Proceedings of VEXTAL'99 - Venezia, San Servolo, V.I.U. - 22-24 November 1999

Modular Specification of Chart Parsing Algorithms Using Abstract Parsing Schemata

Karl-Michael Schneider*

Abstract

Parsing schemata are high-level descriptions of parsing algorithms. This paper is concerned with parsing schemata for different grammar formalisms. We separate the description of parsing steps from that of grammatical properties by means of abstract parsing schemata. We define an abstract Earley schema and prove it correct. We obtain Earley schemata for several grammar formalisms by specifying the grammatical properties in the abstract Earley schema. Our approach offers a clear and well-defined interface between a parsing algorithm and a grammar. Moreover, it provides a precise criterion for the classifaction of parsing algorithms.

Keywords: parsing schema, specification, context-free transition grammar

1 Introduction

Parsing schemata have been proposed mainly by Sikkel [11] as a means for specifying and comparing parsing algorithms for context-free grammars (CFG). A parsing schema describes the steps that a parsing algorithm takes in order to find the derivations of an input string. Formally, a parsing schema is a deduction system $(I, \vdash)$ where I is a set of *parse items* or simply *items*, and $\vdash$ is a *deduction relation*. For example, Fig. 1 shows the parsing schema for Earley's CFG parsing algorithm [2], where $G = (N, \Sigma, P, S)$ is a CFG and $w = a_1 \dots a_n \in \Sigma^*$ is an input string. Intuitively, an item $[A \to \beta \bullet \beta', i, j]$ can be read as saying that a constituent of type A starts at position i (that is, with a_{i+1}) and has been partially recognized until position j (that is, it covers $a_{i+1} \dots a_j$), where β is the recognized part and β' has yet to be recognized. More precisely, $[A \to \beta \bullet \beta', i, j]$ is deducible, notation: $\vdash^* [A \to \beta \bullet \beta', i, j]$, iff for some $\delta \in (N \cup \Sigma)^*$: $S \underset{G}{\Longrightarrow}^* a_1 \dots a_i A \delta$ and $\beta \underset{G}{\Longrightarrow}^* a_{i+1} \dots a_j$. Therefore, $\vdash^* [S \to \gamma \bullet, 0, n]$ iff $S \underset{G}{\Longrightarrow}^* w$; that is, iff $w \in L(G)$.

A parsing schema leaves the order in which parsing steps are performed unspecified. A *chart parser* can be seen as the canonical implementation of a parsing schema [12].

Within the last 35 years several parsing algorithms for CFG have been developed, and parsing schemata constitute a well-defined level of abstraction for the description and comparison of these algorithms [11]. At the same time, on the other hand, a number of new grammar formalisms were conceived, some of which are considered later in this paper. The general goal was always to have a more adequate device for the description of natural languages while at the same time retaining the polynomial time complexity of the parsing problem. For most of these grammars polynomial time parsing algorithms have been presented, usually by adapting algorithms for CFG, but in most cases the relation between a parsing algorithm for CFG and its adaption is stated only informally [10, 14]. This means that a correctness proof for the adapted algorithm must be made from scratch.

In [11] the relations between different parsing schemata for CFG are made precise. This paper aims at establishing a precise relation between parsing schemata for different (but related) grammar formalisms, in particular one that is *correctness preserving* (that is, when a parsing schema for CFG is adapted to a different grammar formalism, its correctness is preserved). Besides CFG, we consider the ID/LP (*Immediate Dominance, Linear Precedence*) grammar format in which vertical and horizontal aspects of syntactic structure are treated separately [4], and ECFG (*Extended Context-Free Grammars*), a generalization of CFG where the generated languages are still context-free [13]. The ID/LP format is used in *Generalized Phrase-Structure Grammar* (GPSG) [4] while ECFG appear, in a restricted form, in *Lexical-Functional Grammar* (LFG) [7].

