数理解析研究所講究録 1205 巻 2001 年 77-82

Simulation of One-Dimensional Cellular Automata by Uniquely Parallel Parsable Grammars

李　佳 (Jia Lee) *
今井克暢 (Katsunobu Imai) †
森田憲一 (Kenichi Morita) ‡
広島大学工学部
(Faculty of Engineering, Hiroshima University)

Abstract

A uniquely parsable grammar (UPG) introduced by Morita et al. (1997) is a special kind of generative grammar where parsing can be performed without backtracking. By extending a UPG, a uniquely parallel parsable grammar (UPPG) was proposed and its unique parallel parsability has been investigated. In this paper, we show any one-dimensional cellular automaton (CA), as a parallel language recognition device, can be simply simulated by a parallel reduction in an equivalent UPPG.

1 Introduction

A uniquely parsable grammar (UPG) [10] is a special kind of generative grammar where parsing can be performed without backtracking. Rewriting rules of a UPG satisfy the following condition: If a suffix of the righthand side of a rule matches with a prefix of that of some other rule, then the overlapping portions remain unchanged by the reverse application of these rules. By this condition, UPGs have a kind of confluence property, and thus parsing can be performed deterministically. By extending a UPG, a uniquely parallel parsable unification grammar (UPPUG) [8] has been proposed. It is a unification grammar (UG) version of a UPG in which parallel parsing is also possible.

A simplified version of a UPPUG is a uniquely parallel parsable grammar (UPPG) such that every function symbol is of arity 0. In order to define parallel reduction (i.e., parallel parsing) properly, a "context index" is associated to each rewriting rule in a UPPG, which explicitly specifies the left- and right-context portions of a rewriting rule. Thus rewriting rules of a UPPG satisfy the following condition: If a suffix of the righthand side of a rule matches with a prefix of that of some other rule, then each of these portions is contained in the context portion of each rule. By this, unique parsability also holds for UPPGs. Furthermore, any number of reverse applications of rules to a string can be performed in parallel without interfering each other.

A cellular automaton is one of the fundamental models of parallel computation. Fast language recognition by one-dimensional cellular automata (CA) in parallel has been widely studied so far [1, 2, 3, 6, 12, 13, 14]. In this paper, we show a parallel language recognition process in a one-dimensional cellular automaton can be simply simulated by a parallel reduction in an equivalent UPPG in real time plus 3 steps in the case of a one-way CA (OCA) and in linear time in the case of a two-way CA.

2 Preliminaries

Note in this paper, we omit the definitions and known results on uniquely parsable grammars (UPG), uniquely parallel parsable grammars (UPPG), and cellular automata that are needed in the following sections. See [7, 8, 10] for the details of UPGs and UPPGs. See also [4, 11] for the basic notions of formal languages.

3 Simulation of CAs by UPPGs

Definition 3.1 *Let $L \subseteq Q_0^+$ be a language and $T : \mathbf{N} \to \mathbf{N}$ be a function. We say L is a $T(n)$-time recognizable CA (OCA) language if it can be recognized by some CA (OCA) $C = (Q, Q_0, Q_f, \#, f_C)$ in*

*lijia@ke.sys.hiroshima-u.ac.jp
†imai@ke.sys.hiroshima-u.ac.jp
‡morita@ke.sys.hiroshima-u.ac.jp

time $T(n)$. Moreover, L is said to be a real-time recognizable CA (OCA) language iff $T(n) = n$. Also L is said to be a linear-time recognizable CA (OCA) language iff $T(n) \leq kn$ for some positive integer k.

Theorem 3.1 *Let L be a real-time OCA language. Then there exists a UPPG G such that $L(G) = L$, and each string $w \in L$ can be parsed in exactly $|w| + 3$ steps by a maximum parallel reduction in G.*

Proof. Assume $C = (Q, Q_0, Q_f, \#, f_C)$ be a one-way cellular automaton that recognizes L in real-time ($L \subseteq Q_0^+$). Suppose $Q = \{a_1, \cdots, a_m\}$ and $Q_0 = \{a_1, \cdots, a_l\}$ ($l < m$). Let $a_1^0 a_2^0 \cdots a_n^0$ be a string in L, and the space-time diagram of recognition of this string by C be as follows.

