Theoretical Computer Science 15 (1981) 117–157
North-Holland Publishing Company

A PARSING AUTOMATA APPROACH TO LR THEORY

Stephan HEILBRUNNER

Fachbereich Informatik, Hochschule der Bundeswehr München, D 8014 Neubiberg, Fed. Rep. Germany

Communicated by M.A. Harrison
Received October 1978
Revised March 1980

Abstract. Two simple but powerful tools are introduced in LR theory, viz. item grammars and parsing automata. Parsing automata are used to define a large class of correct prefix parsers operating in linear time. DeRemer's and Pager's parsing methods turn out to be special cases. LL and LC tests, as well as inclusion theorems for the classes of LL, LC, and LR grammars will be based on parsing automata, too.

1. Introduction

It is not a simple task to give a convincing explanation of the commonly used LR(k) definition. There are detailed tutorials on LR parsing which do not even state it [1, 21]. The technical apparatus for a presentation of LR theory along the now traditional lines of [2] involves a lot of tedious details which may prove the results but certainly obscures the ideas. A comprehensive and formal treatment using these methods is contained in [13]. We claim that basically simple ideas and a few clever tricks are sufficient to explain and prove correct in an intuitively clear and still formal way Knuth's original LR algorithm [25] and DeRemer's [9, 10] and Pager's [27] variants.

We shall see that a grammar is an LR(k) grammar if and only if the straightforward nondeterministic bottom-up parsing algorithm for this grammar can be made deterministic by eliminating superfluous moves. From this a characterization of LR grammars by certain regular sets will be immediate. These sets will be generated by special right-linear grammars, the *item grammars*, from which we derive finite automata, the *parsing automata*, by well-known methods. Parsing automata will be used to define correct prefix parsers operating in linear time. DeRemer's and Pager's methods will turn out to be special cases where only Pager's method requires some effort.

The usual tests for the LL(k) condition (if $k \geq 2$, [2, 31]) and the LC(k) condition (if $k \geq 1$, [30, 33]) are complicated and completely different from the LR(k) test. Therefore, it should not surprise that the formal proof of these tests and of intuitively clear theorems relating the LL(k), LC(k) and LR(k) parsing methods use lengthy

and involved arguments [2, 30, 33]. In this paper we shall give theoretically simple tests for the LL(k) and LC(k) properties which are based on parsing automata and which are very similar to the LR(k) test. These tests make possible new inclusion theorems for the classes of LL, LC and LR grammars and allow simple proofs of the well-known inclusion theorems. Moreover, an efficient and rather surprising LC(1) test will be based on parsing automata.

We use the definitions and notations of [2] with some minor modifications (see Fig. 1). Note $u, v \in \Sigma^k$ and $u = v = \varepsilon$ if $k = 0$. Throughout, $G = (N, \Sigma, \Pi, S)$ is an arbitrary but fixed grammar which does not allow $S \Rightarrow_{rm}^{+} S$ and does not have useless variables. There is a special symbol $\perp \in \Sigma$ which does not appear in the production rules, i.e., $\Pi \subseteq N \times (V - \{\perp\})^*$. All strings analysed by our parsing algorithms are assumed to end in $\perp^k$ so that $\perp$ acts as an end marker. We let $\#$ denote $\perp^k$. If the sequence $\pi \in \Pi^*$ is to be concatenated with the rule $A \to \beta \in \Pi$ we write $A \to \beta\pi$. We write $k{:}w = u$ if $w = uz$ for some z and if $|u| = k$ (see also Fig. 1).

the symbols represent elements of	a, b, c, d	u, v	w, x, y, z	A, B, C, D	X, Y, Z	ε	$\alpha, \beta, \ldots$	π
	Σ	Σ^k	Σ^*	N	V	V^0	V^*	Π^*

Fig. 1

2. LR(k) definition

Considering a pushdown-automaton whose move relation essentially agrees with the relation $\Vdash$ defined by

$$(\alpha\beta, z, \pi) \Vdash (\alpha A, z, A \to \beta\pi) \quad \text{whenever } A \to \beta \in \Pi, \tag{1}$$

$$(\psi, az, \pi) \Vdash (\psi a, z, \pi) \quad \text{for all } a \in \Sigma, \tag{2}$$

is one of the two usual ways of proving that context-free languages can be accepted by pushdown-automata. Moves according to (1) and (2) are called *reductions* and *shifts*, respectively. The relation reflects right-most derivations. Induction proves its essential property (cf. [2, Lemma 2.25]), viz. that it can parse all and only the strings in the language generated by the grammar.

Theorem 2.1.

$$S \underset{rm}{\overset{\pi}{\Rightarrow}} z \;\Leftrightarrow\; (\varepsilon, z, \varepsilon) \overset{*}{\Vdash} (S, \varepsilon, \pi). \tag{3}$$

The relation $\Vdash$ describes the behaviour of a nondeterministic machine. This machine has to be modified if it is to be used for a practical purpose. First, we try to make the machine deterministic by allowing fewer moves, which means that we have to look for subsets $\vdash$ of $\Vdash$ which are partial functions. Next, the machine is simplified

by permitting only k symbols of lookahead for some fixed $k \geq 0$, which is achieved by requiring

$$(\alpha\beta, z, \pi) \vdash (\alpha A, z, A \to \beta\pi) \wedge k{:}z = k{:}y > (\alpha\beta, y, \pi') \vdash (\alpha A, y, A \to \beta\pi'), \tag{4}$$

$$(\psi, az, \pi) \vdash (\psi a, z, \pi) \wedge k{:}az = k{:}by > (\psi, by, \pi') \vdash (\psi b, y, \pi'). \tag{5}$$

Of course, $b \neq a$ is impossible in (5) for $k > 0$. At last, we ensure that these modifications preserve the parsing power of $\Vdash$ by requiring

$$S\# \underset{\text{rm}}{\overset{\pi}{\Rightarrow}} w\# \asymp (\varepsilon, w\#, \varepsilon) \overset{*}{\vdash} (S, \#, \pi) \tag{6}$$

where we append k endmarkers to provide sufficient lookahead. Note that '$\asymp$' of (6) holds for all $\vdash \subseteq \Vdash$.

Definition 2.1. A relation $\vdash \subseteq \Vdash$ is a (*shift-reduce*) *parsing relation* (*using k symbols of lookahead*) if (4)–(6) are satisfied. The symbol $\vdash$ is reserved for parsing relations. A relation is *deterministic* if it is a partial function. A grammar is an *LR(k) grammar* if it allows a deterministic parsing relation. Note that $S \Rightarrow_{\text{rm}}^{+} S$ was excluded.

LR(k) grammars are unambiguous because of (6). We note that all bottom-up parsing methods including precedence and bounded-context techniques use parsing relations. Among the many parsing relations we can single out a relation $\vdash_c$ which is best possible in the following sense: Any relation $\vdash \subseteq \Vdash$ using only k symbols of lookahead is a parsing relation iff $\vdash_c \subseteq \vdash$. Therefore, $\vdash_c$ is the uniquely determined minimal parsing relation and is deterministic iff we are dealing with an LR(k) grammar. In order to find an explicit description of $\vdash_c$ we keep in mind that all parsing relations invert rightmost derivations because of (1) and (6). Therefore, we conjecture that a parsing relation must reduce by $A \to \beta$ whenever a rightmost derivation applies $A \to \beta$ and that a parsing relation must shift whenever shifting can lead to a configuration which calls for a reduction. The next lemma expresses this idea in formal terms.

Lemma 2.2. *For any parsing relation* $\vdash$

$$S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha A w \underset{\text{rm}}{\Rightarrow} \alpha\beta w > (\alpha\beta, w, \pi) \vdash (\alpha A, w, A \to \beta\pi), \tag{7}$$

$$S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \rho B w \underset{\text{rm}}{\Rightarrow} \rho\delta w = \psi a y w > (\psi, ayw, \pi) \vdash (\psi a, yw, \pi). \tag{8}$$

Proof. From (6) one obtains for any parsing relation $\vdash$

$$S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha A w \underset{\text{rm}}{\overset{\pi'}{\Rightarrow}} zw > (\varepsilon, zw, \varepsilon) \overset{*}{\vdash} (\alpha A, w, \pi') \tag{9}$$

by induction on the length of $\Rightarrow_{rm}^*$. As there are no useless variables we may apply (9) to (7) which yields $(\varepsilon, zw, \varepsilon) \vdash^* (\alpha A, w, A \to \beta\pi)$ for some z, π. Obviously, this is impossible unless $(\alpha\beta, w, \pi) \vdash (\alpha A, w, A \to \beta\pi)$. Concerning (8) we find $(\varepsilon, zayw, \varepsilon) \vdash^* (\rho B, w, B \to \delta\pi)$ so that $(\varepsilon, zayw, \varepsilon) \vdash^* (\rho\delta, w, \pi) = (\psi ay, w, \pi)$ which entails $(\psi, ayw, \pi) \vdash (\psi a, yw, \pi)$.

Lemma 2.2 and (4) imply for any parsing relation $\vdash$

$$S \# \overset{*}{\underset{rm}{\Rightarrow}} \alpha A u y \underset{rm}{\Rightarrow} \alpha\beta u y > (\alpha\beta, uz, \pi) \vdash (\alpha A, uz, A \to \beta\pi) \tag{10}$$

which is the reason for the next definition (cf. [5]).

Definition 2.2. The $LR(k)$ *context* of $A \to \beta$ is the set defined by

$$\mathrm{LRC}_k^G(A \to \beta) = \mathrm{LRC}(A \to \beta)$$

$$= \{\alpha\beta u \mid \exists w : S \# \overset{*}{\underset{rm}{\Rightarrow}} \alpha A u w \underset{rm}{\Rightarrow} \alpha\beta u w\}$$

where we recall $\alpha, \beta \in V^*$, $u \in \Sigma^k$, $w \in \Sigma^*$. Moreover, we define

$$\mathrm{LRC}_k^G(\Pi) = \mathrm{LRC}(\Pi) = \bigcup_{A \to \beta \in \Pi} \mathrm{LRC}(A \to \beta).$$

Lemma 2.3. *A relation* $\vdash \subseteq \Vdash$ *which satisfies* (4) *and* (5) *is a parsing relation iff* (11) *and* (12) *hold where*

$$\alpha\beta u \in \mathrm{LRC}(A \to \beta) > (\alpha\beta, uw, \pi) \vdash (\alpha A, uw, A \to \beta\pi), \tag{11}$$

$$\exists x \neq \varepsilon : \psi u x \in \mathrm{LRC}(\Pi) \wedge k : aw = u > (\psi, aw, \pi) \vdash (\psi a, w, \pi). \tag{12}$$

Proof. The if part is proven by straightforward induction. As for the only-if part, (11) follows from (10). Concerning (12) we assume $x \neq \varepsilon \wedge \psi u x \in \mathrm{LRC}(\Pi) \wedge k : aw = u$ and obtain a derivation

$$S \# \overset{*}{\underset{rm}{\Rightarrow}} \rho B z \underset{rm}{\Rightarrow} \rho\delta z = \psi u x z$$

for some strings ρ, B, δ, z. Writing ux as $ux = by$ Lemma 2.2 says $(\psi, byz, \pi) \vdash (\psi b, yz, \pi)$. Eq. (5) and $k : by = k : ux = u = k : aw$ imply $(\psi, aw, \pi) \vdash (\psi a, w, \pi)$.

This lemma gives us the means to define the minimal parsing relation for any given grammar. This relation is called the canonical relation.

Definition 2.3. The (*canonical*) *LR(k) parsing relation* $\vdash_c$ is defined by

$$\alpha\beta u \in \mathrm{LRC}(A \to \beta) \ltimes (\alpha\beta, uw, \pi) \vdash_c (\alpha A, uw, A \to \beta\pi),$$

$$\exists x \neq \varepsilon : \psi ux \in \mathrm{LRC}(\Pi) \wedge k : aw = u \ltimes (\psi, aw, \pi) \vdash_c (\psi a, w, \pi)$$

for all strings $w \in \Sigma^*$, $u \in \Sigma^k$, $\alpha, \psi \in V^*$, $a \in \Sigma$ and all $\pi \in \Pi^*$ and $A \to \beta \in \Pi$.

By Lemma 2.3 the LR(k) parsing relation is a parsing relation, indeed, and satisfies $\vdash_c \subseteq \vdash$ for any parsing relation $\vdash$. Hence, a grammar is LR(k) iff $\vdash_c$ is deterministic, that is, iff

$$\psi u \in \mathrm{LRC}(A \to \beta) \cap \mathrm{LRC}(B \to \delta) > A \to \beta = B \to \delta,$$

$$\psi u \in \mathrm{LRC}(\Pi) \wedge \psi ux \in \mathrm{LRC}(\Pi) > x = \varepsilon.$$

Combining the two formulae gives us the next theorem.

Theorem 2.4. *A grammar is* LR(k) *iff*

$$\psi ux \in \mathrm{LRC}(B \to \delta) \wedge \psi u \in \mathrm{LRC}(A \to \beta) > x = \varepsilon \wedge A \to \beta = B \to \delta. \tag{13}$$

The next theorem makes our LR(k) definition equivalent to the usual definition as given in [12]. Note that (13) and (15) differ only in the letters x and σ. We recall that we require $x \in \Sigma^*$ but allow $\sigma \in V^*$ so that (13) is a trivial consequence of (15). It is a common error to consider (14) and (15) equivalent [2, 5, 9, 20, 25, 32]. This error does not affect the theory of LR parsing since (15) is not needed as will be seen in the sequel.

Theorem 2.5. *Condition* (13) *is equivalent to*

$$S \# \underset{\mathrm{rm}}{\overset{*}{\Rightarrow}} \alpha A u w \underset{\mathrm{rm}}{\Rightarrow} \alpha\beta u w \wedge S \# \underset{\mathrm{rm}}{\overset{*}{\Rightarrow}} \gamma B v x \underset{\mathrm{rm}}{\Rightarrow} \alpha\beta u y >$$

$$> \alpha A = \gamma B \wedge v x = u y. \tag{14}$$

Condition (13) *is equivalent to* (15) *iff* $k \geq 1$.

$$\psi u \sigma \in \mathrm{LRC}(B \to \delta) \wedge \psi u \in \mathrm{LRC}(A \to \beta) > \sigma = \varepsilon \wedge B \to \delta = A \to \beta. \tag{15}$$

Proof. The proofs of similar theorems of [20] may be adapted to show the equivalences (see [18] for details). The non-equivalence claimed for $k = 0$ is proven by the LR(0) grammar with the rules $S \to aB$, $B \to \varepsilon$ because of

$$aB \in \mathrm{LRC}(S \to aB) \wedge a \in \mathrm{LRC}(B \to \varepsilon) \wedge B \neq \varepsilon.$$

3. Parsing automata

Obviously, the key to shift-reduce parsing is the computation of the LR(k) contexts. They are well known to be regular but the proofs of this fact [5, 25, 26] yield

parsing methods of merely theoretical interest. Practical methods involve 'items', 'collections of sets of items' and 'GOTO-functions'. We shall see shortly that these notions describe a deterministic, finite automaton which computes the LR(k) contexts. In this way the gap between the theoretical and the practical LR methods will be closed. For this purpose we introduce a simple formalism dealing with items where an *item* (*of degree k*) is a quadruple (A, α, β, u) such that $A \to \alpha\beta \in \Pi$, $u \in \Sigma^k$. Items are denoted by $[A \to \alpha . \beta, u]$. Empty strings are omitted frequently, so that e.g. $[A \to \alpha . \varepsilon, u] = [A \to \alpha ., u]$ and $[A \to \varepsilon . \beta, \varepsilon] = [A \to . \beta]$.

Definition 3.1. The *item grammar* (*of degree k*) of a grammar G is $(I_k \cup \{[S]\}, N \cup \Sigma, \Pi_k, [S])$ where I_k is the set of items of degree k and

$$\Pi_k = \{[A \to \alpha . X\beta, u] \to X[A \to \alpha X . \beta, u] \mid A \to aX\beta \in \Pi\}$$
$$\cup \{[A \to \alpha . B\beta, u] \to [B \to . \delta, v] \mid v \in \text{first}(\beta u) \wedge A \to \alpha B\beta, B \to \delta \in \Pi\}$$
$$\cup \{[S] \to [S \to . \alpha, \#] \mid S \to \alpha \in \Pi\}$$

where

$$\text{first}(\beta u) = \text{first}_k^G(\beta u) = \{v \mid \exists x : \beta u \underset{\text{rm}}{\overset{*}{\Rightarrow}} vx\}.$$

Item grammars are refined versions of the grammars used by Knuth [25] to compute the LR(k) contexts. They have been used at least since 1973 [14] and were rediscovered independently by a number of persons including the author. They do not seem to have appeared in the literature except for the reports [11, 18] and the manuscript [15]. The corresponding nondeterministic finite automata were used in [24, 35]. The next lemma relates rightmost derivations in a grammar to derivations in this item grammar. We shall use $[\ ], [\ ]_1, \ldots$ to denote items.