In order to relate parsing schemata for different grammar formalisms in a precise way, we present *abstract parsing schemata*; that is, parsing schemata that abstract away from grammatical properties as much as possible. An abstract parsing schema is a parsing schema that leaves the form of the grammar unspecified except for some minimum requirements. A parsing schema for a particular grammar is obtained simply by providing an appropriate specification of the grammatical properties in the abstract

*Department of General Linguistics, University of Passau, Innstr. 40, D-94032 Passau, schneide@phil.uni-passau.de

$$I = \{[A \to \beta\bullet\beta', i, j] \mid A \to \beta\beta' \in P,\ 0 \le i \le j \le n\}$$
$$\vdash [S \to \bullet\gamma, 0, 0] \quad (Init)$$
$$[A \to \beta\bullet B\beta', i, j] \vdash [B \to \bullet\gamma, j, j] \quad (Predict)$$
$$[A \to \beta\bullet a_{j+1}\beta', i, j] \vdash [A \to \beta a_{j+1}\bullet\beta', i, j+1] \quad (Scan)$$
$$[A \to \beta\bullet B\beta', i, j], [B \to \gamma\bullet, j, k] \vdash [A \to \beta B\bullet\beta', i, k] \quad (Compl)$$

Figure 1: The Earley parsing schema.

parsing schema. This process is called *specialization*. In this way we achieve a specification of a parsing algorithm that is modular in the sense that the steps taken by the algorithm when computing derivations are specified independently from its interface to the grammar formalism.

The correctness of a parsing schema can be proven on the level of the abstract parsing schema, and the correctness of any parsing schema derived from it by specialization follows immediately from the construction. Besides, this method yields also a precise criterion for the classification of parsing schemata: A parsing schema for some grammar formalism is an Earley schema precisely if it can be obtained from the abstract Earley parsing schema by specialization. Other so-called "Earley" parsing schemata (e.g. [1]) consist of modified deduction steps and should be called "Earley-like" parsing schemata.

The paper is organized as follows: In Sect. 2 we consider the grammar formalism dependent properties of different Earley schemata. In Sect. 3 we define *context-free transition grammars* (CFTG) as an abstraction of the format of context-free productions and give some examples of specializations. In Sect. 4 we define an Earley schema for CFTG and prove it correct. This is the abstract Earley parsing schema. Finally, an application of the technique to a grammar that is based on descriptions of syntactic structures (as opposed to derivations) is discussed in Sect. 5. Earley's algorithm is used as an example throughout the paper, but the technique can be used to give abstract descriptions of other parsing algorithms as well.

2 Grammar Formalism Dependent Properties

In order to find out which parts of a parsing schema are grammar formalism dependent, consider two other Earley parsing schemata, namely for *Immediate Dominance, Linear Precedence* (ID/LP) grammars [4] (Fig. 2) and Extended Context-Free Grammars (ECFG) [13] (Fig. 3). In Fig. 2, $G = (N, \Sigma, P, LP, S)$ is an ID/LP grammar where P is a set of immediate dominance productions of the form $A \to M$ with M a multiset over $N \cup \Sigma$. $A \to M$ encodes the set of all CF productions of the form $A \to X_1 \ldots X_k$ with $\{X_1, \ldots, X_k\} = M$. LP is a set of linear precedence constraints. For simplicity we represent LP as a subset of $(N \cup \Sigma)^*$. A string β is wellformed with respect to the LP constraints iff $\beta \in LP$. An ID/LP Earley algorithm was presented in [10]. A partial constituent, represented by an item $[A \to \beta\bullet M, i, j]$, is expanded as follows: a symbol $B \in M$ is chosen and B is removed from M and apppended to the string β, provided βB satisfies the LP constraints. This yields the item $[A \to \beta B\bullet M \setminus \{B\}, i, k]$.

