Completeness Conditions for Mixed Strategy Bidirectional Parsing

Graeme Ritchie
Department of Artificial Intelligence
University of Edinburgh
80 South Bridge
Edinburgh EH1 1HN
Scotland
g.d.ritchie@ed.ac.uk

June 12, 1997

Abstract

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 bi-directional 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 which cannot be found by the parser. We formalise a fairly natural notion of mixed strategy bi-directional 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 which 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 categorised 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 [9] 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 $NP \rightarrow NP\ CONJ\ NP$ (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 labelling of rules. The current paper does not take a position on the wisdom or effectiveness of such labelling. Instead, we explore the 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 are also 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 with that symbol as its label is found by the parser. Using a method of parsing known as *active chart parsing* it is straightforward to give a precise meaning to this labelling of rules, since a chart parser can operate either bottom-up or top-down.

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

A property of annotated grammars (*direct analysability*) 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, there are some annotated grammars which are not directly analysable, but which nevertheless lead to complete parsing. A characteristic of (a subset of) this wider class of annotated grammars (*indirect analysability*) is defined, and it is proven that any annotated grammar with this property will allow complete parsing. However, indirect analysability 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 which 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 formalisation in Section 3.3 below). For this informal presentation, and occasionally elsewhere, we shall mark a trigger symbol A by underlining it thus: $\underline{A}$.

In these illustrative examples, the distinguished (initial) symbol of the grammar will always be "S" and terminal symbols will be in lower case. Consider the annotated grammar:

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

It should be intuitively clear that although this grammar generates exactly one sentence, that string cannot be parsed by a parser which follows the annotations as described. The rule $\underline{S} \rightarrow NP\ VP$ cannot be used until an initial NP is recognised, and the rule which might do that, $NP \rightarrow \underline{Art}\ N$, cannot be used until an initial need for an NP is established (which could happen only using $\underline{S} \rightarrow NP\ VP$). 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 which 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 non-trivial 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 non-trivial characteristic of annotated grammars which 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 edge in a chart can be used to combine with any other compatible edge, regardless of whether there is a single coherent tree which will result from it. In particular, an edge which has been inserted in the chart as the result of some rule interaction which 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:

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

The unstrategy-marked version of this grammar would generate the string $jlmk$, with a derivation as follows:

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

This derivation is not permitted by 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:

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

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 generate $jlmk$, with an associated syntax tree which corresponds to the derivation given above (i.e. a tree which makes no direct use of the rules which have been added to the grammar). The way in which the added rules act as a "catalyst" to allow the hitherto blocked derivation 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 non-terminal H cannot be expanded as required, since it is on the LHS of a bottom-up rule, and its first symbol B cannot be recognised because it requires a top-down rule. In both the original grammar and the larger grammar, H is introduced only by the rule $\underline{S} \rightarrow E\ H$, i.e. with E on its immediate left. So the only strings where H can participate in a derivation 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 pre-terminal, it can be recognised directly (with no effect from annotations). In the larger grammar, the bottom-up rule $D \rightarrow \underline{E}\ A$ is then introduced to the parsing, which creates a predictive entry in the parser's structures seeking an A, after the recognised E. The top-down rule $\underline{A} \rightarrow B\ C$ is then introduced, which leads to an entry, at that same point, seeking a B. This causes the top-down rule $\underline{B} \rightarrow P\ Q$ (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 $H \rightarrow \underline{B}\ F$, 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, *direct analysability*, which guarantees completeness, and which would 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 we shall call *indirect analysability*, which also guarantees completeness. However, this 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 shall adopt the standard concepts for syntax trees (see [1, Section 0.5] for a possible approach to formalisation). A *syntax tree* is a rooted, ordered, labelled tree. Each node apart from the root has exactly one *mother* node, and each non-terminal 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)

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

We assume completely standard notions for simple context-free grammars and their associated trees, which will be sketched out below. Following the usual convention (e.g. [1]), we will take a CFG to be a quadruple (V_N, V_T, P, S), consisting of a set V_N of *non-terminal 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 non-terminal symbol and the second is a tuple of symbols (i.e. of the form $(A_0, (A_1, \ldots, A_k))$) 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 $A_0 \rightarrow A_1 A_2 \ldots A_k$ 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 [1, Sect 2.4.2].

3. There are no rules whose right hand side is empty.

Rules of the form $A \rightarrow w$ where $w \in V_T$ will be referred to as *lexical rules*, and other rules as *non-lexical*.

Given a CFG G, a *syntax tree based on* G is an rooted, ordered tree whose non-terminal nodes are labelled with elements of V_N and whose terminal nodes are labelled with elements of V_T. Those nodes which immediately dominate terminal nodes will be referred to as *pre-terminal*; other non-terminal nodes will be referred to as *non-lexical*. Where a tree T spans a terminal sequence $a_1 \ldots a_n$, and M is a node within T which 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 non-terminal node with label A_0 and daughter nodes labelled $A_1, ..., A_k$, there is a rule in P of the form $A_0 \rightarrow A_1, ...A_k$; this rule is said to *license* the node and its daughters. For convenience, we shall distinguish between a tree which is compatible with the rules of the grammar, and a tree which also spans a sentence. A syntax tree is said to be *generated* by a grammar G iff:

1. The root node is labelled 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 one of the following is true:

1. $\omega_1 = \omega_2$

2. $\omega_1 = \delta A \gamma$, $\omega_2 = \delta \alpha \gamma$ and $A \rightarrow \alpha$ is a rule in G.

Similarly, ω_1 *derives* ω_2 (written " $\omega_1 \stackrel{*}{\Rightarrow} \omega_2$") if either

1. $\omega_1 \Rightarrow \omega_2$

2. there is a string ω_3 such that $\omega_1 \Rightarrow \omega_3$ and $\omega_3 \stackrel{*}{\Rightarrow} \omega_2$

3.2 Annotated Trees and Grammars

Definition: 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 P to $\mathcal{P}(\mathcal{N})$ (the set of sets of non-negative 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: A *bidirectionally strategy-marked context free grammar* ("BSCFG" for short) is a pair (G, tr) where G is a CFG and tr is a bidirectional strategy marking of G.

Definition: 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 $tr(r) = \{0\}$.

Definition: Let (T, f) be a pair where T is a syntax tree, and f is a function from the non-lexical nodes of T to the set of subsets of $\{\uparrow, \downarrow\}$. (That is, f associates with each non-lexical node either $\emptyset$, $\{\uparrow\}$, $\{\downarrow\}$ or $\{\uparrow, \downarrow\}$.) Then (T, f) is said to be an *annotated tree*.

An annotated tree will be used here to represent a tree in which markings on the nodes describe which types of rule (bottom-up or top-down) could be used to build it. Informally, if a rule contains $\downarrow$ in its annotation, then it (and its immediate daughters) are licensed by a top-down rule; if a rule contains $\uparrow$ in its annotation, then there is a bottom-up rule which licenses this node, its sibling nodes, and its mother, with this node corresponding to a "trigger" element in the bottom-up rule.

