Nonminimal Derivations in Unification-based Parsing

Noriko Tomuro *
DePaul University

Steven L. Lytinen †
DePaul University

Shieber's abstract parsing algorithm (Shieber, 1992) for unification grammars is an extension of Earley's algorithm (Earley, 1970) for context-free grammars to feature structures. In this paper, we show that, under certain conditions, Shieber's algorithm produces what we call a **nonminimal derivation***: a parse tree which contains additional features that are not in the licensing productions. While Shieber's definition of parse tree allows for such nonminimal derivations, we claim that they should be viewed as invalid. We describe the sources of the nonminimal derivation problem, and propose a precise definition of minimal parse tree, as well as a modification to Shieber's algorithm which ensures minimality, although at some computational cost.*

1 Introduction

Unification grammar is a term often used to describe a family of feature-based grammar formalisms, including GPSG (Gazdar et al., 1985), PATR-II (Shieber, 1986), DCG (Pereira and Warren, 1980), and HPSG (Pollard and Sag, 1994). In an effort to formalize the common elements of unification-style grammars, Shieber (1992) developed a logic for describing them, and used this logic to define an *abstract parsing algorithm.* The algorithm uses the same set of operations as Earley's (1970) algorithm for context-free grammars, but modified for unification grammars.

In this paper, we show that, under certain conditions, Shieber's algorithm produces unintended, spurious parses in addition to the intended ones. We call these spurious parses *nonminimal derivations* (or *nonminimal parse trees*)[1], because they contain extra features which are not in the productions that license the parse. We claim such nonminimal derivations are *invalid.* The basis of our claim is that the unification operation as set union preserves minimality; thus any correct unification-based parsing algorithm should produce parses that contain all and only features from the licensing productions (i.e., *minimal derivations* or *minimal parse trees*). Nonminimal derivations are also undesirable in practice because, given a parse tree, we can not tell whether a particular feature should be in the model or not unless we re-construct the whole tree.

However, despite the nonminimal derivations, Shieber (1992) proved the correctness of his algorithm. As it turned out, his definition of parse tree which his proof relied

* School of Computer Science, Telecommunications and Information Systems, Chicago, IL 60604. E-mail: tomuro@cs.depaul.edu
† School of Computer Science, Telecommunications and Information Systems, Chicago, IL 60604. E-mail: lytinen@cs.depaul.edu

1 In this paper, we use "nonminimal derivations" synonymously with "nonminimal parses". Normally the notions of derivation and parse tree are different. However, in this paper we focus on parse trees as the final result of derivation, thus we mean that a derivation is nonminimal when its result is a nonminimal parse, in contrast to a minimal derivation which produces a minimal parse. Unfortunately, formal definitions of minimal and nonminimal derivations are outside the scope of this short paper; interested readers are encouraged to read (Tomuro, 1999).

$$p_0 = \langle 2, \Phi_0 : \left\{ \begin{array}{l} \langle \text{cat} \rangle \doteq \text{S} \\ \langle 1\ \text{cat} \rangle \doteq \text{NP} \\ \langle 2\ \text{cat} \rangle \doteq \text{VP} \\ \langle \text{head} \rangle \doteq \langle 2\ \text{head} \rangle \\ \langle \text{head subj} \rangle \doteq \langle 1\ \text{head} \rangle \\ \langle \text{head agr} \rangle \doteq \langle 1\ \text{head agr} \rangle \end{array} \right\} \rangle \qquad p_1 = \langle \text{“John”}, \Phi_1 : \left\{ \begin{array}{l} \langle \text{cat} \rangle \doteq \text{NP} \\ \langle \text{head agr pers} \rangle \doteq \text{3rd} \\ \langle \text{head agr num} \rangle \doteq \text{sing} \end{array} \right\} \rangle$$

$$p_2 = \langle 1, \Phi_2 : \left\{ \begin{array}{l} \langle \text{cat} \rangle \doteq \text{VP} \\ \langle 1\ \text{cat} \rangle \doteq \text{V} \\ \langle \text{head} \rangle \doteq \langle 1\ \text{head} \rangle \\ \langle \text{head type} \rangle \doteq \text{intrans} \end{array} \right\} \rangle \qquad p_3 = \langle \text{“sleeps”}, \Phi_3 : \left\{ \begin{array}{l} \langle \text{cat} \rangle \doteq \text{V} \\ \langle \text{head agr pers} \rangle \doteq \text{3rd} \\ \langle \text{head agr num} \rangle \doteq \text{sing} \\ \langle \text{head tense} \rangle \doteq \text{pres} \end{array} \right\} \rangle$$

Figure 1
Examples of productions

on was not constraining enough to disallow nonminimal derivations. To solve this two-fold problem, we propose an alternate definition of minimal parse tree for unification grammars, and present a modification to Shieber's algorithm which ensures minimality.

