

10. Testing Grammars for Parsability

Study the complexity of the decision problems

$P_{C(k)}$: "Given a context-free grammar G ,
is G a $C(k)$ grammar?"

P_C : "Given a context-free grammar G and
a natural number k , is G a $C(k)$ grammar?"

Here " $C(k)$ " stand for "strong $LL(k)$ ", " $LALL(k)$ ", " $LL(k)$ ", " $SLR(k)$ ", " $LALR(k)$ ", " $LR(k)$ ", " $non-LL(k)$ " or " $non-LR(k)$ " etc.

$P_{C(k)}$: k is fixed.

P_C : k is not fixed and problem parameter
called **uniform** $C(k)$ testing problems

Convention

G : a grammar (V, Σ, P, S)

G' : $\$$ -augmented grammar of G

$N = V \setminus \Sigma$

k : a natural number

10.1 Efficient Algorithms for LR(k) and SLR(k) Testing

*A decision problem is solvable in deterministic polynomial time(P) if it has a deterministic solution that runs in time $O(p(n))$
 nondeterministic polynomial time(NP) if it has a partial solution that runs in time $O(p(n))$
 nondeterministic one level exponential time(NE) if it has a partial solution that runs in time $O(2^{p(n)})$,
 where p, q are polynomials.*

Nondeterministic LR(k) machine for G is

state alphabet: $I_k \cup \{q_s\}, q_s \notin I_k$

input alphabet: V

initial state: q_s

set of transition:

(i) $q_s \rightarrow [S \rightarrow \bullet \omega, \epsilon]$

(ii) $[A \rightarrow \alpha \bullet X \beta, y] X \rightarrow [A \rightarrow \alpha X \bullet \beta, y], X \in V$, and

(iii) $[A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B \rightarrow \bullet \omega, z],$

$B \in N, z \in \text{FIRST}_k(\beta y)$

set of final states:

$\{[A \rightarrow \omega \bullet, y] \mid A \rightarrow \omega \in P, y \in k:\Sigma^\}$*

$\cup \{[A \rightarrow \alpha \bullet a \beta, y] \mid A \rightarrow \alpha a \beta \in P, a \in \Sigma, y \in k:\Sigma^\}$*

Here I_k : the set of all k -items of G

Theorem 10.1 A state $[A \rightarrow \alpha \bullet \beta, y]$ in the nondeterministic LR(k) machine of G is accessible upon reading γ iff γ is a viable prefix of G and $[A \rightarrow \alpha \bullet \beta, y]$ is an LR(k)-valid item for γ . In other words,

$$\{[A \rightarrow \alpha \bullet \beta, y] \mid q_s \gamma \Rightarrow^* [A \rightarrow \alpha \bullet \beta, y]\} = \text{VALID}_k(\gamma),$$

$$\forall \gamma \in V^*.$$

The automaton obtained by making the nondeterministic LR(k) machine deterministic is exactly the canonical LR(k) machine.

Fact 10.2 Let M be the nondeterministic LR(k) machine for G . Then

- (1) M has at most $|k: \Sigma^* \bullet / G| + 1 = O(|\Sigma|^k \bullet / G|)$ states.
- (2) M has at most $|P|$ transitions of type (i).
- (3) M has at most $|k: \Sigma^* \bullet / G| = O(|\Sigma|^k \bullet / G|)$ transitions of type (ii).
- (4) M has at most $|k: \Sigma^*|^2 \bullet |P| \bullet / G| = O(|\Sigma|^{2k} \bullet |P| \bullet / G|)$ transitions of type (iii).
- (5) the size of M is $O(|G|^{2k+2})$ or $O(|\Sigma|^{2k} \bullet |P| \bullet / G|)$.

Reduction of the size of the automaton

by introducing additional states of the form $[B,z]$, where B is in N , z is a string in $k:\Sigma^*$.

Any transition of type (iii)

$$[A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B \rightarrow \bullet \omega, z]$$

is split into two transitions :

$$[A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B, z]$$

$$[B, z] \rightarrow [B \rightarrow \bullet \omega, z]$$

$M_{LR(k)}(G)$ (or $M_k(G)$) : the finite automaton
state alphabet :

$$\{[A \rightarrow \alpha \bullet \beta, y] \mid A \rightarrow \alpha \beta \in P, y \in k:\Sigma^*\}$$

$$\cup \{[A, y] \mid A \in N, y \in k:\Sigma^*\}$$

input alphabet : V

initial state : $[S, \epsilon]$

set of transitions :

$$(a) [A, y] \rightarrow [A \rightarrow \bullet \omega, y]$$

$$(b) [A \rightarrow \alpha \bullet X \beta, y] \rightarrow [A \rightarrow \alpha X \bullet \beta, y], \text{ for } X \in V$$

$$(c) [A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B, z], \text{ for } B \in N, z \in \text{FIRST}_k(\beta y)$$

final state:

$$\{F_{reduce}(u) \mid u \in k:\Sigma^*\} \cup \{F_{shift}(u) \mid u \in k:\Sigma^*\},$$

where

$$F_{reduce}(u) = \{[A \rightarrow \omega \bullet, u] \mid A \rightarrow \omega \in P\}$$

$$F_{shift}(u) = \{[A \rightarrow \alpha \bullet a \beta, y] \mid A \rightarrow \alpha a \beta \in P, y \in k:\Sigma^*\},$$

$$a \in \Sigma, u \in \text{FIRST}_k(a\beta y)\}$$

Fact 10.3 The following statements hold in the automaton $M_k(G)$.

(1) # of states is at most

$$2 \cdot |\Sigma|^k \cdot |G| = O(|\Sigma|^k \cdot |G|)$$

(2) # of type (a) transitions is at most

$$|\Sigma|^k \cdot |P| = O(|\Sigma|^k \cdot |P|)$$

(3) # of type (b) transitions is at most

$$|\Sigma|^k \cdot |G| = O(|\Sigma|^k \cdot |G|)$$

(4) # of type (c) transitions is at most

$$|\Sigma|^{2k} \cdot |G| = O(|\Sigma|^{2k} \cdot |G|)$$

(5) the sizes of the final state sets is at most

$$O(|\Sigma|^{2k} \cdot |G|).$$

(6) the size of the automaton $M_k(G)$ is

$$O(|G|^{2k+1}) \text{ or } O(|\Sigma|^{2k} \cdot |G|).$$

Theorem 10.4 A state $[A \rightarrow \alpha \cdot \beta, y]$ in $M_k(G)$ is accessible upon reading γ iff γ is a viable prefix of G and $[A \rightarrow \alpha \cdot \beta, y]$ is an LR(k)-valid item for γ . In other words,

$$\{[A \rightarrow \alpha \cdot \beta, y] \mid [S, \varepsilon] \gamma \Rightarrow^* [A \rightarrow \alpha \cdot \beta, y]\} = \text{VALID}_k(\gamma),$$

$$\forall \gamma \in V^*.$$

Mutually accessible states : both reachable from the initial state upon reading the same string.

Distinct k -items $[A \rightarrow \alpha \bullet \beta, y]$ and $[B \rightarrow \omega \bullet, z]$ of G' exhibit an **LR(k)-conflict** if they exhibit, for k , a reduce-reduce conflict or a shift-reduce conflict. i.e.

(1) $\beta = \varepsilon$ and $y = z$ or

(2) $1:\beta$ is a terminal and z is in $FIRST_k(\beta y)$

Theorem 10.5 Let G be a grammar in which $S \Rightarrow^+ S$ is impossible. Then G is non-LR(k) iff the $\$$ -augmented grammar G' has a pair of distinct k -items I and J that exhibit an LR(k)-conflict and are mutually accessible states in $M_k(G')$. In other words,

G is non-LR(k) iff there are distinct k -items $I = [A \rightarrow \alpha \bullet, y]$ and $J = [B \rightarrow \beta \bullet, y]$, or $I = [A \rightarrow \alpha \bullet a \beta, z]$ and $J = [B \rightarrow \omega \bullet, y]$ with $a \in \Sigma$, $y \in FIRST_k(a\beta z)$, s.t.

$[S', \varepsilon] \gamma \Rightarrow^* I$ and $[S', \varepsilon] \gamma \Rightarrow^* J$

hold in $M_k(G')$ for some $\gamma \in \$V^*$.

Proof)

G is non-LR(k) iff for some string $\gamma \in \$V^*$, $VALID_k(\gamma)$ contains a pair of distinct items I, J that exhibit an LR(k)-conflict. By Theorem 10.4, I and J belong to $VALID_k(\gamma)$ iff they are states in $M_k(G')$ accessible upon reading γ .

Algorithm for testing the LR(k) property.

Step 1. Check whether or not $S \Rightarrow^+ S$ is possible in G .

If yes, output "G is non-LR(k)" and halt.

Step 2. Construct automaton $M_k(G')$ with collection

of final state sets $F_{reduce}(u)$, $u \in k:\Sigma^*\$,$ and

$F_{shift}(u)$, $u \in k:\Sigma^*\$$. Remove from each $F_{shift}(u)$

the items $[S' \rightarrow \bullet \$ \$ \$, \epsilon]$ and $[S' \rightarrow \$ \$ \bullet \$, \epsilon]$

Step 3. Determine in $M_k(G')$ the set A of pairs of mutually accessible states.

Step 4. Check whether or not the set A contains a pair

of distinct items I, J such that for some $u \in k:\Sigma^*\$$

$I, J \in F_{reduce}(u)$, or $I \in F_{reduce}(u)$, $J \in F_{shift}(u)$.

If yes, output "G is non-LR(k)" and halt.

otherwise output "G is LR(k)" and halt.

transition of type (c)

$[A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B, z]$, for $B \in N$, $z \in FIRST_k(\beta y)$

Given item $[A \rightarrow \alpha \bullet B \beta, y]$, determine $z \in FIRST_k(\beta y)$

$$F_{shift}(u) = \{ [A \rightarrow \alpha \bullet a \beta, y] \mid A \rightarrow \alpha a \beta \in P, y \in k:\Sigma^*, \\ a \in \Sigma, u \in FIRST_k(a \beta y) \}$$

for each $u \in k:\Sigma^*\$,$ determine all item $[A \rightarrow \alpha \bullet a \beta, y]$ in which $u \in FIRST_k(a \beta y)$.

Lemma 10.6 Any grammar G can be transformed in time $O(|G|)$ into a grammar $G_{pre} = (V_{pre}, \Sigma, P_{pre}, S_{pre})$ s.t. the following statements hold.

(1) G_{pre} is in canonical two-form, so that the rules in P_{pre} are of the forms

$$A \rightarrow BC, A \rightarrow B, A \rightarrow a.$$

(2) $V \subseteq V_{pre}$, for each $A \in V$ generates in G_{pre} exactly the nonempty terminal strings derived by A in G , i.e.

$$L_{G_{pre}}(A) = L_G(A) \setminus \{\varepsilon\}, \quad \forall A \in V \setminus \Sigma.$$

(3) for each $A \in V$, $\exists A_{pre} \in V_{pre} \setminus \Sigma$ that generates in G_{pre} exactly the nonempty prefixes of terminal strings derived by A in G , i.e.

$$L_{G_{pre}}(A_{pre}) = \{x \in \Sigma^* / x \neq \varepsilon, xy \in L_G(A)\}.$$

(4) for each $A \rightarrow \alpha\beta \in P$, $\alpha \neq \varepsilon$, $\beta \neq \varepsilon$, $\exists [\beta] \in V_{pre} \setminus V$ s.t. $L_{G_{pre}}([\beta]) = L_G([\beta]) \setminus \{\varepsilon\}$.

(5) for each $A \rightarrow \alpha\beta \in P$, $\alpha \neq \varepsilon$, $\beta \neq \varepsilon$, $\exists [\beta]_{pre} \in V_{pre} \setminus V$ s.t. $L_{G_{pre}}([\beta]_{pre}) = \{x \in \Sigma^* / x \neq \varepsilon, xy \in L_G(\beta)\}$

Proof)

Transform G into a canonical two-form grammar $G_1 = (V_1, \Sigma, P_1, S)$, where $V \subseteq V_1$, $L_{G_1}(A) = L_G(A) \setminus \{\epsilon\}$, For each rule $A \rightarrow \alpha\beta$ and nonempty strings α and β there is a nonterminal $[\beta] \in V_1 \setminus V$ such that $L_{G_1}([\beta]) = L_G([\beta]) \setminus \{\epsilon\}$

$$G_{pre} = (V_{pre}, T, P_{pre}, S_{pre})$$

$$V_{pre} = V_1 \cup \{A_{pre} \mid A \text{ is a nonterminal in } V_1\}$$

$$P_{pre} = P_1 \cup \{A_{pre} \rightarrow A \mid A \text{ is a nonterminal in } V_1\}$$

$$\cup \{A_{pre} \rightarrow BC_{pre} \mid A \rightarrow BC \text{ is a rule in } P_1\}$$

$$\cup \{A_{pre} \rightarrow B_{pre} \mid \text{For some } C, A \rightarrow BC \in P_1, \\ C \text{ drives some terminal string}\}$$

$$\cup \{A_{pre} \rightarrow B_{pre} \mid A \rightarrow B \text{ is a rule in } P_1\}$$

G_{pre} satisfies (1) since G_1 does.

$L_G(A) = L_{G_{pre}}(A)$ since $P_{pre} \setminus P_1$ contains no rules for the nonterminal in V_1 .

A_{pre} generates nonempty prefixes of $L(A)$.