Lemma 3.1 (Item lemma).

$$\exists j : [\ ]_1 \overset{j}{\Rightarrow} \gamma[\ ]_2 \wedge j = |\gamma| \ \nLeftrightarrow \ \exists A, \alpha, \beta, u : [\ ]_1 = [A \to \alpha . \gamma\beta, u]$$
$$\wedge [\ ]_2 = [A \to \alpha\gamma . \beta, u] \qquad (1)$$

$$\exists j : [A \to \alpha . \beta, u] \overset{j}{\Rightarrow} \gamma[C \to \rho . \sigma, v] \wedge j \neq |\gamma|$$
$$\nLeftrightarrow \ \exists \omega, w : \beta u \underset{\text{rm}}{\overset{*}{\Rightarrow}} \omega C v w \underset{\text{rm}}{\Rightarrow} \omega\rho\sigma v w = \gamma\sigma v w \qquad (2)$$

where we exclude $[\ ]_1 = [S]$ *in* (1).

Proof. Note that $j < |\gamma|$ is impossible in (2). (1) is obvious. For the proof of '$<$' of (2) we use induction on the length i of $\Rightarrow_{\text{rm}}^*$. The case $i = 0$ is simple. If $i > 0$, the syntactic

variable C shown explicitly, was introduced in some step $n+1 \leq i$ so that we may write

$$\beta u \overset{n}{\underset{\text{rm}}{\Rightarrow}} \psi B y \underset{\text{rm}}{\Rightarrow} \psi\delta C\delta' y \overset{*}{\underset{\text{rm}}{\Rightarrow}} \psi\delta C v w \underset{\text{rm}}{\Rightarrow} \psi\delta\rho\sigma v w. \tag{3}$$

Applying the induction hypothesis to (3) yields

$$[A \to \alpha\,.\,\beta, u] \overset{m}{\Rightarrow} \psi\delta[B \to \delta\,.\,C\delta', k\colon y] \wedge |\psi\delta| > m.$$

The rule $[B \to \delta\,.\,C\delta', k\colon y] \to [C \to .\,\rho\sigma, v]$ is a consequence of (3). The derivation $[C \to .\,\rho\sigma, v] \overset{*}{\Rightarrow} \rho[C \to \rho\,.\,\sigma, v]$ is obvious. This proves '$<$'. The other half of the proof is a straightforward inductive exercise.

Note the similarity of this proof to the proof of Theorem 5.10 of [2] concerning the 'validity' of items. A similar lemma appears in [11], too. The lemma provides the key to computing the LR(k) contexts.

Theorem 3.2. $\psi u \in \mathrm{LRC}(A \to \beta) \Leftrightarrow [S] \overset{*}{\Rightarrow} \psi[A \to \beta., u]$.

As the item grammar is right-linear all the LR(k) contexts are regular sets and can be accepted by finite automata. The simple structure of the item grammar allows us to construct effectively a single finite automaton which can recognize all LR(k) contexts in a certain sense. The method of choice is the well-known subset construction for nondeterministic automata as presented in [16].

Definition 3.2. The *completely specified automaton* is the finite automaton $\mathfrak{A}_{cs} = (\mathfrak{P}(I), V, \delta_{cs}, q_{cs}, \emptyset)$[1] where $\mathfrak{P}(I)$ is the power set of $I = I_k \cup \{[S]\}$ and where

$$q_{cs} = \{[\ ] \mid [S] \overset{*}{\Rightarrow} [\ ]\} \quad \text{and} \quad \delta_{cs}(q, X) = \{[\ ] \mid \exists [\ ]_1 \in q\colon [\ ]_1 \overset{*}{\Rightarrow} X[\ ]\}$$

for all $q \subseteq I$, $X \in V$.

The automaton obtained from $\mathfrak{A}_{cs}$ by deleting inaccessible states[2] and the empty state is denoted by $\mathfrak{L}_c = (Q_c, V, \delta_c, q_c, \emptyset)$ and is called the (*canonical*) *LR(k) automaton*. Fig. 2 gives an example.

A moment's reflection shows that Q_c is the 'canonical collection of sets of items' and that δ_c is the GOTO-function of [2]. The transition graph of the LR(k) automaton is the 'characteristic graph' defined in [6]. Note that we do not need an algorithm to define δ_c and Q_c which will be a great help in the proofs. Nevertheless, an algorithm which computes δ_c and Q_c is an immediate consequence of the usual finite state techniques. We extend the transition functions of finite automata from

[1] The notion of a 'final state' is not needed in the sequel. Hence, the set of final states may be empty.

[2] A state is inaccessible if it is not part of any path emanating from the initial state.

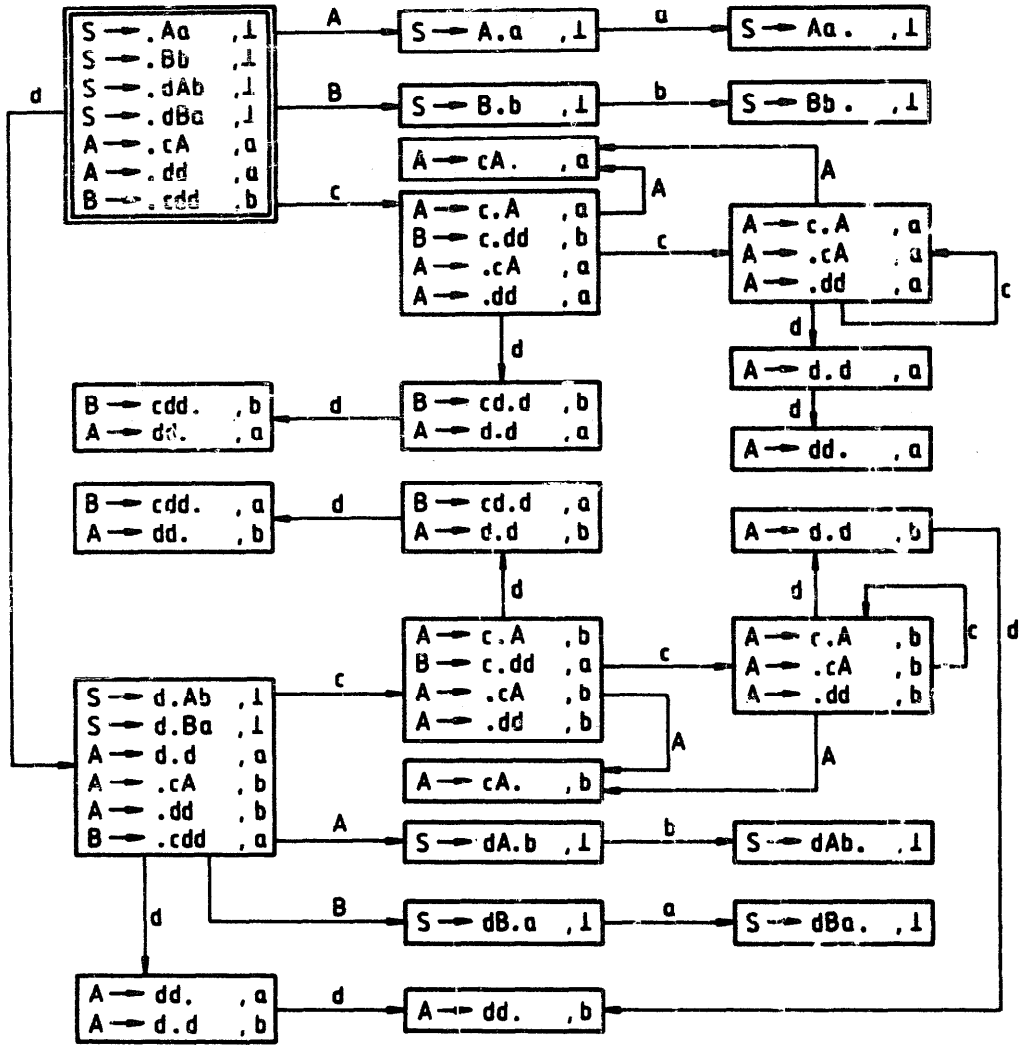

Fig. 2. The LR(1) automaton for the grammar $G1$ with the rules $S \to Aa$, $S \to dAb$, $A \to cA$, $A \to dd$, $S \to Bb$, $S \to dBa$, $B \to cdd$. The initial state is doubly framed. $L(G1) = c^+dda \cup dc^+ddb \cup cddb \cup dcdda$

letters to words in the usual way and note

$$\delta_c^*(q_c, \psi) = \delta_{cs}^*(q_{cs}, \psi) = \{[\ ]\,|\,[S] \overset{*}{\Rightarrow} \psi[\ ]\} \qquad (4)$$

whenever δ_c^* is defined for q_c and ψ. Combining this formula with Theorem 3.2 gives us

$$\psi u \in \mathrm{LRC}(A \to \beta) \Leftrightarrow [A \to \beta\,.\,, u] \in \delta_c^*(q_c, \psi) \qquad (5)$$

so that the LR(k) automaton may be used to compute the LR(k) relation. In order to avoid repetitions we introduce more general automata before drawing the obvious conclusions from (5).

Definition 3.3. A *parsing automaton (of degree k)* is a finite automaton $(Q, V, \delta, q_0, \emptyset)$ without inaccessible states such that $Q \subseteq \mathfrak{P}(I)$, and $q_{cs} \subseteq q_0$, and, for

all $q \in Q, X \in V$

$$\delta_{cs}(q, X) \neq \emptyset \succ \delta_{cs}(q, X) \subseteq \delta(q, X).$$

Of course, this makes the LR(k) automaton a parsing automaton. A set q of items is *consistent* if the two conditions

$$[A \to \beta\,.\,, u] \in q \wedge [B \to \delta\,.\,, u] \in q \succ A \to \beta = B \to \delta,$$

$$[A \to \alpha\,.\,a\beta, v] \in q \wedge [B \to \delta\,.\,, u] \in q \succ u \notin \text{first}(a\beta v)$$

are satisfied. A parsing automaton is *consistent* if all its states are consistent. See Fig. 3 for an example.

Note that this definition encompasses all the parameterized algorithms of [13].

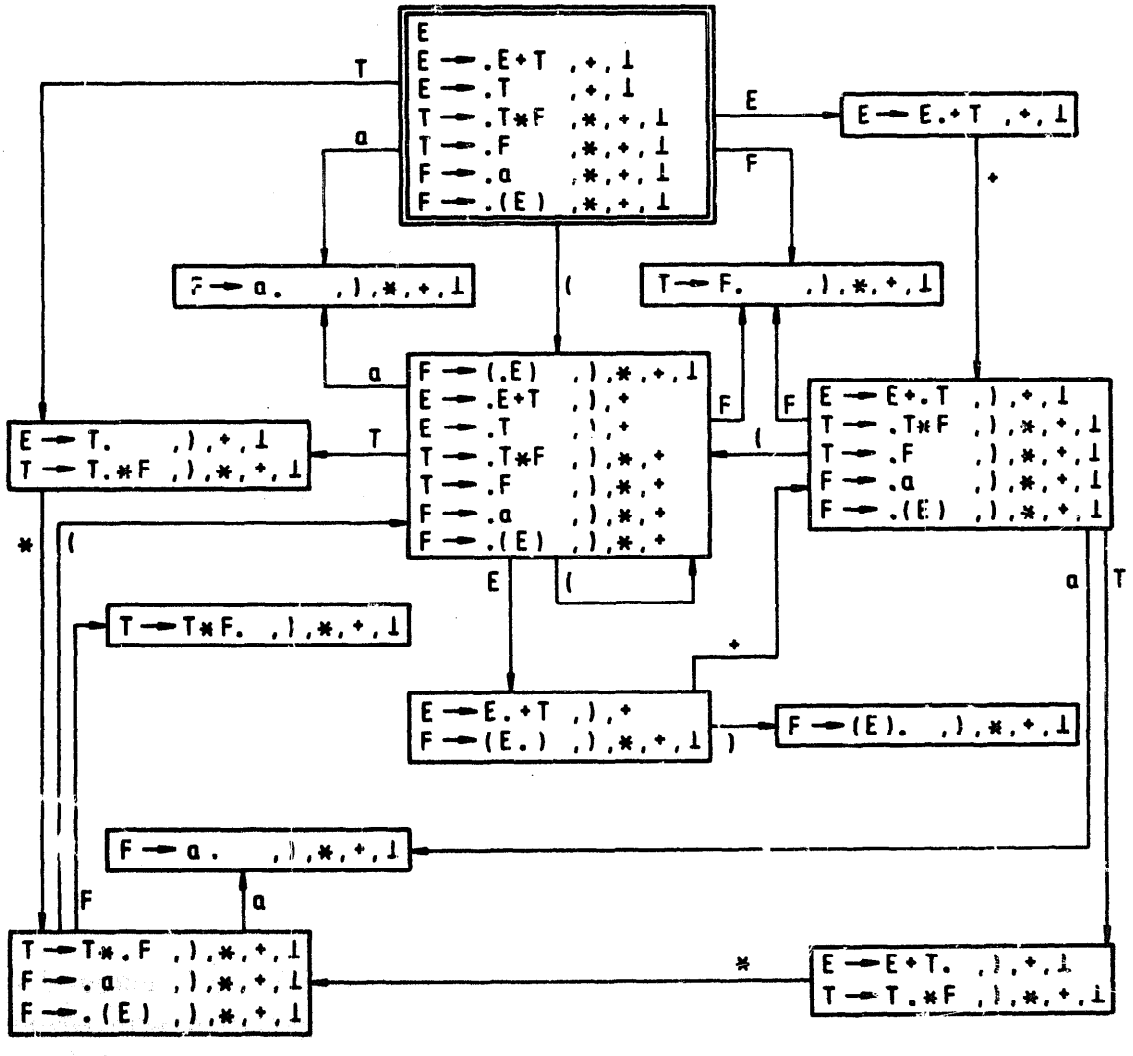

Fig. 3. A parsing automaton of degree one for the grammar with the rules $E \to E+T$, $E \to T$, $T \to T*F$, $T \to F$, $F \to a$, $F \to (E)$. Each line represents several items, e.g. line '$E \to E+.\,T,), +, \perp$' represents $[E \to E+.\,T,)]$, $[E \to E+.\,T, +]$, and $[E \to E+.\,T, \perp]$. The initial state is doubly framed.

For parsing automata we have

$$\delta_c^*(q_c, \psi) = \delta_{cs}^*(q_{cs}, \psi) \subseteq \delta^*(q_0, \psi) \quad \text{whenever } \delta_{cs}^*(q_{cs}, \psi) \neq \emptyset. \tag{6}$$

Lemma 3.3. *For all parsing automata*

$$\psi u \in \mathrm{LRC}(A \to \beta) \succ [A \to \beta\,.\,, u] \in \delta^*(q_0, \psi), \tag{7}$$

$$\exists x \neq \varepsilon : \psi u x \in \mathrm{LRC}(\Pi) \succ \exists [A \to \alpha\,.\,a\beta, v] \in \delta^*(q_0, \psi) : u \in \mathrm{first}(a\beta v). \tag{8}$$

For the LR(k) *automaton the reversed implications hold, too.*

Proof. (7) is a consequence of (5) and (6). For the proof of (8) suppose $x \neq \varepsilon \wedge \psi u x \in \mathrm{LRC}(B \to \delta)$. Then $\psi u x$ may be written as $\psi u x = \psi a y v' \in \mathrm{LRC}(B \to \delta)$ so that Theorem 3.2 and the structure of the item grammar gives us a derivation

$$[S] \overset{*}{\Rightarrow} \psi [A \to \alpha\,.\,a\beta, v] \Rightarrow \psi a [A \to \alpha a\,.\,\beta, v] \overset{*}{\Rightarrow} \psi a y [B \to \delta\,.\,, v']$$

for some A, α, β, y. If $\beta \in \Sigma^*$, then $y = \beta \wedge v = v'$ so that $u \in \mathrm{first}(a\beta v)$ is immediate. Otherwise, we apply the item lemma to find a derivation

$$a\beta v \underset{\mathrm{rm}}{\overset{*}{\Rightarrow}} \omega B v' w \underset{\mathrm{rm}}{\Rightarrow} \omega \delta v' w = a y v' w = u x w$$

and, again, $u \in \mathrm{first}(a\beta v)$. Finally, $[A \to \alpha\,.\,a\beta, v] \in \delta^*(q_0, \psi)$ is a consequence of (4) and (6).

The reversed implication for (7) holds true by (5). Concerning (8), suppose that $[A \to \alpha\,.\,a\beta, v]$ is given accordingly. If $\beta \notin \Sigma^*$ the choice of u gives us a derivation

$$a\beta v \underset{\mathrm{rm}}{\overset{*}{\Rightarrow}} y B v' w \Rightarrow y z v' w \quad \text{where } u = k : yzv' \wedge y \neq \varepsilon$$

because $a\beta \in \Sigma V^+$. Using the item lemma and (7) we conclude

$$[B \to z\,.\,, v'] \in \delta_c^*(\delta_c^*(q_c, \psi), yz) = \delta_c^*(q_c, \psi y z)$$

so that $\psi y z v' \in \mathrm{LRC}(\Pi)$ by (5). Rewriting $\psi y z v'$ as $\psi u x$ verifies our claim. The case $\beta \in \Sigma^*$ is still simpler.