Definition: An annotated tree (T, f) is said to *conform* to a BSCFG (G, tr), if for every non-lexical node M_i (with mother node M_0 and sibling nodes $M_1, \ldots, M_{i-1}$ and $M_{i+1} \ldots M_k$) in T:

1. M_i must have either $\uparrow \in f(N)$ for some daughter N of M_i, or $\downarrow \in f(M_i)$, but not both.
2. if $\uparrow \in f(M_i)$, there is a bottom-up rule r in (G, tr) such that r licenses M_0 and $M_1 \ldots M_k$, and $i \in tr(r)$.
3. if $\downarrow \in f(M_i)$, there is a top-down rule in (G, tr) which licenses M_i and its daughters.

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 generalisation of Earley's algorithm [2], and tutorial expositions of the ideas can be found in [10] or [12]. Although most presentations express the parsing method in an algorithmic form, we shall here abstract from these more procedural statements to provide a declarative definition of how a chart characterises the structure of a sentence according to a grammar. That is, parsing principles which are usually summarised in a form such as "if the chart has property P, add an item of the form I to the chart" will here be stated as "any chart which has property P should contain an item of the form I". That is, we will state parsing principles as well-formedness conditions on complete charts, abstracting away from the sequence of steps used to build them. This will simplify the formal proofs without losing their essential meaning, and will also allow the results to cover alternative implementations which meet the principles in different procedural ways.

An active chart parser operates by maintaining a record of what has been parsed so far, and which items are required to complete constituents of the proposed sentence, in a uniform representation. Each record of information is usually known as an *edge*, and the collection of current edges is the *chart*. In a conventional chart for simple left-to-right parsing, each edge contains a "dotted rule" which shows how far the left-to-right process has reached in using that rule at that point in the parse. For a bidirectional parser, it is necessary to indicate progress from either end of the rule, or both.

Definition: Given a CF grammar 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 $A_0 \rightarrow A_1 \ldots A_n$ and l, r are integers such that $0 \leq l \leq r \leq n$.

Such a rule will be written as:

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

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

Definition: 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 non-negative integers with i $\leq$ j, and d is a double-dotted rule based on G. Such an edge is known as *non-lexical* if the rule is non-lexical, and is a *lexical edge* if the rule is a lexical rule.

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

1. for every $(i, j, r) \in \mathcal{C}$, $i \in \{0, \ldots, n\}$ and $j \in \{0, \ldots, n\}$
2. for every $a_i \in \{a_1, a_2, \ldots, a_n\}$, and for every rule of the form $L \rightarrow a_i \in P$, there is an element $(i-1, i, L \rightarrow \ \bullet a_i \bullet\ ) \in \mathcal{C}$.
3. there are no other lexical edges in $\mathcal{C}$.

Notice that the second stipulation in this definition corresponds to something that would be treated as the initialisation phase of the procedural construction of a chart for a sentence.

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

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 non-empty components 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*. Informally, edges represent constituents found so far by the parser, or rules which have been introduced to the parsing process. The "dots" in a dotted rule indicate progress through the right hand side of the rule, indicating which required constituents have been found so far, and the first two components of the edge indicate the region of the string concerned. Inactive edges represent complete constituents which have been found by the parser and active edges indicate partial constituents which require further constituents to become whole. 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; it represents a constituent none of which has been found yet, but for which a suitable rule has been introduced into the chart. (Sometimes it will be referred to as "an empty active edge for $A_0 \rightarrow A_1 \ldots A_k$".)

The central idea of active chart parsing (sometimes known as the *fundamental rule*) is that wherever there is an adjacent compatible pair of an active and an inactive edge, a further edge should also be in the chart to represent the combined information of these two edges. The notion of "compatibility" required here is that the inactive edge represents a constituent of category A, and the active edge requires a constituent of the type A as the next part of the right hand side of its dotted rule. In a conventional left-to-right chart, the right end (second component) of the active edge and the left end (first component) of the inactive edge must coincide; for bidirectional parsing, the converse is also allowable.

Definition: Let G 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 resolved* iff *both* the following conditions hold:

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

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

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

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

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

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

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

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

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

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

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

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

(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:

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

Proof: Both the cases (i) and (ii) proceed by induction on the number of inactive edges. Both the base case and the inductive step rely on the fact that the chart is bidirectionally resolved, and hence the existence of an adjacent active edge and inactive edge necessitates a "combined" edge. □

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 $(p+t) = n$ so that there are $n-p$ inactive edges to the right, all with labels matching the rule), then there is a complete (inactive) edge of the form

$(i_1, j_n, B_0 \rightarrow \bullet B_1 \ldots B_n \bullet)$

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. The latter is normally done in one of two ways, either bottom-up or top-down. For a strategy-marked grammar, the rule-introduction principle is a combination of these two, but is sensitive to the annotation of the rules.

Definition: 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 $\mathcal{C}$:

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

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

2. (Top-down initialisation) For every rule in G of the form $S \rightarrow B_1 \ldots B_k$, where S is the distinguished symbol of G, 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:

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

where $0 \leq p < n$, 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:

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

   where $0 < q \leq n$, 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:

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

We will frequently have to refer to a chart which is both bidirectionally resolved and bidirectionally mixed strategy explored. For brevity, the term *fully bidirectional* will be used for such charts.

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: Let T be a tree which spans the string $a_1 a_2 \ldots a_n$, and let M_0 be a non-terminal node in T which spans $a_{i+1} \ldots a_j$, is labelled A_0, has k daughters, and these daughters are labelled $A_1, \ldots A_k$ respectively. Then the triple $(i, j, A_0 \rightarrow \bullet A_1 \ldots A_k \bullet)$ is said to be an *(inactive) edge for* M_0, written "$edge(M_0)$"

Notice that for any chart $\mathcal{C}$ based on a string σ, $\mathcal{C}$ will contain an edge for each pre-terminal node of any tree which 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 non-terminal nodes being subject to verification.

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

1. T spans a substring of σ (not necessarily proper);
2. for every non-terminal node N in T, $edge(N) \in \mathcal{C}$.

With these definitions, the notion of a chart being "complete" is very straightforward:

Definition: Using the notation above, a chart $\mathcal{C}$ based on σ is *complete* w.r.t. a grammar G iff for every tree $T \in trees(G)$ which spans σ, $\mathcal{C}$ contains a representation of T.

The central aim of this paper is to explore the conditions under which a BSCFG is guaranteed to lead to a complete chart.

Definition: Given a CFG G, a bi-directional strategy-marking tr of G is said to be *complete* iff for every string $\sigma \in$ L(G), any fully bidirectional chart $\mathcal{C}$ based on σ and using (G, tr) is complete w.r.t. G.

4 A decidable class of complete annotations

As observed in Section 2.1, certain grammar annotations will prevent the parser from finding some valid analyses. It is possible to define a *decidable* property of grammars which detects if all generated trees will have well-behaved (i.e. parsable) annotations, or if there may be potential blocks. This property, which we call *direct analysability*, is a sufficient condition for completeness, but it is not a necessary condition. There is also a weaker (more general) sufficient condition for completeness, *indirect analysability*, which is defined in Section 5 below, but the fact that the stronger condition is decidable makes it worth defining. (The later proofs in Section 5 also make use of some of the concepts related to direct analysability.)