$$\begin{array}{ll}
t = 0 & : a_1^0\, a_2^0 \cdots a_{j-1}^0\, a_j^0 \cdots a_{n-1}^0\, a_n^0 \\
t = 1 & : a_1^1\, a_2^1 \cdots a_{j-1}^1\, a_j^1 \cdots a_{n-1}^1\, a_n^1 \\
t = 2 & : \Delta\, a_2^2 \cdots a_{j-1}^2\, a_j^2 \cdots a_{n-1}^2\, a_n^2 \\
\vdots & \\
t = j & : \Delta\, \Delta \cdots\ \Delta\ \ a_j^j \cdots a_{n-1}^j\, a_n^j \\
\vdots & \\
t = n-1 & : \Delta\, \Delta \cdots\ \Delta\ \ \Delta \cdots a_{n-1}^{n-1}\, a_n^{n-1} \\
t = n & : \Delta\, \Delta \cdots\ \Delta\ \ \Delta \cdots\ \Delta\ \ a_n^n
\end{array}$$

where $a_i^l \in Q$ ($1 \leq i, l \leq n$) and $a_n^n \in Q_f$. Note Δ represents those cells whose states do not influence the real-time recognition result any more, thus can be ignored.

Now, we construct a UPPG G that generates L, and show the language recognition process in C can be easily simulated by a parallel reduction in G. Let $G = (N, Q_0, P, S, \$)$ be a UPPG where

$$N = \{S, S', S''\} \cup \{A_1, \cdots, A_m\} \cup \{*\}.$$

The set P of rules is as follows.

(i) For each $a_i \in Q_f$ include the following rule in P.

$$[\ S \rightarrow S' A_i S'',\ (0,0)\ ]$$

(ii) For each $a_i, a_j, a_h \in Q$ and $f_C(a_i, a_j) = a_h$, including the following rule in P.

$$[\ A_h * A_h \rightarrow A_i A_j *,\ (0,0)\ ]$$

(iii) For each $a_i, a_j, a_h \in Q$ and $f_C(\#, a_i) = a_j$, include the following rules in P.

$$\begin{array}{l}
[\ \$S' A_j \rightarrow \$A_i *,\ \ (1,0)\ ] \\
[\ \ \ S' \rightarrow S' A_h *,\ (1,0)\ ]
\end{array}$$

(iv) For each $a_i \in Q - Q_f$ include the following rules in P.

$$\begin{array}{l}
[\ S''\$ \rightarrow A_i\$,\ \ (0,1)\ ] \\
[\ \ S'' \rightarrow A_i S'',\ (0,1)\ ]
\end{array}$$

(v) For each $a_i \in Q_0$ include the following rule in P.

$$[\ A_i * A_i \rightarrow a_i,\ (0,0)\ ]$$

It is easy to verify that G is a UPPG. Consider the string $a_1^0 a_2^0 \cdots a_n^0$ in L. Using rules in (v) we obtain the following maximum parallel reduction.

$$\begin{array}{l}
\$a_1^0 a_2^0 \cdots a_{j-1}^0 a_j^0 \cdots a_n^0 \$ \\
\underset{G}{\Leftarrow}\ \$A_1^0 * A_1^0 A_2^0 * \cdots A_j^0 * A_j^0 \cdots A_{n-1}^0 A_n^0 * A_n^0 \$
\end{array}$$

Note this sentential form exactly corresponds to the initial configuration in C, where the state a_j^0 of j-th cell is given by a substring $A_j^0 * A_j^0$ for each $1 \leq j \leq n$. After that, by reversely applying rules in (ii) to each substring of the form A_iA_j* in a sentential form simultaneously, state transition of C is properly simulated in G. At the same time, by using rules in (iii) and (iv), we remove those substrings whose corresponding cells in C are not needed in the subsequent computation. Since $a_1^0a_2^0\cdots a_n^0$ is recognized by C in real time, after $n+2$ steps of maximum parallel reduction, we obtain $\$S'A_n^nS''\$$ with $a_n^n \in Q_f$ that can be reduced to $\$S\$$ by reverse application of the rule in (i). The whole parallel parsing process is as follows.

