Chart Parsing of Scattered Context Grammars

F. Popowich*
School of Computing Science, Simon Fraser University
Burnaby, B.C., Canada V5A 1S6

(Received and accepted August 1993)

Abstract—Scattered context grammars are a class of context-sensitive grammars. The rules of these grammars can be viewed as sequences of traditional context-free grammar rules. We show how a chart-parsing algorithm for context-free grammars can be extended to scattered context grammars.

1. SCATTERED CONTEXT GRAMMARS

Although context-free grammars (CFGs) have frequently been used in natural language processing research, this family of grammars is not powerful enough to describe some linguistic phenomena [1]. Scattered Context Grammars (SCGs) are more powerful than CFGs, being a subset of context-sensitive grammars [2]. Chart parsing, which is based on Earley’s algorithm [3], is often used for efficient processing of CFGs. Here, we will show how the techniques used in chart parsing CFGs can be applied to SCGs.

For an SCG possessing a set of symbols $V$ and a set of nonterminal symbols $VN$ ($VN \subset V$), the rules of SCGs are of the form $(A_1, \ldots, A_n) \rightarrow (w_1, \ldots, w_n), n \geq 1$, where $A_i \in VN$ and $w_i \in V^+$. Given a rule of this form, the grammar allows the derivation $x_1A_1x_2A_2 \cdots x_nA_nx_{n+1} \Rightarrow x_1w_1x_2w_2 \cdots x_nw_nx_{n+1}$, where the $x_i \in V^*$.

SCGs are closely related to Static Discontinuity Grammars (SDGs) [4, 5]. Actually, SDGs are a generalization of SCGs in that SDGs have terms from the Herbrand universe as symbols rather than just finite sets of atomic symbols. In SDGs, the grammar rules are of the form $A_1 \rightarrow w_1; \cdots; A_n \rightarrow w_n$. Under the rewrite interpretation of SDG rules, a rule of this form is also defined to license the derivation $x_1A_1x_2A_2 \cdots x_nA_nx_{n+1} \Rightarrow x_1w_1x_2w_2 \cdots x_nw_nx_{n+1}$ that we introduced above for SCGs.

Using the rule format associated with SDGs makes it clear that the grammar rules of an SCG are simply sequences of CFG rules which we will call the subrules of an SCG rule. Application of an SCG rule involves the application of each subrule at some location in the derivation. This is best illustrated with an example. Consider the following grammar$^2$ for the language $a^nb^n c^n$.

\begin{align*}
R_1 & \quad S \rightarrow AS BS CS \\
R_2 & \quad AS \rightarrow a AS ; BS \rightarrow b BS ; CS \rightarrow c CS \\
R_3 & \quad AS \rightarrow a ; BS \rightarrow b ; CS \rightarrow c
\end{align*}

* I would like to thank J. Calder, V. Dahl, D. Fass, S. Kodric and C. Vogel for their comments and discussions on earlier versions of this manuscript. The research presented in this paper is supported by an Operating Grant from the Natural Sciences and Engineering Research Council of Canada (NSERC), and by NSERC Infrastructure and Equipment Grants to the Logic and Functional Programming Laboratory at Simon Fraser University.

1 As discussed in [5], there are different ways of interpreting an SDG rule other than in terms of a rewriting relation $\Rightarrow$. We will be assuming a tree admissibility interpretation in this paper. For the grammars that we will be looking at, the languages generated under the rewrite and tree admissibility interpretation are identical.

2 Where $VN = \{S, AS, BS, CS\}$ and $V = VN \cup \{a, b, c\}$; henceforth, uppercase strings will denote nonterminals and lowercase letters will denote terminal symbols.
The following rewrites can be applied to obtain a derivation of the sentence \( a a b b c c \).

\[
S \Rightarrow_{R1} ASBS CS \Rightarrow_{R2} a ASb BS c CS \Rightarrow_{R3} a a b b c c
\]

2. CHART PARSING

In a chart parsing algorithm for CFGs (cf. [6]), the chart is used to keep track of different states of the parse. States contain information about complete constituents (like lexical items or phrases) and about incomplete constituents (which have expectations for specific complete subconstituents). By keeping track of states, we can avoid duplicating computations.