The approach to developing these concepts proceeds in several stages. Firstly, we shall define a type of annotated tree which is guaranteed to be parsable by a mixed-strategy chart parser as defined above;

then we shall define a class of strategy-marked grammars which are guaranteed to produce at least one annotated tree of this type for each valid tree. It follows that this class of strategy-marked grammars can be parsed completely. Moreover, the criterion for membership of this class of grammar is very simple.

4.1 Reachability

As hinted at in the informal discussion in Section 2.1 above, there is a notion of "reachability" within annotated trees which is relevant to the formalisation. We need the concept of there being a route whereby a rule can be triggered bottom-up from the words in the sentence, with no intervening top-down rules. The next two definitions will (when used along with later definitions) describe that situation.

Definition: In an annotated tree (T, f), a non-lexical node M_0 is said to be *reachable from below* iff there is a daughter M_i of M_0 such that $\uparrow \in f(M_i)$ and one of the following is true:

1. M_i is a pre-terminal node of T;
2. M_i is reachable from below.

Definition: An annotated tree (T, f), is said to be *coherently marked* iff, for every non-lexical node $M \in T$ such that $\uparrow \in f(M_i)$ for some daughter M_i of M, M is reachable from below.

Lemma 2 *If a tree is coherently marked, so are all its subtrees.*

Proof: Follows directly from the definitions.□

4.2 Parsing

To proceed further, we require more definitions relating tree nodes to edges in a chart. This section covers some essential notions which are also needed when we come to explore an alternative condition for completeness (Section 5). The following definitions formalise the notion of a subtree being capable of being built directly by the parser.

Definition: Let T be a tree spanning the string σ, and $\mathcal{C}$ be a chart based on σ. A non-lexical node M in T is said to be:

1. *directly parsed in* $\mathcal{C}$ iff $edge(M) \in \mathcal{C}$ and $edge(M') \in \mathcal{C}$ for all non-terminal nodes M' dominated by M (i.e. $\mathcal{C}$ contains a representation of the subtree rooted at M).
2. *rightwards conditionally parsed in* $\mathcal{C}$ iff if there is an active edge in $\mathcal{C}$ of the form

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

   where A_m is the label of M and j is the start of M, then $edge(M) \in \mathcal{C}$ and $edge(M') \in \mathcal{C}$ for all non-terminal nodes M' dominated by M.
3. *leftwards conditionally parsed in* $\mathcal{C}$ iff, if there is an active edge in $\mathcal{C}$ of the form

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

   where A_k is the label of M and j is the end of M, then $edge(M) \in \mathcal{C}$ and $edge(M') \in \mathcal{C}$ for all non-terminal nodes M' dominated by M.
4. A node N is *conditionally parsed in* $\mathcal{C}$ iff it is *both* leftwards conditionally parsed in $\mathcal{C}$ *and* rightwards conditionally parsed in $\mathcal{C}$.

Definition: Let (T, f) be an annotated tree spanning σ, M_0 a node in T and $\mathcal{C}$ be a chart based on σ. M_0 is said to be *parsable in $\mathcal{C}$ from (T, f)* iff either $\uparrow \in f(M_i)$ for some daughter M_i of M_0 and M_0 is directly parsed in $\mathcal{C}$, or $\downarrow \in f(M_0)$ and M_0 is conditionally parsed in $\mathcal{C}$. (Often the phrase "from (T, f)" will be omitted if it is clear from the context).

The next lemma establishes that the formal notion of "parsable" does indeed mean, intuitively, "able to be parsed".

Lemma 3 *In a tree T, let $M_1, \ldots M_k$ be the daughters of node M_0, such that A_i is the label for $M_i, 0 \leq i \leq k$. Let (T, f) be an annotated tree, conforming to the BSCFG (G, tr). Suppose that $\mathcal{C}$ is a fully bidirectional chart based on the terminal string spanned by T and using (G, tr), that $\mathcal{C}$ contains an active edge of the form:*

$$(i_j, i_p, A_0 \rightarrow A_1 \ldots A_j \bullet A_{j+1} \ldots A_p \ \bullet A_{p+1} \ldots A_k)$$

where i_j is the end of M_j and i_p is the end of M_p, and that for $j < i \leq p$, $\mathcal{C}$ contains a representation of the subtree rooted at M_i. If M_i is parsable in $\mathcal{C}$ from (T, f), for $1 \leq i \leq k$, then there are inactive edges in $\mathcal{C}$ for M_0 and all nodes dominated by it.

Proof: By induction on the daughter nodes of M_0.

Consider the nodes $M_{p+1} \ldots M_k$ (where parsing must proceed rightwards).

Base Case: Since (T, f) conforms to the grammar (G, tr), either there is a daughter N of M_{p+1} for which $\uparrow \in f(N)$, or $\downarrow \in f(M_{p+1})$. Hence, since M_{p+1} is parsable in $\mathcal{C}$, either M_{p+1} is directly parsed in $\mathcal{C}$ or M_{p+1} is conditionally parsed in $\mathcal{C}$. If it is directly parsed, then $\mathcal{C}$ contains a representation of the subtree rooted at M_{p+1}, by definition; if it is conditionally parsed, it is rightwards conditionally parsed, and hence the presence of the edge

$$(i_j, i_p, A_0 \rightarrow A_1 \ldots A_j \bullet A_{j+1} \ldots A_p \ \bullet A_{p+1} \ldots A_k)$$

ensures that $\mathcal{C}$ contains a representation of this subtree.

Inductive Hypothesis: For any $0 < i < n$, each node M_{p+i} has a representation of its subtree in $\mathcal{C}$ (where $0 < n < k - p$).

Inductive Step: The Inductive Hypothesis, together with Lemma 1, entails that there is an active edge derived from the original active edge, spanning the daughters $M_{j+1} \ldots M_{n-1}$ and with its right vertex at the start of M_n. Reasoning similar to that for the Base Case then ensures that there is a representation in $\mathcal{C}$ for the subtree rooted at M_n. This establishes the induction.

The leftwards situation is similar.

The presence of the edges for all the nodes $M_1, \ldots, M_k$, and the Corollary to Lemma 1 ensures that there is an edge

$$(i_0, i_k, A_0 \rightarrow \ \bullet A_1 \ldots A_k \ \bullet \ )$$

i.e. an inactive edge for M_0. □

Notice the two special cases of this, where there is an *empty* active edge at the extreme left

$$(i_0, i_0, A_0 \rightarrow \ \bullet \ \bullet A_1 \ldots A_k)$$

or the similar special case with an empty edge at the extreme right.

Now we need to show that coherent marking of a tree guarantees parsability.

Lemma 4 *Let T be a tree generated by G, spanning the string σ, where (G, tr) is a BSCFG. Let $\mathcal{C}$ be a fully bidirectional chart based on σ and using (G, tr). If there is an annotated tree (T, f) which conforms to this grammar, and which is coherently marked, then every non-lexical node in T is parsable in $\mathcal{C}$ (from (T, f)).*

Proof: By induction on the height of nodes. Suppose (T, f) is an annotated tree meeting the requirements. Suppose $\mathcal{C}$ is a fully bidirectional chart based on the terminal string which T spans, using (G, tr).

