Parsing Schemata for Grammars with Variable Number and Order of Constituents

Karl-Michael Schneider
Department of General Linguistics
University of Passau
Innstr. 40, 94032 Passau, Germany
schneide@phil.uni-passau.de

Abstract

We define state transition grammars (STG) as an intermediate formalism between grammars and parsing algorithms which is intended to separate the description of a parsing strategy from the grammar formalism. This allows to define more general parsing algorithms for larger classes of grammars, including grammars where the number and order of subconstituents defined by a production may not be fixed. Various grammar formalisms are characterized in terms of properties of STG's. We define an Earley parsing schema for STG's and characterize the valid parse items. We also discuss the usability of STG's for head-corner parsing and direct parsing of sets of tree constraints.

1 Introduction

This paper addresses the question of how to define (tabular) parsing algorithms on a greater level of abstraction, in order to apply them to larger classes of grammars (as compared to parsing algorithms for context-free grammars). Such an abstraction is useful because it allows to study properties of parsing algorithms, and to compare different parsing algorithms, independently of the properties of an underlying grammar formalism. While previous attempts to define more general parsers have only aimed at expanding the domain of the nonterminal symbols of a grammar (Pereira and Warren, 1983), this paper aims at a generalization of parsing in a different dimension, namely to include grammars with a flexible constituent structure, i.e., where the sequence of subconstituents specified by a grammar production is not fixed. We consider two grammar formalisms: Extended context-free grammars (ECFG) and ID/LP grammars.

ECFG's (sometimes called *regular right part grammars*) are a generalization of context-free grammars (CFG) in which a grammar production specifies a regular set of sequences of subconstituents of its left-hand side instead of a fixed sequence of subconstituents. The right-hand side of a production can be represented as a regular set, or a regular expression, or a finite automaton, which are all equivalent concepts (Hopcroft and Ullman, 1979). ECFG's are often used by linguistic and programming language grammar writers to represent a (possibly infinite) set of context-free productions as a single production rule (Kaplan and Bresnan, 1982; Woods, 1973). Parsing of ECFG's has been studied for example in Purdom, Jr. and Brown (1981) and Leermakers (1989). Tabular parsing techniques for CFG's can be generalized to ECFG's in a natural way by using the computations of the finite automata in the grammar productions to guide the recognition of new subconstituents.

ID/LP grammars are a variant of CFG's that were introduced into linguistic formalisms to encode word order generalizations (Gazdar et al., 1985). Here, the number of subconstituents of the left-hand side of a production is fixed, but their order can vary. ID rules (*immediate dominance rules*) specify the subconstituents of a constituent but leave their order unspecified. The admissible orderings of subconstituents are specified separately by a set of LP constraints (*linear precedence constraints*).

A simple approach to ID/LP parsing (called indirect parsing) is to fully expand a grammar into a CFG, but this increases the number of productions significantly. Therefore, direct parsing algorithms for ID/LP grammars were proposed (Shieber, 1984). It is also possible to encode an ID/LP grammar as an ECFG by interleaving the ID rules with LP checking with-

out increasing the number of productions. However, for unification ID/LP grammars, expansion into a CFG or encoding as an ECFG is ruled out because the information contained in the ID rules is only partial and has to be instantiated, which can result in an infinite number of productions. Moreover, Seiffert (1991) has observed that, during the recognition of subconstituents, a subconstituent recognized in one step can instantiate features on another subconstituent recognized in a previous step. Therefore, all recognized subconstituents must remain accessible for LP checking (Morawietz, 1995).

We define an intermediate formalism between grammars and parsers (called *state transition grammars*, STG) in which different grammar formalisms, including CFG's, ECFG's, and ID/LP grammars can be represented. Moreover, admissible sequences of subconstituents are defined in a way that allows a parser to access subconstituents that were recognized in previous parsing steps. Next, we describe an Earley algorithm for STG's, using the parsing schemata formalism of Sikkel (1993). This gives us a very high level description of Earley's algorithm, in which the definition of parsing steps is separated from the properties of the grammar formalism. An Earley algorithm for a grammar may be obtained from this description by representing the grammar as an STG.

The paper is organized as follows. In Section 2, we define STG's and give a characterization of various grammar formalisms in terms of properties of STG's. In Section 3 we present an Earley parsing schema for STG's and give a characterization of the valid parse items. In Section 4, we introduce a variant of STG's for head-corner parsing. In Section 5, we discuss the usability of STG's to define parsers for grammars that define constituent structures by means of local tree constraints, i.e., formulae of a (restricted) logical language. Section 6 presents final conclusions.

