A framework for parsing algorithm specification and analysis*

Klaas Sikkel

Gesellschaft für Mathematik und Datenverarbeitung,
Institut für angewandte Informationstechnik (GMD-FIT),
D-53754 Sankt Augustin,
sikkel@gmd.de

Abstract. *Parsing schemata* are defined as an intermediate level of abstraction between context-free grammars and parsers. Clear, concise specifications of radically different parsing algorithms can be expressed as parsing schemata. Moreover, because of the uniformity of these specifications, relations between different parsing algorithms can be formally established. The framework is introced in an informal manner.

Zusammenfassung. *Parsing schemata* bilden eine Abstraktionsebene zwischen kontextfreien Grammatiken und Parsers. Von unterschiedlichsten parsing Algorithmen kann eine kurze aber deutliche Spezifikation gegeben werden mittels einem parsing Schema. Wegen der erreichten Uniformität ist es möglich, Beziehungen zwischen verschiedenen Parsers formal festzulegen. Die Präsentation in diesem Artikel ist informell.

1 Introduction

A wide variety of parsing algorithms can be found in the Computer Science and Computational Linguistics literature. Algorithms differ a lot with respect to languages in which they are expressed, data structures used, degree of formality, class of grammars that can be handled, etc. Things get worse when we consider parallel, rather than sequential algorithms, because many different architectures can be exploited.

One can compare parsers by evaluating run-time performance, but in order to get a deeper insight into the relative merits and deficiencies of different algorithms, one needs a common framework in which they can be described. In this paper we propose a framework that is both sufficiently general to cover a large range of wildly different parser, and sufficiently practical to allow legible specifications of nontrivial algorithms. We introduce the notion of a *parsing schema*. Such a schema describes the basic traits of an algorithm, but abstracts from the technical details needed to make the algoritm work. A schema specifies

- the type of intermediate results used by a parser, i.e., some representation of partial parses,

* This research was carried out at the University of Twente, Department of Computer Science, Enschede, the Netherlands

- steps that allow computation of new intermediate results from given ones.

A parsing schema does *not* specify

- *data structures* for storage and retrieval of such results,
- *control structures* enforcing that all the relevant steps are performed,
- (only for parallel algorithms:) *communication structures* between different processors.

One can see parsing schemata as a separate, well-defined level of abstraction in between grammars and parsing algorithms.

In this paper we give a brief, informal introduction to the parsing schemata framework, largely by means of examples. A more complete, formal treatment can be found in [10]. For the sake of brevity, we consider only context-free parsing. In section 5 a few words will be spent on parsing of unification grammars.

Definitions and some examples of parsing schemata given in Section 2. In 3 we show how different parsing schemata can be related to one another. As a more detailed example, we transform the Earley schema into a Left-Corner schema in Section 4. Related work and various applications of the framework are briefly discussed in 5, conclusions stated in 6.

2 Parsing systems and parsing schemata

The usual notational conventions apply. A grammar is a quadruple $G = (N, \Sigma, P, S)$. We write $A, B, \ldots \in N$ for nonterminals, $a, b, \ldots \in \Sigma$ for terminals, $\alpha, \beta, \ldots \in (N \cup \Sigma)^*$ for strings of arbitrary variables. The letters $i, j, \ldots$ are nonnegative integers.

A parsing schema is defined for a class of grammars, and for all strings over the alphabet of any particular grammar. But in order to simplify things, we will first restrict ourselves to a single given grammar and a single given sentence. For this particular grammar and sentence we define a *parsing system*, which is in fact logical deduction system.