It is important to note that the same spurious parses also occur in context-free parsing, specifically in Earley's algorithm. However, since the only information a constituent carries in context-free grammar is the grammar symbol, the spurious derivations only produce exactly the same results as the normal ones. However, these spurious parses are a problem when the algorithm is extended to unification grammar.

2 Unification Grammar and Parse Trees

Shieber (1992) defines a unification grammar as a 3-tuple $\langle \Sigma, P, p_0 \rangle$, where Σ is the *vocabulary* of the grammar, P is the set of *productions*, and $p_0 \in P$ is the *start production*. Σ contains L, a set of *labels* (feature names); C, a set of *constants* (feature values); and W, a set of terminals. There are two kinds of productions in P: *phrasal* and *lexical*. A phrasal production is a 2-tuple $\langle a, \Phi \rangle$, where a is the *arity* of the rule (the number of right-hand side (RHS) constituents), and Φ is a logical formula. Typically, Φ is a conjunction of equations of the form $p_1 \doteq p_2$ or $p_1 \doteq c$, where $p_1, p_2 \in L^*$ are *paths*, and $c \in C$. In an equation, any path which begins with an integer i $(1 \leq i \leq a)$ represents the i^{th} RHS constituent of the rule.[2] A lexical production is a 2-tuple $\langle w, \Phi \rangle$, where $w \in W$ and Φ is the same as above, except that there are no RHS constituents. Figure 1 shows some example phrasal and lexical productions (p_0 corresponds to the context-free rule S $\rightarrow$ NP VP and is the start production). Then a *model* M relates to a formula Φ by a satisfaction relation $\models$ as usual ($M \models \Phi$), and when Φ is the formula in a production $p = \langle a, \Phi \rangle$, p is said to *license* M.

Based on the logic above, Shieber defines a parse tree and the language of a grammar expressed in his formalism. To define a valid parse tree, he first defines the set of possible parse trees $\Pi = \bigcup_{i \geq 0} \Pi_i$ for a given grammar G, where each Π_i is defined as follows:

Definition
A *parse tree* τ is a model that is a member of the infinite union of sets of bounded-depth parse trees $\Pi = \bigcup_{i \geq 0} \Pi_i$, where each Π_i is defined as:

1.Π_0 is the set of models τ for which there is a lexical production $p = \langle w, \Phi \rangle \in G$

2 Shieber (1992) also uses a path that begins with 0 for the left-hand side (LHS) constituent of a rule. In this paper, we omit the 0 arcs and place the features of the LHS constituent directly at the root. This change does not affect the formalism for the purpose of this paper.

such that $\tau \models \Phi$.

2. $\Pi_i (i > 0)$ is the set of models τ for which there is a phrasal production $p = \langle a, \Phi \rangle \in G$ such that $\tau \models \Phi$ and, for all $1 \leq i \leq a$, $\tau/\langle i \rangle$ is defined and $\tau/\langle i \rangle \in \bigcup_{j<i} \Pi_j$.

In the second condition, the *extraction* operator, denoted by /, retrieves the feature structure found at the end of a particular path; so for instance $\tau/\langle 1 \rangle$ retrieves the first subconstituent on the RHS of the production which licenses τ. In the definition above, Π_0 contains all models which satisfy any lexical production in the grammar, while Π_i contains all models which satisfy a phrasal production, and whose subconstituents are all in $\bigcup_{j \leq i} \Pi_j$.

To specify what constitutes a valid parse for a particular sentence, the next step is to define the *yield* of a parse tree. It is defined recursively as follows: if τ is licensed by some lexical production $p = \langle w, \Phi \rangle$, then the yield of τ is w; or if τ is licensed by some phrasal production $\langle a, \Phi \rangle$ and $\alpha_1, \ldots, \alpha_a$ are the yields of $\tau/\langle 1 \rangle, \ldots, \tau/\langle a \rangle$ respectively, then the yield of τ is $\alpha_1 \ldots \alpha_a$.

