Completeness Conditions for Mixed Strategy Bidirectional Parsing

Graeme Ritchie*
University of Edinburgh

It has been suggested that, in certain circumstances, it might be useful for a grammar writer to annotate which rules are to be used bottom-up and which are to be used top-down within a parser, using a bidirectional variant of the active chart parsing technique. The formal properties of such systems have not been fully explored. One limitation of this mixed strategy technique is that certain annotations of rules can lead to incompleteness; that is, there may be valid analyses of the input string that cannot be found by the parser. We formalize a fairly natural notion of mixed strategy bidirectional parsing for context-free grammars, in which one or more symbols within a rule may be annotated as "triggers," so that the rule is either top-down (triggered from its left-hand side), or bottom-up (triggered from element(s) of its right-hand side). We define a decidable property of annotated grammars, such that any grammar with this property is provably complete. There are, however, some complete annotations of grammars that fall outside this decidable class. We show that membership of this wider class is undecidable. These results suggest that the mixed strategy approach is of rather limited usefulness, regardless of whether it is empirically efficient or not.

1. Overview

Many methods have been explored for parsing context-free grammars; some of these methods are loosely categorized as "top-down" (e.g., recursive descent), some as "bottom-up" (e.g., shift-reduce), and some could be seen as a mixture of these two varieties (e.g., left-corner). All of the well-explored methods assume that the rules in the grammar are handled in a fairly uniform way. In particular, it is not usual for the rules to be separated into two classes—those to be used bottom-up and those to be used top-down. Steel and de Roeck (1987) argue (giving credit to Henry Thompson for some of the ideas) that the performance of a parser could be improved by allowing the grammar writer to do exactly this. The motivation comes from linguistic phenomena where it is intuitively clear that one symbol (linguistic category) in the rule is noticeably more distinctive than others, so that a parser should not waste time trying to match the rule unless that distinctive element is there. For example, a rule such as *NP* → *NP CONJ NP* (where *CONJ* indicates a conjunction, such as *and*) should not be invoked simply because a noun phrase (*NP*), or the start of a noun phrase, has been found. The proposal is that if the linguist is allowed to mark the *CONJ* element as a "trigger," and the parser introduces the rule, bottom-up, only if the trigger has been matched, then parsing would proceed more efficiently.

Steel and de Roeck describe semi-formally a system they have implemented, which they claim benefits from this labeling of rules. The current paper does not take a position on the wisdom or effectiveness of such labeling. Instead, we explore the

* Division of Informatics, 80 South Bridge, Edinburgh EH1 1HN, Scotland.

formal consequences of this proposal. We show that, although the idea may seem superficially plausible, it still has certain formal limitations in the area of completeness and decidability. The proofs may be of some theoretical interest from a formal language viewpoint.

The central ideas are as follows: A conventional context-free grammar is "annotated" by marking at least one symbol in each rule as a trigger. Marking the left-hand-side (LHS) symbol as a trigger indicates that the rule can be used top-down; marking a right-hand-side (RHS) symbol as a trigger means that the rule can be used bottom-up whenever a constituent labeled with that symbol is found by the parser.[1] Using a method of parsing known as active chart parsing, it is straightforward to give a precise meaning to this labeling of rules, since a chart parser can operate either bottom-up or top-down. The scheme examined here is similar to, but different in important ways from, **head-driven parsing** (see Section 7.2).

It is simple to construct an annotated grammar in which there are some analyses that are valid according to the original (unannotated) grammar but that would not be parsed by a chart parser following the annotations. This establishes that not all annotated grammars allow complete parsing.

The main substance of this paper is as follows: A property of annotated grammars (direct analyzability) is defined, which is decidable, and it is proven that any annotated grammar with this property will also allow the parser to produce all the valid analyses licensed by the original grammar. However, some annotated grammars are not directly analyzable, but nevertheless lead to complete parsing. A characteristic of (a subset of) this wider class of annotated grammars (indirect analyzability) is defined, and it is proven that any annotated grammar with this property will allow complete parsing. However, indirect analyzability can be shown to be undecidable.

2. The problems

2.1 Losing Completeness

Before presenting a formal definition of the mechanisms, and proceeding to prove their various properties, it is useful to consider informally a very simple example that shows how this approach can lead to loss of analyses by the parser. As outlined above, the central idea is to allow different rules to be marked as either top-down (LHS trigger symbol) or bottom-up (RHS trigger symbol(s)), or both. Top-down means that the rule can be invoked only if some other rule has established a need for its LHS symbol (or if the LHS symbol is the initial symbol of the grammar). Bottom-up means that the rule can be invoked only if one of the symbols marked as triggers on its RHS has been completely parsed. We shall assume that rules of the form $A \rightarrow w$ where w is a terminal symbol are never annotated, and can be used whenever needed in the parser (all this is made precise in our formalization in Section 3.3 below).

For this informal presentation, and occasionally elsewhere, we shall mark a trigger symbol A by overlining it, thus: $\overline{A}$. In the illustrative examples, the distinguished (initial) symbol of the grammar will always be S and terminal symbols will be in lower case.

1 The term "bottom-up" is adopted here for compatibiity with some other literature on chart parsing, and for lack of a better simple phrase. In fact, there are various possible parsing regimes that are in some sense "bottom-up," and it is arguable that some are "more bottom-up" than those outlined here. Where right-hand-side triggers are restricted to the leftmost symbol (as in Section 5 below), parsing is more like "left-corner" parsing, but this would be a misleading term when triggers are allowed elsewhere.

Consider the annotated grammar (see Section 3 for a definition of grammar):

$$S \rightarrow \overline{NP}\ VP$$
$$\overline{NP} \rightarrow Art\ N$$
$$VP \rightarrow runs$$
$$Art \rightarrow the$$
$$N \rightarrow dog$$

It should be intuitively clear that although this grammar generates exactly one sentence, that string cannot be parsed by a parser that follows the annotations as described. The rule $S \rightarrow \overline{NP}\ VP$ cannot be used until an initial NP is recognized, and the rule that might do that, $\overline{NP} \rightarrow Art\ N$, cannot be used until an initial need for an NP is established (which could happen only using $S \rightarrow \overline{NP}\ VP$). There is a form of deadlock, resulting in incompleteness. It should also be intuitively clear that the presence or absence of such combinations of annotations may not be as obvious as it is here. In a grammar with hundreds of rules, the presence of a combination that blocks an otherwise valid analysis could take some detailed checking. This is a serious flaw, as the annotation method was supposed to alter the *efficiency* of the parser, but not to eliminate strings from its language.

It would be very easy to ensure that annotation does not lose analyses, by stipulating that all rules are marked as top-down, or that all rules are marked as bottom-up with the leftmost symbol as a trigger. The parser would then behave as a conventional top-down (or bottom-up) chart parser, which is known to be complete. However, since the aim is to allow the grammar writer to make a nontrivial annotation of the grammar (in an attempt to allow linguistic knowledge to influence the efficiency of the parsing), we need to be able to check the completeness of arbitrarily annotated grammars. In Section 4 below, we define formally a nontrivial characteristic of annotated grammars that guarantees that they do not lose analyses in this way, and show that this property of grammars is decidable.

2.2 Completeness through Interactions

The situation is even more complicated than Section 2.1 above indicates. One of the crucial aspects of chart parsing (which is central to its simplicity and its efficiency) is that any entry in a chart can be used to combine with any other compatible entry, regardless of whether there is a single coherent tree that will result from it. In particular, an entry that has been inserted in the chart as the result of some rule interaction that does not itself produce a complete sentential tree (i.e., a partial fragment of an analysis) can contribute to some other analysis that happens to require it.

This is best demonstrated by a simple artificial example. Consider the strategy-marked grammar, notation as before:

$$\overline{S} \rightarrow E\ H$$
$$H \rightarrow \overline{B}\ F$$
$$\overline{B} \rightarrow P\ Q$$
$$E \rightarrow j$$
$$P \rightarrow l$$
$$Q \rightarrow m$$
$$F \rightarrow k$$

The un-strategy-marked version of this grammar would generate the string *jlmk*, with

a derivation as follows (see Section 3 for a definition of the relation "⇒"):

$$S \Rightarrow EH \Rightarrow jH \Rightarrow jBF \Rightarrow jPQF \Rightarrow jlQF \Rightarrow jlmF \Rightarrow jlmk$$

The tree described by this derivation cannot be found by a parser following the strategy-marked grammar, for reasons similar to those outlined in Section 2.1 above. Suppose we now add the following rules to the grammar:

$$\overline{S} \rightarrow C\ D$$
$$D \rightarrow \overline{E}\ A$$
$$\overline{A} \rightarrow B\ C$$
$$C \rightarrow x$$

This larger grammar will also generate the string *xjlmx*, but this is not relevant to the argument. What is more interesting is that the extended grammar does now allow the parsing of *jlmk*, with an associated syntax tree that corresponds to the derivation given above (i.e., a tree that makes no direct use of the rules that have been added to the grammar). The way in which the added rules act as a "catalyst" to allow the hitherto blocked analysis is an example of a general phenomenon. Informally, what happens is the following: (A chart parser is assumed here; formal details are given in Section 3.3 below.) With just the smaller grammar, the nonterminal H cannot be expanded as required, since it is on the LHS of a bottom-up rule, and its first symbol B cannot be recognized because it requires a top-down rule. In both the original grammar and the larger grammar, H is introduced only by the rule $\overline{S} \rightarrow E\ H$, i.e., with E on its immediate left. So the only strings where H can participate in an analysis are those where E occurs at the start. Consider the parsing, with the larger grammar, of the string *jlmk* (which does indeed start with an E). As E is preterminal, it can be recognized directly (with no effect from annotations). In the larger grammar, the bottom-up rule $D \rightarrow \overline{E}\ A$ is then introduced to the parsing, which creates a predictive entry in the parser's structures seeking an A, after the recognized E. The top-down rule $\overline{A} \rightarrow B\ C$ is then introduced, which leads to an entry, at that same point, seeking a B. This causes the top-down rule $\overline{B} \rightarrow P\ Q$ (from the original grammar) to be introduced; this is a crucial step. This allows the sequence *lm* to be parsed as a B, thereby causing the introduction of the bottom-up rule $H \rightarrow \overline{B}\ F$, and the subsequent success of the parse.

2.3 What Are the Problems?

The grammars discussed above (Sections 2.1 and 2.2) are examples of various aspects of the problem. We shall show that there is a simple, decidable property of annotated grammars that guarantees completeness, and that could be used to detect the simple blocking illustrated in Section 2.1. However, this property is merely a sufficient condition for completeness, as the larger grammar of Section 2.2 above does not possess it, despite being complete. We shall show that the larger grammar of Section 2.2 has a more general property, which also guarantees completeness. However, the more general property of annotated grammars is undecidable.