Lemma 10.7 Given grammar G , $k > 0$, $u = a_1 \dots a_k \in \Sigma^*$, one can compute in space $O(k^2 \cdot |G|)$ and in time $O(k^3 \cdot |G|)$, $k \times k$ matrix N_u containing sets of symbols of the form $[\beta]$ and $[\beta]_{pre}$ s.t.

$$N_u(i, j) = \{[\beta] / A \rightarrow \alpha\beta \in P, \alpha, \beta \neq \epsilon, \beta \Rightarrow^* a_i \dots a_j\} \cup \{[\beta]_{pre} / A \rightarrow \alpha\beta \in P, \alpha, \beta \neq \epsilon, \beta \Rightarrow^* a_i \dots a_j y, y \in \Sigma^*\}$$

for $1 \leq i \leq j \leq k$.

Proof)

Transformed grammar $G_{pre} = (V_{pre}, \Sigma, P_{pre}, S_{pre})$.

Apply general CFG recognition algorithm

$$\hat{N}_u(i, j) = \{A \in V_{pre} \setminus \Sigma \mid A \Rightarrow^* a_i \dots a_j \text{ in } G_{pre}\}$$

for $1 \leq i \leq j \leq k$.

$\hat{N}_u(i, j)$ contains $[\beta]$ iff $[\beta] \Rightarrow^* a_i \dots a_j$ in G_{pre}
iff $\beta \in V^+$, $A \rightarrow \alpha\beta \in P$, $\alpha \neq \epsilon$, $\beta \Rightarrow^* a_i \dots a_j$ in G .

$\hat{N}_u(i, j)$ contains $[\beta]_{pre}$ iff $[\beta]_{pre} \Rightarrow^* a_i \dots a_j y$ in G_{pre}
iff $\beta \in V^+$, $A \rightarrow \alpha\beta \in P$, $\alpha \neq \epsilon$, $\beta \Rightarrow^* a_i \dots a_j y$ in G .

Remove from \hat{N}_u all symbol that are not of forms $[\beta]$ or $[\beta]_{pre}$.

Lemma 10.8 *The automaton $M_k(G)$ can be constructed in time $O((k+1)^3 \cdot |\Sigma|^{2k} \cdot |G|)$.*

Proof)

State set, type (a) and (b) transitions, $F_{reduce}(u)$, $u \in k:\Sigma^*$ take $O(|\Sigma|^k |G|)$.

type (c) transitions

$([A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B, u], u \in FIRST_k(\beta y))$

case $k = 0$

For any 0-item $[A \rightarrow \alpha \bullet B \beta, \varepsilon]$, add

$[A \rightarrow \alpha \bullet B \beta, \varepsilon] \rightarrow [B, \varepsilon]$,

whenever β drives some terminal string

case $k > 0$

For each string $u = a_1 \dots a_l \in k:\Sigma^*$, determine

$[A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B, u]$.

If β is nullable, add $[A \rightarrow \alpha \bullet B \beta, u] \rightarrow [B, u]$.

When $l > 0$, compute N_u .

For $j=1, \dots, l$ and for $[\beta] \in N_u(1, j)$, add

$[A \rightarrow \alpha \bullet B \beta, a_{j+1} \dots a_l y] \rightarrow [B, u]$,

where $a_{j+1} \dots a_l y \in k:\Sigma^*$ and $k:uy = u$.

Observe that β derives $a_1 \dots a_i$.

For all symbols $[\beta]_{pre} \in N_u(1, l)$, add

$[A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B, u]$,

where $y \in k:\Sigma^*$ s.t. $y = \varepsilon$ whenever $l < k$.

Observe that u is a prefix of some terminal string derived by β .

$F_{shift}(u)$

$(\{[A \rightarrow \alpha \bullet a \beta, y] \mid A \rightarrow \alpha \bullet a \beta \in P, y \in k: \Sigma^*,$
 $a \in \Sigma, u \in FIRST_k(a\beta y)\})$

case $k = 0$

Add all $[A \rightarrow \alpha \bullet B \beta, \varepsilon]$, where $a \in \Sigma$, β drives some terminal string

case $k > 0$

For each string $u = a_1 \dots a_l \in k: \Sigma^*$

if β is nullable and $l > 0$ add $[A \rightarrow \alpha \bullet a_1 \beta, a_2 \dots a_l y]$,
 where $a_2 \dots a_l y \in k: \Sigma^*$ and $k: u y = u$.

if $l = k = 1$ and β drives some terminal string add
 $[A \rightarrow \alpha \bullet a \beta, y]$, where $y \in k: \Sigma^*$.

When $l > 0$, compute N_u .

For $j = 2, \dots, l$ and for $[\beta] \in N_u(2, j)$, add

$[A \rightarrow \alpha \bullet a_1 \beta, a_{j+1} \dots a_l y]$,

where $a_{j+1} \dots a_l y \in k: \Sigma^*$ and $k: u y = u$.

For all symbols $[\beta]_{pre} \in N_u(2, l)$, add

$[A \rightarrow \alpha \bullet a_1 \beta, y]$,

where $y \in k: \Sigma^*$ s.t. $y = \varepsilon$ whenever $l < k$.

Time complexity

$O(|k: \Sigma^*| \cdot k^3 \cdot |G| + |\Sigma|^{2k} \cdot |G|)$

$|k: \Sigma^*| \cdot k^3 \cdot |G|$: computation of N_u , $u \in \Sigma^*$.

$|\Sigma|^{2k} \cdot |G|$: generation of states and transition

Determines the pairs of mutually accessible states

Assume M is a finite automaton with state alphabet Q , input alphabet V , and set of transitions P .

$Q \times Q$ relation:

(p, q) ***mutually-goes-to*** (p', q') , if for some $X \in V$
 P contains the transitions $pX \rightarrow p'$ and $qX \rightarrow q'$.

(p, q) ***by-left-passes-empty*** (p', q) , if P contains
the transition $p \rightarrow p'$.

(p, q) ***by-right-passes-empty*** (p, q') , if P contains
the transition $q \rightarrow q'$.

mutually-accesses = $(\text{mutually-goes-to} \cup$
 $\text{by-left-passes-empty} \cup \text{by-right-passes-empty})^*$

Lemma 10.9 States p and q are mutually accessible iff (q_s, q_s) ***mutually-accesses*** (p, q) , where q_s is initial state.

Lemma 10.10 For any f.a. M , the set of pairs of mutually accessible states can be determined in time $O(|M|^2)$.

Theorem 10.11 (LR(k) test using $M_k(G')$) Grammar

G can be tested for the LR(k) property in deterministic time $O((k+1)^3 \cdot |G|^{4k+2})$.

step 1: $O(|G|)$

step 2: $O((k+1)^3 \cdot |\Sigma|^{2k} \cdot |G|)$

step 3: $O(|\Sigma|^{4k} \cdot |G|^2)$

step 4: linear time

$P_{LR(k)}$ is solvable in deterministic polynomial time

$$O(n^{4k+2})$$

n , size of a problem instance, is proportional to $|G|$ in $P_{LR(k)}$.

$n = |G| + k$ when k is expressed in unary.

For k in unary the uniform (non)-LR(k) testing problem is solvable in deterministic one-level exponential time ($O(2^{(4n+2) \log n})$)

$$(n = \log_2 2^n)$$

$n = |G| + \log k$ when k is expressed in binary.

For k in binary the uniform (non)-LR(k) testing problem is solvable in deterministic two-level exponential time ($O(2^{(4 \cdot 2^n + 2) \log n})$)

A more sophisticated method for LR(k) testing

Key idea : representation of the automaton $M_k(G')$ as a collection of several very small automata.

One automaton for each specific string $u \in k:\Sigma^\$$. denoted by $M_u(G')$*

Let $u \in k:\Sigma^$. Then k -item $[A \rightarrow \alpha \bullet \beta, y]$ is a **u -item**, if y is a suffix of u .*

$FIRST_u(\beta)$ = $\{y \in \Sigma^* \mid \beta \Rightarrow^* yz, xy = u \text{ for some } x, z\}$

denote the set of all suffixes of u that are prefixes of some terminal string derived by β

$M_{LR(u)}(G)$ (or $M_u(G)$, for short) :

state alphabet

$\{[A \rightarrow \alpha \bullet \beta, y] \mid A \rightarrow \alpha \beta \in P, y \text{ is a suffix of } u\}$
 $\cup \{[A, y] \mid A \in N, y \text{ is a suffix of } u\}$

input alphabet: V

initial state: $[S, \epsilon]$

set of transitions :

(a) $[A, y] \rightarrow [A \rightarrow \bullet \omega, y]$

(b) $[A \rightarrow \alpha \bullet X \beta, y] \rightarrow [A \rightarrow \alpha X \bullet \beta, y]$, for $X \in V$

(c) $[A \rightarrow \alpha \bullet B \beta, y] \rightarrow [B, z]$,
 for $B \in N, z \in FIRST_u(\beta y)$

set of final states : $F_{reduce} \cup F_{shift}$, where

$$F_{reduce} = \{[A \rightarrow \omega \bullet, u] \mid A \rightarrow \omega \text{ is a rule of } G\}$$

$$F_{shift} = \{[A \rightarrow \alpha \bullet a \beta, y] \mid A \rightarrow \alpha a \beta \text{ is a rule of } G, \\ a \text{ is a terminal and } u \in FIRST_u(a\beta y)\}$$

Fact 10.12 Let $u \in \Sigma^*$, the following statements hold in the automaton $M_u(G)$.

- (1) the number of states is at most $2 \cdot (|u| + 1) \cdot |G|$
- (2) the number of type (a) transitions is at most $(|u| + 1) \cdot |P|$.
- (3) the number of type (b) transitions is at most $(|u| + 1) \cdot |G|$.
- (4) the number of type (c) transitions is at most $(|u| + 1)^2 \cdot |G|$.
- (5) the size of the automaton is $O((|u| + 1)^2 \cdot |G|)$.

An item $[A \rightarrow \alpha \bullet \beta, y]$ of G is LR(u)-valid string $\gamma \in V^*$ if $S \xRightarrow{rm}^* \delta A z \xRightarrow{rm} \delta \alpha \beta z = \gamma \beta z$ and $y \in FIRST_u(z)$

hold in G for some strings $\delta \in V^*$ and $z \in \Sigma^*$

Fact 10.13 If $[A \rightarrow \alpha \bullet \beta, y]$ is an LR(u)-valid item for γ then γ is a viable prefix, $[A \rightarrow \alpha \bullet \beta, y]$ is u -item, α is a suffix of γ and $y \in FOLLOW_{|y|}(\gamma\beta)$.

Conversely, if γ is a viable prefix, then some item is LR(u)-valid item for γ .

3/2/06

10. Testing Grammars for Parsability 17

$VALID_{LR(u)}(\gamma)$ (or $VALID_u(\gamma)$, for short):
the set of all $LR(u)$ -valid items for γ

For all $n \geq 0$,

$VALID_{u,n}(\gamma)$: set of items $[A \rightarrow \alpha \bullet \beta, y]$ that satisfy

$$S \xRightarrow[rm]{n} \delta A z \xRightarrow[rm]{} \delta \alpha \beta z = \gamma \beta z \quad \text{and } y \in FIRST_u(z)$$

for some $\delta \in V^*$ and $z \in \Sigma^*$

Lemma 10.14 If in grammar G

$$[A \rightarrow \alpha \bullet B \beta, y] \in VALID_{u,n}(\gamma) \text{ and } \beta \xRightarrow{m} v \in \Sigma^*,$$

then $\forall B \rightarrow \omega \in P, z \in FIRST_u(vy)$

$$[B \rightarrow \bullet \omega, z] \in VALID_{u,n+m+1}(\gamma) .$$

Lemma 10.15 If in grammar G

$$[B \rightarrow \bullet \omega, z] \in VALID_{u,n}(\gamma), \quad n > 0,$$

then $\exists A \rightarrow \alpha B \beta \in P, m < n,$

$$[A \rightarrow \alpha \bullet B \beta, y] \in VALID_{u,m}(\gamma), \quad \beta \xRightarrow[rm]{n-m-1} v,$$

and $z \in FIRST_u(vy)$.

Fact 10.16 If $[A \rightarrow \alpha \bullet B \beta, y] \in VALID_{u,n}(\gamma)$ then $\gamma\omega$ is a viable prefix and $[A \rightarrow \alpha B \bullet \beta, y] \in VALID_{u,n}(\gamma\omega)$.

Conversely, if $[A \rightarrow \alpha B \bullet \beta, y] \in VALID_{u,n}(\delta)$ then there is a viable prefix γ s.t. $\delta = \gamma\omega$ and $[A \rightarrow \alpha \bullet B \beta, y] \in VALID_{u,n}(\gamma)$.

Theorem 10.17 A state $[A \rightarrow \alpha \bullet \beta, y]$ in $M_u(G)$ is accessible upon reading γ iff $[A \rightarrow \alpha \bullet \beta, y]$ is an LR(u)-valid item for γ . In other words,

$$\text{VALID}_u(\gamma) = \{ [A \rightarrow \alpha \bullet \beta, y] \mid [S, \varepsilon] \gamma \Rightarrow^* [A \rightarrow \alpha \bullet \beta, y] \text{ in } M_u(G) \}.$$

Proof)

Only if) By induction on m , the length of computation

$$[S, \varepsilon] \gamma \Rightarrow^m [A \rightarrow \alpha \bullet \beta, y] \text{ in } M_u(G)$$

base) $m=1$, we have $\gamma=\varepsilon$, $[A \rightarrow \alpha \bullet \beta, y] = [S, \bullet \beta, \varepsilon]$

$$[S, \bullet \beta, \varepsilon] \in \text{VALID}_u(\varepsilon)$$

induction) If $m > 1$, the computation is either

$$\begin{aligned} \text{i) } [S, \varepsilon] \gamma &= [S, \varepsilon] \gamma' X \Rightarrow^{m-1} [A \rightarrow \alpha' \bullet X \beta, y] X \\ &\Rightarrow [A \rightarrow \alpha' X \bullet \beta, y] = [A \rightarrow \alpha \bullet \beta, y], \text{ or} \end{aligned}$$

$$\begin{aligned} \text{ii) } [S, \varepsilon] \gamma &\Rightarrow^{m-2} [A' \rightarrow \alpha' \bullet A \beta', y'] \Rightarrow [A, y] \\ &\Rightarrow [A \rightarrow \bullet \beta, y] = [A \rightarrow \alpha \bullet \beta, y], \text{ where} \end{aligned}$$

$$y \in \text{FIRST}_u(\beta' y').$$

\therefore By i.h., Fact 10.16(i), and Lemma 10.14(ii), it holds.