An Earley parsing schema for ECFG is shown in Fig. 3. ECFG are a generalization of CFG where the right-hand side of a production defines a regular language[1] over $N \cup \Sigma$. In Fig. 3 the right-hand sides of productions are represented as finite automata. A finite automaton with input alphabet $N \cup \Sigma$ is a tuple $(N \cup \Sigma, Q, q_0, \delta, Q_f)$, with state set Q, initial state q_0, final (or accepting) states Q_f, and transition relation $\delta \subseteq Q \times (N \cup \Sigma) \times Q$ [15]. Without loss of generalization, there are no ε-transitions. We can, without loss of generalization, assume that the automata in the right-hand sides are all disjoint; thus let Q be the set of all states of all automata, Q_f the set of all final states, δ the (disjoint) union of all transition relations, and assume that productions are of the form $A \to q_0$ where q_0 is an initial state. A partial constituent, represented by an item $[A \to \beta\bullet q, i, j]$ where q is a state, is expanded by appending a symbol B to β only if there is a transition from q to some state q' that consumes B; that is, if $(q, B, q') \in \delta$. This yields $[A \to \beta B\bullet q', i, k]$.

The schemata in Figs. 1, 2 and 3 are quite similar. In fact, an *Earley item* is always of the form $[A \to \beta\bullet\Gamma, i, j]$ where A is a (possibly complex) symbol, β is a string of symbols and $0 \le i \le j$. The symbols in β are the children of a partial constituent that this item represents. Constituents are built by successively appending new children to β. The *Init* step deduces an Earley item $[A \to \bullet\Gamma, 0, 0]$ (where β is empty) where $A \to \Gamma$ is a production. Γ can be regarded as an *agenda*, specifying possible ways

1. In [13] the language defined by a right-hand side can even be context-free.

$$I = \{[A \to \beta \bullet M, i, j] \mid \beta = X_1 \ldots X_k,\ A \to M \cup \{X_1, \ldots, X_k\} \in P\}$$
$$\vdash [S \to \bullet M, 0, 0] \quad (Init)$$
$$[A \to \beta \bullet M \cup \{B\}, i, j] \vdash [B \to \bullet M', j, j] \text{ iff } \beta B \in LP \quad (Predict)$$
$$[A \to \beta \bullet M \cup \{a_{j+1}\}, i, j] \vdash [A \to \beta a_{j+1} \bullet M, i, j+1] \text{ iff } \beta a_{j+1} \in LP \quad (Scan)$$
$$[A \to \beta \bullet M \cup \{B\}, i, j], [B \to \gamma \bullet \emptyset, j, k] \vdash [A \to \beta B \bullet M, i, k] \text{ iff } \beta B \in LP \quad (Compl)$$

Figure 2: The ID/LP Earley parsing schema.

$$I = \{[A \to \beta \bullet q, i, j] \mid A \in N,\ q \in Q,\ 0 \leq i \leq j \leq n\}$$
$$\vdash [S \to \bullet q_0, 0, 0] \text{ iff } S \to q_0 \in P \quad (Init)$$
$$[A \to \beta \bullet q, i, j] \vdash [B \to \bullet q_0, j, j] \text{ iff } B \to q_0 \in P, \exists q' :\ (q, B, q') \in \delta \quad (Predict)$$
$$[A \to \beta \bullet q, i, j] \vdash [A \to \beta a_{j+1} \bullet q', i, j+1] \text{ iff } (q, a_{j+1}, q') \in \delta \quad (Scan)$$
$$[A \to \beta \bullet q, i, j], [B \to \gamma \bullet q_f, j, k] \vdash [A \to \beta B \bullet q', i, k] \text{ iff } q_f \in Q_f,\ (q, B, q') \in \delta \quad (Compl)$$

Figure 3: The Earley parsing schema for ECFG.

to continue a partial constituent. This Γ is the only grammar formalism specific part of an Earley item. The realizations for Γ for CFG, ID/LP grammars and ECFG are shown in Table 1. The second item in a *Complete* step represents a complete constituent. It is called a *complete item*. Realizations for Γ in complete items are also shown in Table 1. The *Predict*, *Scan* and *Complete* steps all involve obtaining a symbol B from the agenda Γ as well as a new agenda Γ'. The conditions for this transition are also shown in the table.

3 Context-Free Transition Grammars

In order to separate the description of parsing steps from that of grammatical properties we define *context-free transition grammars* (CFTG) that describe the propagation of information in the Earley items on the greatest possible level of abstraction. In Sect. 4 we define an Earley parsing schema for CFTG. Every true Earley parsing schema for any grammar can be regarded as a particular instance of that CFTG Earley schema.