First, we have to set up the basic formal mechanisms for our definitions.

3. Trees, Grammars, and Charts

3.1 Basic Concepts and Terms

We adopt the standard concepts for syntax trees (see Aho and Ullman [1972, Sect. 0.5] or Partee, ter Meulen, and Wall [1990, Chap. 16] for possible approaches to formalization). A **syntax tree** is a rooted, ordered, labeled tree. Each node apart from the root

has exactly one **mother** node, and each nonterminal node has one or more **daughter** nodes. A tree is said to **span** the sequence of labels associated with the sequence of its terminal nodes (in left-to-right order), and we shall also say that the root node of a (sub)tree spans its sequence of terminal nodes.

Definition 1
The **height** of a node in a tree is defined as follows. A terminal node has height 0; a nonterminal node has height = (1 + maximum height of its daughter nodes).

Definition 2
The **depth** of a node in a tree is defined as follows. A root node has depth 0; a nonroot node has depth = (1 + depth of its parent node).

Following the usual conventions (e.g., Aho and Ullman 1972), we will take a context-free grammar (CFG) to be a quadruple (V_N, V_T, P, S), consisting of a set V_N of **nonterminal** symbols, a set V_T of **terminal** symbols, a set P of **rules** (productions), and a single **distinguished symbol** $S \in V_N$. Set theoretically, rules can be regarded as being ordered pairs where the first element is a nonterminal symbol and the second is a tuple of symbols, i.e., of the form $(A_0, (A_1, \ldots, A_k))$ where $k \geq 0$, but for ease of exposition they will be written as

$$A_0 \rightarrow A_1 \ldots A_k$$

We will make the following simplifying assumptions (which do not lose generality):

1. Each rule in P is either of the form $A_0 \rightarrow A_1 A_2 \ldots A_k$ with $k > 0$, where all the symbols $A_i \in V_N$, or of the form $A \rightarrow w$ where $w \in V_T$.
2. The grammar has no redundant symbols, in the sense that no symbols are "useless" or "inaccessible" as defined by Aho and Ullman (1972, Sect. 2.4.2).

Rules of the form $A \rightarrow w$ where $w \in V_T$ will be referred to as **lexical** rules, and other rules as **nonlexical**. A nonterminal A that appears in a lexical rule will be called **preterminal**, or **lexical**.

Given a CFG G, **a syntax tree based on G** is a rooted, ordered tree whose nonterminal nodes are labeled with elements of V_N and whose terminal nodes are labeled with elements of V_T. Those nodes immediately dominating terminal nodes will be referred to as **preterminal**; other nonterminal nodes will be referred to as **nonlexical**. Where a tree T spans a terminal string $a_1 \ldots a_n$, and M is a node within T that spans $a_i \ldots a_k$, the **start** of M is the index $i - 1$, and the **end** of M is the index k.

A syntax tree based on (V_N, V_T, P, S) is said to be **well-formed** with respect to (V_N, V_T, P, S) if for every nonterminal node with label A_0 and daughter nodes labeled $A_1, \ldots, A_k$, there is a rule in P of the form $A_0 \rightarrow A_1 \ldots A_k$; this rule is said to **license** the node labeled A_0. For convenience, we shall distinguish between a tree that is compatible with the rules of the grammar, and a tree that also spans a sentence. A syntax tree is said to be **generated** by a grammar G iff:

1. The root node is labeled with S (the distinguished symbol).
2. The tree is well-formed w.r.t. G.

We will write $trees(G)$ for the set of all trees generated by G.

The conventional "rewrite" interpretation of CFGs will also be used in some situations (Section 5 below). Given two strings ω_1, ω_2 from $(V_N \cup V_T)^*$, then ω_1 **directly derives** ω_2, written "$\omega_1 \Rightarrow \omega_2$," if $\omega_1 = \delta A \gamma$, $\omega_2 = \delta \alpha \gamma$ and $A \rightarrow \alpha$ is a rule in G. Similarly, ω_1 **derives** ω_2, written "$\omega_1 \stackrel{*}{\Rightarrow} \omega_2$," is the reflexive transitive closure of directly derives. A **derivation** is a sequence of symbol strings $\omega_1, \ldots, \omega_n$ such that $\omega_i \Rightarrow \omega_{i+1}$ for all $1 \leq i < n$. A **rightmost** derivation is one in which each step from ω_i to ω_{i+1} is made by replacing the furthest right nonterminal symbol in ω_i using some rule (i.e., γ in the above definition of directly derives is entirely made up of terminal symbols) (cf. Aho and Ullman 1972).

3.2 Annotated Grammars

Since we are allowing trigger elements of a rule to occur anywhere on the RHS of a rule, it is necessary to allow the parser to explore outwards in either direction (leftwards or rightwards) from a constituent that has been parsed. Hence the parsing schemes defined below are referred to as **bidirectional**, to reflect this fact. This does not allude to the two "directions" of top-down or bottom-up.

Definition 3

Let G be a context-free grammar (V_N, V_T, P, S). A **bidirectional strategy marking of** G is a (total) function tr from the nonlexical rules in P to $\mathcal{P}(N)$ (the set of sets of nonnegative integers) such that for any rule r of the form $A_0 \rightarrow A_1 \ldots A_k$:

1. $tr(r) \neq \emptyset$
2. $0 \leq i \leq k$ for every $i \in tr(r)$

Informally, tr indicates which element(s) of the rule can trigger it. If $0 \in tr(r)$, the LHS of the rule is a trigger; that is, it can be used top-down. If $j \in tr(r)$, where $j > 0$, then element j of the RHS can act as a trigger, bottom-up. The value of $tr(r)$ is a *set* of integers in order to allow a rule to have more than one possible trigger; in particular, it is allowable for a rule to be used either top-down or bottom-up.

Definition 4

A **bidirectionally strategy-marked context free-grammar** (BSCFG) is a pair (G, tr) where G is a CFG and tr is a bidirectional strategy marking of G.

Definition 5

Let $((V_N, V_T, P, S), tr)$ be a bidirectionally strategy-marked context-free grammar. Then a rule $r \in P$ is said to be:

1. **top-down**, if $0 \in tr(r)$.
2. **bottom-up**, if there is an $i > 0$ such that $i \in tr(r)$.
3. **purely bottom-up**, if $0 \notin tr(r)$.
4. **purely top-down**, if $tr(r) = \{0\}$.

3.3 Active Charts

The techniques and structures known as active charts have been in use for parsing (at least in the area of natural language processing) since the early 1970s. The method

is a generalization of Earley's algorithm (Earley 1970), and tutorial expositions of the ideas can be found in Thompson and Ritchie (1984) or Winograd (1983). In keeping with more recent presentations (e.g., Shieber, Schabes, and Pereira 1995; Sikkel and op den Akker 1996) we define the parsing principles as well-formedness conditions on complete charts, abstracting away from the sequence of steps used to build them.

Definition 6
Given a CFG G of the form (V_N, V_T, P, S) a **double-dotted rule** based on G is a triple (p, l, r) where p is a rule in P of the form $A_0 \rightarrow A_1 \ldots A_k$ and l, r are integers such that $0 \leq l \leq r \leq k$.

Such a rule will be written as:

$$A_0 \rightarrow A_1 \ldots A_l \bullet A_{l+1} \ldots A_r \bullet A_{r+1} \ldots A_k$$

for ease of exposition and similarity to previous literature. Where either $l = 0$ or $r = k$, the empty portions will be omitted from the expression.

Definition 7
Given a CFG $G = (V_N, V_T, P, S)$, an **edge based on** G is a triple (i, j, d) where i and j are nonnegative integers with $i \leq j$, and d is a double-dotted rule based on G.

An edge is said to be **lexical** or **nonlexical** according to whether or not the rule is lexical. An edge of the form $(i, j, A_0 \rightarrow A_1 \ldots A_{q-1} \bullet A_q \ldots A_p \bullet A_{p+1} \ldots A_k)$ where either $q > 1$ or $p < k$ (i.e., with a nonempty component at either end) is referred to as an **active edge**, and an edge of the form $(i, j, A_0 \rightarrow \bullet A_1 \ldots A_k \bullet)$ is an **inactive edge**. An active edge $(i, i, A_0 \rightarrow \bullet \bullet A_1 \ldots A_k)$ or $(i, i, A_0 \rightarrow A_1 \ldots A_k \bullet \bullet)$ is referred to as an **empty** active edge. (Sometimes it will be referred to as "an empty active edge for $A_0 \rightarrow A_1 \ldots A_k$.")

Definition 8
Given a CFG $G = (V_N, V_T, P, S)$ and a string $a_1, \ldots, a_n$ from V_T^*, a **chart based on** $a_1, \ldots, a_n$ **and using G** is a set C of edges based on G that meets the following conditions:

1. for every $(i, j, r) \in C$, $i \in \{0, \ldots, n\}$ and $j \in \{0, \ldots, n\}$
2. for $a_i \in V_T$, $(i-1, i, L \rightarrow \bullet a_i \bullet) \in C$ iff $a_i \in \{a_1, a_2, \ldots, a_n\}$ and $L \rightarrow a_i \in P$.

The terminology of the last three definitions will also be used for a BSCFG (G, tr).