BASE CASE: Consider a node M_0 of height 2 (the lowest non-terminal which is not a pre-terminal). Suppose M_0 is coherently marked. Since M_0 is of height 2, all its daughters must be of height 1 (pre-terminal). By definition of $\mathcal{C}$ as a chart for the string spanned by T, there are inactive edges in $\mathcal{C}$ for all these daughter nodes.

(a) Suppose $\uparrow \in f(M_j)$ for some daughter M_j of M_0. Since f conforms to (G, tr), there is a bottom-up rule $A_0 \rightarrow A_1 \ldots A_k$ licensing M_0 and its daughters, with $j \in tr(A_0 \rightarrow A_1 \ldots A_k)$. Since $\mathcal{C}$ is fully bidirectional, and there is an inactive edge in $\mathcal{C}$ for M_j, there is an active edge

$$(i, p, A_0 \rightarrow A_1 \ldots A_{j-1} \bullet A_j \bullet A_{j+1} \ldots A_k)$$

in $\mathcal{C}$ for this rule, spanning the edge for M_j. Since there are inactive edges in $\mathcal{C}$ for all the daughter nodes $M_1, \ldots, M_k$, by the Corollary to Lemma 1 there is an inactive edge in $\mathcal{C}$ for M_0. There are also edges for all the non-terminal nodes M_i dominated by M_0. Hence M_0 is directly parsed, and (since $\uparrow \in f(M_j)$), parsable.

(b) Suppose $\downarrow \in f(M_0)$. Suppose there is an active edge

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

in $\mathcal{C}$ where j is the start of M_0, and A_m is M_0's label. Since f conforms to (G, tr), there is a top-down rule $A_m \rightarrow B_1 \ldots B_n$ licensing M_0 and its daughters. Since $\mathcal{C}$ is fully bidirectional, there is an empty active edge

$$(j, j, A_m \rightarrow \bullet \bullet B_1 \ldots B_n)$$

in $\mathcal{C}$ for this rule at j, the start of M_0. Hence by Corollary to Lemma 1, there is an inactive edge in $\mathcal{C}$ for M_0. There is also an edge in $\mathcal{C}$ for each non-terminal node dominated by M_0. Hence M_0 is rightwards conditionally parsed. A similar argument shows that M_0 is also leftwards conditionally parsed, and hence conditionally parsed. Since $\downarrow \in f(M_0)$, M_0 is parsable.

INDUCTIVE HYPOTHESIS: For any node M_0 in T of height $m < n$, if the subtree rooted at M_0 is coherently marked (within (T, f)), every non-lexical node in that subtree is parsable in $\mathcal{C}$.

INDUCTIVE STEP: Let M_0 be a node of height $n > 2$, the root of a subtree which is coherently marked. All its daughters M_i are of height less than n. They are also, by Lemma 2, roots of subtrees which are coherently marked. Hence by the INDUCTIVE HYPOTHESIS all non-lexical nodes in these daughter subtrees are parsable. Since (T, f) conforms and is coherently marked, either $\downarrow \in f(M_0)$, or there is some daughter M_j of M_0 such that $\uparrow \in f(M_j)$ and M_j is reachable from below. Consider these two cases:

(a) Suppose $\downarrow \in f(M_0)$. To see if M_0 is parsable, have to determine that it is conditionally parsed. So suppose there is an edge in $\mathcal{C}$ of the form

$$(i, j, A_0 \rightarrow A_1 \ldots \bullet A_p \ldots A_{m-1} \bullet A_m \ldots A_k)$$

where j is the start of M_0 and A_m is M_0's label. Since the tree conforms, there is a top-down rule licensing M_0 and its daughters. Since the chart is fully bidirectional, there is an empty active edge in $\mathcal{C}$ for this rule at the start of M_0. So by Lemma 3 above, there are inactive edges in $\mathcal{C}$ for M_0 and all nodes dominated by it. Hence, M_0 is rightwards conditionally parsed in $\mathcal{C}$.

A similar argument shows that M_0 is also leftwards conditionally parsed in $\mathcal{C}$, hence conditionally parsed in $\mathcal{C}$, and hence parsable in $\mathcal{C}$.

(b) $\uparrow \in f(M_j)$, and M_j is reachable from below. If M_j is pre-terminal, there is an inactive edge in $\mathcal{C}$ for it. Suppose M_j is not pre-terminal. As observed above, all the daughters M_i are parsable, so if $\uparrow \in f(M_j)$, then M_j must be directly parsed, in which case there is an inactive edge in $\mathcal{C}$ for it. Since the tree conforms and $\uparrow \in f(M_j)$ there is a bottom-up rule r licensing M_0 and its daughters, with $j \in tr(r)$. Since the chart is fully bidirectional, there is an active edge for this rule spanning M_j. So by Lemma 3 above, there are inactive edges in $\mathcal{C}$ for M_0 and all nodes dominated by it. Hence, M_0 is parsable in $\mathcal{C}$.

This establishes the INDUCTIVE STEP and hence the result. □

Now we can show that a parsable tree will appear in any suitably constructed chart.

Lemma 5 *Let (G, tr) be a BSCFG. Let (T, f) be an annotated tree spanning σ, such $T \in trees(G)$ and (T, f) conforms to (G, tr). Let $\mathcal{C}$ be a fully bidirectional chart based on σ and using (G, tr). If every non-lexical node in T is parsable in $\mathcal{C}$ from (T, f), then $\mathcal{C}$ contains a representation of T.*

Proof: Let M_0 be the root node of T. Since (T, f) conforms to (G, tr), there are exactly two possibilities:

(a) Suppose $\uparrow \in f(M_i)$ for some daughter M_i of M_0. Since all the non-lexical nodes are parsable in $\mathcal{C}$, M_0 must be directly parsed in $\mathcal{C}$. Hence there are inactive edges in $\mathcal{C}$ for M_0 and all the non-terminal nodes it dominates.

(b) Suppose $\downarrow \in f(M_0)$. Since (T, f) conforms to (G, tr), there is a top down rule licensing M_0 and its daughters. Since this rule must have the distinguished symbol on its LHS, and $\mathcal{C}$ is fully bidirectional, there is an empty active edge in $\mathcal{C}$ for this rule at the start of M_0. Since all the nodes in (T, f) are parsable in $\mathcal{C}$, including M_0's daughters, it follows by Lemma 3 that there is an inactive edge in $\mathcal{C}$ for M_0 and each non-terminal node dominated by it. □

Lemma 6 *Let (G, tr) be a BSCFG. Let T be a tree in trees(G). If there is an annotated tree (T, f) which conforms to this grammar, and which is coherently marked, then any fully bidirectional chart $\mathcal{C}$ based on the terminal string spanned by T, contains a representation of T.*

Proof: Follows directly from Lemmas 4 and 5. □

Notice that Lemma 6 essentially states that any tree which can be coherently marked corresponds to a parse which can be found by bidirectional exploration. Hence we have completed the first step in our characterisation of completeness for bidirectional grammars — we have a well-defined class of annotated trees which a bidirectional parser can always find.