It is probably easiest to view the chart as a directed graph whose nodes correspond to positions between the words of a sentence, and whose edges contain information about the different possible ways one can attempt to parse the words delimited by the starting and ending position of the edge.

For example, consider the CFG for the language \( ab \) consisting of the productions \( S \rightarrow ASB \) and \( S \rightarrow AB \). We will assume that \( A \) and \( B \) are preterminal categories for the terminal symbols \( a \) and \( b \) (see footnote\(^3\)). The chart constructed for the parse of the sentence \( ab \) contains three nodes, say 1, 2, and 3, where 1 corresponds to the beginning of the sentence and 3 to the position at the end of the sentence. Edges describing complete constituents are called inactive edges. For example, there will be an inactive edge from position 1 to position 3 stating that the constituent is an \( S \) containing an \( A \) and a \( B \). Using the dotted rule notation, we can describe this edge as a sequence containing the starting position, ending position and label of the edge, namely \( (1, 3, S \rightarrow AB) \). Any symbols appearing to the left of the dot on the right hand side of a rule correspond to symbols that have been found. Edges describing incomplete constituents are called active edges. For example, there will be an active edge from position 1 to position 2 corresponding to an \( S \) that has found an \( A \) but is still looking for a \( B \), namely \( (1, 2, S \rightarrow AB) \).

The chart is initialized by introducing an inactive edge for each word in the input sentence. So, assuming a word \( w \) is introduced by the rule of the form \( A \rightarrow w \), edge \( (i, i+1, A \rightarrow w) \) will be associated with word \( i \) in a sentence.

The fundamental rule of chart parsing describes how new edges are created from an active edge meeting an inactive edge:

If the chart contains edges \( (i, j, A \rightarrow W1 \cdot BW2) \) and \( (j, k, B \rightarrow W3) \), where \( A \) and \( B \) are categories and \( W1, W2 \) and \( W3 \) are (possibly empty) sequences of categories or words, then add edge \( (i, k, A \rightarrow W1B \cdot W3) \) to the chart [6, p. 195].

How rules are used by the chart parser differs depending on whether a top-down or a bottom-up search strategy is used. Under the bottom-up strategy, inactive edges are used to create new active edges in a process called rule invocation.

If you are adding edge \( (i, j, C \rightarrow W1) \) to the chart, then for every rule in the grammar of the form \( B \rightarrow C W2 \), add an edge \( (i, i, B \rightarrow C W2) \) to the chart [6, p. 197].

Instead of using inactive edges to create new edges in combination with a grammar rule, the top-down strategy uses active edges. The top-down strategy need not concern us here.\(^4\)

Edges are never removed from the chart. Before an edge is added to the chart, a test is performed to see if the edge already exists. If it does, then it is not added to the chart again.

\(^3\)To simplify our discussion, we will assume that all terminal symbols are introduced via rules involving just the preterminal symbol and the terminal symbol itself.

\(^4\)As described in [5], there is an algorithm to convert SDGs (and thus SCGs) into an equivalent DCG which can be used by a top-down parser. One could then use a (top-down) DCG chart parser on this equivalent grammar. The same techniques that are used in the SDG to DCG conversion algorithm could be directly incorporated into the chart parser to obtain a top-down SDG chart parser.
3. EXTENSIONS TO THE CHART

Chart parsing generalizes from CFGs to SCGs. In the generalization, as before, dotted rules track the state of a constituent. The difference is the dot may appear anywhere on the right hand side of any one subrule of an SCG rule. In the case where the SCG rule has only one subrule, this reduces to the case of dotted rules for traditional CFG rules. During a parse, the dot will "move" rightward through each subrule until it reaches the very end of the rule. Edges in which the dot occurs at the rightmost position of any subrule except the last will be called semiactive edges. They are not inactive since there are still more subrules to be processed, and they are not active since they are not actively looking for a symbol in the next position.