2 State Transition Grammars

We denote nonterminal symbols with A, B, terminal symbols with a, terminal and nonterminal symbols with X, states with Γ, strings of symbols with β, γ, and the empty string with ε. An STG is defined as follows:

Definition 1 (STG). *An STG G is a tuple $(N, \Sigma, \mathcal{M}, \mathcal{M}_F, \vdash_G, P, S)$ where*

- *N is a finite set of nonterminal symbols,*
- *Σ is a finite set of terminal symbols,*
- *$\mathcal{M}$ is a finite set of states,*
- *$\mathcal{M}_F \subseteq \mathcal{M}$ is a set of final states,*
- *$\vdash_G \subseteq (\mathcal{M} \times V)^2$ is a binary relation of the form $(\Gamma, \beta) \vdash_G (\Gamma', \beta X)$, where $V = N \cup \Sigma$,*
- *$P \subseteq N \times \mathcal{M} \setminus \mathcal{M}_F$ is a set of productions written as $A \to \Gamma$, and*
- *$S \in N$ is a start symbol.*

Note that we do not allow final states in the right-hand side of a production. A pair (Γ, β) is called a *configuration*. If Γ is a final state then (Γ, β) is called a final configuration. The reflexive and transitive closure of $\vdash_G$ is denoted with $\vdash_G^*$. The *state projection* of $\vdash_G$ is the binary relation

$$\sigma(\vdash_G) = \{(\Gamma, \Gamma') \mid \exists \beta X : \ (\Gamma, \beta) \vdash_G (\Gamma', \beta X)\}.$$

$\vdash_G$ is called *context-free* iff a transition from (Γ, β) does not depend on β, formally: for all $\beta, \beta', \Gamma, \Gamma', X$: $(\Gamma, \beta) \vdash_G (\Gamma', \beta X)$ iff $(\Gamma, \beta') \vdash_G (\Gamma', \beta' X)$. The set of *terminal states* of G is the set

$$\top(G) = \{\Gamma \mid \forall \Gamma' : \ (\Gamma, \Gamma') \notin \sigma(\vdash_G)\}.$$

The language defined by a state Γ is the set of strings in the final configurations reachable from (Γ, ε):

$$L(\Gamma) = \{\beta \mid \exists\, \Gamma' \in \mathcal{M}_F : \ (\Gamma, \varepsilon) \vdash_G^* (\Gamma', \beta)\}.$$

Note that if $A \to \Gamma$ is a production then $\varepsilon \notin L(\Gamma)$ (i.e., there are no ε-productions). The derivation relation is defined by $\gamma A \delta \Longrightarrow \gamma \beta \delta$ iff for some production $A \to \Gamma$: $\beta \in L(\Gamma)$. The language defined by G is the set of strings in Σ^* that are derivable from the start symbol.

We denote a CFG as a tuple (N, Σ, P, S) where N, Σ, S are as before and $P \subseteq N \times V^+$ is a finite set of productions $A \to \beta$. We assume that there are no ε-productions.

An ECFG can be represented as an extension of a CFG with productions of the form $A \to \mathcal{A}$, where $\mathcal{A} = (V, Q, q_0, \delta, Q_f)$ is a nondeterministic finite automaton (NFA) without ε-transitions,

	$\mathcal{M}$	$\mathcal{M}_F$	$(\Gamma, \beta) \vdash_G (\Gamma', \beta X)$
CFG	$\{\beta' \mid \exists A \to \beta\beta' \in P\}$	$\{\varepsilon\}$	$\Gamma = X\Gamma'$
ECFG	Q	Q_f	$(\Gamma, X, \Gamma') \in \delta$
ID/LP	$\{M \mid \exists A \to M' \in P :\ M \subseteq M'\}$	$\{\emptyset\}$	$\Gamma = \Gamma' \cup \{X\},\ \beta X \in LP$

Table 1: Encoding of grammars in STG's.