Surprisingly, we do not need a corresponding condition for nodes which are built top-down. It is possible to formulate the appropriate condition, but it turns out that any tree which meets the condition of being coherently marked will also meet the appropriate condition for top-down nodes. As will be seen in later proofs, no condition on top-down built nodes is explicitly needed for completeness. 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_0 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 left-daughter categories from some other prediction, or A_0 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.

4.3 Direct analysability

The next stage is to define a characteristic of BSCFGs which guarantees that they generate annotated trees which can be coherently marked (and hence are able to be parsed).

Definition: Given a strategy-marked grammar (G, tr), a category A_0 is *directly analysable* iff for every 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 analysable.

This definition essentially refers to non-terminal symbols from which there is a path to a terminal symbol going via trigger symbols on the right hand side of rules. It is analogous to "reachable from below" in annotated trees, as defined earlier. Notice that it implies that if A_0 is directly analysable, there is no purely top down rule of the form $A_0 \rightarrow A_1 \ldots A_k$.

There is also a counterpart definition for grammars, as follows.

Definition: A BSCFG (G, tr) is *directly analysable* 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 analysable.

Now we can prove that every annotation of a grammar that meets the above condition will allow any tree generated by the original grammar to be annotated in a way which is coherently marked; and hence, as shown earlier, the tree is capable of being bidirectionally parsed.

Lemma 7 *Suppose (G, tr) is an directly analysable BSCFG. Suppose $T \in trees(G)$. Then there exists a marking f for T such that (T, f) conforms to (G, tr) and is coherently marked.*

Proof: Define f as follows. For any non-lexical node M_0 with daughters $M_1, \ldots, M_k$ in T, where each M_i is labelled A_i, there must be a rule $A_0 \rightarrow A_1 \ldots A_k$ in G. If this rule is purely top-down, $\downarrow \in f(M_0)$. If this rule is purely bottom-up, then (since (G, tr) is directly analysable), there must be some A_i which is directly analysable. Then $\uparrow \in f(M_i)$ for each such i. If the rule is both top-down and bottom-up, then $\uparrow \in f(M_i)$ for each $i \in tr(A_0 \rightarrow A_1 \ldots A_k)$ for which A_i is directly analysable; if there are no such values if i, then $\downarrow \in f(M_0)$.

For all M, $f(M)$ is defined to be the least subset of $\{\uparrow, \downarrow\}$ necessary to meet these stipulations.

Now to show f to be the required marking, consider all nodes M for which $\uparrow \in f(M)$. This proceeds by induction on the height of nodes

BASE CASE: If a node is of height 2 (lowest non-lexical) then it is reachable from below since its daughters must be pre-terminal.

INDUCTIVE HYPOTHESIS: For any node M_0 of height less than n, if $\uparrow \in f(M_i)$ for some daughter M_i of M_0, then M_0 is reachable from below.

INDUCTIVE STEP: Let M_0 be a node of height n labelled A_0 with daughters $M_1, .., M_k$, such that $\uparrow \in f(M_i)$. The construction of f produced the condition "$\uparrow \in f(M_i)$" only for nodes M_i whose label was directly analysable. Hence the label A_i of M_i is directly analysable. Hence the rule $A_i \rightarrow B_1 \ldots B_p$ which licenses M_i and its daughters must be bottom-up, and also, for some j, $1 \leq j \leq p$, B_i is directly analysable. The construction above will therefore have ensured that $\uparrow \in f(N_j)$ (the node corresponding to B_j). By the INDUCTIVE HYPOTHESIS, M_i is reachable from below, and hence M_0 is also.

This establishes by induction that any node M_0 for which $\uparrow \in f(M_i)$ for some daughter M_i is reachable from below. By definition, the tree is coherently marked. □

The final step in proving completeness is now trivial.

Theorem 1 *Suppose (G, tr) is an directly analysable BSCFG. 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 $\mathcal{C}$ contains a representation of T.*

Proof: This follows directly from Lemmas 6 and 7. □

Corollary: If a BSCFG (G, tr) is directly analysable, then tr is complete.

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

Theorem 2 *It is decidable whether a given BSCFG is directly analysable.*

Proof: This is straightforward to verify from the definition of "directly analysable". Also, the check can be carried in $O(n^2)$ time where n is the number of non-terminal symbols in the grammar (see Appendix A for an algorithm). □

5 An undecidable class of complete annotations

5.1 Informal outline

Before proceeding to formalise 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 4.3 above). Such a rule leads to the possibility of there being a tree $T \in trees(G)$ which contains a non-terminal node which 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 which was not coherently marked. In the example in Section 2.2 above, the rules

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

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" non-terminal (H in this example) always occurs in a *left context* (i.e. a sequence of symbols to its left) with the following property. Every possible terminal expansion of the left context contains a substring which will, via bottom-up rules, introduce rules which 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 (non-lexical) 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 [10] or [12]). That is, there is a natural subclass of annotated grammars which 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 focussing 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: 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 characterising, for a given non-terminal A, exactly those sequences of symbols which *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 sequences immediately adjacent to it. Also, to get away from the complication of derivations (where the order of expansion of rules can alter the symbol sequences, and several derivations can be equivalent), we shall define this in terms of syntax trees.

The first definition describes the possible sequences of left sister node labels that a given non-terminal can have when immediately dominated by a particular non-terminal.

Definition: Suppose we have a context-free grammar (V_N, V_T, P, S), and two symbols $A_0, A' \in V_N$. Then the *left contexts of A' within A_0* is the set

$$\{A_1 \ldots A_{j-1} \mid A_0 \rightarrow A_1 \ldots A_{j-1} A' \ldots A_k \in P\}$$

To construct the sort of left context sequence we need, we have to consider not only left sister nodes, but the left sisters of the mother node, and so on up the tree. This accumulation of sisters has to be limited to a certain level, and this next definition develops the terminology for describing sequences of dominating nodes.

Definition: Suppose we have a context-free grammar (V_N, V_T, P, S), and a symbol B_1 in V_N. A *derivation path* for B_1 is a sequence $(B_1, \ldots, B_n)$ such that for every i, $2 \leq i \leq n$, there is a rule of the form $B_i \rightarrow \ldots B_{i-1} \ldots$ in the grammar.

Such a path is said to be:

1. *non-repeating* if $B_i \neq B_j$ whenever $i \neq j$.
2. *rooted* if $B_n = S$.
3. *localised* if there is a longer rooted derivation path $(B_1, \ldots, B_m)$ such that $B_n = B_k$ for some $n < k \leq m$.
4. *essential* if it is non-repeating and either rooted or localised.