Definition. A *parsing system* $\mathbb{P}$ for some grammar G and string $a_1 \ldots a_n$ is a triple $\mathbb{P} = \langle \mathcal{I}, H, D \rangle$, in which

- $\mathcal{I}$ is a set of items, called the called the *domain* or *item set* of $\mathbb{P}$;
- H is a set of items (not necessarily a subset of $\mathcal{I}$), the *hypotheses* of $\mathbb{P}$;
- D is a set of so-called *deduction steps* of the form

  $$\eta_1, \ldots, \eta_k \vdash \xi,$$

  with $\eta_i \in \mathcal{I} \cup H$ for $0 \leq i \leq k$ and $\xi \in \mathcal{I}$. The items $\eta_1, \ldots, \eta_k$ are called the *antecedents*, ξ the *consequent* of the deduction step. □

Different types of items can be used for parsing systems and schemata underlying different algorithms. Most examples here use the well-known Earley items.

Implicitly specified by a parsing system $\mathbb{P}$ is the set of *valid items* $\mathcal{V}(\mathbb{P})$, that contains all items of $\mathcal{I}$ (and only those) that can be inferred from H, using deduction steps from D. That is, starting from H, one can systematically search for all deduction steps of which the antecedents are known to be valid, and add their consequents to the set of known valid items. It may take an infinite number of steps to obtain $\mathcal{V}(\mathbb{P})$, however.

As a first example, we will show a parsing schema for a bottom-up variant of the Earley algorithm in which top-down prediction has been eliminated.

Example. the parsing system $\mathbb{P}_{buE}$ for a given context-free grammar G and string $a_1 \ldots a_n$ is defined as follows:

$$\begin{aligned}
\mathcal{I}_{buE} &= \{[A \rightarrow \alpha \bullet \beta, i, j] \mid A \rightarrow \alpha\beta \in P \ \wedge \ 0 \leq i \leq j\};\\
H_{buE} &= \{[a, i-1, i] \mid a = a_i, 1 \leq i \leq n\};\\
D^{Init} &= \{\vdash [A \rightarrow \bullet \gamma, i, i]\},\\
D^{Scan} &= \{[A \rightarrow \alpha \bullet a \beta, i, j], [a, j, j+1] \vdash [A \rightarrow \alpha a \bullet \beta, i, j+1]\},\\
D^{Compl} &= \{[A \rightarrow \alpha \bullet B \beta, i, j], [B \rightarrow \gamma \bullet, j, k] \vdash [A \rightarrow \alpha B \bullet \beta, i, k]\},\\
D_{buE} &= D^{Init} \cup D^{Scan} \cup D^{Compl}.
\end{aligned}$$

A few remarks about this specification:

- Deduction steps D^{Init} are needed to start the deduction of further valid items, hence these have no antecedents. In the definition of D^{Init} there is no need to state explicitly that $A \rightarrow \gamma \in P$ is required; the definition of a parsing system imposes that only items from $\mathcal{I}$ and H can be used in the definition of D.
- D^{Scan} and D^{Compl} conform to the *scan* and *complete* steps of Earley's algorithm.
- The definition of the domain does *not* restrict the position markers in items to the case $i, j \leq n$. This has been done on purpose. It has several advantages that the definitions of $\mathcal{I}$ and D do not depend on the string $a_1 \ldots a_n$. The price for this modularity is infinite, rather than finite sets $\mathcal{I}$ and D. But at this level of abstraction that does not pose any problem — we do not *do* anything, we only define possible algorithmic steps — and in any implementation it will be evident that only those items have to be considered, that have position markers within the appropriate range.

It is easy to verify that $\mathcal{V}(\mathbb{P}_{buE}) = \{[A \rightarrow \alpha \bullet \beta, i, j] \mid \alpha \Rightarrow^* a_{i+1} \ldots a_j\}$. □

Top-down prediction, yielding a parsing system for the canonical Earley algorithm, can be included as follows.