In the previous section we have seen which information is contained in an Earley item and how it is propagated between items:

- productions are of the form $A \to \Gamma$ where Γ can be left completely unspecified.
- for some instances of Γ, an Earley item $[A \to \beta \bullet \Gamma, i, j]$ represents a complete constituent.
- there is some way to make a transition from Γ to a new Γ' that may depend on β and yields a symbol B (see Table 1).

Note that Γ' need not be uniquely determined, as for example in the case of ID/LP grammars and ECFG. Also, there may not be a transition at all from some Γ. Γ will be called a *state*. If $[A \to \beta \bullet \Gamma, i, j]$ represents a complete constituent then Γ is called a *final state*. A state together with a string β is called a *configuration*. Observe that the right-hand side of a dotted production in an Earley item $[A \to \beta \bullet \Gamma, i, j]$ is a configuration. We denote configurations with (Γ, β). Actually, we consider transitions not between two states Γ and Γ' but between two configurations, so we can include the conditions on β and B in the transition relation. Therefore, a transition is denoted as $(\Gamma, \beta) \vdash_c (\Gamma', \beta B)$ where $\vdash_c$ is the transition relation (the subscript c distinguishes it from the deduction relation in a parsing schema). A construction of a constituent consists of a sequence of items

$$[A \to \bullet \Gamma_0, i, i] \vdash [A \to X_1 \bullet \Gamma_1, i, j_1] \vdash [A \to X_1 X_2 \bullet \Gamma_2, i, j_2] \vdash \ldots$$

With this sequence we associate a sequence of configurations

$$(\Gamma_0, \varepsilon) \vdash_c (\Gamma_1, X_1) \vdash_c (\Gamma_2, X_1 X_2) \vdash_c \ldots$$

A construction of a constituent (and the associated sequence of configurations) may end when the constituent is complete, i.e. when Γ_n is a final state.

Observe that a configuration (Γ, β) looks like a configuration (or *instaneous description*) of a finite automaton, where Γ is a state of the automaton and β is the remaining input [15]. $\vdash_c$ is similar to the transition relation between instaneous descriptions of a finite automaton, except that each transition

	CFG	ID/LP	ECFG
Γ	string	multiset	state
Γ (complete item)	ε	$\emptyset$	final state
$\beta\bullet\Gamma \vdash \beta B\bullet\Gamma'$	$\Gamma = B\Gamma'$	$\Gamma = \Gamma' \cup \{B\}$, $\beta B \in LP$	$(\Gamma, B, \Gamma') \in \delta$

Table 1: Grammar formalism specific parts of the Earley parsing schema.

generates a new symbol instead of consuming a symbol from the input. Note also that we do not consider a transition relation between states and symbols, i.e. of the form $\delta \subseteq \mathcal{M} \times V \times \mathcal{M}$, where $\mathcal{M}$ is a set of states and V is a set of symbols—instead the transition relation between configurations is considered to be basic. In finite automata the transition relation between instaneous descriptions is derived from that between states.

Let $\mathcal{M}$ be a set of states and $\mathcal{M}_F$ a set of final states. The reflexive and transitive closure of $\vdash_c$ is denoted with $\vdash_c^*$; that is, $(\Gamma, \beta) \vdash_c^* (\Gamma', \beta')$ iff for some $n \geq 0$, $\beta' = \beta X_1 \dots X_n$ and there are states $\Gamma_1, \dots, \Gamma_{n-1} \in \mathcal{M}$ such that

$$(\Gamma, \beta) \vdash_c (\Gamma_1, \beta X_1) \vdash_c \dots \\ \dots \vdash_c (\Gamma_{n-1}, \beta X_1 \dots X_{n_1}) \vdash_c (\Gamma', \beta').$$

Sequences of transitions *generate* strings in the following sense: A state $\Gamma \in \mathcal{M}$ generates a string β if there is a transition sequence from (Γ, ε) to (Γ', β) where Γ' is a final state (similarly, a finite automaton accepts a string β precisely if there is a sequence of transitions from the intial state to an accepting state that consumes β entirely). The *language* defined by a state Γ is the set of all strings generated by Γ:

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

In a *context-free transition grammar* (CFTG) the productions substitute a state (rather than a string) for a symbol. As a further generalization we let a CFTG have a set of start symbols rather than a single start symbol.