Notice that if $(B_1, \ldots, B_n)$ is a derivation path for B_1 then $(B_i, \ldots, B_n)$ is a derivation path for B_i for all $2 \leq i \leq n$. Also, for any rooted derivation path, there is an essential derivation path which is an initial subsequence of it. We shall make use of these points in what follows. Informally, an essential derivation path defines exactly the level at which left sister nodes are relevant to the proofs, since any occurrence of the chosen non-terminal (B_1 in the above definition) is bound to be dominated by a path of nodes labelled with an essential derivation path.

The left context set of a sequence of non-terminal labels is the nested concatenation of all the possible left sister sequences that could occur at various stages of the sequence.

Definition: For any derivation path $(B_1, \ldots, B_n)$, the *left context set* of the path is:

1. $\{\epsilon\}$ if $n = 1$.
2. if $n > 1$, $\{\omega_1\omega_2 \mid \omega_1 \in$ the left context set of $(B_2, \ldots, B_n)$ and $\omega_2 \in$ left contexts of B_1 within $B_2\}$

This allows us to define just those sequences of symbols which a given symbol must have (one of) to its left:

Definition: For any non-terminal A, the *set of essential left contexts of A* is
$\{\omega \mid \exists$ an essential derivation path R for A and $\omega \in$ the left context set of R $\}$

The following lemma proves that essential left contexts have just the required property. (Notice that the derivation relationship "$\stackrel{*}{\Rightarrow}$" is the conventional one, as in Section 3.1, and is independent of any strategy marking.)

Lemma 8 *Let G be a CFG. Let T be a tree in trees(G). Let M be a non-terminal 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 $\psi_1\gamma\delta\psi_2$ where $\alpha \stackrel{*}{\Rightarrow} \gamma$ for some α in the essential left contexts of the label of M.

Proof: Since T is a tree, there is a path of non-terminal nodes $(N_1, \ldots, N_n)$ where N_i labelled B_i, for $1 \leq i \leq n$, $N_1 = M, N_n$ is the root of T, and N_i is the mother of N_{i-1} for $2 \leq i \leq n$. Since $T \in trees(G)$, there must be a sequence of rules $B_i \rightarrow \rho_i B_{i-1} \beta_i$ such that the node N_i and its daughters are licensed by $B_i \rightarrow \rho_i B_{i-1} \beta_i$ for $2 \leq i \leq n$. Hence $(B_1, \ldots, B_n)$ is a rooted derivation path for B_1. From this it is trivial to form an essential derivation path $(B_1, \ldots, B_k)$ for B_1, and there is also a derivation:

$$
\begin{aligned}
& B_k \\
& \Rightarrow \rho_k B_{k-1} \beta_k \\
& \Rightarrow \rho_k \rho_{k-1} B_{k-2} \beta_{k-1} \beta_k \\
& \ldots \\
& \Rightarrow \rho_k \rho_{k-1} \ldots B_2 \ldots \beta_{k-1} \beta_k \\
& \Rightarrow \rho_k \rho_{k-1} \ldots \rho_2 B_1 \beta_2 \ldots \beta_{k-1} \beta_k \\
& \stackrel{*}{\Rightarrow} \theta
\end{aligned}
$$

where θ is the substring of σ spanned by N_k.

This means that θ is of the form $\gamma\mu$ where $\rho_k \rho_{k-1} \ldots \rho_2 \stackrel{*}{\Rightarrow} \gamma$ and $B_1 \beta_2 \ldots \beta_{k-1} \beta_k \stackrel{*}{\Rightarrow} \mu$. Since B_1 is the label of M, and δ is the terminal string spanned by M, $B_1 \stackrel{*}{\Rightarrow} \delta$. Hence μ is of the form $\delta\phi$, and θ is therefore of the form $\gamma\delta\phi$ where $\rho_k \rho_{k-1} \ldots \rho_2 \stackrel{*}{\Rightarrow} \gamma$. Also $\rho_k \rho_{k-1} \ldots \rho_2$ is an essential left context of B_1, by virtue of the way it was constructed. Since θ is a substring of σ, this establishes the result.□

5.3 Deriving subtrees

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

Definition: In a BSCFG, suppose A is a non-terminal symbol, and σ is a sequence of terminal symbols. Then σ *can be coherently derived from* A if there is an annotated tree (T, f) such that:

1. T spans σ
2. the root of T is labelled A
3. (T, f) conforms to the grammar
4. (T, f) is coherently marked

The next definition requires that the derivation can occur without need for top-down initiation, since the fifth requirement in the definition, coupled with the tree being coherently marked, guarantees a sequence of bottom-up rules from root to string.

Definition: In a strategy marked grammar, suppose A is a non-terminal symbol, and σ is a sequence of terminal symbols. Then σ *can be up-derived from* A (written "$A \Uparrow^* \sigma$") if there is an annotated tree (T, f) such that:

1. T spans σ
2. the root of T is labelled A
3. (T, f) conforms to the grammar
4. (T, f) is coherently marked
5. for some M_i, a daughter of the root M of T, $\uparrow \in f(M_i)$

Now we ensure that all nodes of such trees will appear in a chart.

Lemma 9 *If $A \Uparrow^* \sigma$, and $\mathcal{C}$ is a fully bidirectional chart based on a string $\gamma_1 \sigma \gamma_2$, and using a BSCFG (G, tr), then there are inactive edges in $\mathcal{C}$ for all the nodes of a tree T such that T spans σ, T conforms to (G, tr), and the root of T is labelled A.*

Proof: Follows from Lemma 4 and part(a) of Lemma 5. □

5.4 Left-introducible rules

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

Definition: Let A, B be two non-terminal 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

$A \rightarrow A_0 \ldots$
$A_0 \rightarrow A_1 \ldots$
$\ldots$
$A_n \rightarrow B \ldots$

Lemma 10 *Let A, B be two non-terminal symbols from a LCSCFG (G, tr). If $A \rightsquigarrow B$, then in any bidirectionally mixed-strategy explored chart $\mathcal{C}$ using (G, tr) which contains an active edge $(i, j, A_1 \rightarrow \ \bullet\alpha_1 \bullet A\beta_1)$ there will also be an active edge in $\mathcal{C}$ of the form $(l, j, A_2 \rightarrow \ \bullet\alpha_2 \bullet B\beta_2)$, where either $l = i$ or $l = j$.*

Proof: Straightforward.□

Next a definition of the condition on rules which allow them to enter into the parsing process despite the difficulties outlined in Section 2.1 above.

Definition: In a LCSCFG grammar, a bottom-up rule $B_0 \rightarrow \underline{B_1}\alpha$ is said to be *left-introducible* iff for every θ which is an essential left context of B_0, for every terminal string γ such that $\theta \stackrel{*}{\Rightarrow} \gamma$, there is a bottom-up rule $A_0 \rightarrow \underline{A_1} \ldots A_k$ 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 $1 \leq j \leq i$
4. $A_{i+1} \rightsquigarrow B_1$

Scrutiny of this definition should reveal its relationship to the informal outlines in Sections 2.2 and 5.1 earlier.