If) By induction on $n + |\gamma|$, where

$$[A \rightarrow \alpha \bullet \beta, y] \in \text{VALID}_{u,n}(\gamma).$$

base) $n + |\gamma| = 0$, we have $[A \rightarrow \alpha \bullet \beta, y] = [S \rightarrow \bullet \beta, \varepsilon]$, $\gamma = \varepsilon$.

$[S, \bullet \beta, \varepsilon]$ is the state to which $M_u(G)$ has ε -transition from $[S, \varepsilon]$

induction) If $n + |\gamma| > 0$,

i) $[A \rightarrow \alpha \bullet \beta, y] = [A \rightarrow \alpha' X \bullet \beta, y] \in \text{VALID}_{u,n}(\gamma)$,

ii) $[A \rightarrow \alpha \bullet \beta, y] = [A \rightarrow \bullet \beta, y] \in \text{VALID}_{u,n}(\gamma)$,

where $n > 0$.

By i.h., Fact 10.16(i), and Lemma 10.15(ii), it holds

Let $u \in k:\Sigma^*$. Then distinct items $[A \rightarrow \alpha \bullet \beta, y]$ and $[B \rightarrow \omega \bullet, z]$ exhibit an LR(u)-conflict if either

(1) $\beta = \varepsilon$ and $y = z = u$, or

(2) $1:\beta \in \Sigma$, y is suffix of u , and $z = u \in \text{FIRST}_u(\beta y)$.

Fact 10.18 Distinct items $[A \rightarrow \alpha \bullet \beta, y]$ and $[B \rightarrow \omega \bullet, z]$ exhibit an LR(u)-conflict iff

(1) the items u -items,

(2) they exhibit an LR($|u|$)-conflict, and

(3) $z = u$.

Theorem 10.19 Let G is impossible $S \Rightarrow^+ S$.

Then G is non-LR(k) iff $\exists u \in k:\Sigma^*\$,$ the $\$$ -augmented grammar G' has a pair of distinct u -items I and J that exhibit an LR(u)-conflict and are mutually accessible states in $M_u(G')$.

Proof)

Only if) Assume that G (also G') is non- $LR(k)$.

$\exists \gamma \in \$V^*$ $[A \rightarrow \alpha \bullet \beta, y]$ and $[B \rightarrow \omega \bullet, u]$ in $VALID_k(\gamma)$ that exhibit $LR(k)$ -conflict.

$$S' \xRightarrow{rm}^* \delta_1 A y_1 \xRightarrow{rm} \delta_1 \alpha \beta y_1 = \gamma \beta y_1, k: y_1 = y,$$

$$S' \xRightarrow{rm}^* \delta_2 A y_2 \xRightarrow{rm} \delta_2 \omega y_2 = \gamma y_2, k: y_2 = u \in FIRST_k(\beta y)$$

Then $u \in k: \Sigma^* \$$, $u \in FIRST_u(y_2)$

$\therefore [B \rightarrow \omega \bullet, u] \in VALID_u(\gamma)$

$u \in FIRST_k(\beta y)$ implies that y has a prefix y' s.t.

y' is a suffix of u , $u \in FIRST_k(\beta y')$. $u \in FIRST_k(\beta y)$ also implies $y' \in FIRST_k(\beta y_1)$, $u \in FIRST_u(\beta y')$.

Hence $[A \rightarrow \alpha \bullet \beta, y]$ and $[B \rightarrow \omega \bullet, u]$ exhibit an $LR(u)$ -conflict. They accessible upon reading γ .

If) Assume $u \in k: \Sigma^* \$$ and $[A \rightarrow \alpha \bullet \beta, y]$ and $[B \rightarrow \omega \bullet, u]$ exhibit an $LR(u)$ -conflict. They accessible upon reading γ . Then $u \in FIRST_u(\beta y)$, $[A \rightarrow \alpha \bullet \beta, y]$, $[B \rightarrow \omega \bullet, u]$ in $VALID_u(\gamma)$.

$$S' \xRightarrow{rm}^* \delta_1 A y_1 \xRightarrow{rm} \delta_1 \alpha \beta y_1 = \gamma \beta y_1, y \in FIRST_u(y_1),$$

$$S' \xRightarrow{rm}^* \delta_2 B y_2 \xRightarrow{rm} \delta_2 \omega y_2 = \gamma y_2, u \in FIRST_u(y_2).$$

Here $[A \rightarrow \alpha \bullet \beta, k: y_1]$, $[B \rightarrow \omega \bullet, k: y_2] \in VALID_k(\gamma)$.

$u \in FIRST_u(y_2)$ implies $u = |u|: y_2$. But $u = k: y_2$.

$u \in FIRST_u(\beta y)$ and $y \in FIRST_u(y_1)$ implies

$u \in FIRST_u(\beta y_1)$. So $u \in FIRST_k(\beta y_1)$.

Lemma 10.20 *The automaton $M_u(G)$ can be constructed in simultaneously in space $O((|u|+1)^2 \cdot |G|)$, in time $O((|u|+1)^3 \cdot |G|)$.*

Theorem 10.21 *Grammar G can be tested for the $LR(k)$ property simultaneously in deterministic space $O((k+1)^2 \cdot |G|^2)$ and in time $O((k+1)^3 \cdot |\Sigma|^k \cdot |G|^2)$.*

Corollary 10.22 *For any fixed k , the $LR(k)$ testing problem $P_{LR(k)}$ is solvable simultaneously in deterministic space $O(n^2)$ and in deterministic time $O(n^{k+2})$.*

Corollary 10.23 *The uniform $LR(k)$ testing problem P_{LR} is solvable simultaneously in deterministic polynomial space and in deterministic one-level exponential time when k is expressed in unary, and simultaneously in deterministic two-level exponential time when k is expressed in binary.*

Nondeterministic algorithm for Non-LR(k) testing

Step 1. Check whether or not $S \Rightarrow^+ S$ is possible in G . If yes, output "G is non-LR(k)" and halt.

Step 2. Guess a string $u \in k:\Sigma^\$$.*

Step 3. Construct $M_u(G')$. Remove from F_{shift} the items $[S' \rightarrow \bullet \$ \$ \$, \epsilon]$ and $[S' \rightarrow \$ S \bullet \$, \epsilon]$.

Step 4. Determine in $M_u(G')$ the set A of pairs of mutually accessible states.

Step 5. Check whether or not the set A contains a pair of distinct items I, J s.t. $(I, J) \in F_{reduce} \times F_{reduce}$, $(I, J) \in F_{shift} \times F_{reduce}$. If yes, output "G is non-LR(k)" and halt. Otherwise, halt.

Theorem 10.24 *Grammar G can be tested for the non-LR(k) property simultaneously in nondeterministic space $O(k + |G|)$ and in time $O((k+1)^2 \cdot |G|^2)$.*

Corollary 10.25 *For any fixed k , the non-LR(k) testing problem $P_{non-LR(k)}$ is solvable simultaneously in nondeterministic space $O(n)$ and in nondeterministic time $O(n^2)$.*

Corollary 10.26 *The uniform non-LR(k) testing prob-*

lem $P_{\text{non-LR}}$ is solvable simultaneously in nondeterministic polynomial time when k is expressed in unary, and in nondeterministic one-level exponential time when k is expressed in binary.

Theorem 10.27 Let G is impossible $S \Rightarrow^+ S$.

Then G is non- $SLR(k)$ iff $\exists u \in k:\Sigma^*\$, G' has a pair of distinct u -items $[A \rightarrow \alpha \bullet \beta, y]$ and $[B \rightarrow \omega \bullet, u]$ that exhibit an $LR(u)$ -conflict and are mutually accessible states in $M_u(G')$ and where $[A \rightarrow \alpha \bullet \beta]$ and $[B \rightarrow \omega \bullet]$ are mutually accessible states in $M_\varepsilon(G')$.$

Theorem 10.28 Grammar G can be tested for the $SLR(k)$ property simultaneously in deterministic space $O((k+1)^2 \cdot |G| + |G|^2)$ and in time $O((k+1)^3 \cdot |\Sigma|^k \cdot |G|^2)$.

Corollary 10.29 For any fixed k , the $SLR(k)$ testing problem $P_{SLR(k)}$ is solvable simultaneously in deterministic space $O(n^2)$ and in deterministic time $O(n^{k+2})$.

Theorem 10.30 Grammar G can be tested for the non- $SLR(k)$ property simultaneously in nondeterministic space $O(k + |G|)$ and in time $O((k+1) \cdot |G| + |G|^2)$.

10.2 Efficient Algorithms for LL(k) and SLL(k) testing

The LR-transformed grammar for G is the grammar $G_{LR} = (V \cup P, \Sigma, P_{LR}, S)$, where

$$P_{LR} = \{A \rightarrow (A, \omega)\omega \mid A \rightarrow \omega \in P\} \\ \cup \{(A, \omega) \rightarrow \varepsilon \mid A \rightarrow \omega \in P\}.$$

The size of G_{LR} is at most $3 \cdot |G|$

Lemma 10.31 Let G_{LR} be the LR-transformed grammar for G , and h a homomorphism from the rule strings of G_{LR} to the rule strings of G defined by:

$$h(A \rightarrow (A, \omega)\omega) = \varepsilon, \\ h((A, \omega) \rightarrow \varepsilon) = A \rightarrow \omega.$$

If X is a symbol in V , ϕ a string in V^* , and π a rule string in P^* such that

$$X \xrightarrow[lm]{\pi} \phi \text{ in } G,$$

then G_{LR} has a unique rule string π' such that

$$X \xrightarrow[lm]{\pi'} \phi \text{ in } G_{LR} \text{ and } h(\pi') = \pi.$$

Conversely, if X is a symbol in V , ϕ a string, and π' a rule string of G_{LR} such that

$$X \xrightarrow[lm]{\pi'} \phi \in V^* \text{ in } G_{LR}, \text{ then} \\ X \xrightarrow[lm]{h(\pi')} \phi \text{ in } G.$$

Moreover, if X is a symbol in V , x a string in Σ^* , A a nonterminal in V , and δ a string over the alphabet of G_{LR} such that

$$X \xrightarrow{lm}^* xA\delta,$$

then $\delta \in V^*$.

Lemma 10.32 Let G be a grammar and G_{LR} its LR-transformed grammar. Then the following statements hold for all $k \geq 0$.

(a) G is $LL(k)$ iff G_{LR} is $LL(k)$.

(b) G is $SLL(k)$ iff G_{LR} is $SLL(k)$.

Proof)

Assume that G is not $LL(k)$, then

$$S \xrightarrow{lm}^* xA\delta \xrightarrow{lm} x\omega_1\delta \xrightarrow{lm}^* xy_1 \text{ in } G,$$

$$S \xrightarrow{lm}^* xA\delta \xrightarrow{lm} x\omega_2\delta \xrightarrow{lm}^* xy_2 \text{ in } G,$$

where $\omega_1 \neq \omega_2$ and $k:y_1 = k:y_2$.

By def. $A \rightarrow (A, \omega_1)\omega_1$, $(A, \omega_1) \rightarrow \varepsilon$,

$$A \rightarrow (A, \omega_2)\omega_2, \text{ and } (A, \omega_2) \rightarrow \varepsilon$$

$$S \xrightarrow{lm}^* xA\delta \xrightarrow{lm} x(A, \omega_1)\omega_1\delta \xrightarrow{lm} x\omega_1\delta \xrightarrow{lm}^* xy_1 \text{ in } G_{LR},$$

$$S \xrightarrow{lm}^* xA\delta \xrightarrow{lm} x(A, \omega_2)\omega_2\delta \xrightarrow{lm} x\omega_2\delta \xrightarrow{lm}^* xy_2 \text{ in } G_{LR}.$$

Hence G_{LR} is not $LL(k)$.

Assume that G_{LR} is not $LL(k)$, then

$$S \xRightarrow{lm}^* xA\delta \xRightarrow{lm} x\omega'_1\delta \xRightarrow{lm}^* xy_1 \text{ in } G_{LR},$$

$$S \xRightarrow{lm}^* xA\delta \xRightarrow{lm} x\omega'_2\delta \xRightarrow{lm}^* xy_2 \text{ in } G_{LR},$$

where $\omega'_1 \neq \omega'_2$ and $k:y_1 = k:y_2$.

ω'_1 and ω'_2 must be of the form $(A, \omega_1)\omega_1$, $(A, \omega_2)\omega_2$, where $A \rightarrow \omega_1$ and $A \rightarrow \omega_2$ are rules of G .

Then $\delta \in V^*$

$$S \xRightarrow{lm}^* xA\delta \xRightarrow{lm} x\omega_1\delta \xRightarrow{lm}^* xy_1 \text{ in } G,$$

$$S \xRightarrow{lm}^* xA\delta \xRightarrow{lm} x\omega_2\delta \xRightarrow{lm}^* xy_2 \text{ in } G,$$

where $\omega_1 \neq \omega_2$, which means G is not $LL(k)$.