Definition 9
Let G be a CFG, and let C be a chart based on a string σ and using G. C is said to be **bidirectionally resolved** iff *both* the following conditions hold:

1. **Left Extension:** For every pair of edges:

$$(i, j, A_0 \rightarrow \bullet A_1 \ldots A_m \bullet)$$
$$(j, k, B_0 \rightarrow B_1 \ldots B_q \bullet B_{q+1} \ldots B_p \bullet B_{p+1} \ldots B_v)$$

where $p \leq v, q > 0$ and $A_0 = B_q$, there is also an edge:

$$(i, k, B_0 \rightarrow B_1 \ldots B_{q-1} \bullet B_q \ldots B_p \bullet B_{p+1} \ldots B_v)$$

2. **Right Extension:** For every pair of edges:

$$(i, j, B_0 \rightarrow B_1 \ldots B_q \bullet B_{q+1} \ldots B_p \bullet B_{p+1} \ldots B_v)$$
$$(j, k, A_0 \rightarrow \bullet A_1 \ldots A_m \bullet)$$

where $p < v, q \geq 0$ and $A_0 = B_{p+1}$, there is also an edge:

$$(i, k, B_0 \rightarrow B_1 \ldots B_q \bullet B_{q+1} \ldots B_{p+1} \bullet B_{p+2} \ldots B_v)$$

Lemma 1
Let $\mathcal{C}$ be a bidirectionally resolved chart based on a string σ and using a CFG G, and suppose that $\mathcal{C}$ contains an edge of the form:

$$(i, j, B_0 \rightarrow B_1 \ldots B_q \bullet B_{q+1} \ldots B_p \bullet B_{p+1} \cdots B_v)$$

(i) (Rightwards) If $\mathcal{C}$ contains edges of the form:

$$(i_{p+1}, j_{p+1}, B_{p+1} \rightarrow \bullet \omega_{p+1} \bullet)$$
$$\ldots$$
$$(i_{p+t}, j_{p+t}, B_{p+t} \rightarrow \bullet \omega_{p+t} \bullet)$$

where $(p + t) \leq v$, $i_{k+1} = j_k$ where $(p + 1) \leq k < (p + t)$ and $i_{p+1} = j$, then $\mathcal{C}$ also contains an edge of the form:

$$(i, j_{p+t}, B_0 \rightarrow B_1 \ldots B_q \bullet B_{q+1} \ldots B_{p+t} \bullet B_{p+t+1} \ldots B_v)$$

(ii) (Leftwards) If $\mathcal{C}$ contains edges of the form:

$$(i_{(q-t)}, j_{(q-t)}, B_{(q-t)} \rightarrow \bullet \omega_{(q-t)} \bullet)$$
$$\ldots$$
$$(i_q, j_q, B_q \rightarrow \bullet \omega_q \bullet)$$

where $0 \leq t < q$, $i_{k+1} = j_k$ where $(q - t) \leq k < q$ and $j_q = i$, then $\mathcal{C}$ also contains an edge of the form:

$$(i_{(q-t)}, j, B_0 \rightarrow B_1 \ldots B_{(q-t-1)} \bullet B_{(q-t)} \ldots B_p \bullet B_{p+1} \ldots B_v)$$

Proof
Both the cases (i) and (ii) proceed by induction on the number of inactive edges.

Corollary
If $\mathcal{C}$ is as described, and it contains a full set of edges as given at both sides (i.e., $t = (q - 1)$, so that there are q inactive edges to the left, and $(p + t) = v$ so that there are $(v - p)$ inactive edges to the right, all with labels matching the rule), then there is a complete (inactive) edge of the form $(i_1, j_v, B_0 \rightarrow \bullet B_1 \ldots B_v \bullet)$.

A chart parser is driven by two principles: one is that of edge combination, as given in the above definition of bidirectionally resolved, and the other is the introduction of rules into the chart. For a strategy-marked grammar, the rule-introduction principle is sensitive to the annotation of the rules.

Definition 10
Let (G, tr) be a BSCFG, and let $\mathcal{C}$ be a chart based on a string σ and using G. $\mathcal{C}$ is said to be **bidirectionally mixed strategy explored** iff *all* the following conditions hold:

1. (Bottom-up activation) For every edge:

$$(i, j, A_0 \rightarrow \bullet A_1 \ldots A_m \bullet)$$

there is an edge in $\mathcal{C}$:

$$(i, j, B_0 \rightarrow B_1 \ldots B_{q-1} \bullet B_q \bullet B_{q+1} \ldots B_v)$$

for every rule r in G of the form $B_0 \rightarrow B_1 \ldots B_v$ such that $q \in tr(r)$ and $B_q = A_0$,

2. (Top-down initialization) For every rule r in G of the form $S \rightarrow B_1 \ldots B_k$, where S is the distinguished symbol of G and $0 \in tr(r)$, there is an edge in $\mathcal{C}$ of the form:

$$(0, 0, S \rightarrow \bullet \bullet B_1 \ldots B_k)$$

3. (Top-down activation, right) For every edge:

$$(i, j, B_0 \rightarrow B_1 \ldots B_q \bullet B_{q+1} \ldots B_p \bullet B_{p+1} \ldots B_v)$$

where $0 \leq p < v$, and every rule r in G of the form $A_0 \rightarrow A_1 \ldots A_k$ for which $B_{p+1} = A_0$ and $0 \in tr(r)$, there is also an edge in $\mathcal{C}$ of the form:

$$(j, j, A_0 \rightarrow \bullet \bullet A_1 \ldots A_k)$$

4. (Top-down activation, left) For every edge:

$$(i, j, B_0 \rightarrow B_1 \ldots B_q \bullet B_{q+1} \ldots B_p \bullet B_{p+1} \ldots B_v)$$

where $0 < q \leq v$, and every rule r in G of the form $A_0 \rightarrow A_1 \ldots A_k$ for which $B_q = A_0$ and $0 \in tr(r)$, there is also an edge in $\mathcal{C}$ of the form:

$$(i, i, A_0 \rightarrow A_1 \ldots A_k \bullet \bullet)$$

For brevity, the term **fully bidirectional** will be used for a chart that is both bidirectionally resolved and bidirectionally mixed strategy explored.

To explore the issue of completeness (i.e., whether a parsing mechanism finds all valid analyses) we need to define how the edges in a chart correspond to those in a syntax tree.

Definition 11
A chart $\mathcal{C}$ based on $a_1 a_2 \ldots a_n$ is said to **contain a representation of** a syntax tree T, iff:

1. T spans a substring (not necessarily proper) of $a_1 a_2 \ldots a_n$; *and*
2. for every nonterminal node N in T, spanning $a_{i+1} \ldots a_j$, labeled A_0, with k daughters labeled $A_1, \ldots, A_k$ in order, $\mathcal{C}$ contains an edge

$$(i, j, A_0 \rightarrow \bullet A_1 \ldots A_k \bullet)$$

(This includes the case where $k = 1$ and $A_1 \in V_T$.)

Notice that for any chart $\mathcal{C}$ based on a string σ, $\mathcal{C}$ will contain an edge for each preterminal node of any tree that spans σ, by virtue of the definition of a chart being "based on" a string. Hence later discussions of parsing and completeness can assume the presence of these edges in the relevant charts, with only the presence of edges for other nonterminal nodes being subject to verification.

Definition 12
Given a CFG G, a bidirectional strategy-marking tr of G is said to be **complete** iff for every tree $T \in trees(G)$ that spans a string σ, any fully bidirectional chart $\mathcal{C}$ based on σ and using (G, tr) contains a representation of T.

4. A Decidable Class of Complete Annotations

In this section we define a decidable property of annotated grammars that guarantees that a parser following the annotations will not miss analyses in the manner outlined in Section 2.1. There is also a weaker (more general) sufficient condition for completeness, which is defined in Section 5 below, but which is undecidable. The fact that the stronger condition is decidable makes it worth defining, and some of the proofs in Section 5 make use of some concepts from the current section.

4.1 Reachability

Another notion that has to be formalized is the way in which a syntax tree can be parsed from a string of terminal symbols in a purely bottom-up manner.

Definition 13
Let (G, tr) be a BSCFG. In a syntax tree T generated by G, a nonlexical node M_0, with daughters $M_1, \ldots, M_k$, is said to be **reachable from below** iff M_0 is licensed by a bottom-up rule r and there is a j, $1 \leq j \leq k$, such that $j \in tr(r)$ and one of the following is true:

1. M_j is a preterminal node of T;
2. M_j is reachable from below.

Definition 14
Let (G, tr) be a BSCFG. A syntax tree T generated by G, is said to be **fully reachable** iff every nonlexical node M in T licensed by a purely bottom-up rule is reachable from below.

4.2 Direct Analyzability

Now we need to define a property of grammars that will guarantee that generated trees are fully reachable in the above sense. This can be done in three stages: first, define a property of nonterminal symbols; then, use that to define a property of grammars; lastly, prove that any grammar with this property generates only fully reachable trees.

A first approximation to the definition for the property of nonterminals would be the following:

Draft Definition:
Given a BSCFG (G, tr), a nonterminal symbol A_0 is **directly analyzable** iff every rule r of the form $A_0 \rightarrow \ldots$ is *either* lexical, *or* of the form $A_0 \rightarrow A_1 \ldots A_k$ with at least one $i \in tr(r), i > 0$, for which A_i is directly analyzable.

The subsequent definition for grammars is then:

Definition 15
A BSCFG (G, tr) is **directly analyzable** iff it meets the following condition: for every purely bottom-up rule r of the form $A_0 \rightarrow A_1 \ldots A_k$, there is at least one $i \in tr(r), i > 0$, for which A_i is directly analyzable.

The draft definition captures the essential idea in a fairly natural and clear way, but it has a slight technical problem. Consider the toy grammar given below:

$$S \rightarrow \overline{A}\ B$$
$$A \rightarrow C\overline{A}$$
$$C \rightarrow x$$
$$B \rightarrow y$$
$$A \rightarrow z$$

In this grammar, the nonterminal A is not classed as directly analyzable. This is because there is a cycle from A to itself via trigger symbols in bottom-up rules.[2] It would be equally consistent with the draft definition to state that A is directly analyzable, or to stipulate that it is *not* directly analyzable. There is a sense in which the draft definition is underspecified, and gives only partial coverage of the items being classified (nonterminals). To extend this definition to total coverage, a more elaborate construction is needed (borrowed from theoretical computer science; cf. Stoy [1981, Chap. 6]).[3] First, for any BSCFG (G, tr) we define an **analyzability predicate** as any function g from nonterminal symbols to the set $\{true, false\}$ that assigns *true* to a category A_0 iff every rule r of the form $A_0 \rightarrow \ldots$ is *either* lexical, *or* of the form $A_0 \rightarrow A_1 \ldots A_k$, with at least one $i \in tr(r), i > 0$, for which $g(A_i) = true$. Call the set of all such functions $\mathcal{AP}(G, tr)$. For any two $g, h \in \mathcal{AP}(G, tr)$, define the relation "$\sqsubseteq$" by $h \sqsubseteq g$ iff $h(A) = true \supset g(A) = true$. This relation is easily shown to be reflexive, transitive, and antisymmetric, and hence $(\mathcal{AP}(G, tr), \sqsubseteq)$ forms a partially ordered set (Maclane and Birkhoff 1967, 59; Stoy 1981, 82). Then for any set $g_1, \ldots, g_n$ of elements of $\mathcal{AP}(G, tr)$, the function g' (in $\mathcal{AP}(G, tr)$) given by

$$g'(A) = true \text{ iff either } g_1(A) = true \text{ or } \ldots\ g_n(A) = true$$

is at least upper bound (Maclane and Birkhoff 1967; Stoy 1981) for $g_1, \ldots, g_n$ with respect to $\sqsubseteq$. Since $\mathcal{AP}(G, tr)$ is finite, the presence of a l.u.b. for any subset means it has a maximum element, which we will call $APMAX_{(G,tr)}$. This predicate $APMAX_{(G,tr)}$ will assign *true* to a symbol A if there is *some* analyzability predicate (for (G, tr)) that makes this assignment.[4] Then define a nonterminal A (from G) to be directly analyzable iff $APMAX_{(G,tr)}(A) = true$. Intuitively, any nonterminal that the draft definition might

2 Thanks to Alistair Willis for pointing out this problem.
3 Thanks to Suresh Manandhar for suggesting this approach.
4 Stoy (1981, 79–80) illustrates the use of a *minimum* element from an ordered set of possible functions, but here we have chosen to use the maximum.

leave as undefined with respect to being directly analyzable is classed by this new definition as being directly analyzable. Instead of the draft definition, we can now have the following complete definition:

Definition 16
Given a BSCFG (G, tr), a nonterminal symbol A_0 is **directly analyzable** iff $APMAX_{(G,tr)}(A_0) = true$, where $APMAX_{(G,tr)}$ is as constructed above.