Finally, Shieber defines a valid parse tree $\tau \in \Pi$ for sentence $w_1 \ldots w_n$ as follows:

1. The yield of τ is $w_1 \ldots w_n$
2. τ is licensed by the start production p_0

Notice this definition allows extra features in a parse tree, because a parse tree τ is defined by the satisfaction relation ($\tau \models \Phi$), which allows the existence of features in the model which are not in the licensing production's formula. Given this definition, for any valid parse tree τ, we can construct another parse tree τ' by simply adding an arbitrary (non-numeric) feature to any node in τ. Such a parse tree τ' is *nonminimal* because extra features are nonminimal with respect to the *minimal* features in the licensing productions. We will return to the issue of minimal and nonminimal parse trees in section 4.

3 The Abstract Parsing Algorithm

Based on the logic described above, Shieber defines an abstract parsing algorithm as a set of 4 logical deduction rules. Each rule derives a new *item*, from previous items and/or productions in the grammar. An item is a 5-tuple $\langle i, j, p, M, d \rangle$, where i and j are indices into the sentence and specify which words in the sentence have been used to construct the item; p is the production used to construct the item; M is a model, and d is the position of the "dot"; i.e., how many subconstituents in p have been completed so far.

The logical rules of the abstract algorithm are shown in Figure 2. The *Initial item* rule produces the first item, and is constructed from the start production p_0. It spans none of the input (i and j are both 0), and its model is the *minimal model* (mm) of p_0.

The *Prediction* rule is essentially the top-down rewriting of the expectation (a subconstituent just after the dot) in a prior item. In this rule, the extraction of $M/\langle d{+}1 \rangle$ retrieves the $d{+}1^{st}$ submodel in M (i.e., expectation). The function ρ, which is left underspecified as a parameter in the abstract algorithm, filters out some features predefined in the various instantiations of the algorithm. Here, it is applied to the expectation, by which it effectively controls the top-down predictive power of the algorithm and provides flexibility to the instantiated algorithms. Then the expectation is unified with a production (Φ') which can consistently rewrite it. By this operation, some features in the expectation may be propagated down in the production.

The remaining two rules advance the dot in a prior item, by unifying the subconstituent to the right of the dot with either a lexical item from the input string (the

$$\text{INITIAL ITEM:} \quad \frac{}{\langle 0, 0, p_0, mm(\Phi_0), 0\rangle}$$

$$\text{PREDICTION:} \quad \frac{\langle i, j, p = \langle a, \Phi\rangle, M, d\rangle}{\langle j, j, p', \rho(M/\langle d{+}1\rangle) \sqcup mm(\Phi'), 0\rangle} \quad \text{where } d < a \text{ and } p' = \langle a', \Phi'\rangle \in P$$

$$\text{SCANNING:} \quad \frac{\langle i, j, p = \langle a, \Phi\rangle, M, d\rangle}{\langle i, j{+}1, p, M \sqcup (mm(\Phi') \setminus \langle d{+}1\rangle), d{+}1\rangle} \quad \text{where } d < a \text{ and } \langle w_{j+1}, \Phi'\rangle \in P$$

$$\text{COMPLETION:} \quad \frac{\langle i, j, p = \langle a, \Phi\rangle, M, d\rangle \qquad \langle j, k, p' = \langle a', \Phi'\rangle, M', a'\rangle}{\langle i, k, p, M \sqcup (M' \setminus \langle d{+}1\rangle), d{+}1\rangle} \quad \text{where } d < a$$

Figure 2
Shieber's parsing operations

$$\begin{array}{lcl} I_0 & = & \langle 0, 0, p_0, mm(\Phi_0), 0\rangle \\ I_1 & = & \langle 0, 1, p_0, M_1, 1\rangle \\ I_2 & = & \langle 1, 1, p_2, M_2, 0\rangle \\ I_3 & = & \langle 1, 2, p_2, M_3, 1\rangle \\ I_4 & = & \langle 0, 2, p_0, M_4, 2\rangle \end{array}$$

***M*$_4$:**

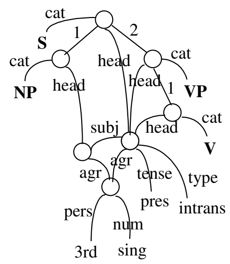

Figure 3
Items produced in the parse of "John sleeps", and the final parse

Scanning rule) or some other completed higher-level item (the *Completion* rule). Both rules perform the correct unification by utilizing the *embedding* operator (signified by \), which places a model M under a path p ($M \backslash p$).