Theorem 3.4. *A grammar is* LR(k) *iff its* LR(k) *automaton is consistent. The* LR(k) *automaton is consistent iff there is a consistent parsing automaton.*

This theorem is an immediate consequence of Lemma 3.3 and Theorem 2.4. Combining Lemmata 2.3 and 3.3 allows us to associate parsing relations with parsing automata. These parsing relations require repeated computations of δ^* which consumes too much time for practical purposes. The following relation $\models$ avoids the use of δ^* by stacking states of the parsing automaton. Instead of using configurations

$(X_1 \cdots X_n, z, \pi)$ as used by parsing relations this relation is based on configurations

$$(X_1 \cdots X_n, q_0 q_1 \cdots q_n, z, \pi) \quad \text{where } \forall i\colon \delta(q_{i-1}, X_i) = q_i$$

so that $\delta^*(q_0, X_1 \cdots X_n) = q_n$.

Definition 3.4. We define the relation $\models$ by

$$(\psi, q_0 q_1 \cdots q_i q_{i+1} \cdots q_n, uz, \pi) \models (\alpha A, q_0 q_1 \cdots q_i q, uz, A \to \beta\pi) \nLeftrightarrow$$
$$\nLeftrightarrow [A \to \beta\,.\,, u] \in q_n \wedge |\beta| = |q_{i+1} \cdots q_n| \wedge \delta(q_i, A) = q \wedge \psi = \alpha\beta$$

and by

$$(\psi, q_0 q_1 \cdots q_n, az, \pi) \models (\psi a, q_0 q_1 \cdots q_n q, z, \pi) \nLeftrightarrow$$
$$\nLeftrightarrow \delta(q_n, a) = q \wedge \exists A, \alpha, \beta, v\colon [A \to \alpha\,.\,a\beta, v] \in q_n \wedge \text{first}(az) \subseteq \text{first}(a\beta v).$$

In this definition q_0 is the initial state of the parsing automaton. The relation $\models$ is deterministic if the parsing automaton is consistent. The relation parses correctly all sentences in the language which may be verified by straightforward induction. The next theorem formalizes this statement. It holds true for all parsing automata.

Theorem 3.5.

$$S \underset{\text{rm}}{\Rightarrow} \alpha \overset{\pi}{\underset{\text{rm}}{\Rightarrow}} w \nLeftrightarrow \exists n\ \exists q_1, \ldots, q_n\colon [S \to \alpha\,.\,, \#] \in q_n$$
$$\wedge (\varepsilon, q_0, w\#, \varepsilon) \overset{*}{\models} (\alpha, q_0 q_1 \cdots q_n, \#, \pi).$$

4. Core-restricted parsing automata

Usually, LR parsing algorithms do not stack vocabulary symbols, that is, they employ configurations $(q_0 q_1 \cdots q_n, z, \pi)$ rather than $(\psi, q_0 q_1 \cdots q_n, z, \pi)$ which is a major improvement. The relation $\models$ of Section 3 needs ψ only when checking whether β is a suffix of ψ while reducing by a rule $A \to \beta$. We do not need ψ any more and can avoid stacking symbols, therefore, if we restrict ourselves to parsing automata which satisfy

$$[A \to \alpha\,.\,\beta, u] \in \delta^*(q_0, \psi) \succ \exists \gamma\colon \psi = \gamma\alpha. \tag{1}$$

The next definition introduces a class of automata which guarantee (1).

Definition 4.1. Let $\mathfrak{A}_c^0 = (Q_c^0, V, \delta_c^0, q_c^0, \emptyset)$ be the LR(0) automaton and define the function $core : I_k \to I_0$ by

$$\text{core}([A \to \alpha\,.\,\beta, u]) = [A \to \alpha\,.\,\beta, \varepsilon] = [A \to \alpha\,.\,\beta].$$

A parsing automaton $\mathfrak{A}$ is *core-restricted* if $\text{core}(q_0) = q_c^0$ and if core generates a homomorphism from $\mathfrak{A}$ onto $\mathfrak{A}_c^0$, i.e., if

$$\text{core}(\delta(q, X)) = \delta_c^0(\text{core}(q), X) \quad \text{for all } q \in Q, X \in V. \tag{2}$$

For all $k \geq 0$, the LR(k) automaton is core-restricted which can be seen by inspecting its definition (Definition 3.2). By induction (2) implies

$$\text{core}(\delta^*(q_0, \psi)) = \delta_c^{0*}(\text{core}(q_0), \psi) = \delta_c^{0*}(q_c^0, \psi). \tag{3}$$

Example 4.1. Fig. 3 shows a core-restricted parsing automaton, whereas the parsing automaton of Fig. 4 is not, since

$$\text{core}\left(\delta\left(\begin{bmatrix} S \to a\,.\,Aa, \perp \\ A \to .\,cd, \quad a \end{bmatrix}, c\right)\right) = \text{core}\left(\begin{bmatrix} A \to c\,.\,d, a \\ B \to c\,., \quad b \end{bmatrix}\right) = \begin{bmatrix} A \to c\,.\,d \\ B \to c\,. \end{bmatrix}$$

and

$$\delta_c^0\left(\text{core}\left(\begin{bmatrix} S \to a\,.\,Aa, \perp \\ A \to .\,cd, \quad a \end{bmatrix}\right), c\right) = \delta_c^0\left(\begin{bmatrix} S \to a\,.\,Aa \\ A \to .\,cd \end{bmatrix}, c\right) = \boxed{A \to c\,.\,d}.$$

Nevertheless, the parsing automaton of Fig. 4 is consistent and satisfies (1).

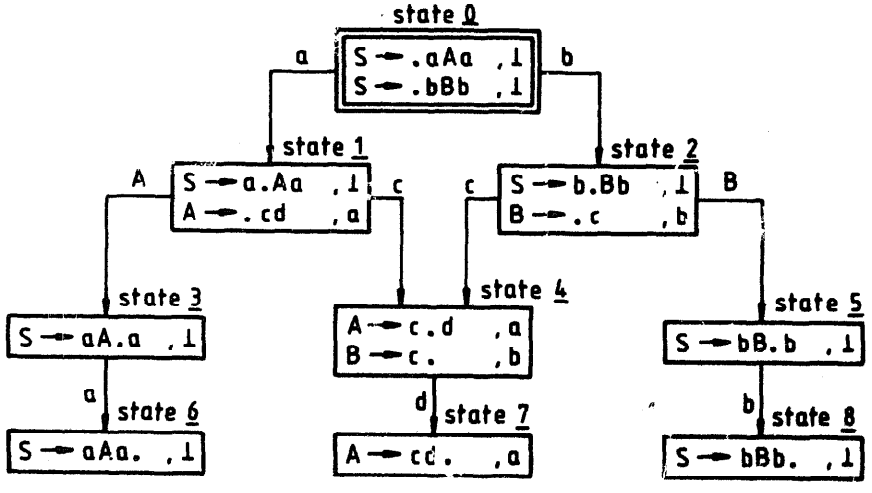

Fig. 4. A not core-restricted parsing automaton for the grammar $G2$ with the rules $S \to aAa$, $A \to cd$, $S \to bBb$, $B \to c$.

To prove (1) for core-restricted automata suppose $[A \to \alpha\,.\,\beta, u] \in \delta^*(q_0, \psi)$. Then

$$[A \to \alpha\,.\,\beta] = \text{core}([A \to \alpha\,.\,\beta, u]) \in \text{core}(\delta^*(q_0, \psi)) = \delta_c^{0*}(q_c^0, \psi)$$

and

$$[S] \overset{*}{\Rightarrow} \psi[A \to \alpha\,.\,\beta] \quad \text{and} \quad \exists \gamma: \psi = \gamma\alpha \tag{4}$$

by the properties of LR(k) automata and the item lemma.

Core-restricted automata generate parsers with good error detection capabilities. These parsers read a portion of the input string only if it is a prefix of a word in the language, that is, these parsers are correct prefix parsers.

Example 4.2. We refer to grammar $G2$ of Fig. 4 and compute for the relation $\vDash$ defined after Theorem 3.4

$$(\varepsilon, \mathbf{0}, bcda, \varepsilon) \vDash (b, \mathbf{02}, cda, \varepsilon) \vDash (bc, \mathbf{024}, da, \varepsilon) \vDash (bcd, \mathbf{0247}, a, \varepsilon).$$

But bcd is not a prefix of a word in the language, i.e., $bcd\Sigma^* \cap L(G2) = \emptyset$ where $L(G2) = \{acda, bcb\}$.

In more formal terms, we claim for parsers generated by core-restricted parsing automata

$$(\varepsilon, q_0, yw, \varepsilon) \overset{*}{\vDash} (\psi, q_0 \cdots q_n, w, \pi) \succ \exists z : S \# \underset{\text{rm}}{\overset{*}{\Rightarrow}} yz. \tag{5}$$

For the proof of (5) recall $\psi \Rightarrow_{\text{rm}}^{\pi} y$ so that it is sufficient to prove $\psi\sigma \in \text{LRC}_0(\Pi)$ for some σ. Note that $\delta^*(q_0, \psi) = q_n$ is defined and not empty. If $[A \to \alpha . \beta, u] \in \delta^*(q_0, \psi)$, then (4) and Theorem 3.2 imply $\psi\beta \in \text{LRC}_0(\Pi)$.

We shall now discuss another advantage of core-restricted parsing automata: if consistent they generate parsers operating in linear time, i.e., the number of steps executed by such a parser up to any moment of its operation is bounded by a linear function of the length of the portion of the input string which was read up to this moment. Note that operating in linear time implies halting on every input string in linear time whether or not this input string belongs to the language of the underlying grammar. We shall formalize this statement in the next definition and prove it in Theorem 4.4.

Definition 4.2. The relation $\vDash$ (see Definition 3.4) *operates in linear time* if there are integers c, d such that

$$(\varepsilon, q_0, wz, \varepsilon) \overset{j}{\vDash} (\psi, \ldots, z, \pi) \text{ implies } j \leqslant c|w| + d.$$

Example 4.3. We note that core-restriction is an essential condition by considering the consistent parsing automaton of Fig. 5.

For the relation $\vDash$ associated with this automaton we have

$$(\varepsilon, \mathbf{0}, aa, \varepsilon) \vDash (a, \mathbf{01}, a, \varepsilon) \overset{i}{\vDash} (aB^i, \mathbf{012}^i, a, (B \to \varepsilon)^i)$$

which means that $\vDash$ does not operate in linear time.

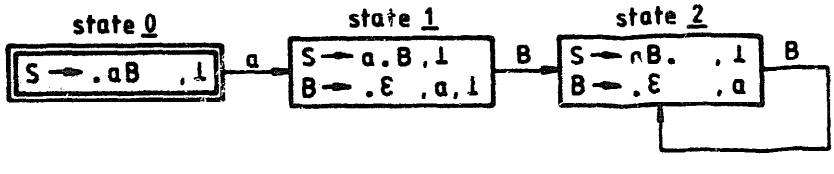

Fig. 5. A parsing automaton for the grammar with the rules $S \to aB$, $B \to \varepsilon$.

The proof of the announced theorem requires a theorem on context-free grammars which is rather obvious but could not be found in the literature. For completeness, we present its proof.

Theorem 4.1. *For every cycle-free grammar there are integers a, b such that*

$$A \overset{j}{\Rightarrow} w \text{ implies } j \leq a|w| + b. \tag{6}$$

Proof. First, we show that the number

$$e = \max(\{m \mid \exists A \in N: A \overset{m}{\Rightarrow} \varepsilon\} \cup \{1\}) \tag{7}$$

is well defined. It is not hard to prove the following implication for all $A \in N$

$$(\forall n\ \exists m > n: A \overset{m}{\Rightarrow} \varepsilon) \text{ implies } \exists B: (A \overset{+}{\Rightarrow} B \wedge \forall n\ \exists m > n: B \overset{m}{\Rightarrow} \varepsilon). \tag{8}$$

Using (8) one readily obtains a sequence $A_0 \Rightarrow^+ A_1 \Rightarrow^+ \cdots \Rightarrow^+ A_{|N|} \Rightarrow^+ \varepsilon$ if e is not well defined by (7). This sequence must contain a cycle, which is a contradiction.

Let l be the length of the longest right side of a production rule and define $p = |N| + 1$. We prove that

$$X \overset{j}{\Rightarrow} w \neq \varepsilon \text{ implies } j \leq (1 + le)[(2p-1)(|w|-1) + (p-1)] \tag{9}$$

by induction on $|w|$. We do the induction step first. Suppose, $|w| \geq 2$. Then, for some $n \geq 1$, $m \geq 2$ there are strings and numbers such that

$$X = A_1 \Rightarrow \alpha_2 A_2 \beta_2 \wedge \alpha_2 \beta_2 \overset{s_2}{\Rightarrow} \varepsilon \wedge A_2 \overset{*}{\Rightarrow} w \wedge \cdots$$

$$\wedge A_{n-1} \Rightarrow a_n A_n \beta_n \wedge \alpha_n \beta_n \overset{s_n}{\Rightarrow} \varepsilon \wedge A_n \overset{*}{\Rightarrow} w \tag{10}$$

and

$$A_n \Rightarrow \gamma_1 X_1 \gamma_2 \cdots \gamma_m X_m \gamma_{m+1} \wedge \gamma_1 \cdots \gamma_{m+1} \overset{s_{n+1}}{\Longrightarrow} \varepsilon \wedge X_1$$

$$\overset{t_1}{\Rightarrow} w_1 \neq \varepsilon \wedge \cdots \wedge X_m \overset{t_m}{\Rightarrow} w_m \neq \varepsilon \wedge w_1 \cdots w_m = w$$

$$\wedge j = (1 + s_2) + (1 + s_3) + \cdots + (1 + s_{n+1}) + t_1 + \cdots + t_m. \tag{11}$$

If $n \geq |N| + 1 = p$, then $A_1 \Rightarrow^+ A_2 \Rightarrow^+ \cdots \Rightarrow^+ A_{|N|+1}$ contains a cycle. Therefore, $n \leq p - 1$. From (7) we learn $(1 + s_i) \leq (1 + le)$. Because of (11) and $m \geq 2$ we have

$$|w_1| + \cdots + |w_m| = |w| \quad \text{and} \quad \forall i: 0 < |w_i| < |w|$$

so that the induction hypothesis can be applied. We compute

$$\begin{aligned} j &\leq (1+le)n + (1+le)[(2p-1)(|w_1|-1)+(p-1)+\cdots \\ &\quad +(2p-1)(|w_m|-1)+(p-1)] \\ &= (1+le)[(2p-1)(|w_1|+\cdots+|w_m|)-mp+n] \\ &\leq (1+le)[(2p-1)|w|-2p+p-1] \\ &\leq (1+le)[(2p-1)(|w|-1)+p-1]. \end{aligned}$$

This completes the induction step. The case $|w| = 1 \wedge j \neq 0$ is similar. We just replace (11) by the following claim

$$A_n \Rightarrow \alpha_{n+1} w \beta_{n+1} \wedge \alpha_{n+1}\beta_{n+1} \overset{s_{n+1}}{\Longrightarrow} \varepsilon \wedge j = (1+s_2)+(1+s_3)+\cdots+(1+s_{n+1})$$

so that

$$j \leq n(1+le) \leq (p-1)(1+le) = (1+le)[(2p-1)(|w|-1)+(p-1)].$$

In order to complete the proof of the theorem we use (7) if $|w| = 0$ and claim that it is sufficient to choose $a = (1+le)(2p-1)$ and $b = e$.