Notice that it follows from the construction of $APMAX_{(G,tr)}$ that A_0 is directly analyzable iff every rule r of the form $A_0 \rightarrow \ldots$ is *either* lexical, *or* of the form $A_0 \rightarrow A_1 \ldots A_k$ with at least one $i \in tr(r), i > 0$, for which A_i is directly analyzable. That is, the draft definition, which was not sufficiently self-contained to be a definition, is now derivable *as a theorem* from the more rigorous definition. This means that we can use the logical equivalence stated in the draft definition in subsequent proofs.

Lemma 2
Let (G, tr) be a BSCFG that is directly analyzable. Let T be a tree in $trees(G)$. Then T is fully reachable.

Proof
It is straightforward to prove the following preliminary result, using induction on the height of nodes and the logical equivalence stated in the draft definition above:

Any node in T that has a directly analyzable label is reachable from below.

It is then easy to show that any node in T that is licensed by a purely bottom-up rule is reachable from below. □

4.3 Parsing

Lemma 3
Let (G, tr) be a BSCFG. Let T be a fully reachable tree in $trees(G)$, spanning the string σ. Let $\mathcal{C}$ be a fully bidirectional chart based on σ and using (G, tr). Then for any node M in T, labeled A:

1. If M is licensed by a purely bottom-up rule, then $\mathcal{C}$ contains a representation of the subtree rooted at M.
2. If M is licensed by a top-down rule, and there is in $\mathcal{C}$ an active edge

$$(t, g, B_0 \rightarrow B_1 \ldots B_{p-1} \bullet B_p \ldots B_{q-1} \bullet B_q \ldots B_v)$$

where either $B_{p-1} = A$ and t is the end of M, or $B_q = A$ and g is the start of M, then $\mathcal{C}$ contains a representation of the subtree rooted at M.

Proof
By induction on the height of nodes.

Inductive Hypothesis: For any $0 < d' < d$, if node M in T is of height d', the conditions listed in the lemma hold.

Base Case: Suppose M is of height 1 (i.e., preterminal). Then $\mathcal{C}$ contains a representation of the subtree rooted at M, regardless of the antecedent conditions.

Inductive Step: Suppose M is of height d, where $d > 1$, and is labeled A.

(a) Suppose M, with daughter nodes $M_1, \ldots, M_k$, is licensed by a purely bottom-up rule $A \rightarrow A_1 \ldots A_k$. Then, since T is fully reachable, there is a j, $1 \leq j \leq k$ such that M_j is reachable from below; that is, M_j is either lexical or licensed by a bottom-up rule. Therefore, by the Inductive Hypothesis, $\mathcal{C}$ contains a representation of the subtree rooted at M_j. Since $\mathcal{C}$ is fully bidirectional, it contains an edge spanning M_j of the form:

$$(l, h, A \rightarrow A_1 \ldots A_{j-1} \bullet A_j \bullet A_{j+1} \ldots A_k)$$

Now consider the nodes M_i, for $j+1 \leq i \leq k$. The Inductive Hypothesis applies to each of these nodes. Hence it can be proved by induction on i that there is a representation in $\mathcal{C}$ of the subtree rooted at M_i for all $j+1 \leq i \leq k$ (cf. Lemma 1). Similarly, it can be proved that there are representations in $\mathcal{C}$ for the subtrees rooted at $M_1, \ldots, M_{j-1}$. By the corollary to Lemma 1, there is a representation for the tree rooted at M in $\mathcal{C}$.

(b) Suppose M is licensed by a top-down rule. Suppose there is an edge

$$(t, g, B_0 \rightarrow B_1 \ldots B_{p-1} \bullet B_p \ldots B_{q-1} \bullet B_q \ldots B_v)$$

where $B_q = A$ and g is the start of M (a similar argument holds in the case where $B_{p-1} = A$ and t is the end of M). Since the chart is fully bidirectional, there must also be an empty active edge

$$(g, g, A \rightarrow \bullet \bullet A_1 \ldots A_k)$$

By a similar argument to that in case (a) above, it follows that there are representations in $\mathcal{C}$ for all the nodes $M_1, \ldots, M_k$ and thence for M.

This establishes the main induction. □

Lemma 4
Let (G, tr) be a BSCFG that is directly analyzable. Let T be a tree in $trees(G)$, spanning the string σ. Let $\mathcal{C}$ be a fully bidirectional chart based on σ and using (G, tr). Then for any node M in T, labeled A:

1. If M is licensed by a purely bottom-up rule, then $\mathcal{C}$ contains a representation of the subtree rooted at M.
2. If M is licensed by a top-down rule, and there is in $\mathcal{C}$ an active edge

$$(t, g, B_0 \rightarrow B_1 \ldots B_{p-1} \bullet B_p \ldots B_{q-1} \bullet B_q \ldots B_v)$$

where either $B_{p-1} = A$ and t is the end of M, or $B_q = A$ and g is the start of M, then $\mathcal{C}$ contains a representation of the subtree rooted at M.

Proof
Follows from Lemmas 2 and 3. □

Being reachable from below can be seen as a condition on nodes that can be built bottom-up. Surprisingly, we do not need a corresponding condition for nodes that are built top-down. It is possible to formulate the appropriate condition, but it turns out that any tree that meets the condition of being fully reachable will also meet the appropriate condition for top-down nodes. It is hard to give an informal, intuitive explanation for this, but roughly speaking, the reason is as follows. For a top-down rule to be invoked, it must be used in a position at which some prediction of its LHS symbol A will be introduced (by some other rule). This can happen either as a cascade of predictions from above, using a sequence of top-down rules, or because a rule has been introduced and has caused a sequence of predictions to be made, either left-to-right or right-to-left, as its RHS symbols are parsed. For either of these to happen, either there must be a clear path of daughter categories from some other prediction, or A must be on the RHS of a rule that is somehow introduced. The daughter condition of "reachable from below" simultaneously imposes these conditions on the top-down rules.

4.4 Completeness

The final step in proving completeness is now simple.

Theorem 1
If a BSCFG (G, tr) is directly analyzable, then tr is complete.

Proof
Let T be a tree in $trees(G)$, spanning the string σ, with root node M_0 labeled S (the distinguished symbol of G). Let $\mathcal{C}$ be a fully bidirectional chart based on σ and using (G, tr).

(a) If M_0 is licensed by a bottom-up rule, then by Lemma 4, $\mathcal{C}$ contains a representation of the tree rooted at M_0.

(b) If M_0 is licensed by a top-down rule $S \rightarrow A_1, \ldots, A_k$, then $\mathcal{C}$ must contain an empty active edge of the form:

$$(0, 0, S \rightarrow \bullet\bullet A_1 \ldots A_k)$$

It follows from repeated applications of Lemma 4 and Lemma 1 (similar to part (b) of the Inductive Step of Lemma 3) that $\mathcal{C}$ contains a representation of the subtrees rooted at the daughters of M_0, and thence of the subtree rooted at M_0. □

Thus we have proved that all BSCFGs that meet the condition of being directly analyzable can be bidirectionally parsed without any valid trees being omitted.

Theorem 2
It is decidable whether a given BSCFG is directly analyzable.

Proof
This is straightforward to verify from the definition of directly analyzable (see Appendix A for an algorithm).

5. An Undecidable Class of Complete Annotations

5.1 Informal Outline

Before proceeding to formalize the mechanisms underlying the problem presented in Section 2.2, it is useful to set out informally the relevant factors in that example. A strategy-marked grammar (G, tr) causes problems only if there is some purely bottom-up rule of the form $A_0 \rightarrow A_1 \ldots A_k$ such that every trigger symbol A_i requires a purely top-down rule somewhere in its expansion (see Section 4.2 above). Such a rule leads to the possibility of there being a tree $T \in trees(G)$ that contains a nonterminal node that can only be built by a bottom-up rule, and whose trigger daughter can only be built using a top-down rule. This would give rise to a tree that was not fully reachable. In the example in Section 2.2 above, the rules

$$H \rightarrow \overline{B}\ F$$
$$\overline{B} \rightarrow P\ Q$$

create this situation. The "upper" symbol H cannot be parsed because the "lower" symbol B cannot be parsed. What salvages this difficulty is the fact that the "upper" nonterminal (H in this example) always occurs in a **left context**, (i.e., a string of symbols to its left) with the following property: Every possible terminal expansion of the left context contains a substring that will, via bottom-up rules, introduce rules that are bound to result in the introduction of an active edge, which starts at the point where the "upper" symbol (H) is needed and which is seeking the "lower" symbol (B).

The illustrative grammars in Sections 2.1 and 2.2 are of a particular subclass of grammars—those where $tr(r) = \{0\}$ or $tr(r) = \{1\}$ for any (nonlexical) rule r. This is equivalent to partitioning the rules into two subgroups—top-down and bottom-up—where the bottom-up rules are always triggered in a left-corner manner, much as in conventional "bottom-up" chart parsers (such as those in Thompson and Ritchie [1984] or Winograd [1983]). That is, there is a natural subclass of annotated grammars that do not rely on the bidirectional exploration of the chart, but allow this limited form of mixed strategy left-to-right exploration.

The definitions and proofs of the earlier sections apply to this subclass. It is also clear, from Section 2.2, that the issue of "completeness by interaction" can be illustrated within this limited subclass. In the remainder of Section 5 below, it is proved that detecting the possibility of such rule interactions is undecidable even for this limited subclass of grammars. It follows that it must be undecidable for the more general class, where any annotation is permitted. The advantages of focusing on this more limited subclass are twofold: it shows that restricting the annotations in this way would not ease the undecidability problem, and it simplifies the proofs (which are already tediously complex).