We illustrate these operators with a simple step-by-step example parse. Consider the grammar which consists of the rules presented in Figure 1. Using this grammar, Figure 3 shows the parse of the sentence "John sleeps." First, the Initial Item operator is applied, producing item I_0, whose model is mm(Φ_0). Next, the Scanning operator scans the word "John", producing I_1. The Prediction operator then produces I_2. Next, the word "sleeps" is scanned (since the first subconstituent of the model in I_2 is a V), producing I_3. Finally, since the item in I_3 is complete ($d = 1$, the arity of production p_2), Completion is applied to items I_1 and I_3, producing I_4. Model M_4 is the final parse of the sentence.

4 Nonminimal Derivations

Earlier in section 2, we noted that Shieber's definition of parse trees allows them to be nonminimal. We consider these to be *invalid* based on a principle that, since the unification operation as set union preserves minimality (as proved in (Shieber, 1992)), repeated applications of unification using licensing productions should result in parses that contain features only from those productions and nothing more. In this section, we formally define *minimal* and *nonminimal parse trees*, and show an example in which nonminimal parse trees are produced by Shieber's algorithm.

Our definition of minimal parse tree is to a large extent similar to Shieber's definition, but to ensure minimality, our definition uses the equality relation instead of $\models$, and inductively specifies a minimal parse tree bottom-up.

Definition
Given a grammar G, a *minimal parse tree* τ admitted by G is a model that is a member

$$p_4 = \langle 2, \Phi_4 : \left\{ \begin{array}{l} \langle \text{cat} \rangle \doteq \text{VP} \\ \langle 1 \text{ cat} \rangle \doteq \text{VP} \\ \langle 2 \text{ cat} \rangle \doteq \text{ADV} \\ \langle \text{head} \rangle \doteq \langle 1 \text{ head} \rangle \\ \langle \text{head modified} \rangle \doteq \text{true} \end{array} \right\} \rangle$$

Figure 4
A phrasal production which results in a nonminimal derivation

$$\begin{array}{lcl} I'_2 & = & \langle 1, 1, p_4, M'_2, 0 \rangle \\ I''_2 & = & \langle 1, 1, p_2, M''_2, 0 \rangle \\ I'_3 & = & \langle 1, 2, p_2, M'_3, 1 \rangle \\ I'_4 & = & \langle 0, 2, p_0, M'_4, 2 \rangle \end{array}$$

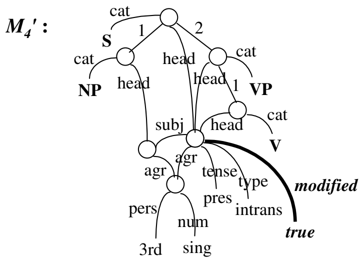

Figure 5
Nonminimal derivation of "John sleeps"

of the infinite union of sets of bounded-depth parse trees $\Pi' = \bigcup_{i \geq 0} \Pi'_i$, where each Π'_i is defined as:

1.For each lexical production $p = \langle w, \Phi \rangle \in G$, $mm(\Phi) \in \Pi'_0$.
2.For each phrasal production $p = \langle a, \Phi \rangle \in G$, let $\tau_1, \ldots, \tau_a \in \bigcup_{j<i} \Pi'_j$. If $\tau = mm(\Phi) \sqcup \tau_1 \backslash \langle 1 \rangle \sqcup \ldots \sqcup \tau_1 \backslash \langle a \rangle$, then $\tau \in \Pi'_i$.