Example. The parsing system $\mathbb{P}_{Earley}$ for a given context-free grammar G and string $a_1 \ldots a_n$ is defined by $\mathcal{I}$ and H as in $\mathbb{P}_{buE}$ above and by D_{Earley} as follows:

$$D^{Init} = \{\vdash [S \rightarrow \bullet \gamma, 0, 0]\},$$

$$
\begin{aligned}
D^{Pred} &= \{[A{\rightarrow}\alpha{\bullet}B\beta, i, j] \vdash [B{\rightarrow}{\bullet}\gamma, j, j]\},\\
D^{Scan} &= \{[A{\rightarrow}\alpha{\bullet}a\beta, i, j], [a, j, j+1] \vdash [A{\rightarrow}\alpha a{\bullet}\beta, i, j+1]\},\\
D^{Compl} &= \{[A{\rightarrow}\alpha{\bullet}B\beta, i, j], [B{\rightarrow}\gamma{\bullet}, j, k] \vdash [A{\rightarrow}\alpha B{\bullet}\beta, i, k]\},\\
D_{Earley} &= D^{Init} \cup D^{Scan} \cup D^{Compl} \cup D^{Pred}.
\end{aligned}
$$

Items of the form $[A{\rightarrow}{\bullet}\gamma, i, i]$ can be deduced only if there is a "need" to do so, i.e., an item $[B{\rightarrow}\alpha{\bullet}A\beta, h, i]$ has been found to be valid. The set of valid items is given by

$$\mathcal{V}(\mathbb{P}_{Earley}) = \{[A{\rightarrow}\alpha{\bullet}\beta, i, j] \in \mathcal{V}(\mathbb{P}_{buE}) \mid S{\Rightarrow}^* a_1 \ldots a_i A\gamma \text{ for some } \gamma\}.$$

Top-down filtering reduces the number of items, but also reduces the possibilities for parallel processing. Earley's algorithm is essentially left-to-right. □

A parsing system has been defined for a fixed grammar and string. In two steps we will extend this to a *parsing schema* for arbitrary grammmars and strings.

Definition. An *uninstantiated parsing system* for a grammar G is triple $\langle \mathcal{I}, \mathcal{H}, D\rangle$ with $\mathcal{H}$ a function that assigns a set of hypotheses to each string $a_1 \ldots a_n \in \Sigma^*$, such that $\langle \mathcal{I}, \mathcal{H}(a_1 \ldots a_n), D\rangle$ is a parsing system. □

In all examples it holds that $\mathcal{H}(a_1 \ldots a_n) = \{[a, i-1, i] \mid a = a_i, 1 \leq i \leq n\}$; in the sequel we will no longer explicitly mention the hypotheses.

Definition. A *parsing schema* for some (sub)class of context-free grammars $\mathcal{CG}$ is a function that assigns an uninstantiated parsing system to every grammar $G \in \mathcal{CG}$. □

Examples.
The parsing schema **buE** is defined by $\mathbf{buE}(G)(a_1 \ldots a_n) = \mathbb{P}_{buE}$ as above.
The parsing schema **Earley** is defined by $\mathbf{Earley}(G)(a_1 \ldots a_n) = \mathbb{P}_{Earley}$ as above. □

Both parsing schemata are defined for the entire class of context-free grammars. The following example, the schema **CYK** (after the algorithm of Cocke, Younger and Kasami [13]) is defined only for grammars in Chomsky Normal Form (CNF). That is, all productions are of the form $A{\rightarrow}BC$ or $A{\rightarrow}a$.

Example. We assign a parsing system $\mathbf{CYK}(G)(a_1 \ldots a_n)$ for arbitrary G in CNF, with $\mathcal{I}$ and D defined by:

$$
\begin{aligned}
\mathcal{I}_{CYK} &= \{[A, i, j] \mid A \in N \ \wedge \ 0 \leq i < j\};\\
D^{(1)} &= \{[a, i-1, i] \vdash [A, i-1, i] \mid A{\rightarrow}a \in P\},\\
D^{(2)} &= \{[B, i, j], [C, j, k] \ \vdash \ [A, i, k] \mid A{\rightarrow}BC \in P\},\\
D_{CYK} &= D^{(1)} \cup D^{(2)}.
\end{aligned}
$$

□

3 Relations between parsing schemata

Several types of relations between parsing schemata can be defined.
A parsing schema $\mathbf{P}_2$ is a *refinement* of a schema $\mathbf{P}_1$ if

- single deduction steps in $\mathbf{P}_1$ correspond to sequences of deduction steps in $\mathbf{P}_2$ (and, most likely, $\mathbf{P}_2$ contains additional items for the additional intermediate results);
- single items in $\mathbf{P}_1$ correspond to multiple items in $\mathbf{P}_2$.