with input alphabet V, state set Q, initial state q_0, final (or accepting) states Q_f, and transition relation $\delta \subseteq Q \times V \times Q$ (Hopcroft and Ullman, 1979). $\mathcal{A}$ accepts a string β iff for some final state $q \in Q_f$, $(q_0, \beta, q) \in \delta^*$. Furthermore, we assume that $q_0 \notin Q_f$, i.e., $\mathcal{A}$ does not accept the empty word. We can assume without loss of generalization that the automata in the right-hand sides of a grammar are all disjoint. Then we can represent an ECFG as a tuple $(N, \Sigma, Q, Q_f, \delta, P, S)$ where $N, \Sigma, Q, Q_f, \delta, S$ are as before and $P \subseteq N \times Q$ is a finite set of productions $A \to q_0$ (q_0 is an initial state). For any production $p = A \to q_0$ let $\mathcal{A}^p = (V, Q, q_0, \delta, Q_f)$ be the NFA with initial state q_0. The derivation relation is defined by $\gamma A \delta \Longrightarrow \gamma\beta\delta$ iff for some production $p = A \to q_0$, $\mathcal{A}^p$ accepts β.

An ID/LP grammar is represented as a tuple (N, Σ, P, LP, S) where N, Σ, S are as before and P is a finite set of productions (ID rules) $A \to M$, where $A \in N$ and M is a multiset over V, and LP is a set of linear precedence constraints. We are not concerned with details of the LP constraints here. We write $\beta \in LP$ to denote that the string β satisfies all the constraints in LP. The derivation relation is defined by $\gamma A \delta \Longrightarrow \gamma\beta\delta$ iff $\beta = X_1 \ldots X_k$ and $A \to \{X_1, \ldots, X_k\} \in P$ and $\beta \in LP$.

CFG's, ECFG's and ID/LP grammars can be characterized by appropriate restrictions on the transition relation and the final states of an STG:[1]

- CFG: $\vdash_G$ is context-free and deterministic, $\sigma(\vdash_G)$ is acyclic, $\mathcal{M}_F = \top(G)$.
- ECFG: $\vdash_G$ is context-free.
- ID/LP: $\sigma(\vdash_G)$ is acyclic, $\mathcal{M}_F = \top(G)$, for all Γ: if $\beta, \gamma \in L(\Gamma)$ then γ is a permutation of β.

For instance, if G is an STG that satisfies the conditions for CFG's, then a CFG G' can be constructed as follows: For every production $A \to q_0$ in G, let $A \to \beta$ be a production in G' where $L(q_0) = \{\beta\}$. Then the derivation relations of G and G' coincide. Similarly for the other grammar types. Conversely, if a grammar is of a given type, then it can be represented as an STG satisfying the conditions for that type, by specifying the states and transition relation, as shown in Table 1 ($\cup$ denotes multiset union).

3 Earley Parsing

Parsing schemata were proposed by Sikkel (1993) as a framework for the specification (and comparison) of tabular parsing algorithms. Parsing schemata provide a well-defined level of abstraction by abstracting from control structures (i.e., ordering of operations) and data structures. A parsing schema can be implemented as a tabular parsing algorithm in a canonical way (Sikkel, 1998).

A parsing schema for a grammar class is a function that assigns each grammar and each input string a deduction system, called a *parsing system*. A parsing schema is usually defined by presenting a parsing system. A parsing system consists of a finite set $\mathcal{I}$ of parse items, a finite set $\mathcal{H}$ of hypotheses, which encode the input string, and a finite set $\mathcal{D}$ of deduction steps of the form $x_1, \ldots, x_n \vdash x$ where $x_i \in \mathcal{I} \cup \mathcal{H}$ and $x \in \mathcal{I}$. The hypotheses can be represented as deduction steps with empty premises, so we can assume that all x_i are items, and represent a parsing system as a pair $(\mathcal{I}, \mathcal{D})$.

Correctness of a parsing system is defined with respect to some item semantics. Every item denotes a particular derivation of some substring of the input string. A parsing system is correct if an item is deducible precisely if it denotes an admissible derivation. Items that denote admissible derivations are called *correct*.