It is obvious that Π' is a subset of Π in Shieber's definition. Then, a *nonminimal parse tree* is defined as a model which is a member of the difference of the two sets $(\Pi - \Pi')$.[3]

Here is a simple example in which a nonminimal parse is produced in Shieber's algorithm. Say that we add the production in Figure 4 to the grammar in the previous section. The intent of this production is to mark the verb with the *modified* feature if an adverb follows. Using this grammar, Shieber's algorithm will produce a nonminimal parse for the sentence "John sleeps," in addition to the minimal parse shown in the previous section.[4] The nonminimal parse, shown in Figure 5, arises as follows: after scanning "John", Prediction can produce items I'_2 and I''_2, first using production p_4 (thus inserting $\langle \text{head modified} \rangle \doteq \text{true}$ into the model), and then p_2. Scanning the word "sleeps" then produces I'_3 from I''_2. Completion then can be applied directly to I_1 and I'_3 by skipping a completion using I'_2 and I'_3, thereby producing item I'_4. The *modified* feature remains in I'_4, even though an adverb was never encountered in the sentence. The final parse M'_4, shown in Figure 5, is clearly nonminimal according to our definition because of this feature.

As a note, the example grammar can be changed to prevent the nonminimal parse, by moving the *modified* feature off of the *head* path in Φ_4 (i.e., $\langle \text{modified} \rangle \doteq \text{true}$ instead of $\langle \text{head modified} \rangle \doteq \text{true}$).[5] However, the point of the example is not to argue whether or

3 As a note, using *subsumption* (which we will discuss in section 5) here does not work, for instance by saying "a model τ'' is a nonminimal parse tree if $\tau'' \in \Pi$ and there exists $\tau' \in \Pi$ such that $\tau' \leq \tau''$", because some τ'''s are minimal. See the example in section 5.
4 Here, we are assuming that the filtering function ρ is the identity function.
5 Note that adding $\langle \text{head modified} \rangle \doteq \text{false}$ to Φ_2 (VP $\rightarrow$ V) or Φ_3 ("sleeps") is not feasible, because they can not specify the *modified* feature at their level.

not well-designed grammars will produce erroneous parses: a formally defined parser (see a discussion below) should in principle produce correct parses *regardless* of the grammar used; otherwise, the grammar formalism (i.e., Shieber's logic for unification grammars) must be revised and properly constrained to allow only the kinds of productions with which the parser produces correct results.

In general, nonminimal derivations may arise whenever two or more predictions which are not mutually exclusive can be produced at the same point in the sentence; i.e., two prediction items $\langle i, i, p, M, 0\rangle$ and $\langle i, i, p', M', 0\rangle$ are produced such that $M \neq M'$ and M and M' are unifiable. In the example, items $I_2 = \langle 1, 1, p_2, M_2, 0\rangle$ and $I_2' = \langle 1, 1, p_4, M_2', 0\rangle$ (as well as I_2 and $I_2'' = \langle 1, 1, p_2, M_2'', 0\rangle$) are two such items. Since the two predictions did not have any conflicting features from the beginning, a situation may occur where a completion spawned from one prediction can fill the other prediction without causing conflict. When this happens, features that were in the other prediction but not the original one become nonminimal in the resulting model.

As for the sources which may cause nonminimal situations, we speculate the following. First, nonminimal derivations occur when a prediction is filled by a complete item which did not spawn from the prediction. This mismatch will not happen if parsing is done in one direction only (e.g. pure top-down or bottom-up parsing). Thus, the mixed-direction parsing strategy is a contributing factor.

Second, wrong complete items are retrieved because Shieber's item-based algorithm makes all partial results available during parsing, as if they are kept in a global structure (such as a *chart* in chart parsing). But if the accessibility of items were somehow restricted, prediction-completion mismatch would not happen. In this respect, other chart-based algorithms for unification grammars which adopt mixed-direction parsing strategy, including head-corner parsing (van Noord, 1997) and left-corner parsing (Alshawi, 1992), are subject to the same problem.

Third, extra features can only appear when the grammar contains rules which interact in a certain way (such as rules p_2 and p_4 above). If the grammar contained no such rules, or if ρ (the filtering function applied in Prediction) filtered out those features, even the prediction-completion mismatch would not produce nonminimal derivations.

As we stated in the beginning of this section, we consider nonminimal parses to be invalid on the basis of minimality. Then it immediately follows that any parsing algorithm which produces nonminimal parses is considered to be unsound; in particular, Shieber's algorithm is unsound. However, since nonminimal parse trees have the same yield as their minimal counterparts, his algorithm does indeed *recognize* exactly the language of a given grammar. So, Shieber's algorithm is sound as a recognizer,[6] but not as a transducer or *parser* (as is referred to in (van Noord, 1997)) where the correctness of output models (i.e., parse trees) is critical. In other words, Shieber's algorithm is correct *up to* licensing, but incorrect on the basis of a stronger criteria of minimality. Then to guarantee correctness based on minimality, we need another algorithm; such an algorithm is exactly the solution to the nonminimal derivation problem.