A parsing schema $\mathbf{P}_2$ is called an *extension* of a schema $\mathbf{P}_1$ if it is defined for a larger class of grammars. A *generalization* is a combination of refinement and extension (in which either relation can be the identity relation).

We will give a simple, informal example. For formal definitions and a proof of the transitivity of these relations (nontrivial for refinement), see [10].

Example: **buE** is a generalization of **CYK**. This is accomplished as follows.

- Firstly, the CYK items $[A, i, j]$ are replaced by final Earley items $[A \rightarrow \alpha\bullet, i, j]$. Note that a single CYK item is replaced, in general, by a set of Earley items, as there can be different productions with left-hand side A. $D^{(2)}$ does now contain deduction steps of the form $[B \rightarrow \beta\bullet, i, j], [C \rightarrow \gamma\bullet, j, k] \vdash [A \rightarrow BC\bullet, i, k]$.
- Secondly, we introduce the remaining Earley items, split each CYK deduction step into the appropriate set of *scan* and *complete* steps, and add *init* steps as in **buE**.
- With the above two simple transformations we have obtained a parsing system $\mathbb{P}_{buE}$ for each grammar in CNF and each string. As a trivial third step, we extend the schema to the entire class of context-free grammars. □

Refinement means that more deduction steps have to be performed, in order to find all valid items. This is useful when it leads to a *qualitative* change in the algorithm. **buE** is more useful than **CYK** because it handles a larger class of grammars within the same complexity bounds. For a more mundane *quantitative* improvement, one can try to reduce the number of deduction steps needed to obtain all valid items. This is called *filtering*. One can differentiate between *static filtering*, i.e., deleting irrelevant items and deduction steps; *dynamic filtering*, i.e., adding antecedents (= extra conditions) to deduction steps, and *step contraction*, i.e., replacing sequences of steps by single steps. Again, see [10] for formal definitions.

Example: **Earley** is obtained from **buE** by applying a (dynamic) filter, as follows. Deduction steps $\vdash [B \rightarrow \bullet\beta, i, i]$ are replaced by deduction steps $[A \rightarrow \alpha\bullet B\gamma, h, i] \vdash [B \rightarrow \bullet\beta, i, i]$, except for the remaining *init* steps in $\mathbb{P}_{Earley}$. □

4 A more involved example

The above examples are not really new, in the sense that the facts are generally known. The improvement due to the parsing schemata framework is the con-

ciseness and clarity with which these can be stated. The relation between CYK and Earley has been stated thoroughly but clearly in hardly a single page. As a more exciting example, we derive a schema for Left-Corner parsers by a few modifications to the Earley schema.

Let us go back to **buE** first, and consider an item of the form $[A \rightarrow B \bullet \beta, i, j]$. If the item is valid, it can be deduced by a deduction step $[A \rightarrow \bullet B\beta, i, i], [B \rightarrow \gamma \bullet, i, j] \vdash [A \rightarrow B \bullet \beta, i, j]$. The item $[A \rightarrow \bullet B\beta, i, i]$ does not play any significant role here, it is valid by definition. Hence, no harm is done if we delete it and replace the *complete* step by a *left-corner* step:

$$[B \rightarrow \gamma \bullet, i, j] \ \vdash \ [A \rightarrow B \bullet \beta, i, j].$$

A similar argument applies to items of the form $[A \rightarrow a \bullet \beta, i, j]$ and the appropriate *scan* step.