[1] These conditions define normal-forms of STG's; that is, for STG's that do not satisfy the conditions for some type there can nevertheless be strongly equivalent grammars of that type. These STG's are regarded as degenerate and are not further considered.

$$\mathcal{I} = \{[A \to \beta\bullet\Gamma, i, j] \mid A \in N,\ \beta \in V^*,\ \Gamma \in \mathcal{M},\ |\beta| \leq n,\ 0 \leq i \leq j \leq n\}$$

$$\mathcal{D}^{Init} = \frac{}{[S \to \bullet\Gamma, 0, 0]} \qquad S \to \Gamma \in P$$

$$\mathcal{D}^{Predict} = \frac{[A \to \beta\bullet\Gamma, i, j]}{[B \to \bullet\Gamma_0, j, j]} \qquad \exists\,\Gamma' :\ (\Gamma, \beta) \vdash_G (\Gamma', \beta B),\ B \to \Gamma_0 \in P$$

$$\mathcal{D}^{Scan} = \frac{[A \to \beta\bullet\Gamma, i, j]}{[A \to \beta a_{j+1}\bullet\Gamma', i, j+1]} \qquad (\Gamma, \beta) \vdash_G (\Gamma', \beta a_{j+1})$$

$$\mathcal{D}^{Compl} = \frac{\begin{array}{c}[A \to \beta\bullet\Gamma, i, j] \\ [B \to \gamma\bullet\Gamma_f, j, k]\end{array}}{[A \to \beta B\bullet\Gamma', i, k]} \qquad \Gamma_f \in \mathcal{M}_F,\ (\Gamma, \beta) \vdash_G (\Gamma', \beta B)$$

Figure 1: The Earley parsing schema for an STG G and input string $w = a_1 \ldots a_n$.

STG's constitute a level of abstraction between grammars and parsing schemata because they can be used to encode various classes of grammars, whereas the mechanism for recognizing admissible sequences of subconstituents by a parsing algorithm is built into the grammar. Therefore, STG's allow to define the parsing steps separately from the mechanism in a grammar that specifies admissible sequences of subconstituents.

A generalization of Earley's algorithm for CFG's (Earley, 1970) to STG's is described by the parsing schema shown in Fig. 1. An item $[A \to \beta\bullet\Gamma, i, j]$ denotes an A-constituent that is partially recognized from position i through j in the input string, where β is the sequence of recognized subconstituents of A, and a sequence of transitions that recognizes β can lead to state Γ. Note that the length of β can be restricted to the length of the input string because there are no ε-productions.

In order to give a precise definition of the semantics of the items, we define a derivation relation which is capable of describing the partial recognition of constituents. This relation is defined on pairs (γ, Δ) where $\gamma \in V^*$ and Δ is a finite sequence of states (a pair (γ, Δ) could be called a *super configuration*). γ represents the front (or yield) of a partial derivation, while Δ contains one state for every partially recognized constituent.

Definition 2. *The Earley derivation relation is defined by the clauses:*

- $(\gamma A, \Delta) \mathrel{|\!\sim} (\gamma\beta, \Gamma\Delta)$ *iff* $\exists A \to \Gamma' \in P$*:* $(\Gamma', \varepsilon) \vdash_G^* (\Gamma, \beta)$.
- $(\gamma A\delta, \Delta) \mathrel{|\!\sim} (\gamma\beta\delta, \Delta)$ *iff* $\gamma A\delta \Longrightarrow \gamma\beta\delta$.

The first clause describes the partial recognition of an A-constituent, where β is the recognized part and the state Γ is reached when β is recognized. The second clause describes the complete recognition of an A-constituent; in this case, the final state is discarded. Each step in the derivation of a super configuration (γ, Δ) corresponds to a sequence of deduction steps in the parsing schema. As a consequence of the second clause we have that $w \in L(G)$ iff $(S, \varepsilon) \mathrel{|\!\sim}^* (w, \varepsilon)$. Note that $\mathrel{|\!\sim}$ is too weak to describe the recognition of the next subconstituent of a partially recognized constituent, but it is sufficient to define the semantics of the items in Fig. 1. The following theorem is a generalization of the definition of the semantics of Earley items for CFG's (Sikkel, 1993) ($a_1 \ldots a_n$ is the input string):

Theorem 1 (Correctness).
$\vdash^* [A \to \beta\bullet\Gamma, i, j]$ *iff the conditions are satisfied:*

- *for some* Δ, $(S, \varepsilon) \mathrel{|\!\sim}^* (a_1 \ldots a_i A, \Delta)$.
- $(A, \varepsilon) \mathrel{|\!\sim} (\beta, \Gamma)$.
- $\beta \Longrightarrow^* a_{i+1} \ldots a_j$.

The first and third condition are sometimes called *top-down* and *bottom-up* condition, respectively. The second condition refers to the partial recognition of the A-constituent.