$$
\begin{aligned}
& \$a_1^0a_2^0\cdots a_{j-1}^0a_j^0\cdots a_n^0\$ \\
\Leftarrow\ & \$A_1^0 * A_1^0A_2^0 * \cdots A_{j-1}^0 * A_{j-1}^0A_j^0 * \cdots A_{n-1}^0A_n^0 * A_n^0\$ \\
\Leftarrow\ & \$S'A_1^1A_2^1 * A_2^1\cdots A_{j-1}^1 * A_{j-1}^1A_j^1 * A_j^1\cdots A_{n-1}^1A_n^1 * A_n^1S''\$ \\
\Leftarrow\ & \$S'A_2^2 * A_2^2\cdots A_{j-1}^2 * A_{j-1}^2A_j^2 * A_j^2\cdots A_{n-1}^2A_n^2 * A_n^2S''\$ \\
& \vdots \\
\Leftarrow\ & \$S'A_j^j * A_j^j\cdots A_{n-1}^jA_n^j * A_n^jS''\$ \\
& \vdots \\
\Leftarrow\ & \$S'A_{n-1}^{n-1} * A_{n-1}A_n^{n-1} * A_n^{n-1}S''\$ \\
\Leftarrow\ & \$S'A_n^n * A_n^nS''\$ \\
\Leftarrow\ & \$S'A_n^nS''\$ \\
\Leftarrow\ & \$S\$
\end{aligned}
$$

By above $a_1^0a_2^0\cdots a_n^0 \in L(G)$, thus $L \subseteq L(G)$.

Conversely, suppose $a_1^0a_2^0\cdots a_n^0$ is an arbitrary string in $L(G)$. According to a theorem and a corollary shown in [8] (Theorem 3.2 and Corollary 3.1), the following statement holds: $a_1^0a_2^0\cdots a_n^0 \in L(G)$ iff

$$
\begin{aligned}
& \$a_1^0a_2^0\cdots a_{j-1}^0a_j^0\cdots a_n^0\$ && (1)\\
\underset{G}{\Leftarrow}\ & \$A_1^0 * A_1^0A_2^0 * \cdots A_{j-1}^0 * A_{j-1}^0A_j^0 * \cdots A_{n-1}^0A_n^0 * A_n^0\$ && (2)\\
\overset{*}{\underset{G}{\Leftarrow}}\ & \$S\$ && (3)
\end{aligned}
$$

Note the sentential form (2) exactly corresponds to the initial configuration of C at time 0 with the same input string. After that, the rules in (ii) can be used to reduce this sentential form. Reverse application of these rules in parallel properly reflects the state transition of each cell in C by the local transition function f_C. Furthermore, the rules in (iii) and (iv) are always reversely applicable to the sentential form during the process of maximum parallel reduction, which removes the leftmost substring A_i* and the rightmost A_i at each step. Since $\$a_1^0a_2^0\cdots a_n^0 \in L(G)$, the following maximum parallel reduction exists.

$$
\begin{aligned}
& \$a_1^0a_2^0\cdots a_{j-1}^0a_j^0\cdots a_n^0\$ && (4)\\
\underset{G}{\Leftarrow}\ & \$A_1^0 * A_1^0A_2^0 * \cdots A_{j-1}^0 * A_{j-1}^0A_j^0 * \cdots A_{n-1}^0A_n^0 * A_n^0\$ && (5)\\
\overset{n}{\underset{G}{\Leftarrow}}\ & \$S'A_n^n * A_n^nS''\$ && (6)
\end{aligned}
$$

where $A_n^n \in Q_f$. Thus, the input string can be recognized by C in exactly n steps, i.e. $a_1^0a_2^0\cdots a_n^0 \in L$ is obtained. This derives $L \supseteq L(G)$. Combining the above two arguments, we have $L = L(G)$.

In the above UPPG G, each string $w \in L$ is parsed in $|w|+3$ steps by a maximum parallel reduction. Hence we have the theorem. □

Note the method used in Theorem 3.1 to construct a UPPG that generates a given real-time OCA language can be easily extended to construct a UPPG to generate a linear-time OCA language. As shown by Bucher and Culik (1984) [1], the class of linear-time OCA languages is equivalent to that of $2n$-time OCA languages, where n is the length of input. Thus we have the following theorem.