Lemma 11 *Let (G, tr) be a LCSCFG grammar. Let (T, f) be an annotated tree, where T is generated by G, and M_0 a non-lexical node in T whose leftmost daughter is M_1, with M_0 labelled B_0, M_1 labelled 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 and its daughters in T is left-introducible. Then in any fully bidirectional chart based on the terminal string spanned by T, there is an active edge of the form*

$(l, m, A \rightarrow \ \bullet\alpha \bullet B_1\beta)$

(i.e an edge at the start of M_0, M_1, seeking B_1).

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 8, σ_1 is of the form $\psi\gamma$ where $\theta \stackrel{*}{\Rightarrow} \gamma$ for some $\theta \in$ the essential left contexts of A_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, $1 \leq j \leq i$
- $A_{i+1} \rightsquigarrow B_1$

By definition of $A_1 \Uparrow^* \rho_1$, there is a tree conforming to (G, tr) with root label A_1 spanning ρ_1. By Lemma 9, there are inactive edges in $\mathcal{C}$ for all the nodes of this tree, including its root. Since $A_0 \rightarrow A_1 \ldots A_k$ is bottom-up, and the chart is mixed-strategy explored, 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 labelled A_1), and j' is its end. By Lemma 4 and Lemma 3, there are inactive edges in $\mathcal{C}$ labelled $A_2, \ldots, A_i$, 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 $A_{i+1} \rightsquigarrow B_1$, by Lemma 10 there is an active edge

$$(l, m, A \rightarrow \ \bullet\alpha \bullet B_1\beta)$$

(i.e. seeking B_1). □

Now we have to relate these notions to patterns of annotations of trees, and thence to strategy-marked grammars.

5.5 Indirect reachability and indirect coherence

First we need a generalisation of "reachable from below" that allows the possibility of an left-introducible rule causing a form of reachability.

Definition: Let (G, tr) be a LCSCFG, $T \in trees(G)$, and (T, f) an annotated tree which conforms to this grammar. Then a node M_0 in T, with leftmost daughter M_1, is *indirectly reachable from below* iff one of the following holds:

(a) M_0 is reachable from below;

(b) $f(M_1) = \{\uparrow\}$ and M_1 is indirectly reachable from below;

(c) $f(M_1) = \{\uparrow, \downarrow\}$ and the rule licensing M_0 and its daughters is left-introducible.

Now we have an analogue of "coherently marked ", which describes just those trees which are annotated in an orderly manner for mixed-strategy parsing.

Definition: Let (G, tr) be a LCSCFG, $T \in trees(G)$, and (T, f) an annotated tree which conforms to this grammar. Then (T, f) is *indirectly coherent* iff every node M_0 such that $\uparrow \in f(M_1)$ (where M_1 is the leftmost daughter of M_0) is indirectly reachable from below.

Lemma 12 *If a tree is indirectly coherent, so are all its subtrees.*

Proof: Follows directly from the definition. □

The following lemma is analogous to Lemma 4 above.

Lemma 13 *Let (G, tr) be a LCSCFG grammar, and let $T \in trees(G)$. Suppose $\mathcal{C}$ is a fully bidirectional chart based on the terminal string which T spans, using (G, tr). If there is an annotated tree (T, f) which conforms to (G, tr) and is indirectly coherent, then every non-lexical node in T is parsable in $\mathcal{C}$.*

Proof: Comparable to that of Lemma 4. □.

Corollary : With (G, tr), T as above, if there is an annotated tree (T, f) which conforms to (G, tr) and is indirectly coherent, then every non-lexical node in T is parsable in *every* fully bidirectional chart based on the terminal string which T spans, using (G, tr).

The following lemma is analogous to Lemma 6 above.

Lemma 14 *Let (G, tr) be a LCSCFG. Let $T \in trees(G)$. If there is an annotated tree (T, f) which conforms to this grammar, and which is indirectly coherent, then any fully bidirectional chart $\mathcal{C}$ based on the terminal string spanned by T, contains a representation of T.*

Proof: Follows from Lemma 5 and Corollary to Lemma 13 above. □

5.6 Indirect analysability

In Section 4 we defined "direct-analysability" as a condition on grammars which would lead to coherently marked trees. Now we need a comparable property which will guarantee indirectly coherent trees.

Definition: Let (G, tr) be a LCSCFG. A non-terminal symbol A_0 in G is said to be *indirectly analysable* iff both of the following hold:

1. there is no purely top-down rule of the form $A_0 \rightarrow \alpha$;
2. for every rule of the form $A_0 \rightarrow A_1\alpha$, at least one of the following holds:
   (a) $A_0 \rightarrow A_1\alpha$ is left-introducible.
   (b) A_1 is indirectly analysable;

Definition: A LCSCFG (G, tr) is *indirectly analysable* iff every purely bottom-up rule $A_0 \rightarrow A_1 \ldots A_k$ in it meets the following condition:
either $A_0 \rightarrow A_1 \ldots A_k$ is left-introducible, *or* A_1 is indirectly analysable.

The next lemma ensures that a grammar with the property of indirect analysability can generate trees with orderly annotations, and is analogous to Lemma 7 above (with a comparable proof).

Lemma 15 *Suppose (G, tr) is an indirectly analysable LCSCFG. Suppose $T \in trees(G)$. Then there exists a marking f for T such that (T, f) conforms to (G, tr) and is indirectly coherent.*

Proof: Define f so that, for any node M, $f(M)$ is the least subset of $\{\downarrow, \uparrow\}$ which will ensure the following conditions:

> For any non-lexical node M_0 with daughters $M_1, \ldots, M_k$ in T, licensed by a rule $A_0 \rightarrow A_1 \ldots A_k$ in G: if this rule is purely top-down, $\downarrow \in f(M_0)$; if this rule is purely bottom-up, or if A_0 is indirectly analysable, then $\uparrow \in f(M_1)$; otherwise, $\downarrow \in f(M_0)$.

To show that (T, f) is indirectly coherent, proceed by induction on the height of nodes, considering nodes M_0 where $\uparrow \in f(M_1)$, where $f(M_1)$ is the leftmost daughter of M_0. □

Theorem 3 *Suppose (G, tr) is an indirectly analysable 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 $\mathcal{C}$ contains a representation of T.*

Proof: Follows from Lemma 14 and Lemma 15. □

Corollary : If a LCSCFG (G, tr) is indirectly analysable, then tr is complete.

5.7 Undecidability

So far, we have established that the condition of *indirect analysability* suffices to ensure completeness. Unfortunately, indirect analysability is not a decidable property of annotated grammars, as we now show.

Theorem 4 *In a LCSCFG, it is undecidable whether a given rule is left-introducible.*

Proof: Suppose that there were a decision procedure for left-introducibility. This could then be used to construct a decision procedure which determines for any two CFGs G_1, G_2 whether every member of $L(G_1)$ ends in a substring which is a member of $L(G_2)$. This is an undecidable problem; hence, the left-introducibility question is also undecidable. The construction proceeds as follows.

Suppose we have the two arbitrary CFGs G_1 and G_2 over the same alphabet V_T, and assume that their non-terminal 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 bottom-up rules (i.e. $tr(r) = \{1\}$) are all the rules of G_2, together with

$S' \rightarrow \underline{S_1} B_1$
$B_1 \rightarrow \underline{B_2} B_3$
$B_6 \rightarrow \underline{S_2} B_2$

where each S_i is the distinguished symbol of G_i. The top-down rules (i.e. $tr(r) = \{0\}$) are all the rules of G_1 together with

$\underline{B_2} \rightarrow B_4 B_5$

Also we include lexical rules:

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

for some terminal symbols a, b, c.