$[E \to \bullet q_1, 0, 0]$	$(E,\varepsilon) \mid\!\sim^* (E,\varepsilon),\ (E,\varepsilon) \mid\!\sim (\varepsilon, q_1)$
$[T \to \bullet q_3, 0, 0]$	$(E,\varepsilon) \mid\!\sim (T, q_2),\ (T,\varepsilon) \mid\!\sim (\varepsilon, q_3)$ $(E,\varepsilon) \mid\!\sim (T,\varepsilon)$
$[F \to \bullet q_5, 0, 0]$	$(E,\varepsilon) \mid\!\sim (T, q_2) \mid\!\sim (F, q_4 q_2),\ (F,\varepsilon) \mid\!\sim (\varepsilon, q_5)$ $(E,\varepsilon) \mid\!\sim (T, q_2) \mid\!\sim (F, q_2)$ $(E,\varepsilon) \mid\!\sim (T,\varepsilon) \mid\!\sim (F, q_4)$ $(E,\varepsilon) \mid\!\sim (T,\varepsilon) \mid\!\sim (F,\varepsilon)$
$[F \to a \bullet q_6, 0, 1]$	$(E,\varepsilon) \mid\!\sim (T, q_2) \mid\!\sim (F, q_4 q_2),\ (F,\varepsilon) \mid\!\sim (a, q_6)$ $(E,\varepsilon) \mid\!\sim (T, q_2) \mid\!\sim (F, q_2)$ $(E,\varepsilon) \mid\!\sim (T,\varepsilon) \mid\!\sim (F, q_4)$ $(E,\varepsilon) \mid\!\sim (T,\varepsilon) \mid\!\sim (F,\varepsilon)$
$[T \to F \bullet q_4, 0, 1]$	$(E,\varepsilon) \mid\!\sim (T, q_2),\ (T,\varepsilon) \mid\!\sim (F, q_4),\ F \Longrightarrow^* a$ $(E,\varepsilon) \mid\!\sim (T,\varepsilon)$
$[F \to a \bullet q_6, 2, 3]$	$(E,\varepsilon) \mid\!\sim (T, q_2) \mid\!\sim (F * F, q_4 q_2) \mid\!\sim (a * F, q_4 q_2),\ (F,\varepsilon) \mid\!\sim (a, q_6)$ $(E,\varepsilon) \mid\!\sim (T, q_2) \mid\!\sim (F * F, q_2) \mid\!\sim (a * F, q_2)$ $(E,\varepsilon) \mid\!\sim (T,\varepsilon) \mid\!\sim (F * F, q_4) \mid\!\sim (a * F, q_4)$ $(E,\varepsilon) \mid\!\sim (T,\varepsilon) \mid\!\sim (F * F,\varepsilon) \mid\!\sim (a * F,\varepsilon)$
$[E \to T * T \bullet q_2, 0, 3]$	$(E,\varepsilon) \mid\!\sim^* (E,\varepsilon),\ (E,\varepsilon) \mid\!\sim (T * T, q_2),\ T * T \Longrightarrow^* a * a$

Table 2: Valid parse items and derivable super configurations for $a * a$.

Example 1. Consider the following STG:

$$G = (\{E, T, F\}, \{a, +, *\}, \{q_1, \ldots, q_6\},$$
$$\{q_2, q_4, q_6\}, \vdash_G, P, E),$$
$$P = \{E \to q_1,\ T \to q_3,\ F \to q_5\}$$

with the following transitions (for all β):

$$(q_1, \beta) \vdash_G (q_2, \beta T), \quad (q_2, \beta) \vdash_G (q_1, \beta +),$$
$$(q_3, \beta) \vdash_G (q_4, \beta F), \quad (q_4, \beta) \vdash_G (q_3, \beta *),$$
$$(q_5, \beta) \vdash_G (q_6, \beta a).$$

Table 2 shows some valid parse items for the recognition of the string $a * a$, together with the conditions according to Theorem 1.

4 Bidirectional Parsing

STG's describe the recognition of admissible sequences of subconstituents in unidirectional parsing algorithms, like Earley's algorithm. Bidirectional parsing strategies, e.g., head-corner strategies, start the recognition of a sequence of subconstituents at some position in the middle of the sequence and proceed to both sides. We can define appropriate STG's for bidirectional parsing strategies as follows.