Definition 1 (CFTG). *A CFTG 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,*
- *$\mathcal{M} \subseteq \mathcal{M}_F$ is a set of final states,*
- *$\vdash_G \subseteq (\mathcal{M} \times (N \cup \Sigma))^2$ is a binary relation of the form $(\Gamma, \beta) \vdash_G (\Gamma', \beta B)$,*
- *$P \subseteq N \times \mathcal{M}$ is a set of productions written as $A \to \Gamma$ and*
- *$S \subseteq N$ is a set of start symbols.*

The state Γ in a production $A \to \Gamma$ can be regarded as an initial state. In a CFTG derivation $A \to \Gamma$ is applied to a string by substituting some string $\beta \in L(\Gamma)$ for A. Let $V = N \cup \Sigma$. The *derivation relation* $\underset{G}{\Longrightarrow} \subseteq V^* \times V^*$ is defined by $\gamma A \delta \underset{G}{\Longrightarrow} \gamma\beta\delta$ iff for some production $A \to \Gamma \in P$, $\beta \in L(\Gamma)$. $\underset{G}{\Longrightarrow}^*$ is the reflexive and transitive closure of $\underset{G}{\Longrightarrow}$. The *language* defined by a CFTG is the set of strings of terminal symbols that are derivable from some start symbol:

$$L(G) = \{w \in \Sigma^* \mid \exists A \in S : \ A \underset{G}{\Longrightarrow}^* w\}.$$

A CFTG is said to *represent* a grammar G' (where G' is a CFG, ID/LP grammar, etc.) iff G and G' have the same derivation trees. A CFTG G that represents a grammar G' is called a *CFTG representation* of G'. For the following examples we refer the reader to the considerations in Sect. 2 and, in particular, to Table 1.

Example 1. Let $G = (N, \Sigma, P, S)$ be a CFG without ε-productions. Then $(N, \Sigma, \mathcal{M}, \mathcal{M}_F, \vdash_G, P, \{S\})$ is a CFTG representation of G where $\mathcal{M} = \{\beta' \mid \exists \beta : A \to \beta\beta' \in P\}$, $\mathcal{M}_F = \{\varepsilon\}$ and $(X\beta', \beta) \vdash_G (\beta', \beta X)$ iff $X \in N \cup \Sigma$ and $X\beta' \in \mathcal{M}$.

Example 2. Let $G = (N, \Sigma, P, LP, S)$ be an ID/LP grammar where $LP \subseteq (N \cup \Sigma)^*$ (cf. Sect. 2). Then $(N, \Sigma, \mathcal{M}, \mathcal{M}_F, \vdash_G, P, \{S\})$ is a CFTG representation of G where $\mathcal{M} = \{M \mid \exists A \to M' \in P, \ M \subseteq M'\}$, $\mathcal{M}_F = \{\emptyset\}$ and $(M, \beta) \vdash_G (M', \beta X)$ iff $M = M' \cup \{X\}$ and $\beta X \in LP$.

Example 3. Let $G = (N, \Sigma, P, S)$ be an ECFG (cf. Sect. 2). Let Q be the set of all states of all automata in G, Q_f the set of all final states and $\delta \subseteq Q \times (N \cup \Sigma) \times Q$ the (disjoint) union of the transition relations. Then $(N, \Sigma, Q, Q_f, \vdash_G, P, \{S\})$ is a CFTG representation of G where $(q, \beta) \vdash_G (q', \beta X)$ iff $(q, X, q') \in \delta$.

