, 10, 18, 23]), there are other cases of bad antecedent choice which are not ruled out by $\mathcal{A}_s$, but are instead left to $\mathcal{A}_t$ to deal with. Consider figure 3, which might be an underspecified tree for a parse of, say, the empty string. This tree, t, is

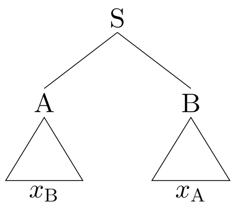


Figure 3: Infinite regress

accepted by $\mathcal{A}_s$ (x_{B} has an antecedent which does not contain it, as does x_{A}).[7] However, $L(\mathcal{A}_t) = \emptyset$. This is problematic, as it entails that we cannot answer the membership problem by determining whether $L(\mathcal{A}_s) = \emptyset$, as is standard. Instead, we need to evaluate $\mathcal{A}_t$ for each $t \in L(\mathcal{A}_s)$, until we find some t such that $L(\mathcal{A}_t) \neq \emptyset$. In the next section, we explore how to give a direct characterization of the well-formed parse trees for elliptical sentences; in effect, deforesting the two-step procedure described here.

5 Towards a direct characterization of the licensing conditions on ellipsis

The problem unearthed in the previous section was that our regular characterization of the relative positions of ellipsis sites and their antecedents (embodied in $\mathcal{A}_s$) was not strong enough to correctly rule out unwanted cases. Here we embark on the project of giving a direct characterization of

[7]Note that we cannot improve on this situation; even though there are only finitely many 'ellipsis types' (x_{A}'s), the problem is not that an x_{A} is instantiated as something with an ellipsis site of type A, but rather that an x_{A} is instantiated as something which itself gets instantiated as something that contains *that very occurrance of* x_{A}. Thus we need to keep track of tokens of variables, not of types. As there is no upper bound on the number of variable tokens in a given sentence, our current finite-state model cannot rule out infinite regress in the general case.

the licensing conditions on ellipsis, of the set DelId. We will use context-free tree grammars [9, 22]. The appeal of using a CFTG to describe the distribution of deleted structure is that CFTGs are closed under intersection with regular tree languages (under both OI and IO derivation modes, as shown in Fischer [9]). This entails that given a grammar G with regular derivation structures (as have CFGs, LCFRSs, MGs, TAGs, etc [16, 27]), and a CFTG G' which enforces certain constraints on ellipsis, we can effectively construct a CFTG with defines just those well-formed derivation structures in G that meet the conditions on ellipsis licensing enforced by G'.

A context-free tree grammar $G = \langle N, T, P, S\rangle$ is a quadruple where

- N and T are ranked alphabets of *non-terminal* and *terminal* symbols respectively. $S \in N^{(0)}$ is the *start symbol.*
- P is a finite set of *productions* of the form

$$A(x_1, \ldots, x_k) \Rightarrow C$$

  where $A \in N^{(k)}$, and $C \in T_{N\cup T}(X_k)$

A CFTG G is k-ary just in case $N^{(n)} = \emptyset$, for all $n > k$. Given trees $t, t' \in T_{N\cup T}$, t *derives* t' *in one step* just in case there is a linear non-deleting context $C \in T_{N\cup T}(X_1)$, a production $A(x_1, \ldots, x_k) \Rightarrow D \in P$, and trees $t_1, \ldots, t_k \in T_{N\cup T}$ such that $t = C[A(t_1, \ldots, t_k)]$ and $t' = C[D[t_1, \ldots, t_k]]$. A tree $t \in T_{N\cup T}$ derives another $t' \in T_{N\cup T}$ in one step *in inside-out derivation mode* just in case t derives t' in one step as above, and the $t_1, \ldots, t_k$ do not contain any non-terminals (i.e. $t_1, \ldots, t_k \in T_T$).