Theorem 4.2. *Let the relation $\vDash$ be derived (via Definition 3.4) from a core-restricted parsing automaton for a cycle-free grammar. Then the relation $\vDash$ operates in linear time iff*

$$\neg\exists B, \psi, w: B \overset{*}{\underset{\text{rm}}{\Rightarrow}} \psi B w \wedge \psi \overset{+}{\Rightarrow} \varepsilon. \tag{12}$$

Proof. 'If': Let a and b be the integers according to Theorem 4.1. We claim that there is another integer p such that

$$\alpha\rho\beta \in \mathrm{LRC}_0(\Pi) \wedge \rho \overset{*}{\Rightarrow} \varepsilon \text{ implies } |\rho| < p. \tag{13}$$

We defer the proof of (13) for a moment and assume

$$(\varepsilon, q_0, wz, \varepsilon) \overset{j}{\vDash} (\psi, \ldots, z, \pi).$$

We know $\psi \Rightarrow_{\text{rm}}^{\pi} w$. Therefore, we may write ψ as $\psi = \psi_0 X_1 \psi_1 X_2 \cdots \psi_{n-1} X_n \psi_n$ where

$$X_i \overset{j_i}{\underset{\text{rm}}{\Longrightarrow}} x_i \neq \varepsilon \wedge \psi_i \overset{m_i}{\underset{\text{rm}}{\Longrightarrow}} \varepsilon \wedge x_1 \cdots x_n = w \wedge |\pi| = \sum_{i=1}^{n} j_i + \sum_{i=0}^{n} m_i.$$

From the proof of (5) we know $\psi\sigma \in \mathrm{LRC}_0(\Pi)$ for some σ so that we may apply (13) to conclude $|\psi_i| < p$. Obviously, $j = |w| + |\pi|$. Putting things together we compute

$$\begin{aligned} j &= |x_1 \cdots x_n| + m_0 + j_1 + m_1 + \cdots + j_n + m_n \\ &\leqslant |x_1 \cdots x_n| + pb + (a|x_1| + b) + pb + (a|x_2| + b) + \cdots + pb \\ &= |x_1 \cdots x_n| + (n+1)pb + nb + a(|x_1| + \cdots + |x_n|) \\ &= (1+a)|x_1 \cdots x_n| + n(pb+b) + pb \\ &\leqslant (1 + a + (pb+b))|x_1 \cdots x_n| + pb = (1 + a + pb + b)|w| + pb. \end{aligned}$$

We still have to verify (13). If there is no such constant p, then there must be some $\alpha X_1 \cdots X_n \beta u' \in \mathrm{LRC}(\Pi)$ where $X_i \Rightarrow^* \varepsilon$ and where n is larger than the number of items of degree k. Theorem 3.2 gives us a derivation

$$[S] \overset{*}{\Rightarrow} \alpha[\ ] \overset{*}{\Rightarrow} \alpha X_1[\ ]_1 \overset{*}{\Rightarrow} \alpha X_1 X_2[\ ]_2 \overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} \alpha X_1 \cdots X_n[\ ]_n \overset{*}{\Rightarrow} \cdots.$$

By the choice of n for some $i < j$ we have $[\ ]_i = [\ ]_j$. We may split

$$[\ ]_i \overset{*}{\Rightarrow} X_{i+1} \cdots X_j[\ ]_j$$

into

$$[\ ]_i \overset{*}{\Rightarrow} \gamma[A \to \rho\,.\,B\beta\sigma, u] \Rightarrow \gamma[B \to .\,\delta, v] \overset{*}{\Rightarrow} \gamma\gamma'[\ ]_j$$

because of the structure of the item grammars. We note $\gamma\gamma' \Rightarrow^+ \varepsilon$. $[\ ]_i = [\ ]_j$ permits a continuation

$$\gamma\gamma'[\ ]_j \Rightarrow \gamma\gamma'\gamma[A \to \rho\,.\,B\beta\sigma, u]$$

so that

$$[B \to .\,\delta, v] \Rightarrow \gamma'\gamma[A \to \rho\,.\,B\beta\sigma, u]$$

and $\gamma'\gamma \Rightarrow^+ \varepsilon$. The item lemma implies $Bv \Rightarrow \delta v \Rightarrow^*_{\mathrm{rm}} \gamma'\gamma B\sigma y$ for some y which contradicts (12).

'Only if': Suppose

$$S \neq \overset{*}{\underset{\mathrm{rm}}{\Rightarrow}} \alpha B y \wedge B \overset{*}{\underset{\mathrm{rm}}{\Rightarrow}} \psi B w \wedge \psi \neq \varepsilon \wedge \psi \overset{*}{\Rightarrow} \varepsilon$$

so that

$$\forall n: S \neq \overset{*}{\underset{\mathrm{rm}}{\Rightarrow}} \alpha \psi^n B w^n y.$$

Choose x such that $\alpha \Rightarrow^* x$. Then

$$\forall n\ \exists j\ \exists z\ \exists \pi\ \exists m\ \exists q_1, \ldots, q_m: (\varepsilon, q_0, xz, \varepsilon) \overset{j}{\vDash} (\alpha\psi^n, q_0 \cdots q_m, z, \pi) \wedge j \geqslant n.$$

This proves that $\vDash$ does not operate in linear time.

The next definition describes a large class of grammars which satisfy (12).

Definition 4.3. Let $\Phi = \{F_1, \ldots, F_r\}$ be a finite partition of Σ^*. A grammar is an *LR(Φ) grammar* if

$$S \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha A w \underset{\text{rm}}{\Rightarrow} \alpha\beta w \wedge S \underset{\text{rm}}{\overset{*}{\Rightarrow}} \gamma B x \underset{\text{rm}}{\Rightarrow} \alpha\beta y \wedge \exists i: \{w, y\} \subseteq F_i$$

implies $\alpha A = \gamma B \wedge x = y$.

The definition is from [7] except that we omit the requirement that Φ consists of regular sets. It is well known [7] that every LR(k) grammar is LR(Φ) for some Φ.

Lemma 4.3. LR(Φ) *grammars satisfy* (12) *of Theorem* 4.2.

Proof. Suppose $B \Rightarrow_{\text{rm}}^* \psi B w \wedge \psi \Rightarrow^+ \varepsilon$. If $w = \varepsilon$, then $B \Rightarrow^+ B$ which contradicts unambiguity. Hence, $w \neq \varepsilon$. There are strings α, z such that for all $m, n \geq 1$ we have

$$S \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha B z \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha \psi^n B w^n z \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha \psi^n \psi^m B w^m w^n z. \tag{14}$$

Useless variables are excluded so that $B \Rightarrow_{\text{rm}}^* x$ and $\psi^m B \Rightarrow_{\text{rm}}^* x$ for some $x \in \Sigma^*$. Because of $\psi \neq \varepsilon$ these two derivations are different and can be split such that we have

$$B \underset{\text{rm}}{\overset{*}{\Rightarrow}} \sigma C y \underset{\text{rm}}{\Rightarrow} \sigma\gamma y \underset{\text{rm}}{\overset{*}{\Rightarrow}} x \wedge \psi^m B \underset{\text{rm}}{\overset{*}{\Rightarrow}} \rho D y' \underset{\text{rm}}{\Rightarrow} \rho\delta y' \underset{\text{rm}}{\overset{*}{\Rightarrow}} x \tag{15}$$

where $\rho\delta y' = \sigma\gamma y$ and $C \to \gamma \neq D \to \delta$. Inserting (15) into (14) yields

$$S \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha\psi^n\sigma C y w^n z \underset{\text{rm}}{\Rightarrow} \alpha\psi^n\sigma\gamma y w^n z \wedge S \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha\psi^n\rho D y' w^m w^n z \underset{\text{rm}}{\Rightarrow} \alpha\psi^n\sigma\gamma y w^m w^n z.$$

A contradiction is immediate if we can choose m, n such that $\{y w^n z, y w^m w^n z\} \subseteq F_i$ for some i. Such a choice is obviously possible if we note that Φ is a finite partition and $\{y w^i z \mid i > 0\}$ is an infinite set.

Theorem 4.4. *The relation* $\vDash$ *operates in linear time if it is derived from a core-restricted parsing automaton for an* LR(Φ) *grammar or from a consistent, core-restricted parsing automaton.*

Proof. Theorem 4.2, Lemma 4.3 and Theorem 3.4.

This theorem generalizes theorems from [2] and [13] in several ways. Theorem 5.13 of [2] states that $\vDash$ operates in linear time on words in the language of the

underlying grammar in the special case of a consistent LR(k) automaton. Its proof is rather informal and not correct.[3] The theorems and remarks in Section 9 of [13] imply that, for a consistent, core-restricted parsing automaton, the parsing algorithm described by $\models$ halts on every input. Our theorem encompasses both statements and applies to a larger class of parsing algorithms including those which are discussed in Sections 6 and 7 of [7].

5. Special parsing automata

In this section we shall show that the practical parsing methods of DeRemer and Pager are based on core-restricted parsing automata. These automata are of manageable size for practical grammars whereas LR(k) automata have a prohibitive number of states for $k \geq 1$ (cf. [4]).

First, we shall discuss SLR(k) grammars. For this purpose we introduce the *ahead-function* by

$$\text{ahead}(A) = \text{ahead}_k(A) = \{u \mid \exists \alpha, w: S \# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \alpha A u w\}$$

and define the *ahead-completion acompl*(q^0) of a state $q^0 \in Q_c^0$ by

$$\text{acompl}(q^0) = \{[A \to \alpha \,.\, \beta, u] \mid [A \to \alpha \,.\, \beta] \in q^0 \wedge u \in \text{ahead}(A)\}.$$

Definition 5.1. The *SLR(k)-automaton* $\mathfrak{A}_s = (Q_s, V, \delta_s, q_s, \emptyset)$ is defined by $q_s = \text{acompl}(q_c^0)$ and $Q_s = \{\text{acompl}(q^0) \mid q^0 \in Q_c^0\}$ and

$$\delta_s(\text{acompl}(q^0), X) = \text{acompl}(\delta_c^0(q^0, X)).$$

A grammar is an *SLR(k) grammar* if $\mathfrak{A}_s$ is consistent.

In less formal terms, we may say that the SLR(k) automaton may be obtained from the LR(0) automaton by replacing every item $[A \to \alpha \,.\, \beta]$ by the set $\{[A \to \alpha \,.\, \beta, u] \mid u \in \text{ahead}(A)\}$ without changing the structure of the automaton.

We note $\text{ahead}(A) \neq \emptyset$, because there are no useless variables, so that acompl: $Q_c^0 \to Q_s$ is a bijection, and δ_s is well defined. Using the item lemma it is not hard to verify that $\mathfrak{A}_s$ is a parsing automaton. $\mathfrak{A}_s$ is core-restricted by definition so that the SLR(k) parsing method has all the desirable properties derived in Section 4. Our SLR(k) definition agrees with the one given in [13] where its relation to other SLR(k) definitions is discussed.

If we define the *SLR(k) contexts* by $\text{SLRC}(A \to \beta) = \text{LRC}_0(A \to \beta) \cdot \text{ahead}(A)$, then Lemma 3.3 as applied to the LR(0) automaton implies

$$\psi u \in \text{SLRC}(A \to \beta) \Leftrightarrow [A \to \beta \,.\,, u] \in \delta_s^*(q_s, \psi).$$

[3] The characteristic of a situation must be defined in another way. The author is grateful to a referee for these hints concerning Theorem 5.13 of [2].

This formula suggests an SLR(k) criterion similar to the LR(k) criterion of Theorem 2.4. Following Backhouse [5] we call a grammar a *BSLR(k) grammar* if

$$\psi u \in \mathrm{SLRC}(A \to \beta) \wedge \psi u x \in \mathrm{SLRC}(B \to \delta) \succ x = \varepsilon \wedge A \to \beta = B \to \delta$$

holds. The remarks following Theorem 4.5 of [5] claim that a grammar is BSLR(k) iff it is SLR(k). Unfortunately, this is not true. There is an SLR(2) grammar which is not BSLR(2) as may be seen by computing the SLR(2) automaton for the grammar with the rules $S \to aB$, $S \to dBbd$, $S \to Aab$, $A \to \varepsilon$, $B \to \varepsilon$. This automaton is consistent, whereas $ab \in \mathrm{SLRC}_2(A \to \varepsilon)$ and $abd \in \mathrm{SLRC}_2(B \to \varepsilon)$.

Using the methods of [23] one can even show that it is undecidable whether there is a k such that an arbitrary SLR(2) grammar is BSLR(k). The next theorem describes the exact relationship between the two grammar classes. A proof is given in [18].

Theorem 5.1. *A grammar is* BSLR(1) *iff it is* SLR(1). *Every* BSLR(k) *grammar is* SLR(k).

Next we consider the LALR(k) method. The LALR(k) automaton is obtained from the LR(k) automaton by merging states with equal cores. To be precise, define the *lookahead-completion* of any $q^0 \in Q_c^0$ by

$$\mathrm{lcompl}(q^0) = \{[\ ] \mid \exists q \in Q_c\colon [\ ] \in q \wedge \mathrm{core}(q) = q^0\} = \bigcup_{\substack{q \in Q_c \\ \mathrm{core}(q) = q^0}} q$$

where Q_c belongs to the LR(k) automaton.

Example 5.1. We refer to Fig. 2 and compute

$$\mathrm{lcompl}\left(\begin{array}{|l|}\hline A \to c\,.\,A \\ A \to .\,cA \\ A \to .\,dd \\ \hline\end{array}\right) = \begin{array}{|ll|}\hline A \to c\,.\,A, & a \\ A \to .\,cA, & a \\ A \to .\,dd, & a \\ \hline\end{array} \cup \begin{array}{|ll|}\hline A \to c\,.\,A, & b \\ A \to .\,cA, & b \\ A \to .\,dd, & b \\ \hline\end{array} = \begin{array}{|ll|}\hline A \to c\,.\,A, & a, b \\ A \to .\,cA, & a, b \\ A \to .\,dd, & a, b \\ \hline\end{array},$$

$$\mathrm{lcompl}\left(\begin{array}{|l|}\hline B \to cdd\,. \\ A \to dd\,. \\ \hline\end{array}\right) = \begin{array}{|ll|}\hline B \to cdd\,., & a \\ A \to dd\,., & b \\ \hline\end{array} \cup \begin{array}{|ll|}\hline B \to cdd\,., & b \\ A \to dd\,., & a \\ \hline\end{array} = \begin{array}{|ll|}\hline B \to cdd\,., & a, b \\ A \to dd\,., & a, b \\ \hline\end{array}.$$

Note that the last set of items is not consistent so that $G1$ of Fig. 2 is not an LALR(1) grammar according to the following definition.

Definition 5.2. The *LALR(k) automaton* $\mathfrak{A}_1 = (Q_1, V, \delta_1, q_1, \emptyset)$ is defined by $q_1 = \mathrm{lcompl}(q_c^0) = q_c$ and $Q_1 = \{\mathrm{lcompl}(q^0) \mid q^0 \in Q_c^0\}$ and by

$$\delta_1(\mathrm{lcompl}(q^0), X) = \mathrm{lcompl}(\delta_c^0(q^0, X)).$$

A grammar is an *LALR(k) grammar* if its LALR(k) automaton is consistent.

The formula $q^0 = \text{core}(\text{lcompl}(q^0))$ is immediate so that δ_1 is well defined. A simple sequence of set manipulations shows that $\mathfrak{A}_1$ is a parsing automaton. Core-restriction is immediate, too. Therefore, all the statements of Section 4 apply to LALR(k) automata. It is not immediate that the LALR(k) automaton may be computed without knowing the LR(k) automaton. We shall provide an algorithm later on (Algorithm 5.A). The next lemma can be used to verify that Fig. 3 shows an LALR(1) automaton.

Lemma 5.2. *A parsing automaton* $\mathfrak{A} = (Q, V, \delta, q_0, \emptyset)$ *is the* LALR(k) *automaton iff*

(i) *it is isomorphic to the* LR(0) *automaton (via the function core in the sense of automata theory),*

(ii) *its initial state agrees with the initial state of the* LR(k) *automaton, and*

(iii)

$$[\ ] \in q \text{ implies } \exists \psi\colon \delta^*(q_0, \psi) = q \wedge [S] \overset{*}{\Rightarrow} \psi[\ ]$$

for all $q \in Q$.

Proof. Assume that (i)–(iii) are true. The reverse implication of (iii) is true, too, because of the definition of parsing automata. We define the inverse isomorphism $\text{compl}: Q_c^0 \to Q$ by $\text{compl}(\text{core}(q^0)) = q^0$ and compute

$$\begin{aligned}
[\ ] \in \text{compl}(q^0) &\Leftrightarrow \exists \psi\colon \delta^*(q_c, \psi) = \text{compl}(q^0) \wedge [S] \overset{*}{\Rightarrow} \psi[\ ] \\
&\Leftrightarrow \exists \psi\colon \delta_c^{0*}(\text{core}(q_c), \psi) = \text{core}(\text{compl}(q^0)) \wedge [\ ] \in \delta_{cs}^*(q_c, \psi) \\
&\Leftrightarrow \exists \psi\colon \text{core}(\delta_c^*(q_c, \psi)) = q^0 \wedge [\ ] \in \delta_c^*(q_c, \psi) \\
&\Leftrightarrow \exists q \in Q_c\colon \text{core}(q) = q^0 \wedge [\ ] \in q \\
&\Leftrightarrow [\ ] \in \text{lcompl}(q^0).
\end{aligned}$$

This proves compl = lcompl as required. The only-if part is still simpler.

For the remainder of this section we assume $k = 1$.

Definition 5.3. A set q of items of degree one has a *shift-reduce conflict* if q contains items $[A \to \alpha\,.\,a\beta, b]$ and $[B \to \delta\,.\,, a]$.

It has a *reduce-reduce conflict* if it contains items $[A \to \beta\,.\,, a] \neq [B \to \delta\,.\,, a]$.

Lemma 5.3. *The* LALR(1) *automaton has a shift-reduce conflict iff the* LR(1) *automaton has a shift-reduce conflict.*

Proof. Let $\{[A \to \alpha\,.\,a\beta, b], [B \to \delta\,.\,, a]\} \subseteq q \in Q_1$. Then $[B \to \delta\,.\,, a] \in q'$ for some $q' \in Q_c$ with $\text{core}(q') = \text{core}(q)$. By definition of core there is some c such that $[A \to \alpha\,.\,a\beta, c] \in q'$.

The lemma does not generalize to LALR(2) automata as may be seen by inspecting the grammar with the rules $S \to Aa$, $S \to Bbb$, $S \to dC$, $A \to cb$, $B \to c$, $C \to Ab$, $C \to Bba$.

Lemma 5.3 leads to Pager's parsing method [27]. Suppose that we are given an LR(1) grammar with an inconsistent LALR(1) automaton. The inconsistency must be due to a reduce-reduce conflict (because of Lemma 5.3) and was introduced by merging two consistent states of the LR(1) automaton. Can we prevent mergers which cause reduce-reduce conflicts? To provide some motivation for a formal discussion of this question we present the parameterized, nondeterministic Algorithm 5.A which computes parsing automata for arbitrary grammars. In this algorithm the merging process is controlled by a relation R on sets of items.