Definition 3. *A headed, bidirectional STG G is like an STG except that P is a finite set of productions of the form $A \to (\Gamma, X, \Lambda)$, where $A \in N$ and $X \in V$ and $\Gamma, \Lambda \in \mathcal{M}$.*

The two states in a production account for the bidirectional expansion of a constituent. The derivation relation for a headed, bidirectional STG is defined by $\gamma A \delta \Longrightarrow \gamma \beta^l X \beta^r \delta$ iff for some production $A \to (\Gamma, X, \Lambda)$: $(\beta^l)^{-1} \in L(\Gamma)$ and $\beta^r \in L(\Lambda)$ ($(\beta^l)^{-1}$ denotes the inversion of β^l). Note that Γ defines the left part of an admissible sequence from right to left.

A bottom-up head-corner parsing schema uses items of the form $[A \to \Gamma \bullet \beta \bullet \Lambda, i, j]$ (Schneider, 2000). The semantics of these items is given by the following clauses:

- for some production $A \to (\Gamma_0, X, \Lambda_0)$, some β^l, β^r: $\beta = \beta^l X \beta^r$ and $(\Gamma_0, \varepsilon) \vdash_G (\Gamma, (\beta^l)^{-1})$ and $(\Lambda_0, \varepsilon) \vdash_G (\Lambda, \beta^r)$.
- $\beta \Longrightarrow^* a_{i+1} \ldots a_j$.

5 Local Tree Constraints

In this section we discuss the usability of STG's for the design of direct parsing algorithms for grammars that use a set of well-formedness conditions, or constraints, expressed in a logical language, to define the admissible syntactic structures (i.e., trees), in contrast to grammars that are based on a derivation mechanism

(i.e., production rules). Declarative characterizations of syntactic structures provide a means to formalize grammatical frameworks, and thus to compare theories expressed in different formalisms. There are also applications in theoretical explorations of the complexity of linguistic theories, based on results which relate language classes to definability of structures in certain logical languages (Rogers, 2000).

From a model-theoretic point of view, such a grammar is an axiomatization of a class of structures, and a well-formed syntactic structure is a model of the grammar (Blackburn et al., 1993). The connection between models and strings is established via a yield function, which assigns each syntactic structure a string of terminal symbols. The parsing problem can then be stated as the problem: Given a string w and a grammar G, find the models $\mathcal{M}$ with $\mathcal{M} \models G$ and $\mathit{yield}(\mathcal{M}) = w$.

In many cases, there are effective methods to translate logical formulae into equivalent tree automata (Rogers, 2000) or rule-based grammars (Palm, 1997). Thus, a possible way to approach the parsing problem is to translate a set of tree constraints into a grammar and use standard parsing methods. However, depending on the expressive power of the logical language, the complexity of the translation often limits this approach in practice.

In this section, we consider the possibility to apply tabular parsing methods directly to grammars that consist of sets of tree constraints. The idea is to interleave the translation of formulae into production rules with the recognition of subconstituents. It should be noted that this approach suffers from the same complexity limitations as the pure translation.

In Schneider (1999), we used a fragment of a propositional bimodal language to express local constraints on syntactic structures. The two modal operators $\langle\downarrow\rangle$ and $\langle\rightarrow\rangle$ refer to the leftmost child and the right sibling, respectively, of a node in a tree. Furthermore, the nesting of $\langle\downarrow\rangle$ is limited to depth one. A so-called *modal grammar* consists of a formula that represents the conjunction of a set of constraints that must be satisfied at every node of a tree. In addition, a second formula represents a condition for the root of a tree.

In Schneider (1999), we have also shown how an extension of a standard method for automatic proof search in modal logic (so-called *analytic labelled tableaux*) in conjunction with dynamic programming techniques can be employed to parse input strings according to a modal grammar. Basically, a labelled tableau procedure is used to construct a labelled tableau, i.e., a tree labelled with formulae, by breaking formulae up into subformulae; this tableau may then be used to construct a model for the original formula. The extended tableau procedure constructs an infinite tableau that allows to obtain all admissible trees (i.e., models of the grammar).