Things get more interesting — and slightly more complicated — if we apply the same transformation to **Earley**, rather than **buE**. It is *not* the case that $[A \rightarrow \bullet B\beta, i, i]$ is always valid. Therefore, the replacement of $[A \rightarrow \bullet B\beta, i, i]$, $[B \rightarrow \gamma \bullet, i, j] \vdash [A \rightarrow B \bullet \beta, i, j]$ by a deduction $[B \rightarrow \gamma \bullet, i, j] \ \vdash \ [A \rightarrow B \bullet \beta, i, j]$ should be allowed *only in those cases* where $[A \rightarrow \bullet B\beta, i, i]$ is actually valid. Under which conditions is this the case?

The item $[A \rightarrow \bullet B\beta, i, i]$ is predicted by **Earley** only if there is some valid item of the form $[C \rightarrow \alpha \bullet A\delta, h, i]$. But if, by chance, $\alpha = \varepsilon$, then this is one of the very items that we seek to eliminate! In that case we continue the search for an item that licences the validity of $[C \rightarrow \bullet A\delta, i, i]$. This search can end in two ways: either we find some item with the dot not in leftmost position, or (only in case $i = 0$) we may move all the way up to $[S \rightarrow \bullet \gamma, 0, 0]$.

Definition. The *left corner* is the leftmost symbol in the right-hand side of a production. $A \rightarrow X\alpha$ has left corner X; an empty production $A \rightarrow \varepsilon$ has left corner ε. The relation $>_\ell$ on $N \times (N \cup \Sigma \cup \{\varepsilon\})$ is defined by

$$A >_\ell U \text{ if there is a production } p = A \rightarrow \alpha \in P \text{ with } U \text{ the left corner of } p.$$

The transitive and reflexive closure of $>_\ell$ is denoted $>_\ell^*$. □

It is clear that $[A \rightarrow \bullet B\beta, i, i]$ will be recognized by the Earley algorithm if there is some valid item $[C \rightarrow \alpha \bullet E\delta, h, i]$ with $E >_\ell^* A$. Moreover, there is such an item with $\alpha \neq \varepsilon$, unless, perhaps, $i = 0$ and $C = S$. For this exceptional case we retain items $[S \rightarrow \bullet \gamma, 0, 0]$ as usual. The discarded *complete* steps are replaced by *left-corner* steps as follows:

$$[C \rightarrow \alpha \bullet E\delta, h, i], [B \rightarrow \gamma \bullet, i, j] \ \vdash \ [A \rightarrow B \bullet \beta, i, j] \text{ only if } E >_\ell^* A,$$

see Figure 1; similarly for consequents of the form $[A \rightarrow a \bullet \beta, j-1, j]$.
Thus we obtain the following parsing schema **LC**, as usual by defining a parsing system $\mathbb{P}_{LC}$ for an arbitrary grammar:

$$\begin{aligned} \mathcal{I}_{LC} \quad = \ & \{[A \rightarrow X\alpha \bullet \beta, i, j] \mid A \rightarrow X\alpha\beta \in P \wedge 0 \leq i \leq j\} \\ & \cup \ \{[A \rightarrow \bullet, j, j] \mid A \rightarrow \varepsilon \in P \wedge j \geq 0\} \\ & \cup \ \{[S \rightarrow \bullet \gamma, 0, 0] \mid S \rightarrow \gamma \in P\}; \end{aligned}$$