Since an SCG rule consists of a sequence of CFG rules, the application of the SCG rule can be achieved through the successive application of each of its subrules. Each subrule can be used to find a constituent of the final parse. We then need to ensure that if the constituent associated with one subrule is used in a complete parse, then the constituents associated with the other subrules are also used (and that each constituent is used only once). As we saw earlier, in the sentence a a b b c c, we use both rule R2 to find an AS, a BS and a CS, and rule R3 to find an AS, a BS and a CS. We need to ensure that when we use R1 to form an S from AS, BS and CS, that each of the constituents comes from the same rule (and not, for instance, AS from R2 but BS and CS from R3). This effect can be obtained by associating a list of pending subrules with each edge of the chart. Each element of the pending list will contain subrules that must be used at some point later in the parse. For example, if we used the first subrule of R2 to find an AS, then we had better use the remaining two subrules later in the parse. Whenever a multipart rule is invoked, after the constituent associated with the first subrule is found, the remaining subrules are placed in the pending list. Additionally, a used list will be associated with each edge. The role of this list is to keep track of the invocation of rules that were pending elsewhere in the chart. In our algorithm, it is not actually necessary to explicitly keep track of the subrules appearing to the left of the one with the dot. So an edge will be of the form $(i, j, C \rightarrow W_1, \alpha, \beta)$ where the $\rho_k$ ($1 \leq k \leq m$) are subrules ($\rho_1$ contains the dot), and $\alpha$ (the pending list) and $\beta$ (the used list) are sequences of SCG rules. When edges are combined according to the fundamental rule to make an new edge, subrules that have been used to make a complete constituent are removed from pending and used lists of the new edge as we shall see in the next section. The pending lists and used lists are required to be empty in edges associated with complete parses.

4. EXTENSIONS TO THE ALGORITHM

The rule invocation process for SCGs can make use of any one of the grammar rules, or it can make use of the one of the subrules to the right of the dot in a semiactive edge. This is reflected in the following extended definition of rule invocation (RI).

**RI** If you are adding an (inactive) edge $(i, j, C \rightarrow W_1, \alpha, \beta)$ to the chart, then

1. for every rule in the grammar of the form $B \rightarrow C W_2; \rho_2; \ldots; \rho_m$, add an (active) edge $(i, i, B \rightarrow C W_2; \rho_2; \ldots; \rho_m, \langle\rangle, \langle\rangle)$ to the chart;
2. for every (semiactive) edge $(p, q, \rho_1; B \rightarrow C W_2; \ldots; \rho_m, \langle\rangle, \langle\rangle)$ where $q \leq i, m > 1$, add an (active) edge $(i, i, B \rightarrow C W_2; \ldots; \rho_m, \langle\rangle, \langle B \rightarrow C W_2; \ldots; \rho_m\rangle)$ to the chart.

RI.1 is straightforward invocation of a new rule, and RI.2 is invocation of a partially used rule (which is also placed in the used list). RI.2 applies rule invocation to the semiactive edges, and effectively moves the dot from the end of one subrule to the beginning of the next subrule.

We also need to extend the operation associated with the fundamental rule (FR) to include a treatment for the pending and used lists.
Figure 1. Scattered Context Grammar for $a^nb^nc^n$.

- **R1** $S \rightarrow AS BS CS$
- **R2** $AS \rightarrow A AS$ ; $BS \rightarrow B BS$ ; $CS \rightarrow C CS$
- **R3** $AS \rightarrow a$
- **R4** $BS \rightarrow b$
- **R5** $CS \rightarrow c$
- **R6** $A \rightarrow a$
- **R7** $B \rightarrow b$
- **R8** $C \rightarrow c$

Figure 2. Selection of edges from chart after initialization.