5 Practical Techniques

Before presenting our solution to the nonminimal derivation problem, we discuss several possible practical techniques to get around the problem in implemented systems. These are known techniques which have been applied to solve other problems in unification-

6 In fact, Shieber hints his interpretation of parsing as "The process of parsing (more properly, recognition)..." (Shieber, 1992, page 78).

INITIAL ITEM: $$\frac{}{\langle id, nil, \langle 0, 0, p_0, mm(\Phi_0), 0\rangle\rangle}$$ where id is a new symbol

PREDICTION: $$\frac{\langle id, pid, \langle i, j, p = \langle a, \Phi\rangle, M, d\rangle\rangle}{\langle id', id, \langle j, j, p', \rho(M/\langle d{+}1\rangle) \sqcup mm(\Phi'), 0\rangle\rangle}$$ where id' is a new symbol, and $d < a$ and $p' = \langle a', \Phi'\rangle \in P$

SCANNING: $$\frac{\langle id, pid, \langle i, j, p = \langle a, \Phi\rangle, M, d\rangle\rangle}{\langle id, pid, \langle i, j{+}1, p, M \sqcup mm(\Phi') \setminus \langle d{+}1\rangle, d{+}1\rangle\rangle}$$ where $d < a$ and $\langle w_{j+1}, \Phi'\rangle \in P$

COMPLETION: $$\frac{\langle id, pid, \langle i, j, p, M, d\rangle\rangle \quad \langle id'', id, \langle j, k, p', M', a'\rangle\rangle}{\langle id, pid, \langle i, k, p, M \sqcup (M' \setminus \langle d{+}1\rangle), d{+}1\rangle\rangle}$$ where $d < a$

Figure 6
Modified Shieber's parsing operations

based systems. However, most of them only offer partial solutions to the nonminimal derivation problem. First, whenever Shieber's algorithm produces a nonminimal derivation, it also produces a corresponding minimal derivation (Tomuro, 1999). Thus, one possible solution is to use *subsumption* to discard items that are more specific than any other items that are produced. Subsumption has often been used in unification-based systems to *pack* items or models (e.g. (Alshawi, 1992)). However, simple subsumption may filter out valid parses for some grammars, thus sacrificing completeness.[7]

Another possibility is to filter out problematic features in the Prediction step by using the function ρ. However, automatic detection of such features (i.e., automatic derivation of ρ) is undecidable for the same reason as the *prediction nontermination* problem (caused by left recursion) for unification grammars (Shieber, 1985). Manual detection is also problematic: when a grammar is large, particularly if semantic features are included, complete detection is nearly impossible. As for the techniques developed so far which (partially) solve prediction nontermination (e.g. (Shieber, 1985), (Haas, 1989), (Samuelsson, 1993)), they do not apply to nonminimal derivations because nonminimal derivations may arise without left recursion or recursion in general.[8] [9] One way is to define ρ to filter all features except the context-free backbone of predictions. However, this severely restricts the range of possible instantiations of Shieber's algorithm.[10]

A third possibility is to manually fix the grammar so that nonminimal derivations do not occur, as we noted earlier in section 4. However, this approach is problematic for the same reason as the manual derivation of ρ mentioned above.

7 For example, when there are two predictions $M1$ and $M2$ for category C and a production p where $M1 = \{\langle \text{cat}\rangle \doteq \text{C}, \langle x\rangle \doteq a\}$, $M2 = \{\langle \text{cat}\rangle \doteq \text{C}, \langle y\rangle \doteq b\}$, and $p = \langle 1, \{\langle \text{cat}\rangle \doteq \text{C}, \langle 1\ \text{cat}\rangle \doteq \text{D}, \langle x\rangle \doteq a\}\rangle$ respectively, the resulting model $M2' = \{\langle \text{cat}\rangle \doteq \text{C}, \langle 1\ \text{cat}\rangle \doteq \text{D}, \langle x\rangle \doteq a, \langle y\rangle \doteq b\}$ will have strictly more information than the other resulting model $M1' = \{\langle \text{cat}\rangle \doteq \text{C}, \langle 1\ \text{cat}\rangle \doteq \text{D}, \langle x\rangle \doteq a\}$, although both models are minimal.