Definition 17

A **left-corner strategy-marked** context-free grammar (LCSCFG) is a BSCFG (G, tr) such that $tr(r) \subseteq \{0, 1\}$ for every rule r in G.

This definition allows a rule to be both bottom-up and top-down marked, rather than enforcing a strict partitioning. In the following proofs, we will define constructs for BSCFGs where possible, simply for generality, but where it matters we shall confine attention to LCSCFGs, thereby narrowing the range of contexts relevant to parsing a particular symbol.

5.2 Left Contexts

Following from the informal discussion in Section 5.1 above, we need to define more precisely the notion of a left context of a symbol. What we want is a way of characterizing, for a given nonterminal A, exactly those strings of symbols that *must* appear immediately to the left of A in any valid derivation in which A appears. These need not be all that is to the left of A in a derivation, but it must be the case that A cannot appear without having one of these left context strings immediately adjacent to it.

In the following definitions, the derivation relationship "$\stackrel{*}{\Rightarrow}$" is the conventional one, and is independent of any strategy marking; the relationship "$\stackrel{R}{\Rightarrow}$" indicates a rightmost derivation (see Section 3.1 earlier).

Definition 18

Suppose we have a context-free grammar (V_N, V_T, P, S), and a sequence of symbols $B_1, \ldots, B_t$ in V_N, where there are rules $B_i \rightarrow \rho_{i-1} B_{i-1} \beta_{i-1}$ for $2 \leq i \leq t$ $(\rho_i, \beta_i \in V_N^*)$. Suppose we have a rightmost derivation of the form:

$$\begin{aligned} B_t &\stackrel{R}{\Rightarrow} \rho_{t-1} B_{t-1} \omega_{t-1} \\ &\stackrel{R}{\Rightarrow} \rho_{t-1}\rho_{t-2} B_{t-2} \omega_{t-2} \\ &\ldots \\ &\stackrel{R}{\Rightarrow} \rho_{t-1}\rho_{t-2} \ldots B_2 \omega_2 \\ &\stackrel{R}{\Rightarrow} \rho_{t-1}\rho_{t-2} \ldots \rho_1 B_1 \omega_1 \end{aligned}$$

(all the $\omega_i \in V_T^*$). This derivation is said to be:

1. **nonrepeating** if $B_i \neq B_j$ whenever $i \neq j$.
2. **rooted** if $B_t = S$.
3. **localized** if there is a longer sequence of nonterminal symbols $B_1, \ldots, B_m$ and a rooted rightmost derivation $B_m \stackrel{R}{\Rightarrow} \rho_{m-1} \ldots \rho_1 B_1 \omega_1'$ such that $B_t = B_k$ for some $t < k \leq m$.
4. **essential** if it is nonrepeating and either rooted or localized.

Also, the derivation is said to be **for** B_1 **from** B_t, and the string $\rho_{t-1} \ldots \rho_1$ is said to be the **left context sequence** of this derivation.

Definition 19

For any nonterminal A, the **set of essential left contexts of A** is

$$\{\sigma \in V_T^* \mid \exists \text{ an essential rightmost derivation } D \text{ for } A \text{ and } \psi \text{ is the left context sequence of } D \text{ and } \psi \stackrel{*}{\Rightarrow} \sigma\}$$

The following lemma proves that essential left contexts have just the required property.

Lemma 5

Let G be a CFG. Let T be a tree in trees(G). Let M be a nonterminal node in T. Let σ be the terminal string spanned by T, and δ the portion of σ spanned by M. Then σ is of the form $\phi_1 \gamma \delta \phi_2$ for some γ in the essential left contexts of the label of M.

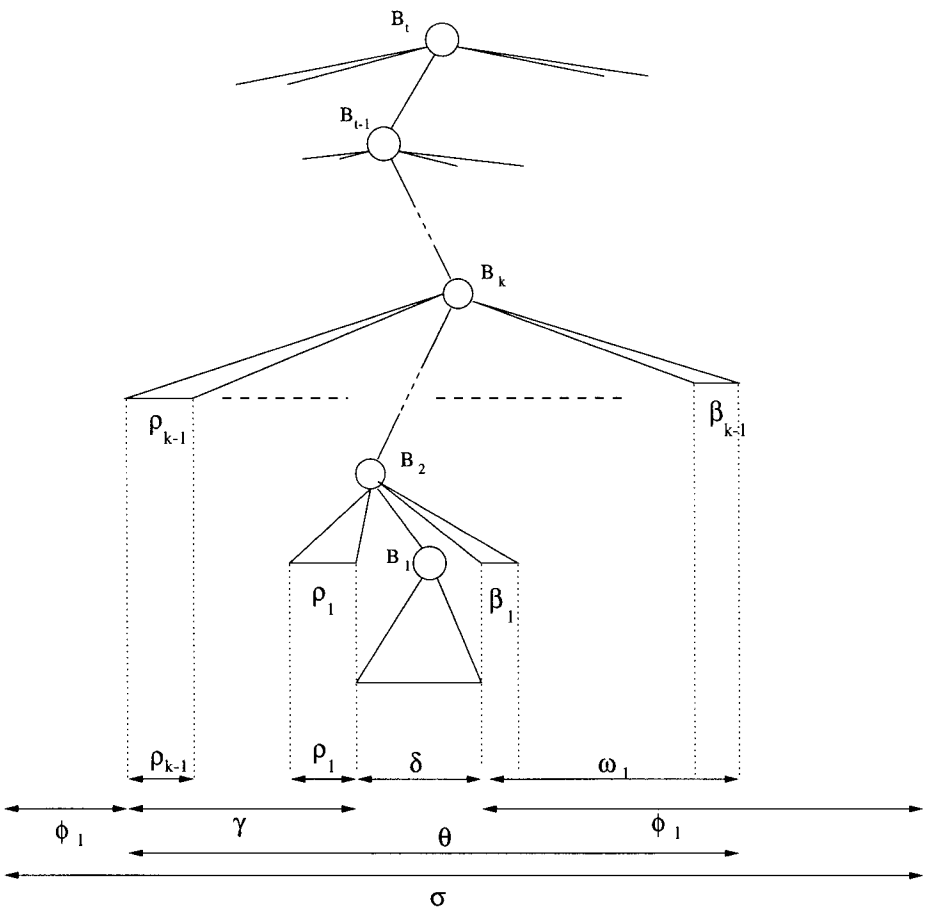

Figure 1
Situation described in Lemma 5.

Proof
(See Figure 1 for an intuitive picture.) Since T is a tree, there is a path of nonterminal nodes $(N_1, \ldots, N_t)$ where N_i is labeled B_i, for $1 \leq i \leq t$, $N_1 = M, N_t$ is the root of T, and N_i is the mother of N_{i-1} for $2 \leq i \leq t$. Since $T \in trees(G)$, there must be a sequence of rules $B_i \rightarrow \rho_{i-1}B_{i-1}\beta_{i-1}$ $(\rho_i, \beta_i \in V_N^*)$ such that the node N_i is licensed by $B_i \rightarrow \rho_{i-1}B_{i-1}\beta_{i-1}$ for $2 \leq i \leq t$. Hence there is a rooted rightmost derivation for B_1 from B_t. From this it is trivial to form an essential rightmost derivation for B_1 from some symbol B_k (where $k \leq t$):

$$B_k \stackrel{R}{\Rightarrow} \rho_{k-1} \ldots \rho_1 B_1 \omega_1$$

where $\omega_1 \in V_T^*$ and

$$\rho_{k-1} \ldots \rho_1 B_1 \omega_1 \stackrel{*}{\Rightarrow} \theta$$

where θ is the substring of σ spanned by N_k.

This means that θ is of the form $\gamma\delta\omega_1$ where $\rho_{k-1} \ldots \rho_1 \stackrel{*}{\Rightarrow} \gamma$ and $B_1 \stackrel{*}{\Rightarrow} \delta$ (since B_1 is the label of M, and δ is the terminal string spanned by M). Then γ is an essential left context of B_1, by virtue of the way $\rho_{k-1} \ldots \rho_1$ was constructed. Since θ is a substring of σ, this establishes the result. □

5.3 Bottom-up Derivations

Definition 20
In a BSCFG (G, tr), suppose A is a nonterminal symbol, and σ is a string of terminal symbols. Then σ **can be coherently derived from** A **(with tree** T**)** iff T is a syntax tree generated by G such that:

1. T spans σ
2. the root of T is labeled A
3. T is fully reachable.

The next definition requires that the derivation can occur without need for top-down initiation.

Definition 21
Let (G, tr) be a BSCFG. Suppose A is a nonterminal symbol, and σ is a string of terminal symbols, from G . Then σ **can be up-derived from** A (written "$A \Uparrow^* \sigma$") using (G, tr) iff:

1. σ can be coherently derived from A with tree T;
2. the root of T is reachable from below.

It is clear that all nodes of such trees will appear in a chart, as formalized in Lemma 6.

Lemma 6
Let (G, tr) be a BSCFG. If $A \Uparrow^* \sigma$ using (G, tr), and $\mathcal{C}$ is a fully bidirectional chart based on a string $\gamma_1\sigma\gamma_2$, and using (G, tr), then $\mathcal{C}$ contains a representation of a tree T such that T spans σ and the root of T is labeled A.

Proof
Follows from Lemma 3. □

5.4 Left-Introducible Rules

In characterizing formally the situation outlined informally in Section 5.1 above, the following definition allows a more succinct statement.

Definition 22
Let A, B be two nonterminal symbols from a LCSCFG. A *introduces* B *from above* (written "$A \rightsquigarrow B$") if either $A = B$, or there is a sequence of top-down rules

$$
\begin{aligned}
&A \rightarrow A_0 \ldots \\
&A_0 \rightarrow A_1 \ldots \\
&\ldots \\
&A_t \rightarrow B \ldots
\end{aligned}
$$

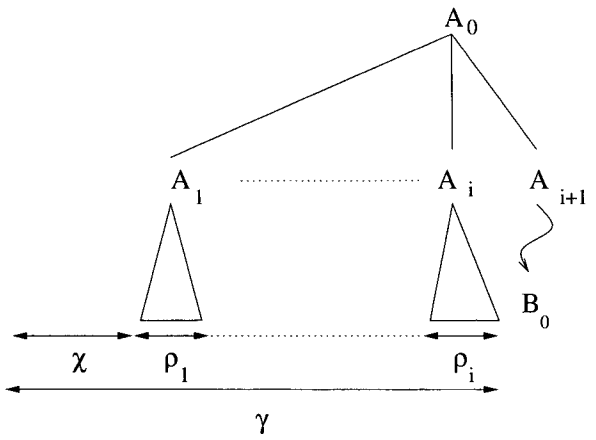

Figure 2
Left-introducibility.