- **FR** If the chart contains edges $(i, j; A \rightarrow W_1 B W_2, \alpha_A, \beta_A)$ and $(j, k; B \rightarrow W_3, \alpha_I, \beta_I)$, where $A, B, W_1, W_2$ and $W_3$ are as defined above, then add edge $(i, k; A \rightarrow W_1 B \cdot W_2, \alpha', \beta')$ to the chart. $\alpha'$ contains all occurrences of rules from $\alpha_I$ and all those from $\alpha_A$ except those that also appear in $\beta_I$ (an occurrence of a rule in both $\alpha_A$ and $\beta_I$ is said to be dismissed and thus is not included in $\alpha'$). $\beta'$ contains all rules from $\beta_A$ and those in $\beta_I$ except those occurrences that were dismissed.$^5$

To illustrate how occurrences of rules are dismissed, if $\alpha_A$ contains two occurrences of rule $R$, $\beta_i$ contains one occurrence of $R$, and $\alpha_I$ and $\beta_A$ are empty, then $\alpha'$ will contain only one occurrence of $R$ and $\beta'$ will be empty.

We also need to incorporate a new rule which we will call subrule completion (SC) for adding semiactive edges to the chart. The first part will create inactive edges from semiactive edges, putting the subrules to the right of the one with the dot into the pending list. The notation $(X \mid Y)$ is used to describe the sequence resulting from prepending the rule $X$ to the sequence of rules $Y$. The second part of SC is complementary to RI.2.

<table>
<thead>
<tr>
<th>Edge</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1: $(1, 2; A \rightarrow a, {}, {})$</td>
<td>2: $(2, 3; AS \rightarrow a, {}, {})$</td>
</tr>
<tr>
<td>3: $(3, 4; B \rightarrow b, {}, {})$</td>
<td>4: $(4, 5; BS \rightarrow b, {}, {})$</td>
</tr>
<tr>
<td>5: $(5, 6; C \rightarrow c, {}, {})$</td>
<td>6: $(6, 7; CS \rightarrow c, {}, {})$</td>
</tr>
</tbody>
</table>

To illustrate how occurrences of rules are dismissed, if $\alpha_A$ contains two occurrences of rule $R$, $\beta_i$ contains one occurrence of $R$, and $\alpha_I$ and $\beta_A$ are empty, then $\alpha'$ will contain only one occurrence of $R$ and $\beta'$ will be empty.

We also need to incorporate a new rule which we will call subrule completion (SC) for adding semiactive edges to the chart. The first part will create inactive edges from semiactive edges, putting the subrules to the right of the one with the dot into the pending list. The notation $(X \mid Y)$ is used to describe the sequence resulting from prepending the rule $X$ to the sequence of rules $Y$. The second part of SC is complementary to RI.2.

**SC** If you are adding a (semiactive) edge $(p, q; A \rightarrow W_1; B \rightarrow C W_2; \ldots ; \rho_m; \alpha, \beta)$ to the chart then:

1. add an inactive edge $(p, q; A \rightarrow W_1; (B \rightarrow C W_2; \ldots ; \rho_m) \mid \alpha, \beta)$ to the chart, and
2. for every inactive edge $(i, j; C \rightarrow W_3; \alpha', \beta')$ where $q \leq i$ in the chart, add an edge $(i, i; B \rightarrow C W_2; \ldots ; \rho_m, \{\}, (B \rightarrow C W_2; \ldots ; \rho_m))$ to the chart.

5. EXAMPLE

To illustrate the bottom-up chart parsing algorithm for SCGs, we will use the grammar shown in Figure 1 for the $a^nb^nc^n$ language, which contains only one multipart grammar rule. Rules R3–R8 from Figure 1 are only used in the initialization of the chart. Initial edges have empty used and pending lists. After initialization, the chart will include the edges shown in Figure 2, amongst others. Figure 3 shows a selection of the edges used to achieve a parse for the sentence $aabbcc$. In parsing this sentence, a total of 42 edges are actually created.