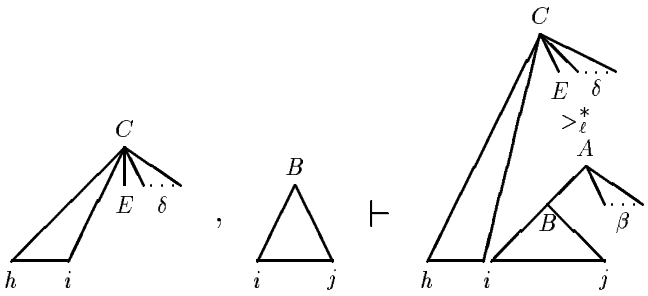

Fig. 1. The (*predictive*) *left-corner* step

$$
\begin{aligned}
D^{Init} &= \{\vdash [S \rightarrow \bullet\gamma, 0, 0]\},\\
D^{LC(A)} &= \{[C \rightarrow \gamma\bullet E\delta, h, i], [A \rightarrow \alpha\bullet, i, j] \vdash [B \rightarrow A\bullet\beta, i, j] \mid E >_\ell^* B\},\\
D^{LC(a)} &= \{[C \rightarrow \gamma\bullet E\delta, h, i], [a, i, i+1] \vdash [B \rightarrow a\bullet\beta, i, i+1] \mid E >_\ell^* B\},\\
D^{LC(\varepsilon)} &= \{[C \rightarrow \gamma\bullet E\delta, h, i], \vdash [B \rightarrow \bullet, i, i] \mid E >_\ell^* B\},\\
D^{Scan} &= \{[A \rightarrow \alpha\bullet a\beta, i, j], [a, j, j+1] \vdash [A \rightarrow \alpha a\bullet\beta, i, j+1]\},\\
D^{Compl} &= \{[A \rightarrow \alpha\bullet B\beta, i, j], [B \rightarrow \gamma\bullet, j, k] \vdash [A \rightarrow \alpha B\bullet\beta, i, k]\},\\
D_{LC} &= D^{Init} \cup D^{LC(a)} \cup D^{LC(A)} \cup D^{LC(\varepsilon)} \cup D^{Scan} \cup D^{Compl}.
\end{aligned}
$$

The general idea of a *left-corner* step involves slightly different details for non-terminal, terminal, and empty left corners.

A conceptual and (when it comes down to implementing the schema) practical simplification can be obtained by introducing a new type of item. In order to apply a *left-corner* step we have to look for *some* item of the form $[C \rightarrow \alpha\bullet E\delta, h, i]$, with arbitrary C, α, β, and h. We can introduce a special *predict item*, denoted $[i, E]$, to indicate that E has been predicted as a feasible constituent at position i. The extra work in deducing (and administering) predict items is offset by the simplification of the *left-corner* steps. So, finally, we obtain the simplified Left-Corner schema, **sLC**, defined by a parsing system $\mathbb{P}_{sLC}$ for arbitrary G as follows:

$$
\begin{aligned}
\mathcal{I}^{Pred} &= \{[i, A] \mid A \in N \ \wedge \ i \geq 0\},\\
\mathcal{I}^{LC(i)} &= \{[B \rightarrow X\alpha\bullet\beta, i, j] \mid B \rightarrow X\alpha\beta \in P \ \wedge 0 \leq i \leq j\},\\
\mathcal{I}^{LC(ii)} &= \{[B \rightarrow \bullet, j, j] \mid B \rightarrow \varepsilon \in P \ \wedge j \geq 0\},\\
\mathcal{I}_{sLC} &= \mathcal{I}^{Pred} \cup \mathcal{I}^{LC(i)} \cup \mathcal{I}^{LC(ii)};\\
D^{Init} &= \{\vdash \ [0, S]\},\\
D^{LC(A)} &= \{[i, C], \ [A \rightarrow \gamma\bullet, i, j] \ \vdash \ [B \rightarrow A\bullet\beta, i, j] \ \mid \ C >_\ell^* B\},
\end{aligned}
$$

$$
\begin{aligned}
D^{LC(a)} &= \{[i,C],\ [a,i,i+1] \vdash [B{\rightarrow}a\bullet\beta,i,i+1] \mid C >_\ell^* B\},\\
D^{LC(\varepsilon)} &= \{[i,C] \vdash [B{\rightarrow}\bullet,i,i] \mid C >_\ell^* B\},\\
D^{Pred} &= \{[B{\rightarrow}\alpha\bullet C\beta,i,j] \vdash [j,C]\},\\
D^{Scan} &= \{[B{\rightarrow}\alpha\bullet a\beta,i,j],\ [a,j,j+1] \vdash [B{\rightarrow}\alpha a\bullet\beta,i,j+1]\},\\
D^{Compl} &= \{[B{\rightarrow}\alpha\bullet A\beta,i,j],\ [A{\rightarrow}\gamma\bullet,j,k] \vdash [B{\rightarrow}\alpha A\bullet\beta,i,k]\},\\
D_{sLC} &= D^{Init} \cup D^{LC(a)} \cup D^{LC(A)} \cup D^{LC(\varepsilon)} \cup D^{Pred} \cup D^{Scan} \cup D^{Compl}.
\end{aligned}
$$