Algorithm 5.A

$Q := \{q_c\}$;

while $\exists X\ \exists q \in Q \neg \exists q' \in Q : \delta_{cs}(q, X) \subseteq q' \wedge \text{core}(\delta_{cs}(q, X)) = \text{core}(q')$

do choose such X, q and let $q' := \delta_{cs}(q, X)$;
if for some $q'' \in Q$ we have $q'Rq''$
then $Q := Q \backslash \{q''\} \cup \{q' \cup q''\}$
else $Q := Q \cup \{q'\}$ **fi**

od
for all X **for all** $q \in Q$ such that $\delta_{cs}(q, X) \neq \emptyset$ **do**
choose $q' \in Q$ such that $\delta_{cs}(q, X) \subseteq q' \wedge \text{core}(\delta_{cs}(q, X)) = \text{core}(q')$
and let $\delta(q, X) := q'$;

do

Algorithm 5.A computes core-restricted automata whenever qRq' implies $\text{core}(q) = \text{core}(q')$. Lemma 5.2 can be used to verify that it computes the LALR(k) automaton if R is defined by 'qRq' iff $\text{core}(q) = \text{core}(q')$'. In order to prove properties for this algorithm we associate sets $Q(i, R)$ with each relation R inductively by $Q(0, R) = Q_c$ and by

$$Q(i+1, R) = \{q \cup q' \mid q, q' \in Q(i, R) \wedge qRq'\} \cup Q(i, R).$$

We add

$$Q(R) = \bigcup_{i=0}^{\infty} Q(i, R)$$

and note $Q \subseteq Q(R)$ for all Q generated by Algorithm 5.A.

Hence, any parsing automaton computed by this algorithm is consistent if $Q(R)$ is. With these notions we may precisely formulate the problem of dangerous mergers:

Can we compute (with reasonable efficiency) relations R such that $Q(R)$ is free of conflicts for LR(1) grammars? We shall now give a formal presentation of Pager's answer [27] in the framework of parsing automata and item grammars.

Definition 5.4. A set q of items has a *potential (reduce-reduce) conflict* if there is a string ψ such that $\delta_{cs}^*(q, \psi)$ has a reduce-reduce conflict. In this case we write $PC(q)$. A relation R is *compatible* if qRq' implies

$$\text{core}(q) = \text{core}(q') \wedge (\neg PC(q) \wedge \neg PC(q') > \neg PC(q \cup q')).$$

A parsing automaton is *compatible* if its set of states is contained in some $Q(R)$ for a compatible relation R.

Example 5.2. We refer to grammar $G1$ of Fig. 2 and compute

$$\delta_{cs}^*\left(\left|\begin{array}{ll} A \to c\,.\,A, & a, b \\ B \to c\,.\,dd, & b, a \\ A \to .\,cA, & a, b \\ A \to .\,dd, & a, b \end{array}\right|, \text{dd}\right) = \left|\begin{array}{ll} B \to cdd\,., & b, a \\ A \to dd\,., & a, b \end{array}\right|.$$

The resulting set of items has a reduce-reduce conflict between $[B \to cdd\,., a]$ and $[A \to dd\,., a]$. Hence, merging the sets $\{[A \to c\,.\,A, a], [B \to c\,.\,dd, b], [A \to .\,cA, a], [A \to .\,dd, a]\}$ and $\{[A \to c\,.\,A, b], [B \to c\,.\,dd, a], [A \to .\,cA, b], [A \to .\,dd, b]\}$ introduces a potential conflict. If we apply Algorithm 5.A we must not merge these two sets if inconsistent states are to be avoided.

Compatible relations do not introduce new potential conflicts. Moreover, the next lemma states that in this way they prevent the introduction of conflicts.

Lemma 5.4. *For compatible relations $Q(R)$ is consistent iff the* LR(1) *automaton is consistent. A compatible automaton of degree one is consistent iff the* LR(1) *automaton is consistent.*

Proof. Assume that Q_c is consistent and that $q \in Q(R)$ has a conflict. It must be a reduce-reduce conflict (cf. the proof of Lemma 5.3). Hence, $PC(q)$, and $PC(q')$ for some $q' \in Q_c$. Thus, $\delta_{cs}^*(q', \psi) \in Q_c$ has a conflict for some ψ which is a contradiction. Considering $Q_c \subseteq Q(R)$ completes the proof.

Of course, we still do not know how to compute compatible relations. A partial answer is provided by the next lemma.

Lemma 5.5. *If* $\text{core}(q_1) = \text{core}(q_2) \wedge \neg PC(q_1) \wedge \neg PC(q_2)$, *then*

$$PC(q_1 \cup q_2) \Leftrightarrow \exists [A \to \alpha\,.\,\beta, a] \in q_1\, \exists [B \to \rho\,.\,\sigma, b] \in q_2:$$

$$a = b \wedge [A \to \alpha \,.\, \beta] \neq [B \to \rho \,.\, \sigma]$$

$$\wedge \exists [C \to \gamma \,.\,] \neq [D \to \delta \,.\,], \psi:$$

$$[A \to \alpha \,.\, \beta, \perp] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, \perp]$$

$$\wedge [B \to \rho \,.\, \sigma, \perp] \overset{*}{\Rightarrow} \psi[D \to \delta \,.\,, \perp].$$

Example 5.3. We refer to $G1$ and to the previous example. We note

$$[B \to c \,.\, dd, b] \in \boxed{\begin{array}{l} A \to c \,.\, A, \ a \\ B \to c \,.\, dd, b \\ A \to .\, cA, \ a \\ A \to .\, dd, \ a \end{array}} \wedge [A \to .\, dd, b] \in \boxed{\begin{array}{l} A \to c \,.\, A, \ b \\ B \to c \,.\, dd, a \\ A \to .\, cA, \ b \\ A \to .\, dd, \ b \end{array}}$$

and

$$[B \to c \,.\, dd] \neq [A \to .\, dd] \wedge [B \to cdd \,.\,] \neq [A \to dd \,.\,]$$

and

$$[B \to c \,.\, dd, \perp] \overset{*}{\Rightarrow} dd[B \to cdd \,.\,, \perp] \wedge [A \to .\, dd, \perp] \overset{*}{\Rightarrow} dd[A \to dd \,.\,, \perp]$$

which fulfils the right-hand side of the condition of Lemma 5.5.

Proof of Lemma 5.5. We use three formulae which are consequences of $\Pi \subseteq N \times (N \cup \Sigma \backslash \{\perp\})^*$ and the item lemma:

$$[A \to \alpha \,.\, \beta, a] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, d] \wedge d \neq a$$

$$> \forall c: [A \to \alpha \,.\, \beta, c] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, d], \tag{1}$$

$$[A \to \alpha \,.\, \beta, \perp] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, \perp] > \forall a: [A \to \alpha \,.\, \beta, a] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, a], \tag{2}$$

$$[A \to \alpha \,.\, \beta, a] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, a] \wedge \neg([A \to \alpha \,.\, \beta, \perp] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, \perp])$$

$$> \forall c: [A \to \alpha \,.\, \beta, c] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, a]. \tag{3}$$

A simple application of (2) proves '$<$' of the lemma. On the other hand, if $PC(q_1 \cup q_2) \wedge \neg PC(q_1) \wedge \neg PC(q_2)$, then there are strings and items such that

$$[A \to \alpha \,.\, \beta, a] \neq [B \to \rho \,.\, \sigma, b] \wedge [C \to \gamma \,.\,] \neq [D \to \delta \,.\,], \tag{4}$$

$$[A \to \alpha \,.\, \beta, a] \in q_1 \wedge [A \to \alpha \,.\, \beta, a] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, d], \tag{5}$$

$$[B \to \rho \,.\, \sigma, b] \in q_2 \wedge [B \to \rho \,.\, \sigma, b] \overset{*}{\Rightarrow} \psi[D \to \delta \,.\,, d]. \tag{6}$$

First, we assume $a \neq d$. Eq. (1) and $\text{core}(q_1) = \text{core}(q_2)$ and (5) imply

$$\exists c: [A \to \alpha \,.\, \beta, c] \in q_2 \wedge [A \to \alpha \,.\, \beta, c] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, d]. \tag{7}$$

But (6) $\wedge$ (7) contradicts $\neg PC(q_2)$. The case $b \neq d$ is symmetric. Hence, $a = b = d$. In addition, $[A \to \alpha \,.\, \beta] \neq [B \to \rho \,.\, \sigma]$ because of (4). Assume $\neg([A \to \alpha \,.\, \beta, \bot] \Rightarrow^* \psi[C \to \gamma \,.\,, \bot])$. Then $\text{core}(q_1) = \text{core}(q_2)$ and (5) and (3) – recall $a = d$ – yield (7) which is again a contradiction. A symmetric argument applies to $\neg([B \to \rho \,.\, \sigma, \bot] \Rightarrow^* \psi[D \to \delta \,.\,, \bot])$.

In order to test the condition of Lemma 5.5 we define a relation μ by

$$\begin{bmatrix} A \to \alpha \,.\, X\beta \\ B \to \rho \,.\, X\sigma \end{bmatrix} \mu \begin{bmatrix} A \to \alpha X \,.\, \beta \\ B \to \rho X \,.\, \sigma \end{bmatrix} \quad \text{for all } X,$$

$$\begin{bmatrix} A \to \alpha \,.\, C\beta \\ B \to \rho \,.\, \sigma \end{bmatrix} \mu \begin{bmatrix} C \to \,.\, \gamma \\ B \to \rho \,.\, \sigma \end{bmatrix} \quad \text{whenever } \beta \overset{*}{\Rightarrow} \varepsilon,$$

$$\begin{bmatrix} A \to \alpha \,.\, \beta \\ B \to \rho \,.\, D\sigma \end{bmatrix} \mu \begin{bmatrix} A \to \alpha \,.\, \beta \\ D \to \,.\, \delta \end{bmatrix} \quad \text{whenever } \sigma \overset{*}{\Rightarrow} \varepsilon.$$

Straightforward induction shows

$$\exists \psi: [A \to \alpha \,.\, \beta, \bot] \overset{*}{\Rightarrow} \psi[C \to \gamma \,.\,, \bot] \wedge [B \to \rho \,.\, \sigma, \bot] \overset{*}{\Rightarrow} \psi[D \to \delta \,.\,, \bot]$$
$$\Leftrightarrow \begin{bmatrix} A \to \alpha \,.\, \beta \\ B \to \rho \,.\, \sigma \end{bmatrix} \mu^* \begin{bmatrix} C \to \gamma \,. \\ D \to \delta \,. \end{bmatrix}.$$

We note that the recursive procedure suggested in [27] computes μ^*. Any transitive closure algorithm may be used for this purpose as well. Let R_s be defined such that $q_1 R_s q_2$ iff $\text{core}(q_1) = \text{core}(q_2)$ and

$$\forall [A \to \alpha \,.\, \beta, a] \in q_1 \; \forall [B \to \rho \,.\, \sigma, b] \in q_2:$$
$$a \neq b \vee [A \to \alpha \,.\, \beta] = [B \to \rho \,.\, \sigma] \vee$$
$$\neg \exists [C \to \gamma \,.\,] \neq [D \to \delta \,.\,]: \begin{bmatrix} A \to \alpha \,.\, \beta \\ B \to \rho \,.\, \sigma \end{bmatrix} \mu^* \begin{bmatrix} C \to \gamma \,. \\ D \to \delta \,. \end{bmatrix}.$$

Lemma 5.5 and the above observation show that R_s is compatible so that $Q(F_s)$ is consistent iff we start with an LR(1) grammar. Obviously, $PC(q_1 \cup q_2)$ if $\neg q_1 R_s q_2$ and $\text{core}(q_1) = \text{core}(q_2)$. Hence, R_s forces Algorithm 5.A to compute a minimal consistent core-restricted automaton. Of course, this is the LALR(1) automaton for LALR(1) grammars. A simple proof shows that it is sufficient to test for '$\text{ess}(q_1) R_s \text{ess}(q_2)$' instead of '$q_1 R_s q_2$' in the algorithm where $\text{ess}(q) = \{[A \to \alpha \,.\, \beta, a] \in q \mid \alpha \neq \varepsilon\}$.

The next lemma may be used to define a compatible merging relation (called 'weak' in [27]) which is simpler than R_s.

Lemma 5.6. *If*

$$\text{core}(q_1) = \text{core}(q_2) \wedge \neg PC(q_1) \wedge \neg PC(q_2) \wedge PC(q_1 \cup q_2),$$

then

$$\exists [A \to \alpha \,.\, \beta, a] \in q_1, [B \to \rho \,.\, \sigma, b] \in q_2:$$

$$a = b \wedge [A \to \alpha \,.\, \beta] \neq [B \to \rho \,.\, \sigma] \wedge$$

$$\neg \exists i \; \exists c: [A \to \alpha \,.\, \beta, c] \in q_i \wedge [B \to \rho \,.\, \sigma, c] \in q_i.$$

The proof is not difficult if Lemma 5.5 and the method of its proof are used. Again the test may be restricted to $\text{ess}(q_1)$ and $\text{ess}(q_2)$.

Algorithm 5.A was programmed by a student in PL/I and run on a Burroughs B7700. Execution times for practical grammars (ALGOL, EULER, PL/I) were about 3 minutes which included a lot of time spent on preparing a readable listing of the computed automaton. Testing for $q_1 R_s q_2$ amounted to less than 5% of the execution times. This verifies Pager's claim that R_s gives rise to a practical general method for constructing LR(1) parsers.

6. LL(k) grammars

In this section we shall apply our theory of item grammars to LL(k) grammars. We begin with a simple lemma connecting leftmost and rightmost derivations. As usual, we define

$$L(\psi) = \{x \in \Sigma^* \mid \psi \overset{*}{\Rightarrow} x\}.$$

Lemma 6.1.

$$B \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\psi \text{ implies } \exists \omega: (\omega \overset{*}{\Rightarrow} w \wedge \forall z \in L(\psi): B \underset{\text{rm}}{\overset{*}{\Rightarrow}} \omega A z), \tag{1}$$

$$B \underset{\text{rm}}{\overset{*}{\Rightarrow}} \omega A z \text{ implies } \exists \psi: (\psi \overset{*}{\Rightarrow} z \wedge \forall w \in L(\omega): B \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\psi). \tag{2}$$

Proof. Use induction on the length of the derivations for B.

Lemma 6.1 is similar to lemmata of [22] and [33].

Definition 6.1. A grammar is an *LL(k) grammar* if

$$S \not\underset{\text{lm}}{\overset{*}{\Rightarrow}} wC\omega \wedge C \to \gamma \in \Pi \wedge C \to \delta \in \Pi \wedge \text{first}(\gamma\omega) \cap \text{first}(\delta\omega) \neq \emptyset$$

implies $\gamma = \delta$.

Lemma 6.2. *For* LL(k) *grammars, if*

$$[S] \overset{*}{\Rightarrow} \psi[A \to \alpha \,.\, \beta, u] \wedge [S] \overset{*}{\Rightarrow} \psi[B \to \rho \,.\, \sigma, v] \wedge \text{first}(\beta u) \cap \text{first}(\sigma v) \neq \emptyset,$$

then

$$[A \to \alpha \,.\, \beta] \overset{*}{\Rightarrow} [B \to \rho \,.\, \sigma] \vee [B \to \rho \,.\, \sigma] \overset{*}{\Rightarrow} [A \to \alpha \,.\, \beta].$$

Proof. The given derivations can be refined such that

$$\begin{aligned} &[S] \Rightarrow \psi_1[\ ]_1 \Rightarrow \psi_1\psi_2[\ ]_2 \Rightarrow \cdots \Rightarrow \psi_1 \cdots \psi_n[\ ]_n = \psi[A \to \alpha \,.\, \beta, u], \\ &[S] \Rightarrow \psi'_1[\ ]'_1 \Rightarrow \psi'_1\psi'_2[\ ]'_2 \Rightarrow \cdots \Rightarrow \psi'_1 \cdots \psi'_{n'}[\ ]'_{n'} = \psi[B \to \rho \,.\, \sigma, v] \end{aligned} \tag{3}$$

where

$$[\ ]_i = [A_i \to \alpha_i \,.\, \beta_i, u_i] \quad \text{and} \quad [\ ]'_i = [B_i \to \rho_i \,.\, \sigma_i, v_i].$$

Suppose that there is a first index i such that

$$[A_i \to \alpha_i \,.\, \beta_i] \neq [B_i \to \rho_i \,.\, \sigma_i].$$

Then

$$\psi_1 = \psi'_1 \wedge \cdots \wedge \psi_{i-1} = \psi'_{i-1} \quad \text{and} \quad \psi_i \cdots \psi_n = \psi'_i \cdots \psi'_{n'}.$$

Next we show $\psi_i = \psi'_i = \varepsilon$. The choice of i and the structure of the item grammar make $\alpha_i \neq \varepsilon \wedge \rho_i \neq \varepsilon$ impossible. Assume $\rho_i = \varepsilon \wedge \alpha_i \neq \varepsilon$. Then

$$\begin{aligned} &[A_{i-1} \to \alpha_{i-1} \,.\, \beta_{i-1}, u_{i-1}] \Rightarrow \psi_i[A_i \to \alpha_i \,.\, \beta_i, u_i] \wedge \psi_i \neq \varepsilon, \\ &[\ ]'_{i-1} = [A_{i-1} \to \alpha_{i-1} \,.\, \beta_{i-1}, v_{i-1}] \Rightarrow \psi'_i[B_i \to \,.\, \sigma_i, v_i] \wedge \psi'_i = \varepsilon. \end{aligned}$$

Hence, β_{i-1} and consequently $\psi_i \cdots \psi_n = \psi'_i \cdots \psi'_{n'} = \psi'_{i+1} \cdots \psi'_{n'}$ start with B_i. Therefore, there is some j such that

$$[B_i \to \,.\, \sigma_i, v_i] \overset{*}{\Rightarrow} [B_{j-1} \to \,.\, \sigma_{j-1}, v_{j-1}] \Rightarrow B_i[B_j \to \rho_j \,.\, \sigma_j, v_j]$$

which means that σ_{j-1} starts with B_i so that B_i is left-recursive. This is impossible for LL(k) grammars (see e.g. [2, Lemma 8.3]). Thus, $\rho_i = \varepsilon \wedge \alpha_i \neq \varepsilon$ is ruled out. Likewise, $\rho_i \neq \varepsilon \wedge \alpha_i = \varepsilon$ can be ruled out.