We consider, as our first example, the 0-ary context-free tree grammar $G_{T_\Sigma} = \langle\{S^{(0)}\}, \{f^{(2)}, a^{(0)}\}, \{S \Rightarrow a,\ S \Rightarrow f(S, S)\}, S\rangle$, which derives the tree language T_Σ, for $\Sigma = \{f^{(2)}, a^{(0)}\}$. Writing $\rightarrow$ for the derives in one step relation, we derive the tree $f(f(a, a), a)$ as follows:

$$S \rightarrow f(S, S) \rightarrow f(S, a) \rightarrow f(f(S, S), a) \rightarrow f(f(S, a), a) \rightarrow f(f(a, a), a)$$

It is well known that all and only recognizable tree languages are defined by 0-ary CFTGs (which are then called regular tree grammars).

Consider the unary CFTG $G_{ww} = \langle\{D^{(1)}, S^{(0)}\}, \{f^{(2)}, a^{(0)}\}, P, S\rangle$, where P contains the following productions

$$\begin{array}{c} S \Rightarrow D(S) \qquad S \Rightarrow a \\ D(x_1) \Rightarrow f(x_1, x_1) \end{array}$$

In inside-out derivation mode, G_{ww} derives the tree language $BinTree \subset T_\Sigma$ of complete and balanced binary trees.

$$S \rightarrow D(S) \rightarrow D(D(S)) \rightarrow D(D(a)) \rightarrow D(f(a, a)) \rightarrow f(f(a, a), f(a, a))$$

We examine the step $D(D(a)) \to D(f(a,a))$. Recall that a tree t derives t' in one step just in case there is a linear non-deleting context C^1, a production $\rho = A(x_1,\ldots,x_k) \Rightarrow C'$, and trees $t_1,\ldots,t_k$ such that $t = C[A(t_1,\ldots,t_k)]$ and $t' = C[C'[t_1,\ldots,t_k]]$. In our example, $C = D(x_1)$, $\rho = D(x_1) \Rightarrow f(x_1,x_1)$, and $t_1 = a$. Then $t = C[D(a)]$, and $t' = C[f(x_1,x_1)[a]] = C[f(a,a)]$. Note that, as we are in inside-out derivation mode, we could not have chosen $C = x_1$, and $t_1 = D(a)$ (which would yield $t = C[D(D(a))] = D(D(a)) \to f(D(a),D(a)) = C[f(D(a),D(a))] = C[f(x_1,x_1)[D(a)]]$), as t_1 is not a tree over T_T, containing as it does the non-terminal D.

As a first attempt to directly characterizing the set DelId, of trees over $T_{\Sigma\cup\{\text{DELETE}\}}$ which satisfy conditions 1 and 2, we define the monadic CFTG $G_1 = \langle N,T,P,S\rangle$, where

- $N = \{S^{(0)}, A^{(1)}, B^{(1)}, D^{(1)}\}$
- $T = \Sigma \cup \{\text{DELETE}\}$
- P contains the following productions:
  - $S \Rightarrow \sigma^{(n)}(S,\ldots,S)$
  - $S \Rightarrow D(S)$
  - $D(x_1) \Rightarrow \sigma^{(n)}(S_1,\ldots,S_n)$, where exactly one S_i is $A(x_1)$, at least one is $B(x_1)$, and the rest are S
  - $A(x_1) \Rightarrow x_1$
  - $A(x_1) \Rightarrow \sigma^{(n)}(S_1,\ldots,S_n)$, where exactly one S_i is $A(x_1)$, and the rest can be either S or $B(x_1)$
  - $B(x_1) \Rightarrow \text{DELETE}(x_1)$
  - $B(x_1) \Rightarrow \sigma^{(n)}(S_1,\ldots,S_n)$, where at least one S_i is $B(x_1)$, and the rest are S

The inside-out derivations of G_1 can be described in the following way. When a subtree of the shape $D(t)$ appears, for $t \in T_{\Sigma\cup\{\text{DELETE}\}}$, it is rewritten as, without loss of generality, $f(A(t),B(t))$. A subtree of the form $A(t)$ expands as the set of trees in which the subtree t occurs at least once (in particular, $A(t) = \{C[t] : C \in \text{Con}^1\}$). A subtree of the form $B(t)$, on the other hand, expands as the set of trees in which the subtree t occurs at least once as the child of a node labeled DELETE. In other words, a subtree of the shape $D(t)$ will derive the minimal subtree containing all ellipsis sites (generated by $B(t)$ and $A(t)$) together with the occurance of t serving as their antecedent (generated by $A(t)$). It is clear that G_1 only generates well-formed (according to conditions 1 and 2) elliptical structures.