Lemma 10.33 Let G_{LR} be the LR-transformed grammar for a reduced grammar G and let $k \geq 0$. If G_{LR} is $LR(k)$, then it is also $LL(k)$.

Proof) Assume G_{LR} is not $LL(k)$, then

$$S \xRightarrow{lm}^* xA\delta \xRightarrow{lm} x(A, \omega_1)\omega_1\delta \xRightarrow{lm} x\omega_1\delta \xRightarrow{lm}^* xv_1y_1,$$

$$S \xRightarrow{lm}^* xA\delta \xRightarrow{lm} x(A, \omega_2)\omega_2\delta \xRightarrow{lm} x\omega_2\delta \xRightarrow{lm}^* xv_2y_2,$$

where $\omega_1 \neq \omega_2$, $k:v_1y_1 = k:v_2y_2$, and ω_1 derives v_1 , ω_2 derives v_2 , and δ derives y_1 and y_2 .

$$S \xRightarrow{rm}^* \gamma Ay_1 \xRightarrow{rm} \gamma(A, \omega_1)\omega_1y_1 \xRightarrow{rm}^* \gamma(A, \omega_1)v_1y_1 \xRightarrow{rm} \gamma v_1y_1,$$

$$S \xRightarrow{rm}^* \gamma Ay_2 \xRightarrow{rm} \gamma(A, \omega_2)\omega_2y_2 \xRightarrow{rm}^* \gamma(A, \omega_2)v_2y_2 \xRightarrow{rm} \gamma v_2y_2.$$

Since $k:v_1y_1 = k:v_2y_2$ but $(A, \omega_1) \rightarrow \varepsilon$ and $(A, \omega_2) \rightarrow \varepsilon$ are distinct, G_{LR} is not $LR(k)$.

For any reduced grammar G , the LR-transformed grammar G_{LR} is LR(k) iff G is LL(k).

Corollary 10.35 Assume only reduced grammars are considered. Then the problem of LL(k) testing reduces in linear time to the problem of LR(k) testing, and the problem of non-LL(k) testing reduces in linear time to the problem of non-LR(k) testing.

second approach to LL(k) testing

Analogy of $M_{LR(u)}(G)$

We define $M_{LL(u)}(G)$ (or $M_u(G)$) as the FA with state alphabet

$$\begin{aligned} & \{[A \rightarrow \alpha \bullet \beta, y] \mid A \rightarrow \alpha \beta \in P, y \text{ is suffix of } u\} \\ & \cup \{[A, y] \mid A \in N, y \text{ is suffix of } u\}, \end{aligned}$$

input alphabet V , initial state $[S, \epsilon]$, and with set of transitions consisting of all rules of the forms:

(a) $[A, y] \rightarrow [A \rightarrow \omega \bullet, y]$,

(b) $[A \rightarrow \alpha X \bullet \beta, y] X \rightarrow [A \rightarrow \alpha \bullet X \beta, z]$, for $X \in V$,
 $z \in \text{FIRST}_u(Xy)$

(c) $[A \rightarrow \alpha B \bullet \beta, y] \rightarrow [B, y]$, for $B \in N$.

The set of final states of $M_u(G)$ is

$$F_{\text{produce}} = \{[A \rightarrow \omega \bullet, u] \mid A \rightarrow \omega \in P\}.$$

String α may derive some terminal string in type (c).

Fact 10.36 Let G be a grammar, $u \in \Sigma^*$, the following statements hold for the automaton $M_{LL(u)}(G)$.

- (1) the number of states is at most $2 \cdot (|u| + 1) \cdot |G|$
- (2) the number of type (a) transitions is at most $(|u| + 1) \cdot |P|$.
- (3) the number of type (b) transitions is at most $(|u| + 1)^2 \cdot |G|$.
- (4) the number of type (c) transitions is at most $(|u| + 1) \cdot |G|$.
- (5) the size of the automaton is $O((|u| + 1)^2 \cdot |G|)$.

Let $u \in \Sigma^*$. We say that an item $[A \rightarrow \alpha \bullet \beta, y]$ of G is **LL(u)-valid** for string $\gamma \in V^*$ if

$$S \xrightarrow{lm}^* xA\delta \xrightarrow{lm} x\alpha\beta\delta = x\alpha\gamma^R \text{ and } y \in \text{FIRST}_u(\gamma^R)$$

hold in G for some strings $x \in \Sigma^*$ and $\delta \in V^*$.

Fact 10.37 If $[A \rightarrow \alpha \bullet \beta, y]$ is an LL(u)-valid item for γ then γ is a viable suffix, $[A \rightarrow \alpha \bullet \beta, y]$ is a u -item, β^R is a suffix of γ , $y \in \text{FIRST}_{|y|}(\beta \text{ FOLLOW}_{|y|}(A))$.

Conversely, if γ is a viable suffix, then some item is LL(u)-valid item for γ , provided that the grammar is reduced.

We denote by $VALID_{LL(u)}(\gamma)$ (or $VALID_u(\gamma)$) the set of $LL(u)$ -valid items for γ .

$$S \xRightarrow[lm]{n} xA\delta \xRightarrow[lm]{} x\alpha\beta\delta = x\alpha\gamma^R \text{ and } y \in FIRST_u(\gamma^R)$$

for some $x \in \Sigma^*$ and $\delta \in V^*$.

Lemma 10.38 If in grammar G

$$[A \rightarrow \alpha B \bullet \beta, y] \in VALID_{u,n}(\gamma) \text{ and } \alpha \xRightarrow{m} v \in \Sigma^*,$$

then $\forall B \rightarrow \omega \in P$

$$[B \rightarrow \bullet \omega, y] \in VALID_{u,n+m+1}(\gamma) .$$

Lemma 10.39 If in grammar G ,

$$[B \rightarrow \bullet \omega, y] \in VALID_{u,n}(\gamma), \quad n > 0,$$

then $\exists A \rightarrow \alpha B \beta \in P$, string v in T^* , $m < n$,

$$[A \rightarrow \alpha B \bullet \beta, y] \in VALID_{u,m}(\gamma) \text{ and } \alpha \xRightarrow[rm]{n-m-1} v.$$

Lemma 10.40 If $[A \rightarrow \alpha \omega \bullet \beta, y] \in VALID_{u,n}(\gamma)$, then

$\gamma \omega^R$ is a viable suffix, $[A \rightarrow \alpha \bullet \omega \beta, z] \in VALID_{u,n}(\gamma \omega^R)$.

Conversely, if $[A \rightarrow \alpha \bullet \omega \beta, z] \in VALID_{u,n}(\delta)$ then there

is a viable suffix γ s.t. $\delta = \gamma \omega^R$ and $[A \rightarrow \alpha \omega \bullet \beta, y]$

$\in VALID_{u,n}(\gamma)$, where $z \in FIRST_u(\omega y)$.

Theorem 10.41 A state $[A \rightarrow \alpha \bullet \beta, y]$ in $M_u(G)$ is accessible upon reading γ iff $[A \rightarrow \alpha \bullet \beta, y]$ is an $LL(u)$ -valid item for γ . In other words,

$$VALID_u(\gamma) = \{ [A \rightarrow \alpha \bullet \beta, y] \mid [S, \varepsilon] \gamma \Rightarrow^* [A \rightarrow \alpha \bullet \beta, y] \text{ in } M_u(G) \} .$$

Let $u \in k: \Sigma^* \$$. We say that items $[A_1 \rightarrow \bullet \omega_1, y_1]$ and $[A_2 \rightarrow \bullet \omega_2, y_2]$ exhibit an $LL(u)$ -conflict if $A_1 = A_2$, $\omega_1 \neq \omega_2$, and $y_1 = y_2 = u$.

Theorem 10.42 Let G is be a grammar, G' its $\$$ -augmented grammar, and k a natural number. Then G is non-SLL(k) iff $\exists u \in k: \Sigma^* \$$, and accessible states I, J in $M_u(G')$ that exhibit an $LL(u)$ -conflict.

Theorem 10.43 Let G be a grammar, G' its $\$$ -augmented grammar, and k a natural number. Then G is non-LL(k) iff $\exists u \in k:\Sigma^*\$, a string $\gamma \in V^*$, and states $[A, y_1], [A, y_2], [A \rightarrow \bullet \omega_1, u], [A \rightarrow \bullet \omega_2, u]$ in $M_u(G')$ such that following statements hold.$

- (1) $[A, y_1]$ and $[A, y_2]$ are both accessible upon reading γ .
- (2) $[A \rightarrow \bullet \omega_1, u]$ is reachable from $[A, y_1]$ upon reading ω_1^R .
- (3) $[A \rightarrow \bullet \omega_2, u]$ is reachable from $[A, y_2]$ upon reading ω_2^R .
- (4) The items $[A \rightarrow \bullet \omega_1, u]$ and $[A \rightarrow \bullet \omega_2, u]$ exhibit an LL(u)-conflict, that is, $\omega_1 \neq \omega_2$.

Lemma 10.44 Given a grammar G and a string $u \in \Sigma^*$, the automaton $M_u(G)$ can be constructed in simultaneously in space $O((|u|+1)^2 \cdot |G|)$, in time $O((|u|+1)^3 \cdot |G|)$.

Theorem 10.45 Grammar G can be tested for the SLL(k) property simultaneously in deterministic space $O((k+1)^2 \cdot |G|)$ and in time $O((k+1)^3 \cdot |T|^k \cdot |G|)$.

Corollary 10.46 For any fixed k , the $SLL(k)$ testing problem $P_{SLL(k)}$ is solvable simultaneously in deterministic space $O(n)$ and in deterministic time $O(n^{k+1})$.

Theorem 10.47 Grammar G can be tested for the non- $SLL(k)$ property simultaneously in nondeterministic space $O(k + |G|)$ and in time $O((k+1) \cdot |G|)$.

Corollary 10.48 For any fixed k , the non- $SLL(k)$ testing problem $P_{non-SLL(k)}$ is solvable in nondeterministic time $O(n)$.

Corollary 10.49 The uniform non- $SLL(k)$ testing problem $P_{non-SLL}$ is solvable in nondeterministic polynomial time when k is expressed in unary, and in nondeterministic one-level exponential time when k is expressed in binary.

Theorem 10.50 Grammar G can be tested for the $LL(k)$ property simultaneously in deterministic space $O((k+1)^2 \cdot |G|^2)$ and in deterministic time $O((k+1)^4 \cdot |T|^k \cdot |G|^2)$.

Theorem 10.51 Grammar G can be tested for the non- $LL(k)$ property simultaneously in nondeterministic space $O(k + |G|)$ and in nondeterministic time $O((k+1) + |G|^2)$.

We define $M_{u\text{-set}}(G)$ as the FA with state alphabet

$$\{[A \rightarrow \alpha \bullet \beta, W] \mid A \rightarrow \alpha \beta \in P, W \subseteq \text{SUFFIX}(u)\} \\ \cup \{[A, W] \mid A \in N, W \subseteq \text{SUFFIX}(u)\},$$

input alphabet V , initial state $[S, \varepsilon]$, and with set of transitions consisting of all rules of the forms:

(a) $[A, W] \rightarrow [A \rightarrow \omega \bullet, W]$,

(b) $[A \rightarrow \alpha X \bullet \beta, W] X \rightarrow [A \rightarrow \alpha \bullet X \beta, \text{FIRST}_u(XW)]$,

for $X \in V$

(c) $[A \rightarrow \alpha B \bullet \beta, W] \rightarrow [B, W]$, for $B \in N$.

The set of final states of $M_{u\text{-set}}(G)$ is

$$F_{\text{produce}} = \{[A \rightarrow \omega \bullet, W] \mid A \rightarrow \omega \in P, \\ u \in W \subseteq \text{SUFFIX}(u)\}.$$

Fact 10.52 Let $G = (V, \Sigma, P, S)$ be a grammar, a string $u \in \Sigma^*$, the following statements hold in the automaton $M_{u\text{-set}}(G)$.

(1) the number of states is at most $2 \cdot 2^{(|u|+1)} \cdot |G|$

(2) the number of type (a) transitions is at most

$$2^{(|u|+1)} \cdot |P|.$$

(3) the number of type (b) transitions is at most

$$2^{(|u|+1)} \cdot |G|.$$

(4) the number of type (c) transitions is at most

$$2^{(|u|+1)} \cdot |G|.$$

(5) the size of the automaton is $O(2^{|u|} \cdot |G|)$.

Lemma 10.53 If $[A \rightarrow \alpha \omega \bullet \beta, W] \in \text{VALID}_{u\text{-set},n}(\gamma)$, then $\gamma \omega^R$ is a viable suffix and

$[A \rightarrow \alpha \bullet \omega \beta, \text{FIRST}_u(\omega W)] \in \text{VALID}_{u\text{-set},n}(\gamma \omega^R)$.

Conversely, if $[A \rightarrow \alpha \bullet \omega \beta, W'] \in \text{VALID}_{u\text{-set},n}(\delta)$ then there is a viable suffix γ s.t. $\delta = \gamma \omega^R$ and $[A \rightarrow \alpha \omega \bullet \beta, W] \in \text{VALID}_{u\text{-set},n}(\gamma)$, where $W = \text{FIRST}_u(\omega W')$.