Theorem 3.2 *For a linear-time OCA language L, there exists a UPPG G such that $L(G) = L$, and each string $w \in L$ can be parsed in exactly $2|w| + 3$ steps by a maximum parallel reduction in G.*

Moreover, according to Umeo et al. [15] (see also [1, 2]), each CA that recognizes a language L in real time can be simulated by an equivalent OCA in $2n$-time. Hence the next corollary holds.

Corollary 3.1 *For a real-time CA language L, there exists a UPPG G that generates L, and each string $w \in L$ can be parsed in exactly $2|w| + 3$ steps by a maximum parallel reduction in G.*

Theorem 3.3 *Let L be a $T(n)$-time CA language. Then there exists a UPPG that generates L, and each string $w \in L$ can be parsed in $2(T(|w|) + |w|)$ steps by a parallel reduction in G.*

Proof. Assume a CA $C = (Q, Q_0, Q_f, \#, f_C)$ recognizes L in time $T(n)$ where n is the length of an input string. Suppose $Q = \{a_1, \cdots, a_m\}$ and $Q_0 = \{a_1, \cdots, a_l\}$ $(l < m)$. Let $G = (N, Q_0, P, S, \$)$ be a UPPG where

$$N = \{S, [,], (,)\} \cup \{A_1, \cdots, A_m\}.$$

P is defined as follows.

(i) For each $a_i, a_j \in Q$ and $a_l \in Q_f$, include the following rules in P.

$$\begin{array}{l} [\ S\$ \rightarrow [A_iA_l\$,\ (0,1)\ ] \\ [\ \ S \rightarrow (A_jS,\ \ \ (0,1)\ ] \\ [\ \ S \rightarrow [A_iA_jS,\ (0,1)\ ] \\ [\ \$S \rightarrow \$A_iA_jS,\ (0,1)\ ] \end{array}$$

(ii) For each $a_i, a_j, a_h, a_l, a_k \in Q$, $f_C(a_i, a_j, a_k) = a_h$ and $f_C(\#, a_j, a_k) = a_l$, include the following rules in P.

$$\begin{array}{l} [\ A_h)A_h(A_h \rightarrow [A_iA_jA_k],\ (0,0)\ ] \\ [\ \ \ \$A_l(A_l \rightarrow \$A_jA_k],\ \ (1,0)\ ] \end{array}$$

(iii) For each $a_i, a_l \in Q$, $a_j \in Q - Q_f$ and $f_C(a_i, a_j, \#) = a_l$, include the following rule in P.

$$[\ A_l)A_i\$ \rightarrow [A_iA_j\$,\ (0,1)\ ]$$

(iv) For each $a_i, a_j \in Q$, include the following rule in P.

$$[\ A_j][A_i \rightarrow (A_iA_j),\ (0,0)\ ]$$

(v) For each $a_i \in Q_0$, include the following rules in P.

$$\begin{array}{l} [\ A_i)A_i(A_i \rightarrow a_i,\ \ \ \ \ \ (0,0)\ ] \\ [\ \ \ \ \ \$A_i \rightarrow \$A_i)A_i,\ (1,1)\ ] \\ [\ \ \ \ \ A_i\$ \rightarrow A_i(A_i\$,\ (1,1)\ ] \end{array}$$

The idea to prove $L(G) = L$ is similar to that in Theorem 3.1. Let $a_1^0 a_2^0 \cdots a_n^0$ be a string in L. The space-time diagram of recognition of this string in C is given below where $a_n^{T(n)} \in Q_f$.

$$\begin{array}{lcccccccc} t=0 & : & a_1^0 & a_2^0 & \cdots & a_{j-1}^0 & a_j^0 & \cdots & a_{n-1}^0 & a_n^0 \\ t=1 & : & a_1^1 & a_2^1 & \cdots & a_{j-1}^1 & a_j^1 & \cdots & a_{n-1}^1 & a_n^1 \\ t=2 & : & a_1^2 & a_2^2 & \cdots & a_{j-1}^2 & a_j^2 & \cdots & a_{n-1}^2 & a_n^2 \\ \vdots & & & & & & & & & \\ t=j & : & a_1^j & a_2^j & \cdots & a_{j-1}^j & a_j^j & \cdots & a_{n-1}^j & a_n^j \\ \vdots & & & & & & & & & \\ t=T(n)-1 & : & a_1^{T(n)-1} & a_2^{T(n)-1} & \cdots & a_{j-1}^{T(n)-1} & a_j^{T(n)-1} & \cdots & a_{n-1}^{T(n)-1} & a_n^{T(n)-1} \\ t=T(n) & : & a_1^{T(n)} & a_2^{T(n)} & \cdots & a_{j-1}^{T(n)} & a_j^{T(n)} & \cdots & a_{n-1}^{T(n)} & a_n^{T(n)} \end{array}$$