Theorem 5

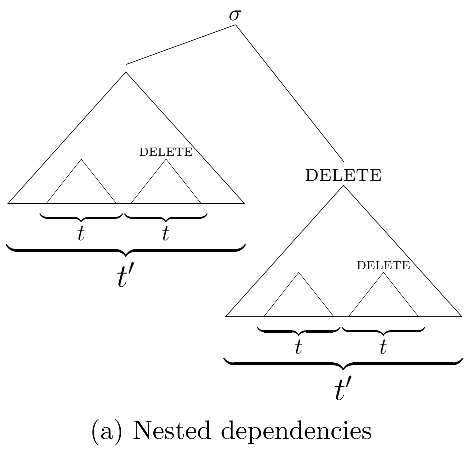
$$L(G_1) \subset \textsc{DelId}$$

Proof: To see the inclusion, let $t \in L(G_1)$. Clearly, t satisfies condition 1, as any subtree t' whose parent is DELETE must have been introduced by a derivation step which rewrote $B(t')$ into $\textsc{delete}(t')$, and $B(x_1)$ can be initially introduced into a sentential form only with a $D(x_1)$ production or an $A(x_1)$ production, both of which guarantee the existence of another occurance of x_1, one which is not dominated by DELETE (condition 2).

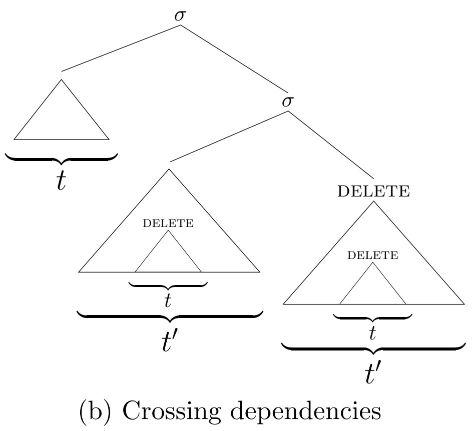
To see that the inclusion is proper, note that G_1 cannot generate trees in which an elliptical constituent contains another ellipsis term's antecedent, or elided part, but not both (figure 4b). Consider a hypothetical derivation of the structure in figure 4b:

$$\begin{aligned} S \to D(S) \to^* D(t) \to \sigma(A(t), B(t)) \to \sigma(t, B(t)) \\ \not\to^* \sigma(t, D(t')) \to \sigma(t, \sigma(A(t'), B(t'))) \to^2 \sigma(t, \sigma(t', \textsc{delete}(t'))) \end{aligned}$$

That $B(t) \not\to^* D(t')$ is immediate from the fact that all productions in G_1 with left-hand side $B(x_1)$ rewrite into trees whose roots are drawn from the terminal vocabulary. □



(a) Nested dependencies



(b) Crossing dependencies

Figure 4: Types of identity dependencies

Although G_1 cannot describe 'crossing' identity dependencies, as in figure 4b, it *is* able to describe 'nested' identity dependencies, as in 4a, of unbounded depth; a term of the form $D(C[S])$, where S in $C[S]$ is accessible in IO mode, may rewrite as $D(C[D(S)])$.

Given this state of affairs, we might ask as to whether this peculiar property of G_1 is mirrored in any straight-forward way the situation in natural language, i.e. whether natural languages disallow elliptical dependencies from having the form in figure 4b. However, it seems that such dependencies are quite natural:[8]

(7) Although currently only John has run the marathon, Bill wants to, and Mary says she does too.

In sentence 7, "*run the marathon*" plays the role of the subtree t in figure 4b, and "*want to ~~run the marathon~~*" plays the role of t'.

Given the naturalness of sentences like 7, which exhibit the kind of dependency shown in figure 4b, it is reasonable to conclude that G_1 does not provide an adequate model of the phenomenon of ellipsis in all of its generality. However, it is relatively straightforward to modify G_1 so as to allow generation of structures with a single level of crossing identity dependencies, as in figure 4b and sentence 7.