4 An Earley Parsing Schema for CFTG

An Earley parsing schema for a CFTG is shown in Fig. 4. In an item $[A \to \beta\bullet\Gamma, i, j]$, the state Γ encodes the information about how a partial constituent can be expanded. In order to give a precise definition of

$$I = \{[A \to \beta \bullet \Gamma, i, j] \mid A \in N,\ \beta \in (N \cup \Sigma)^*,\ \Gamma \in \mathcal{M},\ 0 \le i \le j \le n\}$$

$$\vdash [S \to \bullet \Gamma, 0, 0] \text{ iff } S \to \Gamma \in P \quad (Init)$$

$$[A \to \beta \bullet \Gamma, i, j] \vdash [B \to \bullet \Gamma_0, j, j] \text{ iff } \exists \Gamma' :\ (\Gamma, \beta) \vdash_G (\Gamma', \beta B) \quad (Predict)$$

$$[A \to \beta \bullet \Gamma, i, j] \vdash [A \to \beta a_{j+1} \bullet \Gamma', i, j+1] \text{ iff } (\Gamma, \beta) \vdash_G (\Gamma', \beta a_{j+1}) \quad (Scan)$$

$$[A \to \beta \bullet \Gamma, i, j], [B \to \gamma \bullet \Gamma_f, j, k] \vdash [A \to \beta B \bullet \Gamma', i, k] \text{ iff } \Gamma_f \in \mathcal{M}_F,\ (\Gamma, \beta) \vdash_G (\Gamma', \beta B) \quad (Compl)$$

Figure 4: The Earley parsing schema for CFTG.

the semantics of the items we need a more general notion of derivation that is capable of describing the derivation of partial constituents. A *partial derivation step* is a derivation step where a symbol A in a string is replaced with a configuration (Γ, β) such that for some production $A \to \Gamma'$: $(\Gamma', \varepsilon) \vdash_G^* (\Gamma, \beta)$. If Γ is a final state, it can be discarded, so that in this case we obtain the usual derivation relation $\underset{G}{\Longrightarrow}$.

Note that the Earley algorithm builds constituents always from left to right. By induction on the deduction steps in Fig. 4 it follows that partial derivation steps are performed only on rightmost symbols in a string. Therefore we can collect all the states that are introduced by partial derivation steps in a sequence of states; that is, we consider pairs (γ, Δ) where Δ is a sequence of states. A pair (γ, Δ) is called a *super configuration*.[2] When a partial derivation step is performed on a sequence $(\gamma A, \Delta)$ where A is going to be replaced with (Γ, β), Γ is tacked to the left edge of Δ, so we obtain $(\gamma\beta, \Gamma\Delta)$. If Γ is a final state we can also obtain $(\gamma\beta, \Delta)$.

Non-rightmost symbols can only be replaced with strings using the conventional derivation relation. The extended derivation relation is also denoted with $\underset{G}{\Longrightarrow}$. Which one is used is always clear from the context.

Let $G = (N, \Sigma, m, P, S)$. The extended derivation relation is defined by the clauses:

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

Theorem 1. *1.* $(\gamma, \Delta) \underset{G}{\Longrightarrow}^* (\gamma', \Delta)$ *iff* $\gamma \underset{G}{\Longrightarrow}^* \gamma'$.

2. $w \in L(G)$ *iff for some* $A \in S$: $(A, \varepsilon) \underset{G}{\Longrightarrow}^* (w, \varepsilon)$.

Proof. 1. follows from the definitions by induction on the length of derivations. 2. is a direct consequence of 1. □

The following theorem is given without proof; the proof is similar to the correctness proof given in [12].

Theorem 2 (Correctness). $\vdash^* [A \to \beta \bullet \Gamma, i, j]$ *in the parsing schema in Fig. 4 iff the conditions are satisfied:*

- *for some* $A_0 \in S$, *for some* Δ: $(A_0, \varepsilon) \underset{G}{\Longrightarrow}^* (a_1 \dots a_i A, \Delta)$.
- $(A, \varepsilon) \underset{G}{\Longrightarrow} (\beta, \Gamma)$.
- $\beta \underset{G}{\Longrightarrow}^* a_{i+1} \dots a_j$.

5 Further Applications

The ideas expressed in linguistic theories put forward in the 1980s by Noam Chomsky, the so-called *Theory of Government & Binding* [5] suggest that syntactic structures of natural languages should be *described* rather than *derived*. This entails that syntactic structures should not be seen as derivation trees of production-based grammars but as structures in some appropriate class (e.g. trees over some appropriate label domain) that satisfy certain wellformedness conditions; the latter constitute the grammar.