In this strategy-marked grammar, consider whether the rule $B_1 \rightarrow \underline{B_2} B_3$ is left-introducible. The essential left contexts of B_1 is the set $\{ S_1 \}$. The only symbol X for which $X \rightsquigarrow B_2$ is B_2 itself. Hence the only rule which meets the schema for left-introducibility is $B_6 \rightarrow \underline{S_2} B_2$. So $B_1 \rightarrow \underline{B_2} B_3$ 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). □

Corollary : For an arbitrary LCSCFG grammar (G, tr), it is undecidable whether G is indirectly analysable or not.

6 Some further complications

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

It is not worthwhile formalising and analysing 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 analysable, and might seem to cause "blocking" as discussed in Section 2.1 earlier, is redeemed by the interaction with other rules in the grammar. This is similar to the phenomenon analysed in Section 5 above, but whereas the analysis above dealt with a configuration of rules which can be parsed bottom up to the left of the problematic rule, there is an analogous condition on subtrees to the left which can be parsed top down.

The following grammar illustrates this phenomenon.

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

Here, the grammar is not indirectly analysable, as the purely bottom up rule $K \rightarrow \underline{Q}\ D$ has a trigger category Q which is not indirectly analysable. (The rule $S \rightarrow \underline{H}\ K$ is also problematic.) However, the only situation in which $K \rightarrow \underline{Q}\ D$ would be needed would be to parse a string $prftvd$. Since $S \rightsquigarrow H$, there will be an empty active edge introduced for $\underline{H} \rightarrow E\ F$ 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 $\underline{Z} \rightarrow H\ Q$, leading to the introduction of an empty active edge for $\underline{Q} \rightarrow T\ V$ 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 distinguished symbol of the grammar.

It is likely that some generalisation 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 bi-directional schemes

As mentioned in Section 1 above, the ideas here were developed from a semi-formal proposal by Steel and de Roeck [9]. The formalisation given here is a slight generalisation, 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 formalise 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 [4, 5, 6] have developed various detailed and rigorous systems of chart-based parsing, include one [4] which 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 which 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 adapted to achieve greater efficiency.

7.2 Extended Generalised Left Corner Parsing

Stabler [8] 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 generalisation 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 generalised 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 recognised before that rule is to be introduced into the parsing process; making this dependent on the parser's stack (which can hold both recognised 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. Superficially, it would seem to be straightforward to define the mixed-strategy parsing method for LCSCFGs within this system, thereby placing it within the scope of Stabler's completeness results. That would lead to the apparent contradiction that Stabler has proved all that such trigger functions yield complete parsers, but Section 2 above shows that some annotations cause incompleteness. The apparently obvious way to achieve mixed-strategy parsing using XGLC parsing would be to have a trigger function which maps each bottom-up rule, regardless of the stack contents, to a one-symbol prefix of its RHS (since this means "introduce this rule whenever that symbol has been recognised"), and maps each top-down rule to a zero prefix, providing that the top of the stack is a prediction of the symbol which appears on the LHS of the rule. However, 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 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. This shows the escape from the contradiction just mentioned. 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.

A further, less crucial, difference between Stabler's formalisation and that defined above, is that the XGLC parser is presented as an exhaustive non-deterministic parser with no attention to efficiency or redundancy issues (quite reasonably, at the level of theoretical abstraction involved). Active charts, although definable in an abstract way, were originally developed with the aim of achieving a certain degree of efficiency and non-redundancy, so the formal model used here, although not directly procedural, is slightly closer to implementational concerns than the formal inferential framework used by Stabler. This difference has no impact on the issues of completeness or decidability, however.

7.3 Relevance to natural language

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 whole 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. Such an eventuality might hold in the study of natural languages, but those interested in constructing compilers for programming languages would regard it as unimportant.

Some preliminary results [11] 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 formalised here. These gains are not great, and it is unclear whether similar improvements could be achieved in large realistic 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 which are directly analysable.

Context free grammar has been used as the basis here, both to simplify the formalisation, 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 which parse natural language. Usually, some much more complex grammatical formalism is used. A particularly common mechanism in recent years has been some form of *unification grammar* [7]. 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 non-terminal 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 analysability" might turn out to be less tractable.

8 Conclusion

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 analysability – which guarantees completeness, it is over-restrictive, in the sense that there are complete annotations which are not directly analysable. There is also a wider class of complete annotations – indirectly analysable – which 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.

Acknowledgements

I would like to thank Anne de Roeck for advice and encouragement in developing these ideas, Alistair Willis and Suresh Manandhar for useful suggestions and Nicolas Nicolov for comments on an earlier draft.

A Computing direct analysability

The algorithm is a simple variant of the use of an AND-OR graph in problem solving, as in [3]. The graph will contain a node for each non-terminal symbol A in the grammar, and any number of OR nodes. Each node has a *label*, which is either a non-terminal 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 labelled OR;
- create an arc *from* N_A *to* N;
- create an arc *from* N *to* N_{A_i} for every $i \in tr(A \rightarrow A_1 \ldots A_k)$.

At this point, each non-terminally-labelled 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 which are not directly analysable; nodes marked* SOLVED *correspond to those which are directly analysable. 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 labelled 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 can be regarded as directly analysable.

3. For every node N_A which 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 analysable.

The above statement is not intended to be maximally efficient.

References

[1] A. Aho and J. Ullman. *The Theory of Parsing, Translation, and Compiling. Volume 1: Parsing.* Prentice-Hall, Englewood Cliffs, NJ, 1972.

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

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

[4] G. Satta and O. Stock. Formal properties and implementation of bidirectional charts. In *Proceedings of Eleventh International Joint Conference on Artificial Intelligence (IJCAI-89)*, pages 1480–1485, 1989.

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

[6] G. Satta and O. Stock. Bidirectional context-free grammar parsing for natural language processing. *Artificial Intelligence*, 69:123–164, 1994.

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

[8] Edward P. Stabler. Parsing for incremental interpretation. Draft paper, UCLA, Los Angeles, 1994.

[9] S. Steel and A. de Roeck. Bidirectional chart parsing. In J. Hallam and C. Mellish, editors, *Advances in Artificial Intelligence*, pages 223–235. John Wiley, 1987.

[10] H. Thompson and G. Ritchie. Implementing natural language parsers. In T. O'Shea and M. Eisenstadt, editors, *Artificial Intelligence: Tools, Techniques and Applications*, chapter 9, pages 245–300. Harper and Row, New York, 1984.

[11] A. Willis. Exploring chart parsing mechanisms. Master's thesis, Department of Artificial Intelligence, University of Edinburgh, Edinburgh, Scotland, 1996.

[12] T. Winograd. *Language as a Cognitive Process. Volume I: Syntax.* Addison-Wesley, Reading, Mass., 1983.