Thus we have shown the close relations between Earley parsing and Left-Corner parsing, by transforming the underlying parsing schemata into each other. It can be shown that the above schema underlies the (generalized) Left-Corner algorithm as it has been described in the literature, cf. [3, 5] (as opposed to *deterministic* LC parsing [7]).

5 Related work and further applications

The notion of a chart parser has been introduced by Kay [1]. Logical deduction as a basis for description of chart parsers is due to Pereira and Warren [6]. But our framework has a rather different emphasis. While the "Parsing as Deduction" approach is primarily interested in connecting the parsing logic with unification-based grammar formalisms, parsing schemata use deduction merely as a convenient notation for describing the relations between different context-free parsing algorithms. The difference in emphasis is illustrated by comparing Shieber's Ph.D. Thesis [9] with the extension of parsing schemata to unification grammars in [10]: the former describes an Earley-like parser for arbitrary unification grammars, whereas the latter describes arbitrary parsing algorithms for a simple PATR-like grammar.

An urgent question that needs to be addressed is: *how general is this framework*? We have simply supposed that parsing algorithms are based on items, but what about algorithms that do not use items? In [10] it is argued at length that items should be understood as congruence classes of partial parse trees. Almost all parsing algorithms are *constructive* in the sense that a parse is created piece by piece. Hence, explicitly or implicitly, a trace is left of those pieces, the intermediate results of the parser. These intermediate results are not necessarily partial parse trees, but it must be objects that denote *relevant properties* of those partial parses. In the CYK algorithm, for example, a recognized item $[A,i,j]$ denotes the existence of *some* tree with root A and yield $a_{i+1}\ldots a_j$. Perhaps there are different trees with the specified root and yield. The internal structure of such a partial parse tree is not relevant for the further proceeding of the algorithm and therefore not represented. So one can see an item as a *partial specification* of a partial parse tree, or alternatively, as a *set of trees*, viz., all trees that conform to this partial specification. Different algorithms use different notations for items, simply because the salient properties of intermediate results

that guide the parsing process are different. All these items can be seen as simplified notations of a more fundamental kind of item that represents sets of trees sharing relevant properties of whatever kind.

An important class of parsers, notably Generalized LR parsers [12], are based on the paradigm of a (nondeterministic) pushdown automaton. In a GLR parser, items are used at *compile time* to compute the parsing table, which makes it look rather different from an item-based parser. But it is no big deal to partly "uncompile" a GLR parser and make the recognized items visible at run time. An LR(0)-based GLR parser has an underlying parsing schema that is identical to the Earley schema, except for some techical restrictions on grammars imposed by the GLR machinery. In the terminology of Section 3: The schema **Earley** is a trivial extension of the schema **GLR(0)**. Having uncovered this close relation, one can cross-fertilize variants of both algorithms; see [11] for a parallel parser that combines Tomita's graph-structured stack with the straightforward parallelization of bottom-up Earley.

The item-based schema for an LC parser of Section 4 is interesting in itself, but it can also can be generalized to a *predictive Head-Corner* parser [8], that does not parse a sentence from left to right but starts with the "most interesting" words, and fills in the gaps later. Recognition of a production starts with a specially designated right-hand-side symbol, called the *head*, and then proceeds both leftwards and rightwards. An LC parser can be seen as a special case in which all productions have their heads in leftmost position.