We arrive at $\rho_i = \alpha_i = \varepsilon \wedge \psi_i = \psi'_i = \varepsilon \wedge A_i = B_i$ and

$$[A_i \to \,.\, \beta_i, u_i] \overset{*}{\Rightarrow} \eta[A \to \alpha \,.\, \beta, u] \wedge [B_i \to \,.\, \sigma_i, v_i] \overset{*}{\Rightarrow} \eta[B \to \rho \,.\, \sigma, v] \tag{4}$$

where $\eta = \psi_{i+1} \cdots \psi_n = \psi'_{i+1} \cdots \psi'_{n'}$. An application of the item lemma to (4) yields z, z' with

$$\beta_i u_i \overset{*}{\underset{\text{rm}}{\Rightarrow}} \eta\beta uz \wedge \sigma_i v_i \overset{*}{\underset{\text{rm}}{\Rightarrow}} \eta\sigma vz'$$

so that

$$\text{first}^t(\beta_i u_i) \cap \text{first}(\sigma_i v_i) \neq \emptyset.$$

Using $[A_j \to \alpha_j . \beta_j] = [B_j \to \rho_j . \sigma_j]$ for $j < i$ one proves by induction

$$\forall j \leq i \; \exists w_j, \omega_j : S\# \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_j A_j \omega_j \wedge \{u_j, v_j\} \subseteq \text{first}(\omega_j).$$

Summing up, we obtain w, ω such that

$$S\# \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA_i\omega \wedge A_i \to \beta_i \in \Pi \wedge A_i \to \sigma_i \in \Pi \wedge \beta_i \neq \sigma_i$$

and

$$\emptyset \neq \text{first}(\beta_i u_i) \cap \text{first}(\sigma_i v_i) \subseteq \text{first}(\beta_i \omega) \cap \text{first}(\sigma_i \omega).$$

The LL(k) condition yields $\beta_i = \sigma_i$.

We conclude that there is no index i with $[A_i \to \alpha_i . \beta_i] \neq [B_i \to \rho_i . \sigma_i]$. Therefore, one of the two derivations of (3) is a prefix of the other one if the u_i's and v_i's are omitted. This proves the lemma.

In order to give an LL(k) criterion in terms of parsing automata we introduce the notion of LL(k) consistency.

Definition 6.2. A set q of items is *LL(k) consistent* if

$$[C \to . \gamma, u] \in q \wedge [C \to . \delta, v] \in q \wedge \text{first}(\gamma u) \cap \text{first}(\delta v) \neq \emptyset$$

implies $\gamma = \delta$. A parsing automaton is *LL(k) consistent* if all its states are.

We note that the position of the dot is the same in this definition as the position of the ε-nonterminals used in Brosgol's LL(k) criterion [6].

Theorem 6.3. *A grammar is* LL(k) *iff its* LR(k) *automaton is* LL(k) *consistent. A grammar is* LL(k) *iff it has a consistent parsing automaton.*

Proof. The second statement is an immediate consequence of the first one if we note that for any parsing automaton $\mathfrak{A}$ and any state q of the LR(k) automaton there is a state of $\mathfrak{A}$ which contains q. For the proof of the first statement we have to show that a grammar is LL(k) iff

$$[S] \overset{*}{\Rightarrow} \psi[A \to . \beta, u] \wedge [S] \overset{*}{\Rightarrow} \psi[A \to . \sigma, v] \wedge \text{first}(\beta u) \cap \text{first}(\sigma v) \neq \emptyset \qquad (5)$$

implies $\beta = \sigma$.

'Only if': Lemma 6.2 says

$$[A \to . \beta] \overset{*}{\Rightarrow} [A \to . \sigma] \vee [A \to . \sigma] \overset{*}{\Rightarrow} [A \to . \beta].$$

If $\beta \neq \sigma$ these derivations have positive lengths and imply that A is left-recursive which is impossible for LL(k) grammars (see e.g. [2, Lemma 8.3]).

Hence, $\beta = \sigma$ which had to be shown.

'If': Suppose

$$S \# \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\psi \wedge A \to \beta \in \Pi \wedge A \to \sigma \in \Pi \wedge \text{first}(\beta\psi) \cap \text{first}(\sigma\psi) \neq \emptyset.$$

There are strings u, x, v, y such that

$$\psi \overset{*}{\Rightarrow} ux \wedge \psi \overset{*}{\Rightarrow} vy \wedge \text{first}(\beta u) \cap \text{first}(\sigma v) \neq \emptyset.$$

Lemma 6.1 and the item lemma give us a string ω such that

$$[S] \overset{*}{\Rightarrow} \omega[A \to . \beta, u] \wedge [S] \overset{*}{\Rightarrow} \omega[A \to . \sigma, v].$$

An application of (5) yields $\beta = \sigma$ as required.

Testing for the LL(k) condition and parsing LL(k) grammars for $k \geq 2$ is not easy. The algorithms given in the literature either require a grammatical transformation (as in [31]) or the computation of 'local follow sets' (as in [2, Section 5.1]). The above theorem provides for a (theoretically) simple LL(k) test. The idea to use parsing automata can be extended to LL(k) parsing. The following relation $\square$ describes a top–down parsing algorithm which is deterministic iff the underlying automaton is LL(k) consistent. The relation is defined by

$$(aw, qaq_1X_1 \cdots q_nX_n) \square (w, q_1X_1 \cdots q_nX_n)$$

for all $a \in \Sigma$, $w \in \Sigma^*$, $X_1, \ldots, X_n \in V$ and all states $q, q_1, \ldots, q_n$ and by

$$(uw, qAq_1X_1 \cdots q_mX_m) \square (uw, \hat{q}_1Y_1 \cdots \hat{q}_{\hat{m}}Y_{\hat{m}}q_1X_1 \cdots q_mX_m) \Leftrightarrow$$

$$\Leftrightarrow \exists v: [A \to . Y_1 \cdots Y_{\hat{m}}\, v] \in q \wedge u \in \text{first}(Y_1 \cdots Y_{\hat{m}} v)$$

$$\wedge \forall i: \delta^*(q, Y_1 \cdots Y_i) = \hat{q}_i.$$

Theorem 6.4. $(w\#, q_0S) \square^* (\#, \varepsilon) \Leftrightarrow w \in L(G)$.

Proof. Induction on j proves that

$$S\# \underset{\text{lm}}{\overset{j}{\Rightarrow}} wAX_1 \cdots X_m\# \underset{\text{lm}}{\Rightarrow} wY_1 \cdots Y_{\hat{m}}X_1 \cdots X_m\#$$

$$= wyZ_1 \cdots Z_n\# \overset{*}{\Rightarrow} wyz\#$$

implies that there are states $q, \hat{q}_1, \ldots, \hat{q}_{\hat{m}}, q_1, \ldots, q_m, q'_1, \ldots, q'_n$ such that

$$(wyz\#, q_0S)\square^* (yz\#, qAq_1X_1 \cdots q_mX_m)$$
$$\square(yz\#, \hat{q}_1Y_1 \cdots \hat{q}_{\hat{m}}Y_{\hat{m}}q_1X_1 \cdots q_mX_m)$$
$$\square^* (z\#, q'_1Z_1 \cdots q'_nZ_n)$$

and such that for all β, u

$$Z_i \to \beta \in \Pi \wedge u \in \text{first}(\beta Z_{i+1} \cdots Z_n\#) \quad \text{implies} \quad [Z_i \to .\,\beta, u] \in q'_i.$$

This proves half of the theorem. The other half is still simpler.

From $\square$ one easily derives a grammar with nonterminals in $Q \times V$ and rules of the forms $qA \to q_1Y_1 \cdots q_mY_m$ and $qa \to a$ which generate $L(G)$. This grammar is strong LL(k) if G is LL(k) and if an LL(k) consistent automaton was used. A construction with the same effect was given in [31]. Its proof can be adapted to our construction.

Our next theorem is a corollary to Lemma 6.2. It is well known and several proofs have been given [3, 17, 29, 31, 33]. The proof in [31] uses an LR(k) definition which is not equivalent to the usual LR(k) definitions (see [12, Theorem 2.25]). The proofs in [17, 29] are informal and very sketchy. The proof in [3] is convincing but only [33] contains a formal (and long) proof.

Theorem 6.5. *Every* LL(k) *grammar is* LR(k).

Proof. We have to show (see Theorem 3.4) that

$$[S] \overset{*}{\Rightarrow} \psi[A \to \alpha\,.\,, u] \wedge [S] \overset{*}{\Rightarrow} \psi[B \to \rho\,.\,\sigma, v] \wedge u \in \text{first}(\sigma v) \wedge \sigma \in \Sigma V^* \cup \{\varepsilon\}$$

implies $[A \to \alpha\,.] = [B \to \rho\,.\,\sigma]$. This is an immediate consequence of Lemma 6.2 and the structure of the item grammar.

It has been claimed [1] that every LL(1) grammar is LALR(1) and even SLR(1) [3, Exercise 7.4.6]. This is not true. The grammar with the rules

$$S \to Aa, A \to C, C \to \varepsilon, S \to Bb, B \to C, S \to dD, D \to Ab, D \to Ba$$

is a counterexample because it is LL(1) and not LALR(1). Applying the methods of [23] to this example one can even prove that it is undecidable whether for an arbitrary LL(1) grammar there is a $k \geq 0$ such that it is LALR(k).

The remainder of this section is devoted to restrictions for LL(1) grammars which make them LR(0) or LALR(1). We need a rather technical lemma.

Lemma 6.6. *For* LL(k) *grammars*

$$\forall A \in N\ \exists x \neq \varepsilon : A \overset{*}{\Rightarrow} x \tag{7}$$

and

$$[C \to . \gamma] \overset{*}{\Rightarrow} [A \to . \varepsilon] \wedge [C \to . \delta] \overset{*}{\Rightarrow} [B \to . \varepsilon] \succ \gamma = \delta \tag{8}$$

imply

$$[S] \overset{*}{\Rightarrow} \psi[A \to \alpha . , u] \wedge [S] \overset{*}{\Rightarrow} \psi[B \to \rho . \sigma, v] \wedge u \in \text{first}_{k-1}(\sigma v) \wedge [A \to \alpha .] \neq [B \to \rho . \sigma] \succ \alpha = \rho = \varepsilon \wedge \sigma \neq \varepsilon. \tag{9}$$

Proof. Starting with the derivations given in (9) we proceed as in the proof of Lemma 6.2 to obtain strings such that $\gamma \neq \delta$ and

$$[C \to . \gamma, \hat{u}] \overset{*}{\Rightarrow} \eta[A \to \alpha . , u] \wedge [C \to . \delta, \hat{v}] \overset{*}{\Rightarrow} \eta[B \to \rho . \sigma, v] \tag{10}$$

and

$$S \overset{*}{\underset{\text{lm}}{\not\Rightarrow}} wC\omega \wedge \text{first}(\eta u) \cap \text{first}(\eta\sigma v) \subseteq \text{first}(\gamma\omega) \cap \text{first}(\delta\omega).$$

A violation of the LL(k) condition is immediate if $\eta \Rightarrow^* x$ for some $x \neq \varepsilon$. Hence, $L(\eta) = \{\varepsilon\}$. If $\eta \neq \varepsilon$ there is a nonterminal A with $L(A) = \{\varepsilon\}$. Therefore, $\eta = \varepsilon$ so that (10) yields $\alpha = \rho = \varepsilon$. If $\sigma = \varepsilon$, then we apply (8) to the cores of (10) and obtain $\gamma = \delta$ which is a contradiction. Therefore, $\sigma \neq \varepsilon$.

Theorem 6.7. *For $k \geq 1$ every ε-free* LL(k) *grammar is* LR($k-1$).

Proof. Note that ε-free LL(k) grammars satisfy (7) and (8) of Lemma 6.6. If the grammar is not LR($k-1$) we have derivations (see Theorem 3.4)

$$[S] \overset{*}{\Rightarrow} \psi[A \to \alpha . , x] \quad \text{and} \quad [S] \overset{*}{\Rightarrow} \psi[B \to \rho . \sigma, y]$$

in the item grammar of degree $k-1$ such that

$$x, y \in \Sigma^{k-1} \wedge x \in \text{first}_{k-1}(\sigma y) \wedge \sigma \in \Sigma V^* \cup \{\varepsilon\} \wedge [A \to \alpha .] \neq [B \to \rho . \sigma].$$

Using the item lemma twice we find letters a, b such that (9) can be applied with $u = xa$ and $v = yb$. We obtain $\alpha = \varepsilon$ so that $A \to \varepsilon$ is a production rule. This is a contradiction.

Theorem 6.8. *An* LL(1) *grammar is* LALR(1) *if every nonterminal produces at least one nonempty word and if*

$$[C \to . \gamma] \overset{*}{\Rightarrow} [A \to . \varepsilon] \wedge [C \to . \delta] \overset{*}{\Rightarrow} [B \to . \varepsilon] \quad \textit{implies} \quad \gamma = \delta.$$

Proof. Suppose that the LALR(1) automaton is not consistent. Then we find a state $\hat{q}$ and items such that

$$[A \to \alpha\,.\,, a] \in \hat{q} \wedge [B \to \rho\,.\,\sigma, \hat{c}] \in \hat{q} \wedge [A \to \alpha\,.\,] \neq [B \to \rho\,.\,\sigma]$$

and

$$\sigma \in \Sigma V^* \cup \{\varepsilon\} \wedge a \in \text{first}(\sigma\hat{c}).$$

By definition of the LALR(1) automaton (Definition 5.2) we find a letter c and a state q of the LR(1) automaton such that

$$[A \to \alpha\,.\,, a] \in q \wedge [B \to \rho\,.\,\sigma, c] \in q \tag{11}$$

which implies

$$[S] \overset{*}{\Rightarrow} \psi[A \to \alpha\,.\,, a] \wedge [S] \overset{*}{\Rightarrow} \psi[B \to \rho\,.\,\sigma, c]$$

for some ψ. Note that $a \in \text{first}_{k-1}(\sigma c)\Sigma$ is trivial because of $k - 1 = 0$. Lemma 6.6 says $\sigma \neq \varepsilon$. In this case $a \in \text{first}(\sigma\hat{c})$ implies $a \in \text{first}(\sigma c)$ so that q is not consistent because of (11). This is a contradiction because the grammar is LR(1) by Theorem 6.5.

It may seem that Theorem 6.8 imposes strong restrictions on LL(1) grammars. However, they are weak enough to apply to the usual LL(1) grammar for arithmetic expressions. This grammar has the rules

$$E \to TE',\ E' \to +TE',\ E' \to \varepsilon,\ T \to FT',\ T' \to *FT',\ T' \to \varepsilon,\ F \to (E),\ F \to a$$

and is obtained by eliminating left-recursion from the grammar in Fig. 3.

Theorem 6.8 does not generalize to $k > 1$ because there is an ε-free LL(2) grammar which is not LALR(k) for all $k \geq 0$. The grammar with the rules

$$S \to Aa,\ A \to c,\ S \to Bb,\ B \to c,\ S \to dC,\ C \to Ab,\ C \to Ba$$

is an example of this kind. It is even a strong LL(2) grammar.

7. LC(k) grammars

A number of theorems and tests for LC(k) grammars have appeared in the literature [2, 6, 8, 30, 33, 34]. However, proofs tend to be long, and difficult to state formally. In fact, no proofs were given until [8, 33]. In this section we see that LC(k) theory is made more tractable by using the parsing automata approach. We use the LC(k) definition of [33, 34] as the basis of our discussion. This definition is equivalent to the definitions used in [6, 8] and differs from the original definition [30] only for a technical reason. (Proofs are given in [19].) We shall not make use of these facts and, therefore, we neither prove the equivalences nor even state the other LC(k) definitions.