Lemma 7
Let A, B be two nonterminal symbols from a LCSCFG (G, tr). If $A \rightsquigarrow B$, then in any bidirectionally mixed strategy explored chart C using (G, tr) that contains an active edge $(i, j, A' \rightarrow \bullet\alpha_1 \bullet A\beta_1)$, there will also be an active edge in C of the form $(l, j, A'' \rightarrow \bullet\alpha_2 \bullet B\beta_2)$, where either $l = i$ or $l = j$.

Proof
Straightforward. The case $l = i$ allows for $A = B$ (and $A' = A''$), and $l = j$ is the more general case where there is a sequence of top-down-invoked active edges linking A to B. □

Next we have a definition of the condition on rules that allows them to enter into the parsing process despite the difficulties outlined in Section 2.1 above.

Definition 23
In a LCSCFG (G, tr), a rule $B_0 \rightarrow B_1\alpha$ is said to be **left-introducible** iff for every γ that is an essential left context of B_0, there is a bottom-up rule $A_0 \rightarrow \overline{A_1} \ldots A_k$ such that

1. $\gamma = \chi\rho_1 \ldots \rho_i$ for some $i < k$
2. $A_1 \Uparrow^* \rho_1$
3. ρ_j can be coherently derived from A_j, for all $1 < j \leq i$
4. $A_{i+1} \rightsquigarrow B_0$

Scrutiny of this definition should reveal its relationship to the informal outlines in Sections 2.2 and 5.1 earlier (see also Figure 2). Notice that for any nonterminal A, if $S \stackrel{*}{\Rightarrow} A \ldots$ then the empty string is an essential left context of A and hence any rule of the form $A \rightarrow \ldots$ cannot be left-introducible.

Lemma 8
Let (G, tr) be a LCSCFG. Let T be an annotated tree generated by G, and M_0 a nonlexical node in T whose leftmost daughter is M_1, with M_0 labeled B_0, M_1 labeled B_1, and where

the start of both M_0 and M_1 is m. Suppose that the rule $B_0 \rightarrow B_1 \ldots$ licensing M_0 in T is left-introducible. Then in any fully bidirectional chart based on the terminal string spanned by T and using (G, tr), there is an active edge of the form

$$(l, m, A \rightarrow \ \bullet\alpha \bullet B_0\beta)$$

(i.e., an edge at the start of M_0, M_1, seeking B_0).

Proof
Let the string spanned by T be σ, with $\sigma = \sigma_1\delta\sigma_2$, where δ is spanned by M_0. Let $\mathcal{C}$ be a fully bidirectional chart based on σ and using (G, tr). By Lemma 5, σ_1 is of the form $\phi\gamma$ where γ is in the essential left contexts of B_0. Since $B_0 \rightarrow B_1 \ldots$ is left-introducible, every such γ has the property that there is a bottom-up rule $A_0 \rightarrow A_1 \ldots A_k$ such that

- $\gamma = \chi\rho_1 \ldots \rho_i$
- $A_1 \Uparrow^* \rho_1$
- ρ_j can be coherently derived from A_j, $1 < j \leq i$
- $A_{i+1} \rightsquigarrow B_0$

It follows from Lemma 6 that, since $A_1 \Uparrow^* \rho_1$, there are inactive edges in $\mathcal{C}$ for all nodes of a tree with root label A_1 spanning ρ_1. Since $A_0 \rightarrow A_1 \ldots A_k$ is bottom-up, this means there is an active edge in $\mathcal{C}$

$$(j, j', A_0 \rightarrow \ \bullet A_1 \bullet A_2 \ldots A_k)$$

where j is the start of the inactive edge for the root of this tree (i.e., the node labeled A_1), and j' is its end. By Lemma 1 and Lemma 4, there are inactive edges in $\mathcal{C}$ labeled $A_2, \ldots, A_i$, corresponding to nodes spanning $\rho_2 \ldots \rho_i$. By Lemma 1, there is an edge

$$(j, m, A_0 \rightarrow \ \bullet A_1 \ldots A_i \bullet A_{i+1} \ldots A_k)$$

where m is the start of δ. Since $A_{i+1} \rightsquigarrow B_0$, by Lemma 7 there is an active edge

$$(l, m, A \rightarrow \ \bullet\alpha \bullet B_0\beta)$$

□

5.5 Indirect Analyzability

In Section 4 we defined direct analyzability as a condition on grammars that would lead to complete parsing. Now we establish a more general property that also leads to completeness.

Definition 24
Let (G, tr) be a LCSCFG. A nonterminal symbol A_0 in G is said to be **indirectly analyzable** iff every rule $A_0 \rightarrow \omega$ is *either* lexical, *or* top-down and left-introducible, *or* bottom-up of the form $A_0 \rightarrow \overline{A_1}\alpha$ where A_1 is indirectly analyzable.[5]

5 Like the definition of directly analyzable in Section 4, this strictly needs a more detailed definition to allow for cycles. This is straightforward to provide, in exactly the manner used in that earlier section, and then the "definition" given here becomes a theorem about indirect analyzability.

Definition 25
A LCSCFG (G, tr) is **indirectly analyzable** iff for every purely bottom-up rule $A_0 \rightarrow \overline{A_1}\alpha$, the nonterminal symbol A_1 is indirectly analyzable.

The next two lemmas ensure that a grammar with the property of indirect analyzability leads to complete parses. The first is just a generalization of Lemma 4.

Lemma 9
Let (G, tr) be a BSCFG that is indirectly analyzable. Let T be a tree in $trees(G)$, spanning the string σ. Let $\mathcal{C}$ be a fully bidirectional chart based on σ and using (G, tr). Then for any node M labeled A in T:

1. If M is licensed by a purely bottom-up rule, then $\mathcal{C}$ contains a representation of the subtree rooted at M.
2. If M is licensed by a top-down rule, and there is in $\mathcal{C}$ an active edge

$$(t, g, B_0 \rightarrow B_1 \ldots B_{p-1} \bullet B_p \ldots B_{q-1} \bullet B_q \ldots A_n)$$

where either $B_{p-1} = A$ and t is the end of M, or $B_q = A$ and g is the start of M, then $\mathcal{C}$ contains a representation of the subtree rooted at M.

Proof
By induction on the height of nodes, in a manner very similar to Lemma 3, except that part (a) of the Inductive Step is as follows:

> Inductive Step(a): Suppose M_0, with daughter nodes $M_1, \ldots, M_k$, is licensed by a purely bottom-up rule $A_0 \rightarrow \overline{A_1} \ldots A_k$. Then, since (G, tr) is indirectly analyzable, this means that A_1 is indirectly analyzable. Hence whatever rule licenses M_1, it must be either lexical, or top-down and left-introducible, or bottom-up with an indirectly analyzable symbol at the start of its RHS. In the lexical and bottom-up cases, the Base Case and Inductive Hypothesis establish that $\mathcal{C}$ contains a representation of the tree rooted at M_1. If the rule is top-down and left-introducible, there is an edge seeking its LHS symbol at the start of M_1, and so, by the Inductive Hypothesis, there is a representation of the tree rooted at M_1 in $\mathcal{C}$. Since $\mathcal{C}$ is fully bidirectional, it contains an edge spanning M_1 of the form:
>
> $$(l, h, A_0 \rightarrow \bullet A_1 \bullet A_2 \ldots A_k)$$
>
> Repeated applications of the Inductive Hypothesis and Lemma 1 (Corollary) establish that there is a representation of the tree rooted at M_0 in $\mathcal{C}$ (i.e., the Inductive Step).

Lemma 10
Suppose (G, tr) is an indirectly analyzable LCSCFG. Suppose $T \in trees(G)$, and $\mathcal{C}$ is a fully bidirectional chart based on the string spanned by T and using (G, tr). Then for any nonroot node M in T, if M is licensed by a top-down rule $A \rightarrow \omega$, then $\mathcal{C}$ contains an active edge at the start of M of the form $(t_1, t_2, B_0 \rightarrow \alpha \bullet \beta \bullet A \ldots)$ (i.e., seeking A).

Proof
By induction on the depth of nodes.

Inductive Hypothesis: Assume that for any node M of depth d', where $0 \leq d' < d$, the lemma holds.

Base Case: Suppose M is of depth 1 (i.e., a daughter of the root node). Suppose the root is licensed by a rule $S \rightarrow A_1 \ldots A_k$, where M_i is the ith daughter of the root ($1 \leq i \leq k$) and $M = M_j$.

(a) Assume this rule is purely bottom-up. Since (G, tr) is indirectly analyzable, A_1 is indirectly analyzable. Consider the rule that licenses M_1. It cannot be left-introducible, as $S \stackrel{R}{\Rightarrow} A_1 \ldots$ (see earlier remark about empty essential left contexts); hence it must be either lexical or bottom-up. By Lemma 9, C contains a representation of the subtree rooted at M_1. Since the rule $S \rightarrow A_1 \ldots A_k$ is bottom-up, C contains an active edge of the form $(0, 0, S \rightarrow \bullet A_1 \bullet A_2 \ldots A_k)$.

(b) Assume this rule is top-down. Then there must be an empty active edge $(0, 0, S \rightarrow \bullet \bullet A_1 \ldots A_k)$.

By repeated applications of Lemma 9 and Lemma 1, there are edges of the form

$$(0, t_i, S \rightarrow \bullet A_1 \ldots A_i \bullet A_{i+1} \ldots A_k)$$

$1 \leq i \leq (j-1)$. The last of these fulfils the condition.

Inductive Step: Let M be a node labeled A of depth $d > 1$, licensed by a top-down rule r. Let its mother node be N, of depth $(d-1)$, and the leftmost daughter of N be M_1. Consider the rule r' (of the form $A_0 \rightarrow A_1 \ldots A_k$), which licenses N.

(a) Suppose r' is purely bottom-up. Then M_1 is labeled with an indirectly analyzable symbol, A_1. Hence for the rule licensing M_1, three cases must be considered:

1. It is lexical. In this case, C contains a representation for M_1.
2. It is top-down and left-introducible. By Lemma 8, there is an edge at the start of M_1 seeking its label. If $M = M_1$, this establishes the Inductive Step in this situation. Otherwise, by Lemma 9, there is a representation in C for the subtree rooted at M_1.
3. It is bottom-up of the form $A_1 \rightarrow B_1 \ldots$ where B_1 is indirectly analyzable. By Lemma 9, there is a representation in C for the subtree rooted at M_1.