Theorem 10.54 A state $[A \rightarrow \alpha \bullet \beta, W]$ in $M_{u\text{-set}}(G)$ is accessible upon reading γ iff $[A \rightarrow \alpha \bullet \beta, W]$ is an $LL(u\text{-set})$ -valid item for γ . In other words,

$$\begin{aligned} \text{VALID}_{u\text{-set}}(\gamma) &= \{ [A \rightarrow \alpha \bullet \beta, W] \mid [S, \{\epsilon\}] \gamma \\ &\Rightarrow^* [A \rightarrow \alpha \bullet \beta, W] \text{ in } M_{u\text{-set}}(G) \}. \end{aligned}$$

Theorem 10.55 G is non- $LL(k)$ iff $\exists u \in k: \Sigma^* \$$, a string $\gamma \in V^*$, and states $[A, W]$, $[A \rightarrow \bullet \omega_1, W_1]$, $[A \rightarrow \bullet \omega_2, W_2]$ in $M_{u\text{-set}}(G')$ s.t. following statements hold.

(1) $[A, W]$ is accessible.

(2) $[A \rightarrow \bullet \omega_1, W_1]$ is reachable from $[A, W]$ upon reading ω_1^R .

(3) $[A \rightarrow \bullet \omega_2, W_2]$ is reachable from $[A, W]$ upon reading ω_2^R .

(4) The items $[A \rightarrow \bullet \omega_1, W_1]$ and $[A \rightarrow \bullet \omega_2, W_2]$ exhibit an $LL(u)$ -conflict, i.e., $\omega_1 \neq \omega_2$ and $u \in W_1 \cap W_2$.

Lemma 10.56 *The automaton $M_{u\text{-set}}(G)$ can be constructed simultaneously in space $O(2^{|u|} \cdot |G|)$, in time $O((|u|+1) \cdot 2^{|u|} \cdot |G|)$.*

Theorem 10.57 *Grammar G can be tested for the $LL(k)$ property simultaneously in deterministic space $O(2^k \cdot |G|)$ and in deterministic time $O((k+1) \cdot 2^k \cdot |T|^k \cdot |G|)$.*

Corollary 10.58 *For any fixed k , the $LL(k)$ testing problem $P_{LL(k)}$ is solvable simultaneously in deterministic space $O(n)$ and in deterministic time $O(n^{k+1})$.*

Theorem 10.59 *Grammar G can be tested for the non- $LL(k)$ property simultaneously in nondeterministic space $O((k+1)^2 \cdot |G|)$ and in nondeterministic time $O((k+1) \cdot 2^k \cdot |G|^2)$.*

Corollary 10.60 *For any fixed k , the non- $LL(k)$ testing problem $P_{\text{non-}LL(k)}$ is solvable in nondeterministic time $O(n)$.*

10.3 Hardness of Uniform LR(k) and LL(k) Testing

derive lower bounds on the complexity of uniform non-C(k) testing

P: solvable in deterministic polynomial time

NP: solvable in nondeterministic polynomial time

NE: solvable in nondeterministic

one level exponential time ($2^{p(n)}$)

PSPACE: solvable in polynomial space

A decision problem P is hard for NP (or NP-hard) if every decision problem in NP reduces in polynomial time to P .

P is complete for NP (or NP-complete) if P is in NP and NP-hard

NE-hard, NE-complete

PSPACE-hard, PSPACE-complete

open problem: whether or not $P = NP$.

$P = NP$ iff some NP-complete problem is in P .

Showing NP-hardness of uniform non-C(k) testing

- 1) *select some specific decision problem which is known to be NP-hard, and reduce this problem to uniform non-C(k) testing.*
- 2) *For any decision problem P in NP, show that there exists a polynomial time-bounded reduction of P to uniform non-C(k) testing.*

Let $M = (V, P)$ be rewriting system and Q, Σ and Γ be subsets of the alphabet V , $q_s \in Q$, $F \subseteq Q$, $B \in \Gamma \setminus \Sigma$, and $\$ \in V \setminus (Q \cup \Gamma)$ s.t. $V = Q \cup \Gamma \cup \{\$\}$, $Q \cap \Gamma = \emptyset$, and $\Sigma \subseteq \Gamma$.

*We say that M is a **Turing machine** with state alphabet Q , input alphabet Σ tape symbol Γ , set of actions P , initial state q_s , set of final states F , blank symbol B , and end marker $\$$, denoted by*

$$M = (Q, \Sigma, \Gamma, P, q_s, F, B, \$),$$

if each rule in P has one of the following forms:

- | | |
|---|---|
| <i>(a) $q_1 a_1 \rightarrow q_2 a_2$</i> | <i>"print a_2"</i> |
| <i>(b) $q_1 a_1 \rightarrow a_2 q_2$</i> | <i>"print a_2 and move to the right"</i> |
| <i>(c) $d q_1 a_1 \rightarrow q_2 d a_2$</i> | <i>"print a_2 and move to the left"</i> |
| <i>(d) $q_1 \\$ \rightarrow q_2 \\$</i> | <i>"record end of tape"</i> |
| <i>(e) $q_1 \\$ \rightarrow q_2 B \\$</i> | <i>"record end of tape and extend"</i> |

A **configuration** of Turing machine M is a string of the form

$$\alpha q \beta,$$

where α and β are tape symbol strings in Γ^* , and q is a state in Q .

The string $\alpha\beta$ is called the **tape contents**, and $1:\beta$ is the **tape symbol scanned** at $\alpha q \beta$.

Configuration $q_s w$ is **initial** for an input string $w \in \Sigma^*$

Configuration $\alpha q \beta$ is **accepting** if q is some final state in F .

A nonaccepting configuration to which no rule in P is applicable is called **error configuration**.

A **computation** of Turing machine M on input string w is any derivation in M from the initial configuration for w .

$$L(M) = \{w \in \Sigma^* \mid q_s w \xrightarrow{*} \alpha q \beta, \alpha, \beta \in \Gamma^*, q \in F\}$$

Turing machine M is **nondeterministic** if to some configuration two actions are applicable.

Proposition 10.61 *Let M be any language recognizer (random-access machine) with input alphabet Σ . Then there exists a Turing machine M' with input alphabet Σ such that the following statements hold for some natural number k .*

- (1) $L(M) = L(M')$
- (2) *If M runs in time $O(T(n))$,
then M' runs in time $O(T(n)^k)$*
- (3) *If M runs simultaneously in time $O(T(n))$
and in space $O(S(n))$, then M' runs
simultaneously in time $O(T(n)^k)$ and
in space $O(S(n)^k)$*
- (4) *If M is deterministic, then so is M'*
- (5) *If M halts on input w , then so does M'*

We shall show that the set of accepting computations of any Turing machine on a fixed input string can be represented as the intersection of two context-free languages.

Let $M = (Q, \Sigma, \Gamma, P, q_s, F, B, \$)$.

Let $C = (\gamma_0, \gamma_1, \dots, \gamma_{n+1})$ be a computation of M on w .

Assume C is nontrivial, meaning that $n+1 \geq 1$. Then
 $\text{repr}(C) = \gamma_0 \# \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \gamma_2 \# \dots \gamma_n \# \gamma_{n+1}^R \# \#$

C and $\text{repr}(C)$ are in one-to-one correspondence with each other.

We shall show that

$\{\text{repr}(C) \mid C \text{ is a nontrivial accepting computation of } M \text{ on } w\} = L(G_1(M)) \cap L(G_2(M, w))$.

$G_1(M)$ is defined with

nonterminals : $\{S_1, A_1, B_1\}$

terminals: $Q \cup \Gamma \cup \{\$, \#\}$

start symbol: S_1

rules

$$1) S_1 \rightarrow \$A_1\$\#S_1,$$

$$2) S_1 \rightarrow \#,$$

$$3) A_1 \rightarrow aA_1a, \forall a \in \Gamma$$

$$4) A_1 \rightarrow \omega_1 B_1 \omega_2^R, \forall \omega_1 \rightarrow \omega_2 \text{ in } P, \text{ no } \$ \text{ in } \omega_1 \omega_2$$

$$5) A_1 \rightarrow \omega_1 \$\#(\omega_2 \$)^R, \forall \omega_1 \$ \rightarrow \omega_2 \$ \text{ in } P$$

$$6) B_1 \rightarrow aB_1a, \forall a \in \Gamma$$

$$7) B_1 \rightarrow \$\#\$.$$

$$\begin{aligned}
L(A_1) &= \{ \alpha \omega_1 \gamma \omega_2^R \alpha^R \mid \alpha \in \Gamma^*, \omega_1 \rightarrow \omega_2 \in P, \\
&\quad \text{no } \$ \text{ in } \omega_1 \omega_2, \gamma \in L(B_1) \} \\
&\cup \{ \alpha \omega_1 \$ \# (\omega_2 \$)^R \alpha^R \mid \alpha \in \Gamma^*, \omega_1 \$ \rightarrow \omega_2 \$ \in P \} \\
&= \{ \alpha \omega_1 \beta \$ \# \$ \beta^R \omega_2^R \alpha^R \mid \alpha, \beta \in \Gamma^*, \omega_1 \rightarrow \omega_2 \in P, \\
&\quad \text{no } \$ \text{ in } \omega_1 \omega_2 \} \\
&\cup \{ \alpha \omega_1 \$ \# \$ \omega_2^R \alpha^R \mid \alpha \in \Gamma^*, \omega_1 \$ \rightarrow \omega_2 \$ \in P \} \\
&= \{ \alpha \omega_1 \beta \$ \# (\alpha \omega_2 \beta \$)^R \mid \alpha, \beta \in \Gamma^*, \omega_1 \rightarrow \omega_2 \in P, \\
&\quad \text{no } \$ \text{ in } \omega_1 \omega_2 \} \\
&\cup \{ \alpha \omega_1 \$ \# (\alpha \omega_2 \$)^R \mid \alpha \in \Gamma^*, \omega_1 \$ \rightarrow \omega_2 \$ \in P \} \\
&= \{ \gamma \$ \# (\delta \$)^R \mid \gamma, \delta \in \Gamma^* Q \Gamma^*, \gamma \$ \Rightarrow \delta \$ \text{ in } M \} \\
L(S_1) &= (\$ L(A_1) \$ \#)^* \# \\
&= \{ \$ \gamma \$ \# (\delta \$)^R \$ \# \mid \gamma, \delta \in \Gamma^* Q \Gamma^*, \gamma \$ \Rightarrow \delta \$ \text{ in } M \}^* \# \\
&= \{ \$ \gamma \$ \# \$ \delta^R \$ \# \mid \gamma, \delta \in \Gamma^* Q \Gamma^*, \gamma \$ \Rightarrow \delta \$ \text{ in } M \}^* \# \\
&= \{ \phi \# \psi^R \# \mid \phi, \psi \in \$ \Gamma^* Q \Gamma^* \$, \phi \Rightarrow \psi \text{ in } M \}^* \# \\
&= \{ \# \} \cup \{ \phi_0 \# \psi_1^R \# \phi_1 \# \psi_2^R \# \dots \phi_n \# \psi_{n+1}^R \# \# \mid n \geq 0, \\
&\quad \phi_i, \psi_{i+1} \in \$ \Gamma^* Q \Gamma^* \$, \phi_i \Rightarrow \psi_{i+1} \text{ in } M \}
\end{aligned}$$

Lemma 10.62 For any Turing machine M ,

$$\begin{aligned}
L(G_1(M)) &= \{ \# \} \cup \{ \phi_0 \# \psi_1^R \# \phi_1 \# \psi_2^R \# \\
&\dots \phi_n \# \psi_{n+1}^R \# \# \mid n \geq 0, \\
&\phi_i \text{ and } \psi_{i+1} \text{ are configurations of } M
\end{aligned}$$

$G_2(M,w)$ is defined with

nonterminals : $\{S_2, A_2, B_2, C, D, E\}$

terminals: $Q \cup \Gamma \cup \{\$, \#\}$

start symbol: S_2

rules

- 1) $S_2 \rightarrow \$q_s w \# A_2 \#$,
- 2) $A_2 \rightarrow \$B_2 \$ \# A_2$,
- 3) $A_2 \rightarrow \$D$,
- 4) $B_2 \rightarrow aB_2a$, $\forall a \in \Gamma$
- 5) $B_2 \rightarrow qCq$, $\forall q \in Q$
- 6) $C \rightarrow aCa$, $\forall a \in \Gamma$
- 7) $C \rightarrow \$\#\$$,
- 8) $D \rightarrow aD$, $\forall a \in \Gamma$
- 9) $D \rightarrow qE$, $\forall q \in F$
- 10) $E \rightarrow aE$, $\forall a \in \Gamma$
- 11) $E \rightarrow \$\#$.

$$L(E) = \Gamma^* \$\#$$

$$L(D) = \Gamma^* FL(E) = \Gamma^* F\Gamma^* \$\#$$

$$L(C) = \{\beta \$\#\$\beta^R \mid \beta \in \Gamma^*\}$$

$$\begin{aligned}
L(B_2) &= \{ \alpha q \gamma q \alpha^R / \alpha \in \Gamma^*, q \in Q, \gamma \in L(C) \} \\
&= \{ \alpha q \beta \beta^R q \alpha^R / \alpha, \beta \in \Gamma^*, q \in Q \} \\
&= \{ \alpha q \beta (\alpha q \beta)^R / \alpha, \beta \in \Gamma^*, q \in Q \} \\
&= \{ \delta^R \delta / \delta \in \Gamma^* Q \Gamma^* \}
\end{aligned}$$

$$\begin{aligned}
L(A_2) &= (\$L(B_2)\$ \#)^* \$L(D) \\
&= \{ \$\delta^R \delta \$ / \delta \in \Gamma^* Q \Gamma^* \}^* \$\Gamma^* F \Gamma^* \$ \\
&= \{ \gamma^R \# \gamma / \gamma \in \$\Gamma^* Q \Gamma^* \$ \}^* \$\Gamma^* F \Gamma^* \$ \\
&= \{ \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \gamma_2 \# \dots \gamma_n^R \# \gamma_{n+1} \# \# \mid n \geq 0, \\
&\quad \gamma_i \text{ is a configuration of } M \text{ for all } i = 1, \dots, n, \\
&\quad \gamma_{n+1} \in \$\Gamma^* F \Gamma^* \$ \}
\end{aligned}$$

$$\begin{aligned}
L(S_2) &= \$q_s w \$ \# L(A_2) \# \\
&= \{ \$q_s w \$ \# \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \gamma_2 \# \dots \gamma_n^R \# \gamma_{n+1} \# \# / \\
&\quad n \geq 0, \gamma_i \in \$\Gamma^* Q \Gamma^* \$, \gamma_{n+1} \in \$\Gamma^* F \Gamma^* \$ \}
\end{aligned}$$

Lemma 10.63 For any Turing machine M and input string w ,

$$\begin{aligned}
L(G_2(M, w)) &= \{ \gamma_0 \# \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \gamma_2 \# \\
&\dots \gamma_n \# \gamma_{n+1}^R \# \# \mid n \geq 0,
\end{aligned}$$

γ_0 is a initial configurations of M for w ,

γ_i is a configuration of M for all $i = 1, \dots, n$,

Theorem 10.64 *Let M be a Turing machine and w an input string. Then*

$L(G_1(M)) \cap L(G_2(M,w)) = \{\text{repr}(\mathbf{C}) \mid \mathbf{C} \text{ is a nontrivial accepting computation of } M \text{ on } w \}$.