Consider the 'hypothetical' derivation of figure 4b as given in the proof of theorem 5, repeated here for convenience.

$$\begin{aligned}&S \to D(S) \to^* D(t) \to \sigma(A(t), B(t)) \to \sigma(t, B(t))\\ &\quad \not\to^* \sigma(t, D(t')) \to \sigma(t, \sigma(A(t'), B(t'))) \to^2 \sigma(t, \sigma(t', \textsc{delete}(t')))\end{aligned}$$

In order to license the currently illegitimate portion of the derivation, we need to pass $B(t)$ as an argument to $D(\cdot)$. More generally, we define G^1_{DI} to be the extension of G_1 with the rules $A(x_1) \Rightarrow D(A(x_1))$ and $B(x_1) \Rightarrow D(B(x_1))$.

$$\begin{aligned}&S \to D(S) \to^* D(t) \to \sigma(A(t), B(t)) \to \sigma(t, B(t))\\ &\quad \to \sigma(t, D(B(t)))\\ &\quad \to^* \sigma(t, D(t')) \to \sigma(t, \sigma(A(t'), B(t'))) \to^2 \sigma(t, \sigma(t', \textsc{delete}(t')))\end{aligned}$$

Although more powerful than G_1, G^1_{DI} still generates only structures in DelId.

Theorem 6

$$L(G_1) \subset L(G^1_{\text{DI}}) \subset \textsc{DelId}$$

The proof of the non-trivial inclusion in theorem 6 is similar to that of theorem 5. The demonstration of the properness of this inclusion convinces us that the distinction made in figure 4 between nested and crossing identity dependencies is a natural one. Grammar G_1 was unable to generate the

[8]The case of nested identity dependencies is also attested in natural language, witness:

Bill suspects that John will go after everyone that he does, and I do too.

structure shown in figure 4b because it didn't pass any information into elliptical structures (the only rule with non-terminal $D^{(1)}$ on the right-hand side was $S \rightarrow D(S)$). This made elliptical structures islands for other ellipsis sites or antecedents. Taking advantage of the fact that a single tree may be passed on as an argument, G^1_{DI} allows $A^{(1)}$ and $B^{(1)}$ to pass their arguments into 'elliptical islands'. Still, due to the fact that all non-terminals are at most monadic, structures with more than a single crossing identity dependency, as in figure 5, are out of reach of monadic CFTGs.

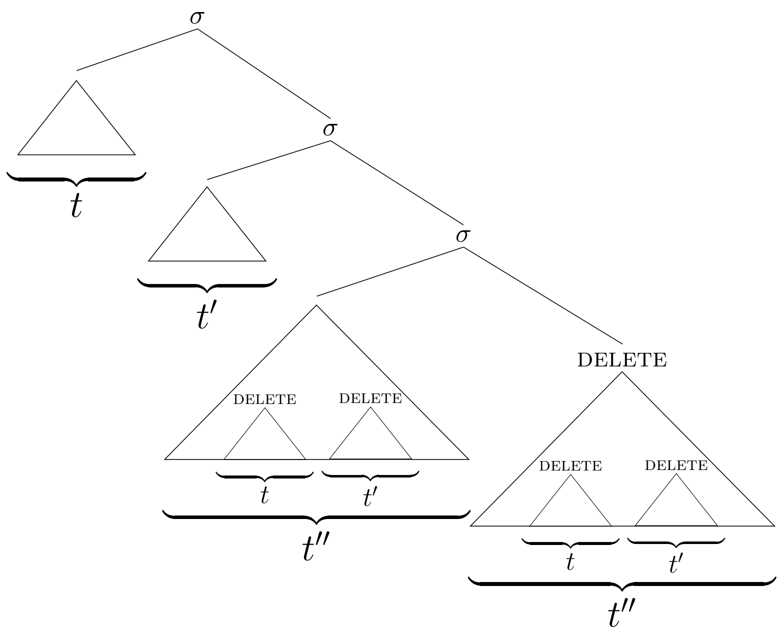


Figure 5: A structure with 2 crossing identity dependencies

As before, we can ask whether natural language avails itself of more than one crossing identity dependency.