The parsing algorithm works as follows. The addition of (inactive) edge 1 to the chart, is subject to rule invocation (RI.1), thus creating (active) edge 7. When this edge is combined (via FR) with edge 1, and the resulting edge is combined (via FR) with edge 2, we get the semiactive edge 8. Since the next subrule of this semiactive edge can be used (via RL.2 or SC.2) by the inactive edge 3, a new active edge 9 is created whose starting position is at the starting position

---

$^5$It is possible to give a more restrictive definition of how rules can be dismissed and thus obtain the rewrite interpretation of SCGs. The current definition, which results in the tree admissibility interpretation, is somewhat simpler and thus has been presented here.
of edge 3. Edge 8 also results in the creation of an inactive edge 10, where the subrules to the right of the dot have been placed into the pending list (via SC.1). Using edge 9 eventually results in another semiactive edge (11) as the dot is moved through the next subrule by appeal to FR. Similarly, this semiactive edge 11 introduces two new edges into the chart, one active edge, 12 (via RI.2 or SC.2), and one inactive edge, 13 (via SC.1). Once edge 12 combines with edge 5 and produces an edge that can combine with edge 6, we get inactive edge 14. At this point, note that we have three important inactive edges: edge 10, which is an AS constituent that spans all the a’s; edge 13, a BS which spans all the b’s; and edge 14, a CS constituent which spans all the c’s. These edges, which have been highlighted in Figure 3, all have rules in either their pending or their used lists. Observe that edge 10 causes the creation of a new active edge, edge 15, through the invocation of rule RI. This new active edge can use edge 10 to satisfy its expectation for an AS, resulting in edge 16. Edge 16 also inherits the pending list of edge 10, according to the FR definition. The fundamental rule then combines active edge 16 with inactive edge 13, to obtain edge 17. Note that the rule BS → B BS; CS → C CS, which was in the pending list of the active edge (16) is not in the pending list of edge 17, since it was in the used list of the inactive edge (13). However, CS → C CS, which appeared in the pending list of 13, is passed onto edge 17. Finally, (active) edge 17 combines with (inactive) edge 14, to yield an inactive edge than spans the entire sentence, edge 18. This edge contains an empty pending list and an empty used list, since the only element in the pending list of the active edge (17) was also the only element in the used list of the inactive edge (14). Edge 18 corresponds to a successful parse of the sentence.

6. DISCUSSION

One can view the use of the pending and used lists as a bottom-up implementation of the same principle that underlies the threading or difference-list technique used in other SDG parsing algorithms [5]. That principle is that multipart rules are used a subrule at a time, and when a new multipart rule is invoked from the grammar, only the first subrule is used while the other subrules are stored away for use elsewhere in the parse. In a successful parse, all parts must be used, and no part may be used more than once.

The simple parsing algorithm presented here does not satisfy a completeness criterion, since there are some grammars for which the parser would not find a valid parse for all inputs. Consider a (contrived) grammar for the language z a n b n, consisting of the rules S → X Y, X → X; Y → a Y b, Y → a b, and X → x. Given a sentence from the language of this grammar, the parser would repeatedly invoke the multipart rule, resulting in edges containing increasing numbers of the subrule Y → a Y b. This behavior can be avoided by modifying the
algorithm so that SC.1 (the creation of an inactive edge from a semiactive edge) is not performed
until the remaining subrules of a multipart rule have been successfully applied. Since it appears
that grammars that cause this behavior can be rewritten in a manner to avoid this problem, just
like infinitely ambiguous context-free grammars have equivalent finitely ambiguous counterparts,
we have chosen not to modify the chart parsing algorithm.

The parsing algorithm can be extended to work for SDGs. Since SDGs are essentially SCCs
whose grammar symbols contain variables, modifications are needed to ensure that variable sub-
stitutions applied to one subrule are also applied to the other subrules from the same rule invo-
cation. Furthermore, the notion of an edge being already in the chart becomes more complex.

REFERENCES

1. G. Gazdar and G.K. Pullum, Computationally relevant properties of natural languages and their grammars,
2. S. Greibach and J. Hopcroft, Scattered context grammars, Journal of Computer and System Sciences 3,
5. V. Dahl and F. Popowich, Parsing and generation with static discontinuity grammars, New Generation Com-
6. G. Gazdar and C. Mellish, Natural Language Processing in Prolog: An Introduction to Computational Lin-