Furthermore, for any natural number $k > |w|+3$

$k:L(G_1(M)) \cap k:L(G_2(M,w)) \subseteq \{k:\text{repr}(\mathbf{C}) \mid \mathbf{C} \text{ is a nontrivial computation of } M \text{ on } w \}$.

Moreover, if $\text{repr}(\mathbf{C})$ belongs to $k:L(G_2(M,w))$, then the computation \mathbf{C} is an accepting computation.

Proof. a) Assume $\Phi \in L(G_1(M)) \cap L(G_2(M,w))$.

Any string in $L(G_1(M))$ is either $\#$ or of the form

$\phi_0 \# \psi_1^R \# \phi_1 \# \psi_2^R \# \dots \# \phi_n \# \psi_{n+1}^R \# \#$,

where $n \geq 0$, ϕ_i and ψ_{i+1} are conf. and $\phi_i \Rightarrow \psi_{i+1}$.

Any string in $L(G_2(M,w))$ is of the form

$\gamma_0 \# \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \gamma_2 \# \dots \# \gamma_m \# \gamma_{m+1}^R \# \#$,

where $m \geq 0$, γ_0 : initial conf., γ_{m+1} : accepting conf.

γ_i : configuration.

Clearly $\Phi \neq \#$.

Two string can be equal if

$n = m$, $\phi_i = \gamma_i$ for $i = 0, \dots, n+1$.

Assume \mathbf{C} be a nontrivial accepting computation of M on w .

$\text{repr}(\mathbf{C}) = \gamma_0 \# \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \gamma_2 \# \dots \# \gamma_n \# \gamma_{n+1}^R \# \#$,

where γ_0 : initial conf., γ_{m+1} : accepting conf.

$\gamma_i \Rightarrow \gamma_{i+1}$, $0 \leq i \leq n$.

$\text{repr}(\mathbf{C}) \in L(G_1(M))$, $\text{repr}(\mathbf{C}) \in L(G_2(M, w))$.

b) Let Φ in $k:L(G_1(M)) \cap k:L(G_2(M, w))$.

Any string in $k:L(G_1(M))$ is either $k:\#$ or of the form

$k:\phi_0 \# \psi_1^R \# \phi_1 \# \psi_2^R \# \dots \# \phi_n \# \psi_{n+1}^R \# \#$,

where $n \geq 0$, ϕ_i and ψ_{i+1} are conf. and $\phi_i \Rightarrow \psi_{i+1}$.

Any string in $k:L(G_2(M, w))$ is of the form

$k:\gamma_0 \# \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \gamma_2 \# \dots \# \gamma_m \# \gamma_{m+1}^R \# \#$,

where $m \geq 0$, γ_0 : $\$q_s w\$$, γ_i : conf. $0 \leq i \leq m+1$.

Since $k > 3$, $\Phi \neq k:\#$.

Since $k > |w| + 3 = |\gamma_0|$, $\phi_0 = \gamma_0$.

Hence Φ must be of the form $k:\text{repr}(\mathbf{C})$

c) Since $\text{repr}(\mathbf{C})$ is end with $\#\#$,

if $\text{repr}(\mathbf{C}) \in k:L(G_2(M, w))$, $\text{repr}(\mathbf{C}) \in L(G_2(M, w))$,

which means that \mathbf{C} is an accepting computation.

Theorem 10.65 (Harmanis, 1967) *Given any Turing machine M and input string w , the pair (M, w) can be transformed in polynomial time into a pair of context-free grammars (G_1, G_2) such that the following statements are logically equivalent.*

- (1) M accepts w .
- (2) $L(G_1) \cap L(G_2) \neq \emptyset$

Proof. We may assume that $q_s \notin F$.

Note that if $q_s \in F$ then $L(M) = \Sigma^*$

set of actions $\{q_s a \rightarrow q_f a \mid a \in \Sigma \cup \{\$\}\}$

When $q_s \notin F$, it follows that every accepting computation must be nontrivial.

Choose $G_1 = G_1(M)$, $G_2 = G_2(M, w)$.

Then M accepts w iff $\{\text{repr}(C) \mid \dots\} \neq \emptyset$
iff $L(G_1) \cap L(G_2) \neq \emptyset$

acceptance problem

P_{accept} : "Does Turing machine M accepts input w ?"

nonemptiness of intersection problem

$P_{\text{non-}\cap}$: "Given two CFG G_1 and G_2 ,
is $L(G_1) \cap L(G_2) \neq \emptyset$?"

A grammar is *s*-grammar when all the rules begins with a terminal, and there is no pair of rules

$$A \rightarrow a\beta_1 / a\beta_2, \text{ where } \beta_1 \neq \beta_2.$$

A nonterminal A of a context-free grammar has the *s*-property if (1) all the rules of A begin with a terminal, (2) there is no pair of rules $A \rightarrow a\beta_1 / a\beta_2$, where $\beta_1 \neq \beta_2$.

We shall show that $G_1(M)$ and $G_2(M,w)$ can be replaced by two *s*-grammars, when M satisfies some additional conditions.

Consider $G_1(M)$.

For A_1 the *s*-property is violated,

$G_1(M)$ has $A_1 \rightarrow dA_1d$ and $A_1 \rightarrow dq_1a_1B_1a_2dq_2$,

where $dq_1a_1 \rightarrow q_2da_2$ is an action of M .

$\hat{G}_1(M)$: resulting of left factoring of $G_1(M)$.

1) rules of S_1 and B_1 are as in $G_1(M)$.

2) $A_1 \rightarrow X[A_1, X]$ for all $X \in \Gamma \cup Q$.

3) $[A_1, a_1] \rightarrow a_2[A_1, a_2]a_1$ for all $a_1, a_2 \in \Gamma$.

4) each rule of the form $A_1 \rightarrow X_1 \dots X_m B_1 Y_1 \dots Y_n$ is replaced by

$$[A_1, X_1] \rightarrow X_2 [A_1, X_1 X_2]$$

...

$$[A_1, X_1 \dots X_{m-1}] \rightarrow X_m B_1 [A_1, X_1 \dots X_m B_1]$$

$$[A_1, X_1 \dots X_m B_1] \rightarrow Y_1 [A_1, X_1 \dots X_m B_1 Y_1]$$

...

$$[A_1, X_1 \dots X_m B_1 Y_1 \dots Y_{n-1}] \rightarrow Y_n.$$

Then $L(G_1(M)) = L(\hat{G}_1(M))$. And

$\hat{G}_1(M)$ is s-grammar.

1) S_1 and B_1 have s-property.

2) $[A_1, \gamma]$, where $|\gamma| \geq 1$, has s-property.

3) rules of $[A_1, q]$, where $q \in Q$, are the forms

$$[A_1, q] \rightarrow a[A_1, qa]$$

4) rules of $[A_1, a]$, where $a \in \Gamma$, are of the forms

$$[A_1, a] \rightarrow b[A_1, b]a, \text{ where } b \in \Gamma \text{ or}$$

$$[A_1, a] \rightarrow q[A_1, aq], \text{ where } q \in Q.$$

Lemma 10.66 For any Turing machine M ,

Lemma 10.69 Any Turing machine $M = (Q, \Sigma, \Gamma, P, q_s, F', B, \$)$ can be transformed in time $O(|M|)$ into a Turing machine $M' = (Q', \Sigma, \Gamma, P', q_s, F', B, \$)$ such that the following statements hold.

- (1) q_s does not belong to F' .
- (2) M' can make no move out of states in F' .
- (3) M' accepts only at the extreme right end of its tape.
- (4) $L(M') = L(M)$.
- (5) If M is $T(n)$ time-bounded, M' is $O(\max\{n, T(n)\})$ time-bounded.
- (6) If M is $S(n)$ space-bounded, M' is $S(n)$ space-bounded.
- (7) If M is simultaneously $T(n)$ time-bounded and $S(n)$ space-bounded, M' is simultaneously $O(\max\{n, T(n)\})$ time-bounded and $S(n)$ space-bounded.

Proof. $Q' = Q \cup \{q' \mid q \in F, q' \notin Q \cup \Gamma\} \cup \{q_f\}$

$$P' = P \cup \{qa \rightarrow q'a \mid q \in F, a \in \Gamma \cup \{\$\}\} \\ \cup \{q'a \rightarrow aq' \mid q \in F, a \in \Gamma\} \\ \cup \{q'\$ \rightarrow q_f \$\}$$

$$F' = \{q_f\}.$$

Consider $G_2(M, w)$.

$G_2(M, w)$ is not $LL(k)$ for any k .

Observe that for any $k \geq 1$,

$A_2 \rightarrow \$B_2\$ \# A_2$ and $A_2 \rightarrow \$D$ are rules

$FIRST_k(\$B_2\$ \# A_2)$

$= k: \{\gamma^R \# \gamma \# / \gamma \in \$\Gamma^* Q \Gamma^* \$\}^+ \$\Gamma^* F \Gamma^* \$ \#$

$FIRST_k(\$D) = k: \$\Gamma^* F \Gamma^* \$ \#$

intersection of the two $FIRST_k$ contains all strings

in $\$ \Gamma^{k-1}$ and in $\$ \Gamma^m F \Gamma^n$, $m, n \geq 0$, $m+n = k-2$.

Remove the above conflict if restriction on M .

$A_2 \rightarrow \$D$ generates reversal of all accepting conf.

Assume M accepts only at the extreme right end ($\$ \gamma q \$$, where $q \in F$).

Then restrict $L(\$D) = \$F \Gamma^* \$ \#$. Remove from

$G_2(M, w)$ all $D \rightarrow aD$, where $a \in \Gamma$.

Now every string in $FIRST_k(\$D)$ begins with $\$q$.

Another conflict, intermediate conf. γ in $\gamma^R \# \gamma \#$ derived by $\$B_2\$ \#$ may contain states belonging to F .

M make no move out of a final state. Restrict

$L(\$B_2\$ \# A_2) = \{\gamma^R \# \gamma \# / \gamma \in \$\Gamma^* (Q \setminus F) \Gamma^* \$\}^+ \$F \Gamma^* \$ \#$

$\hat{G}_2 (M,w)$ is the grammar with
 nonterminals : $\{S_2, A'_2, A_2, B_2, C, E\}$
 terminals: $Q \cup \Gamma \cup \{\$, \#\}$
 start symbol: S_2

rules

- 1) $S_2 \rightarrow \$q_s w \# A_2 \#$,
- 2) $A_2 \rightarrow \$A'_2$
- 3) $A'_2 \rightarrow aB_2 a \$ \# A_2, \forall a \in \Gamma$
- 4) $A'_2 \rightarrow qCq \$ \# A_2, \forall q \in Q \setminus F$
- 5) $A'_2 \rightarrow qE, \forall q \in F$
- 6) $B_2 \rightarrow aB_2 a, \forall a \in \Gamma$
- 7) $B_2 \rightarrow qCq, \forall q \in Q$
- 8) $C \rightarrow aCa, \forall a \in \Gamma$
- 9) $C \rightarrow \$\#\$,$
- 10) $E \rightarrow aE, \forall a \in \Gamma$
- 11) $E \rightarrow \$\#.$

Lemma 10.67 For any Turing machine M and input string w ,

$$L(\hat{G}_2 (M,w)) = \{\gamma_0 \# \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \gamma_2 \# \dots \gamma_n \# \gamma_{n+1}^R \# \# \mid n \geq 0\},$$

γ_0 is a initial configurations of M for w ,

γ_i is a configuration in $\Gamma^*(Q \setminus F)\Gamma^*$ for

all $i = 1, \dots, n$,

γ_{n+1} is a configuration in $\Gamma^*(Q \setminus F)\Gamma^*$ }

Moreover, the grammar $\hat{G}_2(M, w)$ is an s -grammar of size $O(|M| + |w|)$ and can be constructed from M and w in time $O(|M| + |w|)$.

Theorem 10.68 Let M be a Turing machine such that the following statements hold.

- (1) The initial state q_s of M is not a final state.
- (2) M can make no move out of a final state.
- (3) M accepts only at the extreme right end of its tape.