A more theoretical application of parsing schemata is the simplification of correctness proofs. A parsing schema is rather more easy to prove correct, simply because there is much less to prove. The correctness of an algorithm is obtained by showing that it duely implements some correct schema. Notably for LR parsers this is a substantial simplification.

In this limited space we only discussed context-free parsing. It is not difficult to extend parsing schemata to unification grammars, provided that the grammar formalism has some notion of context-free backbone. The trend in unification grammars seems to be that the context-free backbone degenerates; more and more information is put into features. Nagata [4] and Maxwell and Kaplan [2] have pointed out that treating phrase structure grammar as functional constraints convenient for writing down grammars, but not for parsing. Parsing efficiency of unification grammars can be increased by retrieving a context-free backbone that covers more than just the *cat* feature. Context-free parsing, therefore, remains to be of interest for natural language processing.

6 Conclusion

We have given an informal presentation of the parsing schemata framework, that provides a well-defined level of abstraction between context-free grammars and context-free parsing algorithms. It is claimed that the framework covers all constructive parsing algorithms, both sequential and parallel.

Structural relations between parsing algorithms can investigated in a formal but nevertheless conceptually clear manner, by comparing the parsing schemata underlying these algorithms. As a detailed example, we have investigated the relation between Earley chart parsing and (generalized) Left-Corner parsing.

References

1. Kay, M (1980) Algorithm Schemata and Data Structures in Syntactic Processing. Report CSL-80-12, Xerox PARC, Palo Alto, Ca. Reprinted in: Grosz B et al. (1982) *Readings in Natural Language Processing*, Morgan Kaufmann, Los Altos, Ca.
2. Maxwell JT, Kaplan RM (1993) The Interface between Phrasal and Functional Constraints. *Computational Linguistics* **19**: 571–590.
3. Matsumoto Y, Tanaka H, Hirakawa H, Miyoshi H, Yasukawa H (1983) BUP: a bottom-up parser embedded in Prolog. *New Generation Computing* **1**: 145–158.
4. Nagata M (1992) An Empirical Study on Rule Granularity and Unification Interleaving Toward an Efficient Unification-Based Parsing System. *14th Int. Conf. on Computational Linguistics* (*COLING'92*), Nantes, France, 177–183.
5. Nederhof M-J (1993) Generalized Left-Corner Parsing. *6th Meeting of the EACL*, Utrecht, the Netherlands, 305–314.
6. Pereira FCN, Warren, DHD (1983) Parsing as Deduction. *21th Ann. Conf. of the ACL*, Cambridge, Mass., 137–144.
7. Rosenkrantz DJ, Lewis PM (1970) Deterministic Left Corner Parsing. *11th Ann. Symp. on Switching and Automata Theory*, 139–152.
8. Sikkel K, op den Akker R (1993) Predictive Head-Corner Chart Parsing. *3rd Int. Workshop on Parsing Technologies*, Tilburg/Durbuy, NL/Belgium, 267–276.
9. Shieber SM (1992) *Constraint-Based Grammar Formalisms: Parsing and Type Inference for Natural and Computer Languages*. The MIT Press, Cambridge, Mass.
10. Sikkel K (1993) Parsing schemata. Ph.D. Thesis, University of Twente, Enschede, the Netherlands.
11. Sikkel K, Lankhorst M (1992) A Parallel Bottom-Up Tomita Parser. *1. Konferenz Verarbeitung natürlicher Sprache* (*KONVENS'92*), Nürnberg, 238–247.
12. Tomita M (1985) *Efficient Parsing for Natural Language*. Kluwer Academic Publishers, Boston, Mass., 1985.
13. Younger DH (1967) Recognition of context-free languages in time n^3, *Information and Control* **10**: 189–208.

Reference to this paper:
K. Sikkel. A framework for parsing algorithm specification and analysis. in: H. Trost (Hrsg), *Tagungsband KONVENS'94, 2. Konferenz "Verarbeitung natürlicher Sprache", Wien, 1994*. Informatik Xpress 6, Springer Verlag, Berlin, pp. 300-309.

–