The approach can be described as follows: An STG is defined by using certain formulae that appear on the tableau as states, and by defining the transition relation in terms of the tableau rules (i.e., the operations that are used to construct a tableau). The states are formulae of the form

$$X \wedge \bigwedge\langle\downarrow\rangle Q \wedge \bigwedge[\downarrow]Q' \wedge \bigwedge\langle\rightarrow\rangle R \wedge \bigwedge[\rightarrow]R'$$

where X is a propositional variable and $[\downarrow], [\rightarrow]$ are the dual operators to $\langle\downarrow\rangle, \langle\rightarrow\rangle$. X is used as a node label in a tree model. The transition relation can be regarded as a simulation of the application of tableau rules to formulae, and a tabular parser for this STG can be viewed as a tabulation of the (infinite) tableau construction. In particular, it should be noted that this construction makes no reference to any particular parsing strategy.

6 Conclusion

We have defined state transition grammars (STG) as an intermediate formalism between grammars and parsing algorithms. They complement the parsing schemata formalism of Sikkel (1993). A parsing schema abstracts from unimportant algorithmic details and thus, like STG's, represents a well-defined level of abstraction between grammars and parsers. STG's add another abstraction to parsing schemata, namely on the grammar side. Therefore, we argued, a parsing schema defined over a STG represents a very high level description of a tabular parsing algorithm that can be applied to various grammar formalisms. In this paper we concentrated on grammar formalisms with a flexible constituent structure, i.e., where the

number and order of subconstituents specified by a grammar production may not be fixed. In particular, we have discussed extended context-free grammars (ECFG), ID/LP grammars, and grammars in which admissible trees are defined by means of local tree constraints expressed in a simple logical language.

References

Patrick Blackburn, Claire Gardent, and Wilfried Meyer-Viol. 1993. Talking about trees. In *Proc. 5th Conference of the European Chapter of the Association for Computational Linguistics (EACL'93)*, pages 21–29.

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

Gerald Gazdar, Evan H. Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1985. *Generalized Phrase Structure Grammar*. Blackwell, Oxford.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, Amsterdam.

Ronald M. Kaplan and Joan Bresnan. 1982. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*, chapter 4, pages 173–281. MIT Press, Cambridge, MA.

René Leermakers. 1989. How to cover a grammar. In *Proc. 27th Annual Meeting of the Association for Computational Linguistics (ACL'89)*, pages 135–142.

Frank Morawietz. 1995. A unification-based ID/LP parsing schema. In *Proc. 4th Int. Workshop on Parsing Technologies (IWPT'95)*, Prague.

Adi Palm. 1997. *Transforming Tree Constraints into Formal Grammar*. Infix, Sankt Augustin.

Fernando C. N. Pereira and David H. D. Warren. 1983. Parsing as deduction. In *Proc. 21st Annual Meeting of the Association for Computational Linguistics (ACL'83)*, pages 137–144.

Paul Walton Purdom, Jr. and Cynthia A. Brown. 1981. Parsing extended $LR(k)$ grammars. *Acta Informatica*, 15:115–127.

James Rogers. 2000. wMSO theories as grammar formalisms. In *Proc. of 16th Twente Workshop on Language Technology: Algebraic Methods in Language Processing (TWLT 16/AMiLP 2000)*, pages 201–222, Iowa City, Iowa.

Karl-Michael Schneider. 1999. An application of labelled tableaux to parsing. In Neil Murray, editor, *Automatic Reasoning with Analytic Tableaux and Related Methods*, pages 117–131. Tech. Report 99-1, SUNY, N.Y.

Karl-Michael Schneider. 2000. Algebraic construction of parsing schemata. In *Proc. 6th Int. Workshop on Parsing Technologies (IWPT 2000)*, pages 242–253, Trento.

Roland Seiffert. 1991. Unification-ID/LP grammars: Formalization and parsing. In Otthein Herzog and Claus-Rainer Rollinger, editors, *Text Understanding in LILOG*, LNAI 546, pages 63–73. Springer, Berlin.

Stuart M. Shieber. 1984. Direct parsing of ID/LP grammars. *Linguistics and Philosophy*, 7(2):135–154.

Klaas Sikkel. 1993. *Parsing Schemata*. Proefschrift, Universiteit Twente, CIP-Gegevens Koninklijke Bibliotheek, Den Haag.

Klaas Sikkel. 1998. Parsing schemata and correctness of parsing algorithms. *Theoretical Computer Science*, 199(1–2):87–103.

William A. Woods. 1973. An experimental parsing system for transition network grammars. In Randall Rustin, editor, *Natural Language Processing*, pages 111–154. Algorithmic Press, New York.