Definition 7.1. A grammar is an *LC(k) grammar* if

$$S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \psi A w \underset{\text{rm}}{\Rightarrow} \psi X\beta w \wedge \text{first}(\beta w) \cap \text{first}(\sigma y) \neq \emptyset \wedge$$

$$S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \omega B y \underset{\text{rm}}{\Rightarrow} \omega\rho X\sigma y = \psi X\sigma y >$$

$$> \psi A = \omega B \wedge X\beta = \rho X\sigma \tag{1}$$

and

$$S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \psi A u w \underset{\text{rm}}{\Rightarrow} \psi u w \wedge S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \omega B v x \underset{\text{rm}}{\Rightarrow} \psi u y >$$

$$> \psi A = \omega B \wedge v x = u y \tag{2}$$

hold true for all choices of strings.

Note that (2) is part of the LR(k) criterium of Theorem 2.5. We shall see presently that a test for the LC(k) condition can be based on the LR(k) automaton.

Definition 7.2. A set q of items is *LC(k) consistent* if it satisfies (3) and (4).

$$[A \to .\, X\beta, u] \in q \wedge [B \to \gamma .\, X\delta, v] \in q \wedge \text{first}(\beta u) \cap \text{first}(\delta v) \neq \emptyset >$$

$$> [A \to .\, X\beta] = [B \to \gamma .\, X\delta], \tag{3}$$

$$[A \to .\, \varepsilon, u] \in q \wedge [D \to .\, \delta, v] \in q \wedge \delta \in \Sigma V^* \cup \{\varepsilon\} \wedge u \in \text{first}(\delta v) >$$

$$> [A \to .\, \varepsilon] = [D \to .\, \delta]. \tag{4}$$

Otherwise, q has a *left-corner conflict.* A parsing automaton is *LC(k) consistent* if all its states are. Otherwise, it has a *left-corner conflict.*

Note that (4) is part of (the usual) consistency.

Lemma 7.1. LC(k) *consistent* LR(k) *automata satisfy*

$$[A \to \alpha .\, \beta, u] \in q \wedge [B \to \gamma .\, \delta, v] \in q \wedge \alpha \neq \varepsilon \wedge \gamma \neq \varepsilon$$

$$\wedge \text{first}(\beta u) \cap \text{first}(\delta v) \neq \emptyset >$$

$$> [A \to \alpha .\, \beta] = [B \to \gamma .\, \delta]. \tag{5}$$

Proof. We use induction on the length of ψ to show that (5) holds for $\delta_c^*(q_c, \psi)$. The case $\psi = \varepsilon$ is trivial. Otherwise, let $\psi = \omega X$. Given the state $q = \delta_c^*(q_c, \omega X)$ and items according to the hypothesis of (5) we find strings α', γ' such that $\alpha = \alpha' X \wedge \gamma = \gamma' X$. Hence, for $q' = \delta_c^*(q_c, \omega)$ we have

$$[A \to \alpha' .\, X\beta, u] \in q' \wedge [B \to \gamma' .\, X\delta, v] \in q'$$

$$\wedge \text{first}(X\beta u) \cap \text{first}(X\delta v) \neq \emptyset. \tag{6}$$

The induction hypothesis is applicable if $\gamma' \neq \varepsilon \wedge \alpha' \neq \varepsilon$. Otherwise, $\alpha' = \varepsilon$ or $\gamma' = \varepsilon$ and we apply (3) to (6).

Lemma 7.2. LC(k) *consistent* LR(k) *automata are consistent.*

Proof. Suppose that there is a state q of the LR(k) automaton containing items such that

$$[A \to \alpha\,.\,, u] \in q \wedge [B \to \gamma\,.\,\delta, v] \in q \wedge u \in \text{first}(\delta v) \wedge \delta \in \Sigma V^* \cup \{\varepsilon\}.$$

The case $\alpha = \gamma = \varepsilon$ is trivial because of (4). Lemma 7.1 is applicable if $\alpha \neq \varepsilon \wedge \gamma \neq \varepsilon$. Two cases remain to be considered.

Case 1: $\alpha \neq \varepsilon \wedge \gamma = \varepsilon$. Because of $\alpha \neq \varepsilon$ we have $q \neq q_c$ so that there is an item $[C \to \rho\,.\,\sigma, v'] \in q$ such that

$$[C \to \rho\,.\,\sigma, v'] \overset{*}{\Rightarrow} [\,\cdot \to \,\cdot\, v] \wedge \rho \neq \varepsilon.$$

An application of the item lemma gives us $\text{first}(\delta v) \subseteq \text{first}(\sigma v')$ so that $u \in \text{first}(\sigma v')$. Lemma 7.1 gives us $[A \to \alpha\,.\,] = [C \to \rho\,.\,\sigma]$ which implies $\sigma = \varepsilon$ so that $[C \to \rho\,.\,, v'] = [B \to .\,\delta, v]$ contradicting $\rho \neq \varepsilon$.

Case 2: $\alpha = \varepsilon \wedge \gamma \neq \varepsilon$. Again, $q \neq q_c$ so that there is an item $[C \to \rho\,.\,\sigma, u'] \in q$ with

$$[C \to \rho\,.\,\sigma, u'] \overset{j}{\Rightarrow} [A \to .\,\varepsilon, u] \wedge \rho \neq \varepsilon. \tag{7}$$

Again, $u \in \text{first}(\sigma u')$ so that $u \in \text{first}(\sigma u') \cap \text{first}(\delta v)$. From Lemma 7.1 we learn $[C \to \rho\,.\,\sigma] = [B \to \gamma\,.\,\delta]$ so that $\sigma \in \Sigma V^* \cup \{\varepsilon\}$. But, (7) is impossible unless $j > 0$, i.e., $\sigma \in NV^*$. This is a contradiction.

Theorem 7.3. *A grammar is* LC(k) *iff its* LR(k) *automaton is* LC(k) *consistent. A grammar is* LC(k) *iff it has an* LC(k) *consistent parsing automaton.*

Proof. The second statement is a consequence of the first one because for any parsing automaton $\mathfrak{A}$ and any state q of the LR(k) automaton there is a state of $\mathfrak{A}$ which contains q. We turn to the first statement. Using the item lemma it is easy to see that (1) and (3) are equivalent. Only (2) and (4) remain to be considered. Let the grammar be LC(k) and suppose that we are given a state q and items according to (4). Then there is a string ψ such that

$$q = \delta_c^*(q_c, \psi) \quad \text{and} \quad [S] \overset{*}{\Rightarrow} \psi[A \to .\,\varepsilon, u] \wedge [S] \overset{*}{\Rightarrow} \psi[D \to .\,\delta, v]$$

and

$$S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \psi A u w \underset{\text{rm}}{\Rightarrow} \psi u w \wedge S\# \underset{\text{rm}}{\overset{*}{\Rightarrow}} \psi D v z \underset{\text{rm}}{\Rightarrow} \psi \delta v z \tag{8}$$

for some strings w, z because of the item lemma. Recall $u \in \text{first}(\delta v)$. We apply (2) immediately if $\delta \in \Sigma^*$. Otherwise, $\delta = a\gamma$ for some γ where γ contains at least one

nonterminal. For some x, C, z', z'', y we obtain

$$S \neq \underset{\text{rm}}{\overset{+}{\Rightarrow}} \psi a\gamma vz \underset{\text{rm}}{\overset{*}{\Rightarrow}} \psi axCz' \underset{\text{rm}}{\Rightarrow} \psi axz''z' = \psi uy. \tag{9}$$

An application of (2) to (8) and (9) yields $uy = z'$ which contradicts $uy = axz''z'$.

This theorem provides a (theoretically) simple LC(k) test. It does not involve a grammatical transformation as all the other tests given in the literature [6, 31, 33] where the tests of [6, 31] have not been proven. For $k = 1$ we can do more. We provide a practical LC(1) test by proving for all compatible automata (see Definition 5.4) that a grammar is LC(1) iff the automaton is LC(1) consistent. The proof needs a preparatory lemma. For this lemma we shall call a parsing automaton *closed* if

$$[\,] \in q \wedge [\,] \overset{*}{\Rightarrow} [\,]' \quad \text{implies} \quad [\,]' \in q$$

for all its states q. We note that LR(k) automata and compatible automata are closed.

Lemma 7.4. *A closed parsing automaton of degree one is not consistent if there is a state q such that*

$$[A \to \alpha \,.\, X\beta, a] \in q \wedge [B \to \gamma \,.\, X\delta, a] \in q$$

$$\wedge [A \to \alpha \,.\, X\beta] \neq [B \to \gamma \,.\, X\delta] \wedge \beta\delta \overset{*}{\Rightarrow} \varepsilon$$

holds true for some items.

Proof. Obviously, the state $\delta^*(q, X\beta)$ is not consistent if $\beta = \delta$. Therefore, let $\beta \neq \delta$. Then, $\beta\delta \neq \varepsilon$ and for some $n \geq 0$ and some items there are derivations

$$[A \to aX \,.\, \beta, a] = [A_0 \to \alpha_0 X_0 \,.\, \beta_0, a] \overset{*}{\Rightarrow} X_1 [A_1 \to \alpha_1 X_1 \,.\, \beta_1, a]$$

$$\overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} X_1 \cdots X_n [A_n \to \alpha_n X_n \,.\, \beta_n, a],$$

$$[B \to \gamma X \,.\, \delta, a] = [B_0 \to \gamma_0 X_0 \,.\, \delta_0, a] \overset{*}{\Rightarrow} X_1 [B_1 \to \gamma_1 X_1 \,.\, \delta_1, a]$$

$$\overset{*}{\Rightarrow} \cdots \overset{*}{\Rightarrow} X_1 \cdots X_n [B_n \to \gamma_n X_n \,.\, \delta_n, a],$$

such that

$$X_1 \cdots X_n \overset{*}{\Rightarrow} \varepsilon \wedge \beta_n\delta_n \overset{*}{\Rightarrow} \varepsilon \wedge [A_n \to \alpha_n X_n \,.\, \beta_n] \neq [B_n \to \gamma_n X_n \,.\, \delta_n].$$

Suppose that there is no maximal integer $n \geq 0$ such that derivations of this kind exist.

Then there are integers $i<j$ such that

$$[A_i \to \alpha_i X_i . \beta_i, a] \overset{*}{\Rightarrow} X_{i+1} \cdots X_j [A_j \to \alpha_j X_j . \beta_j, a]$$

$$= X_{i+1} \cdots X_j [A_i \to \alpha_i X_i . \beta_i, a]$$

Therefore, we easily obtain (as in the proof of Theorem 4.2) strings ψ, D, w such that $D \Rightarrow^* \psi D w$ and $\psi \Rightarrow^+ \varepsilon$. Now we apply Lemma 4.3 and find that the grammar is not LR(1). Therefore, our parsing automaton is not consistent.

Hence, we may assume that n is maximal. The case $\beta_n = \delta_n$ is similar to the case $\beta = \delta$ considered above. Therefore, assume $\beta_n \neq \delta_n$ and, without loss of generality, let $\delta_n \neq \varepsilon$. Then

$$[B_n \to \beta_n X_n . \delta_n, a] \overset{+}{\Rightarrow} [E \to . \varepsilon, a]$$

for some E. The state $\delta^*(q, X_1 \cdots X_n)$ contains $[A_n \to \alpha_n X_n . \beta_n, a]$ and $[E \to . \varepsilon, a]$ and is inconsistent if $\beta_n = \varepsilon$. Otherwise, if $\beta_n \neq \varepsilon$, then

$$[A_n \to \alpha_n X_n . \beta_n, a] \overset{+}{\Rightarrow} [D \to . \varepsilon, a]$$

for some D. If $D = E$, then n was not maximal. If $D \neq E$, then $\delta(q, X_1 \cdots X_n)$ is inconsistent.

Theorem 7.5. *A compatible parsing automaton of degree one is* LC(1) *consistent iff the grammar is* LC(1).

Proof. The only-if part of the theorem is a consequence of Theorem 7.3. Let the grammar be LC(1). The compatible automaton is consistent by Theorem 7.3, Lemma 7.2 and Lemma 5.4. This proves that (4) of Definition 7.2 is satisfied. In addition, Lemma 7.4 is applicable. We turn to (3) of Definition 7.2 and assume

$$[A \to . X\beta, a] \in q \wedge [B \to \gamma . X\delta, b] \in q \wedge \text{first}(\beta a) \cap \text{first}(\delta b) \neq \emptyset$$

and claim $[A \to . X\beta] = [B \to \gamma . X\delta]$. Lemma 7.4 yields a contradiction if $a = b \wedge (\beta\delta \Rightarrow^* \varepsilon)$. Hence, assume $a \neq b \vee \neg(\beta\delta \Rightarrow^* \varepsilon)$. In this case we claim

$$\forall d\colon \text{first}(\beta d) \cap \text{first}(\delta b) = \emptyset \vee \forall d\colon \text{first}(\beta a) \cap \text{first}(\delta d) \neq \emptyset. \tag{10}$$

We consider three cases for the proof of this claim.

Case 1: $\{a\} \neq \text{first}(\beta a) \cap \text{first}(\delta b)$. This intersection is not empty so that

$$\exists c \in \Sigma, w \in \Sigma^*\colon \beta \overset{*}{\Rightarrow} cw \wedge c \in \text{first}(\delta b)$$

and

$$\forall d\colon \text{first}(\beta d) \cap \text{first}(\delta b) \neq \emptyset.$$

Case 2: $\text{first}(\beta a) \cap \text{first}(\delta b) \neq \{b\}$. Symmetric to Case 1.

Case 3: $\{a\} = \text{first}(\beta a) \cap \text{first}(\delta b) = \{b\}$. This implies $a = b$ so that we arrive at $\neg(\beta \Rightarrow^* \varepsilon) \vee \neg(\delta \Rightarrow^* \varepsilon)$ which is a simple case.

By definition of compatible parsing automata we find states q_1 and q_2 of the LR(1) automaton such that

$$\exists d\colon \{[A \to .\, X\beta, d], [B \to \gamma .\, X\delta, b]\} \subseteq q_1$$

$$\wedge\, \exists d\colon \{[A \to .\, X\beta, a], [B \to \gamma .\, X\delta, d]\} \subseteq q_2. \tag{11}$$

The LR(1) automaton is LC(1) consistent so that (10) $\wedge$ (11) proves the claim.

Corollary 7.6. *An* LALR(1) *grammar is* LC(1) *iff its* LALR(1) *automaton is* LC(1) *consistent.*

This corollary is an immediate consequence of Theorem 7.5. We give two examples which disprove some obvious conjectures related to Corollary 7.6.

The first grammar is LC(1) and not LALR(1). Its LALR(1) automaton is not LC(1) consistent. This is immediate if the states $\delta_1^*(q_1, C) = \delta_1^*(q_1, dC)$ and $\delta_1^*(q_1, CF) = \delta_1^*(q_1, dCF)$ are computed. The grammar has the rules

$$S \to dD,\ S \to Aa,\ S \to Bb,\ A \to CF,\ B \to CE,$$

$$C \to c,\ E \to F,\ F \to \varepsilon,\ D \to Ab,\ D \to Ba.$$

The second grammar is again LC(1) and not LALR(1). However, its LALR(1) automaton is LC(1) consistent. The grammar has the rules

$$S \to dD,\ S \to Aa,\ S \to Bb,\ A \to C,\ B \to C,\ C \to c,\ D \to Ab,\ D \to Ba.$$

This example shows that the next theorem is not a trivial consequence of Corollary 7.6.

Theorem 7.7. *An* ε*-free grammar is* LC(1) *iff its* LALR(1) *automaton is* LC(1) *consistent.*

Proof. The only-if part is a consequence of Theorem 7.3. Let the grammar be LC(1). As there are no ε-rules it is sufficient to prove (3) of Definition 7.2. Assume

$$\{[A \to .\, X\beta, a], [B \to \gamma .\, X\delta, c]\} \subseteq q \wedge \text{first}(\beta a) \cap \text{first}(\delta c) \neq \emptyset$$

for some items and some state q of the LALR(1) automaton. By definition (Definition 5.2) there are states q_B, q_A of the LR(1) automaton such that

$$\{[A \to .\, X\beta, \hat{a}], [B \to \gamma .\, X\delta, c]\} \subseteq q_B \wedge \{[A \to .\, Xb, a], [B \to \gamma .\, Xw, \hat{c}]\} \subseteq q_A$$

for some $\hat{a}, \hat{c}$. If $\beta \neq \varepsilon$ or $\delta \neq \varepsilon$, then $\text{first}(\beta\hat{a}) \cap \text{first}(\delta c) \neq \emptyset$ or $\text{first}(\beta a) \cap \text{first}(\delta\hat{c}) \neq \emptyset$ so that the LC(1) consistency of q_A and q_B implies $[A \to .\, X\beta] = [B \to \gamma .\, X\delta]$ as required.