8 We do not show any particular example here, but if we change the left-recursive VP rule in the earlier example to non-left-recursive, for instance VP $\rightarrow$ VP2 ADV, and add some rules, a nonminimal parse will indeed arise.

9 As a note, some (but not all) cases of prediction nontermination will produce nonminimal derivations. Those cases occur when there is a prediction for a category, and repeated applications of some left-recursive rule(s) generate predictions for the same category that are not mutually exclusive to the original prediction or each other.

10 In Head-corner parsing, (Sikkel, 1997) proposes the use of *transitive features*: the features that propagate only through head arcs. However, this method does not solve nonminimal derivations either, because problematic features may be sub-features of head (such as the example case shown earlier), which will not be filtered.

6 Modified Algorithm

Finally, we propose an algorithm which does not produce nonminimal derivations. It is a modification of Shieber's algorithm which incorporates parent pointers. Figure 6 shows the modified algorithm. In the figure, an item is represented by a nested 3-tuple where the first argument is the self index, the second is the parent index/pointer, and the third is the old 5-tuple used in Shieber's original algorithm. As you can see, a parent pointer is set in Prediction – the resulting item has the index of the antecedent item (id) as its parent. By generating a new symbol for the self index in every Prediction item (id'), parent pointers in those items are threaded to form a prediction path. Then in Completion, the parent pointer is used to restrict the antecedent items: the complete item (on the right) must have the prior expectation (on the left) as its parent (id), thereby ensuring a prediction path to be precisely restored.

While this modified algorithm offers a complete solution on the level of logic, it has some undesirable implications in implemented systems. The most prominent one is that the parent pointer scheme makes implementation of *memoization* rather difficult. Normally, memoization is used to avoid storing duplicate items which are identical; however, in the modified algorithm, many items which are otherwise identical will have different parent pointers, thereby changing the polynomial time algorithm ($O(n^3)$; (Earley, 1970)) to an exponential one. To avoid computational inefficiency, a way must be devised for items which are identical except for parent pointers to share information, especially models, and thus avoid the expense of duplicate identical unification operations. One possibility is to represent the 5-tuple from Shieber's original algorithm by a separate structure and have an index to it in the new 3-tuple item. This way, not only can the items be shared, they can still be memoized in the usual way as well. Another possibility is to adopt an efficiency technique along the line of *selective memoization* (van Noord, 1997). Implementation and empirical analysis is our future research.

Whatever the practical performance will turn out to be, it is important to note that the proposed algorithm is a *formal* solution which guarantees minimality for any grammar defined in Shieber's logic. Moreover the algorithm preserves the same generality and flexibility as Shieber's: a mixed top-down bottom-up parsing with the filtering function ρ to allow various instantiations of the algorithm to characterize their algorithms.

References

Alshawi, H., editor. 1992. *The Core Language Engine*. MIT Press.

Earley, J. 1970. An efficient context-free parsing algorithm. *Communications of the ACM*, 13(2).

Gazdar, G., E. Klein, G. Pullum, and I. Sag. 1985. *Generalized Phrase Structure Grammar*. Blackwell Publishing.

Haas, A. 1989. A parsing algorithm for unification grammar. *Computational Linguistics*, 15(4):219–232.

Pereira, F. and D. Warren. 1980. Definite clause grammars for language analysis. *Artificial Intelligence*, 13:231–278.

Pollard, C. and I. Sag. 1994. *Head-driven Phrase Structure Grammar*. CSLI. The University of Chicago Press.

Samuelsson, C. 1993. Avoiding non-termination in unification grammars. In *Natural Language Understanding and Logic Programming IV*.

Shieber, S. 1985. Using restriction to extend parsing algorithms for complex-feature-based formalisms. In *Proceedings of the 23rd Annual Meeting of the ACL*.

Shieber, S. 1986. *An Introduction to Unification-Based Approaches to Grammar*. CSLI.

Shieber, S. 1992. *Constraint-based Grammar Formalisms*. MIT Press.

Sikkel, K. 1997. *Parsing Schemata*. Springer-Verlag.

Tomuro, N. 1999. *Left-Corner Parsing Algorithm for Unification Grammars*. Ph.D. thesis, DePaul University.

van Noord, G. 1997. An efficient implementation of the head-corner parser. *Computational Linguistics*, 23(3):425–456.