(8) John wants to climb Mt. Kilimanjaro, and Susan wants to sail around the world, and while I know that John will and Susan won't, Bill doesn't.

In example 8, "*climb Mt. Kilimanjaro*" plays the role of t in figure 5, "*sail around the world*" the role of t', and "*know that John will ~~climb Mt. Kilimanjaro~~ and Susan won't ~~sail around the world~~*" the role of t''.

The natural generalization to three such crossing dependencies is obtained by simply adding another conjunct to 8.

(9) John wants to climb Mt. Kilimanjaro, Susan wants to sail around the world, and Mary wants to conquer the known universe, and while I know that John will, Susan won't, and Mary just might, Bill doesn't.

Although sentences with multiple crossing identity dependencies grow long quickly, there doesn't seem to be an obvious 'cut off point' for grammaticality. Moreover, it seems that within a single language arbitrarily many such crossing dependencies may obtain. This is unlike the case with other long-distance dependencies in natural language, where grammar formalisms like minimalist grammars [25] and (set local) multi-component tree adjoining grammars [28] also partition their grammars into those which can handle n such crossing dependencies at a time, but where any given language needs only a fixed number of such.

6 Conclusions

In this paper, we have examined a simple algorithm for parsing elliptical sentences. Although the 'basic idea' embodied in the algorithm appears quite natural, extending throughout the computational [5, 12, 26], linguistic [4], and psycholinguistic communities [11], our algorithm implements a quite different computational theory of ellipsis than has been heretofore proposed. The competence theory underlying the algorithm is most similar to those which postulate that ellipsis is non-pronunciation of a syntactic structure [8, 14, 19] under 'syntactic' identity [3, 8, 20] with an antecedent. In distinction to the theories mentioned above, the present underlying competence theory takes identity to be computed over the derivation itself. This has a number of conceptual and methodological advantages over the alternatives,[9] which are explored in future work (see e.g. [15]). For the present moment, we content ourselves with the observation that, when syntactic structure is taken to be the shape of the derivation itself, 'copying' and 'deletion' theories are clearly notational variants (cf. [24]).

[9]For preciseness' sake, both alternatives, whether 'syntactic' or 'semantic', take the objects over which identity is computed to be trees, albeit over a different alphabet than the derivation trees they are derived from. This has the consequence that the objects which can be deleted are not (necessarily) objects in the derivation tree (this is particularly clear in the literature on 'antecedent contained deletion' [18]). There is often no known characterization of these trees other than as the image of the derivation trees under some function (in the case of minimalist grammars a top-down tree transducer with regular look-ahead). Note that the definition of and attempt to characterize DelId is independent of whether deletion is taken to be over derivational, or other, constituents; all that changes is the vocabulary over which the trees comprising DelId are built.

We took some first steps toward a direct characterization of the licensing conditions on ellipsis under derivational identity. Using context-free tree grammars, we were led to a distinction between two sorts of identity dependencies, nested and crossing, and saw that whereas both appeared amply attested in natural language, only the nested ones were adequately described by CFTGs, with crossing dependencies being approximable only up to a certain arbitrary cut-off point. Having seen where and why CFTGs are limited in their application to this problem, we have taken the first steps toward an exact characterization of the conditions on ellipsis licensing. We must leave it to future work to give an intensional characterization of DelId, noting only that Fischer [9] uses quoted macro grammars (IO grammars with controlled OI capabilities) to define a simple programming language in which variables must be declared before they are used, and shows that the languages of quoted grammars include both the IO and the OI languages.

References

[1] P. R. J. Asveld. Time and space complexity of inside-out macro languages. *International Journal of Computer Mathematics*, 10:3–14, 1981.

[2] L. F. Bouton. Antecedent contained pro-forms. In M. A. Campbell, J. Lindholm, A. Davison, W. Fisher, L. Furbee, J. Lovins, E. Maxwell, and S. Straight, editors, *Papers from the Sixth Regional Meeting of the Chicago Linguistics Society (CLS)*, volume 6, pages 154–167, Chicago, 1970.

[3] S. Chung. Sluicing and the lexicon: The point of no return. paper presented at BLS, 2005.