Therefore, assume $\beta = \delta = \varepsilon$. Then $a = c$. If q_A or q_B is the initial state, then $q_A = q_B$ and LC(1) consistency can be used again. Therefore, neither q_A nor q_B is the

initial state and we find items such that

$$q_B \ni [C_B \to \rho_B \,.\, \sigma_B, a_B] \overset{*}{\Rightarrow} [A \to .\, X, \hat{a}] \wedge$$
$$q_B \ni [D_B \to \eta_B \,.\, \xi_B, c_B] \overset{*}{\Rightarrow} [B \to \gamma \,.\, X, c],$$

$$q_A \ni [C_A \to \rho_A \,.\, \sigma_A, a_A] \overset{*}{\Rightarrow} [A \to .\, X, a] \wedge$$
$$q_B \ni [D_A \to \eta_A \,.\, \xi_A, c_A] \overset{*}{\Rightarrow} [B \to \gamma \,.\, X, \hat{c}]$$

and $\rho_B \neq \varepsilon \wedge \rho_A \neq \varepsilon \wedge \eta_B \neq \varepsilon \wedge \eta_A \neq \varepsilon$. The absence of ε-rules and the item lemma imply

$$\text{first}(\sigma_B a_B) \supseteq \text{first}(X\hat{a}) = \text{first}(Xc) \subseteq \text{first}(\xi_B c_B).$$

Lemma 7.1 yields

$$[C_B \to \rho_B \,.\, \sigma_B] = [D_B \to \eta_B \,.\, \xi_B].$$

Likewise

$$[C_A \to \rho_A \,.\, \sigma_A] = [D_A \to \eta_A \,.\, \xi_A].$$

Because of $\text{core}(q_A) = \text{core}(q_B)$ there are letters b, d such that

$$q_A \ni [C_B \to \rho_B \,.\, \sigma_B, b] \overset{*}{\Rightarrow} [A \to .\, X, d] \in q_A.$$

Using Lemma 7.1 again we find $[C_B \to \rho_B \,.\, \sigma_B] = [C_A \to \rho_A \,.\, \sigma_A]$.

Summing up we obtain items such that

$$q_B \ni [C \to \rho \,.\, \sigma, a_B] \overset{*}{\Rightarrow} [A \to .\, X, \hat{a}] \wedge q_B \ni [C \to \rho \,.\, \sigma, c_B] \overset{*}{\Rightarrow} [B \to \gamma \,.\, X, a],$$

$$q_A \ni [C \to \rho \,.\, \sigma, a_A] \overset{*}{\Rightarrow} [A \to .\, X, a] \wedge q_A \ni [C \to \rho \,.\, \sigma, c_A] \overset{*}{\Rightarrow} [B \to \gamma \,.\, X, \hat{c}].$$

We consider three cases.

Case 1: $[C \to \rho \,.\, \sigma, c_A] \Rightarrow^* [B \to \gamma \,.\, X, a]$. Then $\{[B \to \gamma \,.\, X, a], [A \to .\, X, a]\} \subseteq q_A$ and $[B \to \gamma \,.\, X] = [A \to .\, X]$ because q_A is LC(1) consistent.

Case 2: $[C \to \rho \,.\, \sigma, a_B] \Rightarrow^* [A \to .\, X, a]$. Symmetric to Case 1.

Case 3: $\neg([C \to \rho \,.\, \sigma, c_A] \Rightarrow^* [B \to \gamma \,.\, X, a]) \wedge \neg([C \to \rho \,.\, \sigma, a_B] \Rightarrow^* [A \to .\, X, a])$.

Recalling $[C \to \rho \,.\, \sigma, c_B] \Rightarrow^* [B \to \gamma \,.\, X, a]$ and $[C \to \rho \,.\, \sigma, a_A] \Rightarrow^* [A \to .\, X, a]$ we see that Case 3 is impossible unless $c_B = a \wedge a_A = a$ and $[C \to \rho \,.\, \sigma, \bot] \Rightarrow^* [B \to \gamma \,.\, X, \bot] \wedge [C \to \rho \,.\, \sigma, \bot] \Rightarrow^* [A \to .\, X, \bot]$ (cf. (1)–(3) of the proof of Lemma 5.5). This yields

$$q_A \ni [C \to \rho \,.\, \sigma, a_A] \Rightarrow^* [B \to \gamma \,.\, X, a_A]$$
$$= [B \to \gamma \,.\, X, a] \in q_A \wedge [A \to .\, X, a] \in q_A$$

so that $[B \to \gamma \,.\, X] = [A \to .\, X]$ by the LC(1) consistency of q_A.

We shall close this section with two inclusion theorems for the class of LC(k) grammars. We start with a theorem which was stated in [30] and not proven until [33].

Theorem 7.8. *Every* LL(k) *grammar is* LC(k). *Every* LC(k) *grammar is* LR(k).

Proof. The second statement is a consequence of Lemma 7.2. Let the grammar be LL(k) and consider the LR(k) automaton. The grammar is LR(k) by Theorem 6.5 which proves (4) of Definition 7.2. Given items according to (3) of this definition we find a string ψ such that

$$[S] \overset{*}{\Rightarrow} \psi[A \to . X\beta, u] \wedge [S] \overset{*}{\Rightarrow} \psi[B \to \gamma . X\delta, v] \wedge \text{first}(\beta u) \cap \text{first}(\delta v) \neq \emptyset.$$

Lemma 6.2 says

$$[A \to . X\beta] \overset{*}{\Rightarrow} [B \to \gamma . X\delta] \vee [B \to \gamma . X\delta] \overset{*}{\Rightarrow} [A \to . X\beta].$$

If one of these two derivations has positive length, then X is a left-recursive nonterminal which is a contradiction.

We know (Theorem 6.7) that ε-free LL(1) grammars are LR(0). The last example preceding Theorem 7.7 shows that ε-free LC(1) grammars need not even be LALR(1). However, adding a very weak form of unique invertibility makes ε-free LC(1) grammars LALR(1).

Theorem 7.9. *An* ε*-free* LC(1) *grammar is* LALR(1) *if for all* $A \to \alpha X\beta$, $B \to Y$, $C \to Y \in \Pi$

$$\alpha \neq \varepsilon \wedge [A \to \alpha . X\beta] \overset{*}{\Rightarrow} [B \to . Y] \wedge [A \to \alpha . X\beta] \overset{*}{\Rightarrow} [C \to . Y]$$

$$\textit{implies} \quad B = C. \tag{12}$$

A grammar obviously satisfies (12) if it is uniquely invertible with respect to unit productions, i.e., if

$$B \to Y \in \Pi \wedge C \to Y \in \Pi \quad \text{implies} \quad B = C.$$

This implication holds for the usual grammar for arithmetic expressions given in Fig. 3 so that (12) is not as strong a restriction as it may seem.

Proof of Theorem 7.9. Shift-reduce conflicts are ruled out by Lemma 5.3 and Lemma 7.2. Assume that there is a state q of the LALR(1) automaton with items $[A \to \alpha . , a] \in q$, $[B \to \delta . , a] \in q$. The LALR(1) automaton is core-restricted. Therefore, α is a suffix of δ or δ is a suffix of α (see (1) of Section 4). We consider three cases.

Case 1: $|\alpha| \geq 2 \wedge |\delta| \geq 2$. There are strings β, γ such that $\alpha = \beta XY$ and $\delta = \gamma XY$ for some X, Y. By definition of the LALR(1) automaton (Definition 5.2) we find a state $\hat{q}$ of the LR(1) automaton such that

$$[A \to \beta X.\ Y, \hat{a}] \in \hat{q} \wedge [B \to \gamma X.\ Y, a] \in \hat{q}$$

for some $\hat{a}$. The absence of ε-rules implies $\text{first}(Y\hat{a}) = \text{first}(Ya)$ so that Lemma 7.1 gives us $A \to \beta XY = B \to \gamma XY$ as required.

Case 2: $|\alpha| \geq 2 \wedge |\delta| = 1$. There is a string β such that $\alpha = \beta XY$ and $\delta = Y$ for some X, Y. For some $\hat{a}$ and some state $\hat{q}$ of the LR(1) automaton we have

$$[A \to \beta X.\ Y, \hat{a}] \in \hat{q} \wedge [B \to .\ Y, a] \in \hat{q}.$$

There must be an item $[C \to \rho .\ D\sigma, b] \in \hat{q}$ such that $\rho \neq \varepsilon$ and $[C \to \rho .\ D\sigma, b] \Rightarrow^* [B \to .\ Y, a]$. The item lemma says $\text{first}(Ya) \subseteq \text{first}(D\sigma b)$. Using $\text{first}(Ya) = \text{first}(Y\hat{a})$ we obtain $\text{first}(Y\hat{a}) \cap \text{first}(D\sigma b) \neq \emptyset$. An application of Lemma 7.1 gives us

$$[A \to \beta X.\ Y] = [C \to \rho .\ D\sigma] \quad \text{so that} \quad [A \to \beta X.\ Y, b] \overset{*}{\Rightarrow} [B \to .\ Y, a].$$

As in the proof of Lemma 5.5 we use the obvious facts

$$[A \to \beta X.\ Y, \perp] \overset{*}{\Rightarrow} [B \to .\ Y, \perp] > [A \to \beta X.\ Y, \hat{a}] \overset{*}{\Rightarrow} [B \to .\ Y, \hat{a}] \tag{13}$$

and

$$\neg([A \to \beta X.\ Y, \perp] \overset{*}{\Rightarrow} [B \to .\ Y, \perp]) \wedge [A \to \beta X.\ Y, b] \overset{*}{\Rightarrow} [B \to .\ Y, a] >$$

$$> [A \to \beta X.\ Y, a] \overset{*}{\Rightarrow} [B \to .\ Y, a]. \tag{14}$$

If (13) is applicable, then $\{[A \to \beta X.\ Y, \hat{a}], [B \to .\ Y, \hat{a}]\} \subseteq \hat{q}$ which contradicts the LC(1) consistency of the LR(1) automaton. Otherwise, (14) is applicable. Because of $[A \to \beta XY., a] \in q$ there is a state q' of the LR(1) automaton containing $[A \to \beta X.\ Y, a]$, and $[B \to .\ Y, a]$ because of (14). Again, LC(1) consistency is violated.

Case 3: $|\alpha| = |\delta| = 1$. There is a letter Y such that $\alpha = \delta = Y$. There must be states $\hat{q}, q'$ of the LR(1) automaton and letters $\hat{a}, a'$ such that

$$\{[A \to .\ Y, \hat{a}], [B \to .\ Y, a]\} \subseteq \hat{q} \quad \text{and} \quad \{[A \to .\ Y, a], [B \to .\ Y, a']\} \subseteq q'.$$

If $\hat{q} = q'$ the LC(1) condition implies $A \to Y = B \to Y$ as required. Assume $\hat{q} \neq q'$. Then one of the states $\hat{q}, q'$ is not the initial state of the LR(1) automaton. Let $\hat{q}$ be different from the initial state. There must be items

$$[C \to \rho .\ \sigma, c] \in \hat{q}, [D \to \xi .\ \eta, d] \in \hat{q} \quad \text{such that} \quad \rho \neq \varepsilon \wedge \xi \neq \varepsilon$$

and

$$[C \to \rho .\ \sigma, c] \overset{*}{\Rightarrow} [A \to .\ Y, \hat{a}] \wedge [D \to \xi .\ \eta, d] \overset{*}{\Rightarrow} [B \to .\ Y, a].$$

Using

$$\emptyset \neq \text{first}(Y\hat{a}) \cap \text{first}(Ya) \subseteq \text{first}(\sigma c) \cap \text{first}(\eta d)$$

and Lemma 7.1 we obtain $[C \to \rho . \sigma] = [D \to \xi . \eta]$. Using the assumptions of the theorem we arrive at $A = B$ as required.

8. Conclusions

Item grammars and parsing automata provide a unifying framework for deriving general theorems in a formal way with reasonable effort. On this basis, even the complicated method of [27] may be dealt with formally on a few pages. It seems that a slight extension of the framework may be used for the elimination of unit productions as described in [28, 15]. Notions such as 'viable prefix', 'LR(k) table', 'ε-free first function', 'postponement set', 'config-group', 'lane' may be discarded. Other notions such as 'collection of sets of items', 'GOTO-function' are given simple interpretations.

The LR(k) automaton provides (theoretically) simple LL(k) and LC(k) tests and may be used to avoid 'local follow sets' [2] in the LL(k) parsing algorithm. Efficient LC(1) tests can be based on compatible parsing automata. Item grammars and parsing automata are useful when proving theorems relating LR parsing to LL, LC and other parsing methods such a precedence parsing and strict deterministic parsing [18].

We conclude that item grammars and parsing automata are useful tools in parsing theory.

References

[1] A.V. Aho and S.C. Johnson, LR parsing, *Comput. Surveys* 6(2) (1974) 99–124.

[2] A.V. Aho and J.D. Ullman, *The Theory of Parsing, Translation and Compiling, Vol. I* (Prentice-Hall, Englewood Cliffs, NJ, 1972).

[3] A.V. Aho and J.D. Ullman, *The Theory of Parsing, Translation and Compiling, Vol. II* (Prentice-Hall, Englewood Cliffs, NJ, 1973).

[4] T. Anderson, J. Eve and J.J. Horning, Efficient LR(1) parsers, *Acta Informat.* 2 (1973) 12–39.

[5] R.C. Backhouse, An alternative approach to the improvement of LR(k) parsers, *Acta Informat.* 6 (1976) 277–296.

[6] B.M. Brosgol, Deterministic translation grammars, Ph.D. Thesis and Report TR 3-74, Center for Research in Computing Technology, Harvard University, Cambridge, MA (1974).

[7] K. Culik II and R. Cohen, LR-regular grammars—an extension of LR(k) grammars, *J. Comput. System Sci.* 7 (1973) 66–96.

[8] A. Demers, Generalized left corner parsing, *Conference Record of the 4th ACM Symposium on Principles of Programming Languages* (1977) 170–182.

[9] F.L. DeRemer, Practical translators for LR(k) languages, Project MAC Report MAC TR-65, Cambridge, MA, M.I.T. (1969).

[10] F.L. DeRemer, Simple LR(k) grammars, *Comm. ACM* 14 (1971) 453–460.

[11] P. Deussen, Strategies in acceptors and their relation to LR(k)-/LL(k) theories, Interner Bericht Nr. 8/77, Institut für Informatik, Universität Karlsruhe (1977).
[12] M.M. Geller and M.A. Harrison, On LR(k) grammars and languages, *Theor. Comput. Sci.* **4**(3) (1977) 245–276.
[13] M.M. Geller and M.A. Harrison, Characteristic parsing, a framework for producing compact deterministic parsers, parts I and II, *J. Comput. System Sci.* **14**(3) (1977) 267–317 and 318–343.
[14] G. Goos, Personal communication (1978).
[15] G. Goos, Übersetzerbau, undated manuscript, received January 1978.
[16] D. Gries, *Compiler Construction for Digital Computers* (Wiley, New York, 1971).
[17] M. Griffiths, Toute grammaire LL(k) est LR(k), *R.A.I.R.O.* **8** (B-2) (1974) 55–58.
[18] S. Heilbrunner, Using item grammars to prove LR(k) theorems, Bericht Nr. 7701, Fachbereich Informatik, Hochschule der Bundeswehr München (1977).
[19] S. Heilbrunner, Definition, analysis and transformation of LC(k) grammars, Bericht Nr. 7802, FB Informatik, Hochschule der Bundeswehr München (1978).
[20] J.E. Hopcroft and J.D. Ullman, *Formal Languages and their Relation to Automata* (Addison-Wesley, Reading, MA, 1969).
[21] J.J. Horning, LR grammars and analysers, in: F.L. Bauer and J. Eickel, Eds., *Compiler Construction, an Advanced Course*, Lecture Notes in Computer Science **21** (Springer, Berlin, 1974) 85–108.
[22] H.B. Hunt III and T.G. Szymanski, Lower bounds and reductions between grammar problems, *J. ACM* **25** (1978) 32–51.
[23] H.B. Hunt III and T.G. Szymanski, Corrigendum to "Lower bounds and reductions between grammar problems", *J. ACM* **25** (1978) 687–688, see [22].
[24] H.B. Hunt III, T.G. Szymanski and J.D. Ullmann, On the complexity of LR(k) testing, *Comm. ACM* **18** (1975) 707–716.
[25] D.E. Knuth, On the translation of languages from left to right, *Information and Control* **8** (1965) 607–623.
[26] H. Langmaack, Application of regular canonical systems to grammars translatable from left to right, *Acta Informat.* **1** (1971) 111–114.
[27] D. Pager, A practical general method for constructing LR(k) parsers, *Acta Informat.* **7** (1977) 249–268.
[28] D. Pager, Eliminating unit productions from LR parsers, *Acta Informat.* **9** (1977) 31–59.
[29] C. Pair, Toute grammaire LL(k) est LR(k) (suite), *R.A.I.R.O.* **8** (B-2) (1974) 59–62.
[30] D.J. Rosenkrantz and P.M. Lewis II, Deterministic left corner parsing, *IEEE Conference Record 11th Annual Symposium on Switching and Automata Theory* (1970) 139–152.
[31] D.J. Rosenkrantz and R.E. Stearns, Properties of deterministic top-down grammars, *Information and Control* **17** (1970) 226–256.
[32] A. Salomaa, *Formal Languages, ACM Monograph Series* (Academic Press, New York, 1973).
[33] E. Soisalon-Soininen, Characterization of LL(k) languages by restricted LR(k) grammars, Report A-1977-3, Department of Computer Science, University of Helsinki (1977).
[34] E. Soisalon-Soininen and E. Ukkonen, A characterization of LL(k) languages, in: S. Michaelson and R. Milner, Eds., *Automata, Languages and Programming* (Edinburgh University Press, 1976, 20–30).
[35] T.G. Szymanski, Concerning bounded-right-context grammars, *Theoret. Comput. Science.* **3** (1976) 273–282.