Since there is a representation in C for the subtree rooted at M_1, there is an (inactive) edge in C of the form $(i, j, A_1 \rightarrow \bullet \ldots \bullet)$, where i is the start of M_1 (and hence of N). Since r' is bottom-up and A_1 is the leftmost (trigger) symbol of its RHS, this leads to an active edge of the form $(i, i, A_0 \rightarrow \bullet \bullet A_1 \ldots A_k)$ for r' at the start of M_1 and N. By repeated applications of Lemma 1 and Lemma 9, there is an active edge seeking A at the start of M.

(b) Suppose r' is top-down. By the Inductive Hypothesis, there is an edge at the start of N seeking the label of N. Since r' is top-down, there is also an empty active edge for r' at that point. If $M = M_1$, this establishes the Inductive Step in this situation. Otherwise, by repeated applications of Lemma 1 and Lemma 9, there is an active edge seeking A at the start of M. □

Theorem 3
If a LCSCFG (G, tr) is indirectly analyzable, then tr is complete.

Proof
Follows from Lemma 9 and Lemma 10. □

5.6 Undecidability
We have established that the condition of indirect analyzability suffices to ensure completeness. Unfortunately, indirect analyzability is not a decidable property of annotated grammars, as we now show.

Theorem 4
It is undecidable whether an arbitrary LCSCFG is indirectly analyzable.

Proof
Suppose that there were a decision procedure for indirect analyzability. This could then be used to construct a decision procedure that determines for any two CFGs G_1, G_2 whether every member of $L(G_1)$ ends in a substring that is a member of $L(G_2)$. This is an undecidable problem (see Appendix B); hence, the indirect analyzability question is also undecidable. The construction proceeds as outlined below.

Suppose we have the two arbitrary CFGs G_1 and G_2 over the same alphabet V_T, and assume that their nonterminal alphabets V_N^1, V_N^2 do not intersect. Construct a LCSCFG as follows. The distinguished symbol S' is distinct from all symbols in $V_N^1 \cup V_N^2$. Use symbols $B_1, B_2, \ldots, B_6$ also not in $V_N^1 \cup V_N^2$. The purely bottom-up rules are all the rules of G_2, together with

$$B_1 \rightarrow \overline{B_2} B_3$$
$$B_6 \rightarrow \overline{S_2} B_2$$

where each S_i is the distinguished symbol of G_i. The purely top-down rules are all the rules of G_1 together with

$$\overline{S'} \rightarrow S_1 B_1$$
$$\overline{B_2} \rightarrow B_4 B_5$$

Also we include lexical rules:

$$B_3 \rightarrow a$$
$$B_4 \rightarrow b$$
$$B_5 \rightarrow c$$

for some terminal symbols a, b, c.

This LCSCFG is indirectly analyzable iff (by definition) every purely bottom-up rule has an indirectly analyzable symbol at the start of its RHS (the trigger position). All the rules taken directly from G_2 meet this condition, since all are bottom-up. So, therefore, does the rule $B_6 \rightarrow \overline{S_2} B_2$. All the rules taken directly from G_1, and the rule $\overline{S'} \rightarrow S_1 B_1$, do not affect the condition, since all are top-down. Hence the grammar is indirectly analyzable iff in the rule $B_1 \rightarrow \overline{B_2} B_3$, the trigger symbol B_2 is indirectly

analyzable. This depends on whether the only rule expanding B_2, $\overline{B_2} \rightarrow B_4 B_5$ is left-introducible. The essential left contexts of B_2 is the set $\{\gamma \in V_T^* \mid S_1 \stackrel{*}{\Rightarrow} \gamma\}$. The only symbol X for which $X \rightsquigarrow B_2$ is B_2 itself. Hence the only rule that meets the schema for left-introducibility is $B_6 \rightarrow \overline{S_2} B_2$. So $\overline{B_2} \rightarrow B_4 B_5$ is left-introducible iff every γ such that $S_1 \stackrel{*}{\Rightarrow} \gamma$ is of the form $\psi\rho$ with S_2 coherently derived from ρ. Since all G_2 rules are bottom-up, S_2 is coherently derived from ρ iff $S_2 \stackrel{*}{\Rightarrow} \rho$. Hence the left-introducibility of the rule in question is logically equivalent to $L(G_1) \subseteq V_T^* + L(G_2)$ (where $+$ indicates concatenation). □

6. Some Further Complications

So far, the proofs have shown that direct analyzability is a sufficient condition for completeness, and that indirect analyzability (a more general condition) is also sufficient. The question might be posed—is indirect analyzability *necessary* for completeness? In fact, it is not, as there is at least one other sufficient condition for completeness, not covered by indirect analyzability.

It is not worthwhile formalizing and analyzing these possibilities in detail, but a brief informal outline of one such condition may be helpful. This occurs where a set of rules that is not directly analyzable, and might seem to cause "blocking" as discussed in Section 2.1 earlier, is redeemed by interaction with other rules in the grammar. This is similar to the phenomenon analyzed in Section 5 above, but whereas the analysis above dealt with a configuration of rules that can be parsed bottom-up to the left of the problematic rule, there is an analogous condition on subtrees to the left that can be parsed top-down.

The following grammar illustrates this phenomenon.

$$
\begin{aligned}
S &\rightarrow \overline{H}\ K \\
\overline{S} &\rightarrow Z\ B \\
\overline{H} &\rightarrow E\ F \\
\overline{E} &\rightarrow P\ R \\
\overline{Q} &\rightarrow T\ V \\
\overline{Z} &\rightarrow H\ Q \\
K &\rightarrow \overline{Q}\ D \\
P &\rightarrow p \\
R &\rightarrow r \\
F &\rightarrow f \\
D &\rightarrow d \\
T &\rightarrow t \\
V &\rightarrow v
\end{aligned}
$$

Here, the grammar is not indirectly analyzable, as the purely bottom-up rule $K \rightarrow \overline{Q}\ D$ has a trigger category Q that is not indirectly analyzable. (The rule $S \rightarrow \overline{H}\ K$ is also problematic.) However, the only situation in which $K \rightarrow \overline{Q}\ D$ would be needed would be to parse a string *prftvd*. Since $S \rightsquigarrow H$, there will be an empty active edge introduced for $\overline{H} \rightarrow E\ F$ at the start of the string. This will parse *prf* (top-down) as an H, and this will combine with the active edge already introduced for $\overline{Z} \rightarrow H\ Q$, leading to the introduction of an empty active edge for $\overline{Q} \rightarrow T\ V$ at the start of the correct substring, *tvd*.

Intuitively, this is similar to the phenomenon defined earlier as left-introducible, but with the catalytic sequence of rules being triggered top-down from the distin-

guished symbol of the grammar. It is likely that some generalization could be made to cover this pattern of rules and those described in Section 5, but the undecidability result in Theorem 4 suggests that this would not improve matters—the more general property would also be undecidable.

7. Discussion

7.1 Other Bidirectional Schemes

As mentioned in Section 1 above, the ideas here were developed from a semi-formal proposal by Steel and de Roeck (1987). The formalization given here is a slight generalization, as it allows *multiple* possible triggers on the RHS of a rule, which Steel and de Roeck did not consider. Steel and de Roeck did not formalize their proposal in detail, and did not show how to check if such annotations could lead to the parser missing possible analyses (i.e., becoming incomplete), although they concede that this is an important issue.

Satta and Stock (1989, 1991, 1994) have developed various detailed and rigorous systems of chart-based parsing, including one (Satta and Stock 1989) that allows a form of purely bottom-up bidirectional parsing, but they do not explore the question of mixed strategy invocation of rules. Most of the mechanisms in their bottom-up method are aimed at avoiding redundant edges in the chart, a problem that has been ignored here by working at a more abstract, set-theoretic level. Satta and Stock provide a more algorithmic approach in which such issues are of concern. A practical implementation of the definitions given above might have to consider whether their system could be adopted to achieve greater efficiency. However, Willis (1996) points out that in some situations the scheme given in Satta and Stock (1989) can be *less* efficient (in terms of edges introduced to the chart) than a fairly naive implementation of a mixed strategy chart parser whose grammar is annotated to run bottom-up (essential for comparison with the Satta and Stock algorithm). This seems to be because the Satta and Stock method involves the introduction, when a constituent is found, of an edge for *every* rule with that type of constituent on its RHS.

7.2 Head Parsing

There has been a growth in interest over the past decade or so in head-driven parsing (e.g., Kay 1989). In these approaches, the parsing is guided by the fact that exactly one item on the right-hand side of a grammar rule is the **head** of the construction, in the sense that it is a linguistically important part of the rule. Some of these proposals have been formalized using chart parsing, and their properties explored.

Although some of the head-driven strategies are said to act "top-down," this refers to the parser exploring from a prediction of a specific nonterminal symbol in some region of the input, but not to rules being introduced because the grammar writer has indicated that it is to be introduced top-down in the sense used here. The head markings are always on the right-hand side of the rule, never the left-hand side (since that would not make sense for a linguistic head). Hence, head-driven parsing is, in terms of the approach defined here, a form of bottom-up parsing, and the issues of incompleteness resulting from a mixed strategy algorithm do not arise. The mixed strategy approach here (which was developed independently of the head-driven work) could be seen as a possible generalization of a very simple head-driven parser.

There are similarities between the bidirectional scheme here and the head-corner parser of Sikkel and op den Akker 1996, in which top-down predictions can arise either from the distinguished symbol (predicted to span the whole input) or by working outwards from the specified head constituent (as in the left and right extension

principles of Definition 9 in Section 3.3). They define a transitive reflexive relationship "$>_h^*$", which roughly means that $A>_h^*B$ if there is a chain of rules from A to B such that the left-hand side of each one is the head of the previous rule in the sequence. Sikkel and op den Akker's chart handling principles all have the precondition that the introduction of the new edge can happen only if the region of the input in question is spanned by a predictive edge seeking a symbol A such that $A>_h^*B$, where B is the label of the constituent or prediction being introduced. This is, as they make clear, comparable to using a left-corner oracle to avoid unnecessary edges in a more traditional parser. A similar optimization might be possible for a mixed strategy parser of the sort discussed here, by using the triggers in bottom-up rules in the same way that heads are used by Sikkel and op den Akker.