Then for any input string w

$L(\hat{G}_1(M)) \cap L(\hat{G}_2(M, w)) = \{\text{repr}(\mathbf{C}) \mid \mathbf{C} \text{ is an accepting computation of } M \text{ on } w\}$.

Furthermore, for any natural number $k > |w| + 3$

$k:L(\hat{G}_1(M)) \cap L(\hat{G}_2(M, w)) \subseteq \{\text{repr}(\mathbf{C}) \mid \mathbf{C} \text{ is a nontrivial computation of } M \text{ on } w\}$.

Moreover, if $\text{repr}(\mathbf{C})$ belongs to $k:L(\hat{G}_2(M, w))$, then

Theorem 10.71 *The nonemptiness of intersection problem for s-languages is unsolvable.*

$\hat{G}(M, w)$: uniting the s-grammars $\hat{G}_1(M)$ and $\hat{G}_2(M, w)$, and $S \rightarrow S_1 \mid S_2$.

Theorem 10.72 *Given any Turing machine M and input string w , the pair (M, w) can be transformed in polynomial time into a context-free grammar G such that the following statements are logically equivalent.*

- (1) M accepts w .
- (2) G is ambiguous.

Proof. Let $G = \hat{G}(M, w)$. $\hat{G}_1(M)$ and $\hat{G}_2(M, w)$ can be constructed from M and w in polynomial time.

Then so can $\hat{G}(M, w)$.

The only way $\hat{G}(M, w)$ can be ambiguous is that

$$S \xRightarrow{lm} S_1 \xRightarrow{lm}^* w,$$

$$S \xRightarrow{lm} S_2 \xRightarrow{lm}^* w.$$

sentence w exists iff M accepts w .

ambiguity problem

P_{amb} : "Given a context-free grammar G

Theorem 10.74 If $C = (\gamma_0, \gamma_1 \dots \gamma_t)$, $t \geq 1$, is a computation of Turing machine M on input string w , then

$$|repr(C)| \leq 2t \cdot (|w| + t + 4) + 1.$$

Proof. $repr(C) = \gamma_0 \# \gamma_1^R \# \gamma_1 \# \gamma_2^R \# \dots \# \gamma_{t-1} \# \gamma_t^R \# \#$

Here $|\gamma_0| = |q_s w| = |w| + 3$.

$|\gamma_i| \leq |\gamma_0| + t$, $i = 1, \dots, t$.

$$\begin{aligned} |repr(C)| &= |\gamma_0| + 2|\gamma_1 \#| + \dots + 2|\gamma_{t-1} \#| + |\gamma_t^R \#| + 1 \\ &\leq 2t (|\gamma_0| + 1 + t) + 1 = 2t \cdot (|w| + t + 4) + 1 \end{aligned}$$

Theorem 10.75 Let M be a Turing machine such that the following statements hold.

- (1) The initial state q_s of M is not a final state.
- (2) M can make no move out of a final state.
- (3) m accepts only at the extreme right end of its tape.

Further let w be an input string and assume that there is a natural number $t > |w|$ such that

- (4) M makes no more than t moves on w , that is, M has no computation on w with length greater than t .

Then for all $k \geq 13 \cdot t^2$ the following statements are

logically equivalent.

- (a) M accepts w in time t .
- (b) $G(M,w)$ is ambiguous.
- (c) $G(M,w)$ is not $C(k)$, where $C(k)$ denotes any of the grammar classes $LR(k)$, $LALR(k)$, $SLR(k)$, $LL(k)$, $LALL(k)$, or $SLL(k)$.

Proof. Similar to the proof of T 10.72,

a) implies b), which implies c).

Assume a) is not true.

Then M has on w no accepting computation. All computations on w has length at most t .

Let $\mathbf{C} = (\gamma_0, \gamma_1, \dots, \gamma_{m+1})$, $0 \leq m < t$, be a computation on.

Then $|\text{repr}(\mathbf{C})| \leq 2t(t+t+4) + 1 = 4t^2 + 8t + 1 \leq 13t^2$.
for all $k \geq 13t^2$,

$k:L(S_1) \cap k:L(S_2) \subseteq \{\text{repr}(\mathbf{C}) \mid \mathbf{C} \text{ is } \dots\} = \emptyset$.

$k:L(S_1) \cap k:L(S_2) = \emptyset$. Hence S has $SLL(k)$ property.

$\hat{G}(M,w)$ has $SLL(k)$ property. It is also $LALL(k)$, $LL(k)$, $LALR(k)$, $LR(k)$, and unambiguous.

A function T from the set of natural numbers to the set of positive natural numbers is time-constructible if

Theorem 10.77 (Hunt, Szymanski and Ullman, 1975)
 Let $C(k)$, for all $k \geq 0$, denote the class of $SLL(k)$, $LL(k)$, $SLR(k)$, or $LR(k)$ grammars. Then the problem of uniform non- $C(k)$ testing is NP-complete when k is expressed in unary, and NE-complete when k is expressed in binary. When $C(k)$ denotes the class of $LA-LR(k)$ or $LALL(k)$ grammars, the problem of uniform non- $C(k)$ testing is NP-hard when k is expressed in unary, and NE-hard when k is expressed in binary.

Proof. We have shown that for $C(k)$ the uniform non- $C(k) = SLL(k), LL(k), SLR(k), LR(k)$, testing problem is in NP when k is expressed in unary.

To show that the problem is NP-hard we have to establish polynomial-time reductions to this problem from arbitrary decision problems in NP.

Let \mathbf{P} be any decision problem in NP.

\exists polynomial p and $p(n)$ time-bounded TM M s.t.

M accepts w iff w is a yes-instance of \mathbf{P} .

By P 10.76 we may assume M never makes more than $p(|w|)$ moves on w .

Now any instance w of \mathbf{P} can be transformed into

$$(\hat{G}(M, w), un(13 \cdot p(|w|)^2)),$$

where $un(k)$ denotes the unary representation of k .

By T 10.75,

"Given a context-free grammar G , is there a natural number k such that G is $C(k)$?"

Theorem 10.78 (Knuth, 1965; Rosenkrantz and Stearns, 1970) *Let $C(k)$ denote one of the grammar classes $LR(k)$, $LALR(k)$, $SLR(k)$, $LL(k)$, $LALL(k)$, or $SLL(k)$.*

It is unsolvable whether or not a given context-free grammar G is $C(k)$ for some $k \geq 0$.

10.4 Complexity of LALR(k) and LALL(k) testing

For any fixed $k \geq 1$, $P_{LALR(k)}$ is PSPACE-complete.

For any fixed $k \geq 2$, $P_{LALL(k)}$ is PSPACE-complete.

$NP \subseteq PSPACE$.

Theorem 10.79 For any fixed $k \geq 0$, the problems of non-LALR(k) testing and LALR(k) testing are in PSPACE.

Proof.

1) $Q_1 := Q_2 := \{[S' \rightarrow \bullet \$ \$ \$, \epsilon]\};$

while true do begin

if $CORE(Q_1) = CORE(Q_2)$ then

if $Q_1 \cup Q_2$ contains a pair of distinct

items exhibiting an LR(k)-conflict then

output "G is non-LALR(k)" and halt

guess strings X and $Y \in V \cup \{\$, \epsilon\}$

$Q_1 := GOTO(Q_1, X)$

$Q_2 := GOTO(Q_2, Y)$

end

2) By Savitch's Theorem

$PSPACE = NSPACE$.

Regular expression nonuniversality

P_{nonuniv} : "Given a regular expression E over V ,
is $L(E) \neq V^*$?"

Noncomputations of M on input w means any string that does not represent a valid computation of M on w .

We shall show that given any polynomial p , a $p(n)$ space-bounded M , and w , the pair (M, w) can be transformed in polynomial time into a regular expression $E(M, p, w)$ that denotes the set of those strings that are not representations of accepting computations of M on w .

$V^* \setminus L(E(M, p, w))$ denotes accepting computation.
Then

M accepts w iff $L(E(M, p, w)) \neq V^*$.

Let $M = (Q, \Sigma, \Gamma, P, q_s, F, B, \$)$. represent computation $(\gamma_0, \gamma_1, \dots, \gamma_n)$ as string $\gamma_0 \dots \gamma_n$.

any configuration γ_i is a string in $\$ \Gamma^* Q \Gamma^* \$$.

$$\begin{aligned} E_1(M) = & \varepsilon \cup (\$ \Gamma^* Q \Gamma^* \$)^* (\Gamma \cup Q) V^* \\ & \cup (\$ \Gamma^* Q \Gamma^* \$)^* \$ (\Gamma \cup Q)^* \\ & \cup (\$ \Gamma^* Q \Gamma^* \$)^* \$ \Gamma^* \$ V^* \end{aligned}$$

initial conf. $\$q_s w\$$

$$E_2(M) = \Gamma^+ Q \Gamma^* \$V^* \cup \$(Q \setminus \{q_s\}) \Gamma^* \$V^* \\ \cup \$q_s \Gamma^* (\Gamma \setminus \Sigma) \Gamma^* \$V^*$$

For M and $w = a_1 \dots a_n$ in Σ^n , initial conf. $\$q_s a_1 \dots a_n \$$

$$E_3(M, w) = \$q_s \$V^* \cup \$q_s \Sigma \$V^* \cup \dots \cup \$q_s \Sigma^{n-1} \$V^* \\ \cup \$q_s \Sigma^{n+1} \Sigma^* \$V^* \\ \cup \$q_s (\Sigma \setminus \{a_1\}) \Sigma^* \$V^* \\ \cup \$q_s \Sigma (\Sigma \setminus \{a_2\}) \Sigma^* \$V^* \cup \dots \\ \cup \$q_s \Sigma^{n-1} (\Sigma \setminus \{a_n\}) \Sigma^* \$V^* .$$

accepting conf. $\$\alpha q \beta \$$, where $q \in F$.

$$E_4(M) = V^* \$\Gamma^* (Q \setminus F) \Gamma^* \$$$

$$E_1(M) \cup E_2(M) \cup E_3(M, w) \cup E_4(M)$$

denotes the set of those strings in V^ that are not form $\gamma_0 \dots \gamma_n$, γ_0 is the initial conf., γ_i is a conf., γ_n is accepting conf.*

conf. shorter than $s(|w|)$

tape length, position of state

$$E_6(M, s, w) = \cup \{ V^* \Gamma^m Q \Gamma^n \Gamma^k Q \Gamma^l V^* \mid m, n, k, l \geq 0, \\ 0 \leq m+n \leq s(|w|) - 3, 0 \leq k+l \leq s(|w|) - 3, \\ \text{but none of conditions are satisfied:} \\ (a) k = m \text{ and } l = n > 0, \\ (b) k = m + 1 \text{ and } l = n - 1, \\ (c) k = m - 1 \text{ and } l = n + 1 > 1, \\ (d) k = m \text{ and } l = n = 0, \\ (e) k = m \text{ and } l = n + 1 = 1, \\ (f) k = m - 1 \text{ and } l = n + 1 = 1 \}$$

Restrict our attention to strings denoted by

- (a) $V^* \Gamma^m Q \Gamma^n \Gamma^m Q \Gamma^n V^*$, where $m \geq 0, n > 0$
- (b) $V^* \Gamma^m Q \Gamma^n \Gamma^{m+1} Q \Gamma^{n-1} V^*$, where $m \geq 0, n > 0$
- (c) $V^* \Gamma^m Q \Gamma^n \Gamma^{m-1} Q \Gamma^{n+1} V^*$, where $m \geq 0, n > 0$
- (d) $V^* \Gamma^m Q \Gamma^n \Gamma^m Q V^*$, where $m \geq 0$
- (e) $V^* \Gamma^m Q \Gamma^n \Gamma^m Q \Gamma V^*$, where $m \geq 0$
- (f) $V^* \Gamma^m Q \Gamma^n \Gamma^{m-1} Q \Gamma V^*$, where $m > 0$

In (a), (b), (c), $m + n \leq s(|w|) - 3$,

In (d), (e), (c), $m + n \leq s(|w|) - 3$,

tape symbol changed

$$E_7(M, s, w)$$

$$= \cup \{ V^* \$ \Gamma^m a \Gamma^* Q \Gamma^* \$ \$ \Gamma^m (\Gamma \setminus \{a\}) \Gamma^* Q \Gamma^* \$ V^* / \\ a \in \Gamma, 0 \leq m \leq s(|w|) - 4 \},$$

$E_7(M, s, w)$

$$= \cup \{ V^* \$ \Gamma^* a \Gamma^+ a Q \Gamma^n \$ \$ \Gamma^* Q \Gamma^* (\Gamma \setminus \{a\}) \Gamma^n \$ V^* / \\ a \in \Gamma, 0 \leq m \leq s(|w|) - 4 \},$$

Apply action

Convention: $q_1, q_2 \in Q, a, a_1, a_2, d_1, d_2 \in \Gamma, m \geq 0,$

P denote set of actions of M

$$E_a(M, s, w) = \cup \{ V^* \$ \Gamma^m q_1 a_1 \Gamma^* \$ \$ \Gamma^m q_2 a_2 \Gamma^* \$ V^* / \\ m \leq s(|w|) - 4, q_1 a_1 \rightarrow q_2 a_2 \notin P \}$$

$$E_b(M, s, w) = \cup \{ V^* \$ \Gamma^m q_1 a_1 \Gamma^* \$ \$ \Gamma^m a_2 q_2 \Gamma^* \$ V^* / \\ m \leq s(|w|) - 4, q_1 a_1 \rightarrow a_2 q_2 \notin P \},$$

$$E_c(M, s, w) = \cup \{ V^* \$ \Gamma^m d_1 q_1 a_1 \Gamma^* \$ \$ \Gamma^m q_2 d_2 a_2 \Gamma^* \$ V^* / \\ m \leq s(|w|) - 5, d_1 q_1 a_1 \rightarrow q_2 d_2 a_2 \notin P \}$$

$$E_d(M, s, w) = \cup \{ V^* \$ \Gamma^m q_1 \$ \$ \Gamma^m q_2 \$ V^* / \\ m \leq s(|w|) - 3, q_1 \$ \rightarrow q_2 \$ \notin P \},$$

$$E_e(M, s, w) = \cup \{ V^* \$ \Gamma^m q_1 \$ \$ \Gamma^m q_2 a \$ V^* /$$

Theorem 10.80 *Let p be a polynomial, M a Turing machine that runs in space $p(n)$, and w an input string. Then*

$$L(E(M,p,w)) = V^* \setminus \{ \gamma_0 \gamma_1 \dots \gamma_n \mid n \geq 0,$$

$(\gamma_0, \dots, \gamma_n)$ is an accepting computation of M on w having space complexity at most $p(|w|)$ $\}$,

where V is the alphabet of M . Moreover, the regular expression $E(M,p,w)$ can be constructed from M and w in time polynomial in $|M| + |w|$.