[4] S. Chung, W. A. Ladusaw, and J. McCloskey. Sluicing and logical form. *Natural Language Semantics*, 3(3):239–282, 1995.

[5] M. Dalrymple, S. M. Shieber, and F. C. N. Pereira. Ellipsis and higher-order unification. *Linguistics and Philosophy*, 14(4):399–452, 1991.

[6] M. Egg and K. Erk. A compositional account of VP ellipsis. In F. Van Eynde, L. Hellan, and D. Beermann, editors, *Proceedings of the 8th International Conference on Head-Driven Phrase Structure Grammar*, pages 162–179. CSLI Publications, 2002.

[7] J. Engelfriet and E. M. Schmidt. IO and OI. I. *Journal of Computer and System Sciences*, 15(3):328–353, 1977.

[8] R. Fiengo and R. May. *Indices and Identity.* MIT Press, Cambridge, Massachusetts, 1994.

[9] M. J. Fischer. *Grammars with Macro-like Productions.* PhD thesis, Harvard, 1968.

[10] D. Fox. Antecedent-contained deletion and the copy theory of movement. *Linguistic Inquiry*, 33(1):63–96, Winter 2002.

[11] L. Frazier and C. Clifton Jr. Parsing coordinates and ellipsis: Copy α. *Syntax*, 4(1):1–22, Apr. 2001.

[12] D. Hardt. *Verb Phrase Ellipsis: Form, Meaning, and Processing.* PhD thesis, University of Pennsylvania, 1993.

[13] M. Kanazawa. Parsing and generation as datalog queries. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 176–183, Prague, 2007. Association for Computational Linguistics.

[14] C. Kennedy. Ellipsis and syntactic representation. In K. Schwabe and S. Winkler, editors, *The Interfaces: Deriving and interpreting omitted structures*, volume 61 of *Linguistik Aktuell/Linguistics Today*, pages 29–53. John Benjamins, Amsterdam/Philadelphia, 2003.

[15] G. M. Kobele. Derivational structure and ellipsis. paper presented at the ZAS Syntaxzirkel, June 2007.

[16] G. M. Kobele, C. Retoré, and S. Salvati. An automata theoretic approach to minimalism. ms., LaBRI.

[17] S. Lappin and H.-H. Shih. A generalized reconstruction algorithm for ellipsis resolution. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING)*, volume 2, pages 687–692, Copenhagen, Denmark, 1996.

[18] R. May. *Logical Form: Its Structure and Derivation.* MIT Press, Cambridge, Massachusetts, 1985.

[19] J. Merchant. *The Syntax of Silence: Sluicing, Islands, and the Theory of Ellipsis*, volume 1 of *Oxford Studies in Theoretical Linguistics.* Oxford University Press, New York, 2001.

[20] J. Merchant. Voice and ellipsis. ms., University of Chicago, 2007.

[21] E. Murguía. *Syntactic Identity and Locality Restrictions on Verbal Ellipsis.* PhD thesis, University of Maryland, 2004.

[22] W. C. Rounds. Mappings and grammars on trees. *Mathematical Systems Theory*, 4(3):257–287, 1970.

[23] I. A. Sag. *Deletion and Logical Form.* PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1976.

[24] N. A. Smith. Ellipsis happens and deletion is how. In A. Gualmini, S.-M. Hong, and M. Motomura, editors, *University of Maryland Working Papers in Linguistics*, volume 11, pages 176–191, 2001.

[25] E. P. Stabler. Derivational minimalism. In C. Retoré, editor, *Logical Aspects of Computational Linguistics*, volume 1328 of *Lecture Notes in Computer Science*, pages 68–95. Springer-Verlag, Berlin, 1997.

[26] W. Thompson. Deriving syntactic structure inside ellipsis. In *Proceedings of the Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms (TAG+7)*, pages 204–210, Vancouver, 2004.

[27] K. Vijay-Shanker, D. Weir, and A. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proceedings of the 25th Meeting of the Association for Computational Linguistics*, pages 104–111, 1987.

[28] D. J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms.* PhD thesis, University of Pennsylvania, 1988.