7.3 Extended Generalized Left-Corner Parsing

Stabler (1994) outlines a very general approach to top-down and bottom-up parsing of context-free grammars, in a somewhat different formal framework. Although his theoretical mechanisms are in some ways a generalization of the left-corner strategy-marked grammars discussed in Section 5 above, there is one respect in which they are slightly less general, and which places the chart-based proofs given above outside the scope of his results. Stabler defines a class of **extended generalized left-corner** (XGLC) parsers, by attaching an (extended) trigger function to a CFG. This function maps each pair consisting of a stack configuration and a rule to a prefix of the RHS of that rule. Intuitively, the rule indicates how much of the RHS of the rule has to be recognized before that rule is to be introduced into the parsing process; making this dependent on the parser's stack (which can hold both recognized symbols and predictions of symbols needed) allows some sensitivity to the parsing context. Stabler cites a proof that *all* such parsers are complete with respect to the original CFG. This may seem to conflict with the proofs offered above, but it is crucial that Stabler's trigger functions are defined to be *total* functions—for *any* stack configuration, there must be some prefix of the rule's RHS. To faithfully reproduce the notion of a top-down rule used in the mixed strategy chart system, the trigger function would have to be *partial*, indicating no prefix at all in those cases where the stack did not have the right prediction. It is reasonable to assume that Stabler's completeness proofs rely on the total nature of the trigger function, and thus do not cover the notion of mixed strategy parsing defined here.

7.4 Possible Uses

As mentioned in Section 1 above, the original Steel and de Roeck proposal was put forward as a way of improving the efficiency of parsers for natural languages, such as English. Although they did not have any real statistical evidence that this guidance leads to more efficient parsing, they claimed that it did appear to help, judging by the performance of the parser they had implemented for use in an English-language query interface. That approach is dependent on the grammar writer having some linguistic intuitions about which constituents are best parsed bottom-up and which are best parsed top-down. Alternatively, the rule annotations could be developed from statistics about rule usage in parsing suitably large corpora.

Some preliminary results (Willis 1996) suggest that on small grammars, gains of up to 35% can be made in efficiency (measured in terms of chart entries) by using certain combinations of the mechanisms formalized here. These gains are not great, and it is unclear whether similar improvements could be achieved in realistically large natural language grammars. The formal results in Sections 4 and 5 above suggest that it may not be worthwhile carrying out such experiments, unless grammars are restricted to those that are directly analyzable.

Context-free grammar has been used as the basis here, both to simplify the formalization, to achieve some degree of generality, and in order to relate the work to existing formal language theory. Steel and de Roeck also use a CFG base as an expository device for their ideas. However, it is extremely rare within computational linguistics for a pure CFG to be used in actual systems that parse natural language. Usually, some much more complex grammatical formalizm is used, such as **unification grammar** (Shieber 1986). Many of the methods for parsing unification grammars are closely based on traditional CFG parsing techniques, with enhancements. This means that an obvious extension of the theoretical definitions and results in this paper would be the application of mixed strategy bidirectional parsing to unification grammars. Most of the framework could be retained, since the main difference between a simple unification grammar formalism and CFG is in the way that nonterminal symbols are compared or combined with each other. It is highly improbable that the undecidability result would be overturned, and it is even conceivable that the appropriate counterpart of direct analyzability might turn out to be less tractable.

8. Conclusions

Although the idea of allowing the grammar writer to specify the strategy to be used for each rule in a grammar may seem superficially appealing, the formal evidence presented here is that it is severely limited. In general, grammar annotation may lead to incompleteness. Although there is a decidable property—direct analyzability—that guarantees completeness, it is overrestrictive, in the sense that there are complete annotations that are not directly analyzable. There is also a wider class of complete annotations—indirectly analyzable—that cannot be decidably detected.

There is also some question over the practical effectiveness of the mixed strategy technique, although that issue has not been explored here.

Appendix A: Computing Direct Analyzability

The algorithm is a simple variant of the use of an AND-OR graph in problem solving, as in Nilsson (1971). The graph will contain a node for each nonterminal symbol A in the grammar, and an OR node for each bottom-up rule. Each node has a label, which is either a nonterminal symbol or OR, and may, optionally, have a marking, which is either SOLVED or FAILED.

1. **For each** symbol $A \in V_N$, create a node N_A, and insert arcs and markings as follows:

> **if** there is a purely TD rule of the form $A \rightarrow \alpha$
> **then** mark N_A as FAILED
> **else if** all rules of the form $A \rightarrow \alpha$ are lexical
> **then** mark N_A as SOLVED
> **else**
> **for each** bottom-up rule of the form $A \rightarrow A_1 \ldots A_k$:
> - create a node N labeled OR;
> - create an arc *from* N_A *to* N;
> - create an arc *from* N *to* N_{A_i} for every $i \in tr(A \rightarrow A_1 \ldots A_k)$
> such that $i > 0$.

At this point, each non–terminally labeled node has outgoing arcs for every bottom-up rule that might expand it, and each of these arcs connects to an OR *node, which in turn connects to*

every possible trigger category for that rule. Nodes marked FAILED *correspond to categories that are not directly analyzable; nodes marked* SOLVED *correspond to those that are directly analyzable. Initially, any node marked* SOLVED *or* FAILED *has no outgoing arcs.*

2. **Repeat** until no changes occur in the graph:

for each node N in the graph:

if N is marked FAILED
then delete any arc into N from a node M;
if N is labeled OR, **or** there are no other outgoing arcs from M;
then - mark M as FAILED;
- remove any outgoing arcs from M;
if N is marked SOLVED
then if there is an arc into N from an OR-node M
then - mark M as SOLVED;
- remove any outgoing arcs from M.
else if there is an arc into N from a node M_A
then delete this arc from M_A to N
if this leaves no outgoing arcs from M_A
then mark M_A as SOLVED.
if N is an OR node with no incoming arcs
then delete N and all its outgoing arcs.

The properties remarked above remain invariant during this iteration. The iteration terminates as the graph is finite. On termination, the only arcs left must be in cycles. The categories associated with any nodes in cycles should be taken as directly analyzable.

3. For every node N_A that has an arc (incoming or outgoing) attached to it, mark N_A as SOLVED.

4. If for every purely bottom-up rule $A \rightarrow A_1 \ldots A_k$, there is an $i \in tr(A \rightarrow A_1 \ldots A_k)$ such that N_{A_i} is marked SOLVED, then the grammar is directly analyzable.

The above statement is not intended to be maximally efficient. No formal proof of its correctness is given here, but there is a fairly straightforward relationship to the property of direct analyzability, which is stated as the draft definition in Section 4.2.

Appendix B: Undecidability Proof

Lemma
For any two context-free grammars G_1, G_2, it is undecidable whether every member of $L(G_1)$ ends in a substring that is a member of $L(G_2)$.

Proof
Let G_1 and G_2 be two CFGs over the same alphabet V, with languages $L(G_1)$ and $L(G_2)$ respectively. Let # be a symbol that is not a member of V. Consider the language L_1' given by:

$$\{\#x \mid x \in L(G_1)\}$$

and L_2' given by:

$$\{\#y \mid y \in L(G_2)\}$$

These are both context-free languages; assume that grammars G_1', G_2' generate them. Suppose we have a procedure that would decide, for any two context-free grammars, whether every member of the language of one ends in a substring that is a member of the language of the other. Consider the question whether every member of $L(G_2')$ (i.e., L_2') ends in a substring that is a member of $L(G_1')$ (i.e., L_1'). This is true iff every string of the form $\#y$ in L_2' has a final substring that is in $\{\#x \mid x \in L(G_1)\}$. Since # is not in V, this can be true iff $y \in L(G_1)$. This will be true for every such string in L_2', iff $y \in L(G_1)$ for every $y \in L(G_2)$; i.e., $L(G_2) \subseteq L(G_1)$

That is, a decision procedure for the final substring question would allow the construction of a decision procedure for the subset question for the languages generated by two arbitrary context-free grammars, which in turn would provide a decision procedure for the equivalence of the languages, and that is known to be undecidable (Aho and Ullman 1972, Sect. 2.6.3). □

Acknowledgments

I would like to thank Anne de Roeck, Alistair Willis, and Suresh Manandhar for useful discussions, and Nicolas Nicolov for comments on an earlier draft. The incisive and thorough comments of various anonymous reviewers have greatly improved this paper.

References

Aho, Alfred V. and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling. Volume 1: Parsing*. Prentice-Hall, Englewood Cliffs, NJ.

Earley, Jay. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2):94–102.

Kay, Martin. 1989. Head-driven parsing. In *Proceedings of the International Workshop on Parsing Technologies*, pages 52–62, Carnegie Mellon University, Pittsburgh, PA, August.

Maclane, Saunders and Garrett Birkhoff. 1967. *Algebra*. Macmillan, London.

Nilsson, Nils J. 1971. *Problem-solving methods in artificial intelligence*. McGraw-Hill, New York.

Partee, Barbara H., Alice ter Meulen, and Robert E. Wall. 1990. *Mathematical Methods in Linguistics*. Kluwer Academic, Dordrecht.

Satta, Giorgio and Oliviero Stock. 1989. Formal properties and implementation of bidirectional charts. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1480–1485.

Satta, Giorgio and Oliviero Stock. 1991. A tabular method for island-driven context-free grammar parsing. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-91)*, pages 143–148.

Satta, Giorgio and Oliviero Stock. 1994. Bidirectional context-free grammar parsing for natural language processing. *Artificial Intelligence*, 69:123–164.

Shieber, Stuart. 1986. *An Introduction to Unification Approaches to Grammar*. CSLI Lecture Notes Number 4. Center for the Study of Language and Information.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and Implementation of Deductive Parsing. *Journal of Logic Programming*, 24(1 & 2):3–36.

Sikkel, Klaas and Rieks op den Akker. 1996. Predictive head-corner chart parsing. In Harry Bunt and Masaru Tomita, editors, *Recent Advances in Parsing Technology*. Kluwer Academic, Netherlands, chapter 9, pages 169–182.

Stabler, Edward P. 1994. Parsing for incremental interpretation. Unpublished paper, UCLA, Los Angeles, CA.

Steel, Sam and Anne de Roeck. 1987. Bidirectional chart parsing. In J. Hallam and C. Mellish, editors, *Advances in Artificial Intelligence*. John Wiley, pages 223–235.

Stoy, Joseph E. 1981. *Denotational Semantics: the Scott-Strachey approach to programming language theory*. MIT Press, Cambridge,

MA.
Thompson, Henry and Graeme Ritchie. 1984. Implementing natural language parsers. In T. O'Shea and M. Eisenstadt, editors, *Artificial Intelligence: Tools, Techniques and Applications*. Harper and Row, New York, Chapter 9, pages 245–300.
Willis, Alistair. 1996. *Exploring Chart Parsing Mechanisms*. Master's thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland.
Winograd, Terry. 1983. *Language as a Cognitive Process. Volume I: Syntax*. Addison-Wesley, Reading, MA.