From a model-theoretic point of view, wellformedness conditions are just formulas in a logical language, and a wellformed syntactic structure is a model of the grammar. The connection between models (syntactic structures) and strings (sentences) is established via a yield function[3]. 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 $yield(\mathcal{M}) = w$.

In order to find parsing algorithms for a grammar G that consists of logical formulas, there are basically two possible ways:

- transform G into an equivalent grammar G' with productions, where G and G' are considered equivalent iff the derivation trees of G' are precisely the models of G; then use a parsing algorithm for G', or
- direct parsing of G.

The first alternative has been investigated, for example, in [8]. The second alternative is still an

2. There is some connection here to the *item push-down automaton* of a CFG [15].

3. The yield of a tree is usually defined to be the sequence of its leaves in left to right order.

open problem. However, provided that the logical language in which wellformedness conditions are expressed is sufficiently restricted, we can find an Earley parsing schema by providing a CFTG representation for G. In [9] it is shown how a CFTG representation for a grammar consisting of wellformedness conditions in a restricted multimodal language can be found using standard methods for automatic proof search in modal logic (so-called analytic labelled tableaux).

Notice that a CFTG representation is different from a translation into an equivalent production-based grammar: A CFTG representation of a modal logic grammar simulates the tableau rules in a tableau construction. Therefore a parsing schema for a CFTG representation of a modal logic grammar G can be regarded as an *online* translation of G into an equivalent production-based grammar (that is, the translation is performed during the parsing process).

6 Conclusion

Abstract parsing schemata based on CFTG separate the description of parsing steps from that of grammatical properties, and thus offer a clear and well-defined interface between a parsing algorithm and a grammar. Therefore abstract parsing schemata should be seen as a step towards simplifying the design and development of parsing algorithms. Moreover, they provide a criterion for a precise classification of parsing algorithms.

There are, however, clear limitations for the applicability of the method: We can obtain parsing schemata by specialization from an abstract parsing schema for all grammar formalisms that have a CFTG representation. This excludes some grammar formalisms that have become very popular for the description of natural languages, such as linear indexed grammars [3] and tree adjoining grammars [6]. The development of parsing schemata for these grammar formalisms is a nontrivial problem.

References

[1] Víctor J. Díaz, Vicente Carrillo, and Miguel Toro. A review of earley-based parser for TIG. In *Proc. IEA-98-AIE*, LNCS 1415, pages 732–738, 1998.

[2] Jay Earley. An efficient context-free parsing algorithm. *Comm. ACM*, 13:2:94–102, 1970.

[3] Gerald Gazdar. Applicability of indexed grammars to natural languages. Tech. Rep. CSLI-85-34, Center for Study of Language and Information, Stanford, 1985.

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

[5] James Higginbotham. GB theory: An introduction. In Johan van Benthem and Alice ter Meulen, editors, *Handbook of Logic and Language*, chapter 5, pages 311–360. Elsevier, Amsterdam, 1997.

[6] A. K. Joshi, L. S. Levy, and M. Takahashi. Tree adjoining grammars. *Journal of Computer and System Science*, 10(1):136–163, 1975.

[7] Ronald M. Kaplan and Joan Bresnan. Lexical-functional grammar: A formal system for grammatical representation. In Joan Bresnan, editor, *The Mental Representation of Grammatical Relations*. MIT Press, Cambridge, Mass, 1982.

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

[9] Karl-Michael Schneider. 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., 1999.

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

[11] Klaas Sikkel. *Parsing Schemata. A framework for specification and analysis of parsing algorithms*. Springer, Berlin, 1997.

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

[13] J. W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Science*, 1:317–322, 1967.

[14] K. Vijay-Shanker and David J. Weir. Polynomial parsing of extensions of context-free grammars. In Masaru Tomita, editor, *Current Issues in Parsing Technology*, pages 191–206. Kluwer, Dordrecht, 1991.

[15] Reinhard Wilhelm and Dieter Maurer. *Übersetzerbau: Theorie, Konstruktion, Generierung*. Springer, Berlin, 1992.