Theorem 10.81 (Stockmeyer and Meyer, 1973) P_{nonuniv} regular expression nonuniversality, is PSPACE-hard.

Proof. To show that P_{nonuniv} is PSPACE-hard

- 1) Choose any problem P in PSPACE,
- 2) establish a polynomial-time reduction of

P to P_{nonuniv}

Since P is in PSPACE it has a polynomial space-bounded solution. There exists a polynomial p and $p(n)$ space-bounded Turing machine M s.t.

M accepts input w iff w is a yes-instance of P .

By T 10.80 there exists a polynomial time-bounded algorithm that transforms any input string w of M

Establishing a polynomial-time reduction of r.e. non-universality to non-LALR(k) testing.

*r.e. nonuniversality reduces in polynomial time to
"Given a right-linear grammar G with terminal T and with set of rules contains only rules of the forms $A \rightarrow aB$ and $A \rightarrow \epsilon$, is $L(G) \neq T^*$?"*

We shall show that this problem reduces in polynomial time to non-LALR(k) testing.

Let $k \geq 1$ and $G = (N \cup T, T, P, S)$ be a right-linear grammar. Further assume that the rules in P are of the forms $A \rightarrow aB$ and $A \rightarrow \epsilon$. Define $\hat{G}(G, k)$ as grammar

with nonterminals $\{\hat{S}, E_1, E_2, H_1, H_2, H_3\} \cup N$,

terminals $T \cup \{c, d, f, g, h, (\hat{S}, S)\} \cup P$

start symbol \hat{S}

rules: $\hat{S} \rightarrow E_1 \mid S(\hat{S}, S) \mid gE_2$

$E_1 \rightarrow aE_1$, for all $a \in T$,

$E_1 \rightarrow H_1d^k \mid H_2c^k$,

$E_2 \rightarrow H_1c^k \mid H_2d^k$,

$A \rightarrow aB(A, aB)$, for all $A \rightarrow aB$ in P

$A \rightarrow H_3f^k(A, \epsilon)$, for all $A \rightarrow \epsilon$ in P

Lemma 10.82 *Let γ be a string such that the state $VALID_0(\gamma)$ in the LR(0) machine of the $\$$ -augmented grammar for $\hat{G} (G,k)$ contains a pair of distinct items $[C \rightarrow \alpha \bullet \beta]$ and $[D \rightarrow \omega \bullet]$, where $\alpha \neq \epsilon$. Then $\gamma:1 = h$.*

Proof. *Note that one of α and ω must be a suffix of the other, and that α and ω must be suffixes of γ .*

$\gamma:1 = \alpha:1 = \omega:1$.

Denote $\gamma:1$ by X .

$X \neq \hat{S}'$, \hat{S} , H_1 , H_2 , H_3 , f , g , symbol in $N \cup T$

(No rule $D \rightarrow \omega' Y$, where Y is one of these)

$X \neq E_2$, (\hat{S}, S) , (A, ϵ) , (A, aB) , where $A, B \in N$, $a \in T$

(No two distinct item $[C \rightarrow \alpha Y \bullet \beta]$ and $[D \rightarrow \omega Y \bullet]$ in the same state)

$X \neq c$, d

($[E_i \rightarrow H_j c^m \bullet c^{k-m}]$ and $[E_l \rightarrow H_r c^k \bullet]$ in the same state implies they are equal.)

$X \neq E_1$

($[E_1 \rightarrow a E_1 \bullet]$ and $[E_1 \rightarrow b E_1 \bullet]$, $a \neq b$, cannot simultaneously be in $VALID_0(\gamma)$; $[\hat{S} \rightarrow E_1 \bullet]$ belongs only to $VALID_0(\$E_1)$ and $[E_1 \rightarrow a E_1 \bullet] \notin VALID_0(\$E_1)$.)

Thus $X = h$.

Lemma 10.83 For all $0 \leq n \leq k$ and strings γ the following hold for the $\$$ -augmented grammar of $\hat{G} (G, k)$.

(a) $VALID_n(\gamma)$ contains an item of the form

$$[C \rightarrow \alpha \bullet E_1 \beta, y] \text{ iff } \gamma \in \$T^*$$

(b) $VALID_n(\gamma)$ contains an item of the form

$$[C \rightarrow \alpha \bullet A \beta, y], A \in N, \text{ iff } \gamma \text{ is of the form } \$w$$

where $w \in T^*$ and $S \Rightarrow^* wA$ in G .

(c) $VALID_n(\gamma)$ contains an item of the form

$$[C \rightarrow \alpha \bullet E_2 \beta, y] \text{ iff } C = \hat{S}, \alpha = g, \beta = \varepsilon,$$

$$y = \$, \text{ and } \gamma = \$g.$$

Lemma 10.84 Let $0 \leq n \leq k$ and let γ be a string s.t. in the $\$$ -augmented grammar of $\hat{G} (G, k)$,

$VALID_n(\gamma h) \neq \emptyset$. Then $\gamma \in \$T^* \cup \g . Moreover, $VALID_n(\gamma h)$ equals

$$(1) \{[H_1 \rightarrow h \bullet, d^n], [H_2 \rightarrow h \bullet, c^n], [H_3 \rightarrow h \bullet, f^n]\}$$

if $\gamma \in \$L(G)$,

$$(2) \{[H_1 \rightarrow h \bullet, d^n], [H_2 \rightarrow h \bullet, c^n]\}$$

if $\gamma \in \$T^* \setminus \$L(G)$,

Proof. $VALID_n(\gamma h)$ can only contain the items

$[H_1 \rightarrow h\bullet, d^n], [H_1 \rightarrow h\bullet, c^n], [H_2 \rightarrow h\bullet, c^n],$

$[H_2 \rightarrow h\bullet, d^n],$ and $[H_3 \rightarrow h\bullet, f^n].$

$[H_1 \rightarrow h\bullet, d^n], [H_2 \rightarrow h\bullet, c^n] \in VALID_n(\gamma h)$

iff $[C \rightarrow \alpha\bullet E_1\beta, y] \in VALID_n(\gamma)$ iff $\gamma \in \$T^*$.

$[H_3 \rightarrow h\bullet, f^n] \in VALID_n(\gamma h)$

iff $[C \rightarrow \alpha\bullet A\beta, y] \in VALID_n(\gamma)$, where $A \rightarrow \varepsilon$ in G

iff γ is of the form $\$w$ where $w \in T, S \Rightarrow^* wA$ in G

$[H_1 \rightarrow h\bullet, c^n], [H_2 \rightarrow h\bullet, d^n] \in VALID_n(\gamma h)$

iff $[C \rightarrow \alpha\bullet E_2\beta, y] \in VALID_n(\gamma)$

iff $C = \hat{S}, \alpha = g, \beta = \varepsilon, y = \$, \text{ and } \gamma = \g

Theorem 10.85 Let G be any right-linear grammar with terminal alphabet T and with a set of rules containing only rules of the forms $A \rightarrow aB, A \rightarrow \varepsilon$.

Then for all natural numbers $k \geq 1, \hat{G}(G, k)$ is $LR(1)$.

$\hat{G}(G, k)$ is $LALR(1)$ if $L(G) = T^*$, and

non- $LALR(k)$ if $L(G) \neq T^*$.

Proof. Let γ'_1 and γ'_2 be strings, $[C' \rightarrow \alpha'\bullet\beta', y']$ in

$VALID_1(\gamma'_1)$ and $[C \rightarrow \omega\bullet, z]$ in $VALID_1(\gamma'_2)$ be two

distinct items. Then $VALID_1(\gamma'_1)$ must contain an item $[A \rightarrow \alpha \bullet \beta, y]$ from which distinct $C \rightarrow \omega \bullet$ and $\alpha \neq \epsilon$. Now if $VALID_0(\gamma'_1) = VALID_0(\gamma'_2)$ then $\gamma'_1 = \gamma_1 h$, $\gamma'_2 = \gamma_2 h$. But then shows that $\hat{G}(G, k)$ is LR(1). it is LALR(1) if $L(G) = T^*$. L 10.84 show that if $L(G) \neq T^*$, then it cannot be LALR(k).

Theorem 10.86 For each fixed natural number $k \geq 1$, the problems of non-LALR(k) testing and LALR(k) testing are PSPACE-complete.

Theorem 10.87 For each fixed natural number $k \geq 2$, the problems of non-LALR(k) testing and LALL(k) testing are PSPACE-complete.

Theorem 10.88 Grammar G can be tested for the non-LALR(k) and non-LALL(k) properties simultaneously in nondeterministic space $O(|w| + k)$ and in nondeterministic time $O((k+1) \cdot |G|^2 \cdot 2^{|G|})$

Theorem 10.89 The problem of uniform non-LALR(k), LALR(k), non-LALL(k), and LALL(k) testing are PSPACE-complete when k is expressed in

unary. The problems of non-LALR(k) and non-LALL(k) testing are NE-complete when k is expressed in binary.

Table 1: Complexity of non- $C(k)$ testing

	<i>fixed</i> $k=1$	<i>fixed</i> $k \geq 2$	<i>free</i> k <i>in</i> <i>unary</i>	<i>free</i> k <i>in</i> <i>binary</i>	$\exists k, G$ <i>is</i> $C(k)$
<i>non-SLL(k)</i>	<i>in P</i>	<i>in P</i>	NP-com- plete	NE-com- plete	unsol vable
<i>non-LALL(k)</i>	<i>in P</i>	<i>PSPACE</i> <i>com- plete</i>	<i>PSPACE</i> <i>com- plete</i>	NE-com- plete	unsol vable
<i>non-LL(k)</i>	<i>in P</i>	<i>in P</i>	NP-com- plete	NE-com- plete	unsol vable
<i>non-SLR(k)</i>	<i>in P</i>	<i>in P</i>	NP-com- plete	NE-com- plete	unsol vable
<i>non-LALR(k)</i>	<i>PSPACE</i> <i>com- plete</i>	<i>PSPACE</i> <i>com- plete</i>	<i>PSPACE</i> <i>com- plete</i>	NE-com- plete	unsol vable
<i>non-LR(k)</i>	<i>in P</i>	<i>in P</i>	NP-com- plete	NE-com- plete	unsol vable

Table 1: Upper bounds on the complexity of non- $C(k)$ testing when $k \geq 2$ is fixed.

	<i>deterministic time</i>	<i>deterministic space</i>	<i>nondeterministic time</i>	<i>size of $C(k)$ parser</i>
<i>non-SLL(k)</i>	$O(n^{k+1})$	$O(n)$	$O(n)$	$O(2^{n+2\log n})$
<i>non-LALL(k)</i>	<i>parser construction time</i>	$O(n^2)$	$O(2^{n+2\log n})$	$O(2^{n+(k+1)\log n})$
<i>non-LL(k)</i>	$O(n^{k+1})$	$O(n)$	$O(n)$	$O(2^{n^{k+1}+(k+1)\log n})$
<i>non-SLR(k)</i>	$O(n^{k+2})^a$	$O(n^2)$	$O(n^2)$	$O(2^{n+(k+1)\log n})$
<i>non-LALR(k)</i>	<i>parser construction time</i>	$O(n^2)$	$O(2^{n+2\log n})$	$O(2^{n+(k+1)\log n})$
<i>non-LR(k)</i>	$O(n^{k+2})$	$O(n^2)$	$O(n^2)$	$O(2^{n^{k+1}+(k+1)\log n})$

a. For (non-)SLR(2) testing an $O(n^3)$ algorithm is known.

Table 1: Upper bounds on the Complexity of $C(1)$ testing and on Parser Construction

	<i>C(1) testing deterministic time</i>	<i>C(1) parser size</i>	<i>C(1) parser construction time</i>
<i>SLL(1)</i>	$O(n^2)$	$O(n^2)$	$O(n^2)$
<i>LALL(1)</i>	$O(n^2)$	$O(2^{n+2\log n})$	$O(2^{n+3\log n})$
<i>LL(1)</i>	$O(n^2)$	$O(2^{n^2+2\log n})$	$O(2^{n^2+4\log n})$
<i>SLR(1)</i>	$O(n^2)$	$O(2^{n+2\log n})$	$O(2^{n+3\log n})$
<i>LALR(1)</i>	$O(2^{n+3\log n})$	$O(2^{n+2\log n})$	$O(2^{n+3\log n})$
<i>LR(1)</i>	$O(n^2)$	$O(2^{n^2+2\log n})$	$O(2^{n^2+4\log n})$