where $a_i^j \in Q$ $(1 \leq i \leq n,\ 1 \leq j \leq T(n))$, and $a_n^{T(n)} \in Q_f$. On the other hand, parsing of the same string by a maximum parallel reduction in G is shown below.

$$
\begin{aligned}
& \$a_1^0 a_2^0 \cdots a_{j-1}^0 a_j^0 a_{j+1}^0 \cdots a_{n-1}^0 a_n^0 \$ \\
\Leftarrow\ & \$A_1^0)A_1^0(A_1^0A_2^0)\cdots(A_{j-1}^0A_j^0)A_j^0(A_j^0A_{j+1}^0)\cdots A_{n-1}^0(A_{n-1}^0A_n^0)A_n^0(A_n^0\$ \\
\Leftarrow\ & \$A_1^0A_2^0][A_1^0A_2^0A_3^0]\cdots[A_{j-1}^0A_j^0A_{j+1}^0]\cdots[A_{n-2}^0A_{n-1}^0A_n^0][A_{n-1}^0A_n^0\$ \\
\Leftarrow\ & \$A_1^1(A_1^1A_2^1)A_2^1(\cdots(A_{j-1}^1A_j^1)A_j^1(A_j^1A_{j+1}^1)\cdots A_{n-1}^1(A_{n-1}^1A_n^1)A_n^1\$ \\
\Leftarrow\ & \$A_1^1A_2^1][A_1^1A_2^1A_3^1]\cdots[A_{j-1}^1A_j^1A_{j+1}^1]\cdots[A_{n-2}^1A_{n-1}^1A_n^1][A_{n-1}^1A_n^1\$ \\
& \vdots \\
\Leftarrow\ & \$A_1^j(A_1^jA_2^j)A_2^j(\cdots(A_{j-1}^jA_j^j)A_j^j(A_j^jA_{j+1}^j)\cdots A_{n-1}^j(A_{n-1}^jA_n^j)A_n^j\$ \\
\Leftarrow\ & \$A_1^jA_2^j][A_1^jA_2^jA_3^j]\cdots[A_{j-1}^jA_j^jA_{j+1}^j]\cdots[A_{n-2}^jA_{n-1}^jA_n^j][A_{n-1}^jA_n^j\$ \\
& \vdots \\
\Leftarrow\ & \$A_1^{T(n)}(A_1^{T(n)}A_2^{T(n)})\cdots(A_{j-1}^{T(n)}A_j^{T(n)})A_j^{T(n)}(A_j^{T(n)}A_{j+1}^{T(n)})\cdots(A_{n-1}^{T(n)}A_n^{T(n)})A_n^{T(n)}\$ \\
\Leftarrow\ & \$A_1^{T(n)}A_2^{T(n)}]\cdots[A_{j-1}^{T(n)}A_j^{T(n)}A_{j+1}^{T(n)}]\cdots A_n^{T(n)}][A_{n-1}^{T(n)}A_n^{T(n)}\$ \\
\Leftarrow\ & \$A_1^{T(n)+1}(A_1^{T(n)+1}\cdots A_j^{T(n)+1})A_j^{T(n)+1}(A_j^{T(n)+1}\cdots(A_{n-1}^{T(n)+1}S\$ \\
\Leftarrow\ & \$A_1^{T(n)+1}A_2^{T(n)+1}]\cdots[A_{j-1}^{T(n)+1}A_j^{T(n)+1}A_{j+1}^{T(n)+1}]\cdots S\$ \\
& \vdots \\
\Leftarrow\ & \$A_1^{T(n)+n-1}A_2^{T(n)+n-1}S\$ \\
\Leftarrow\ & \$S\$
\end{aligned}
$$

Note the j-th configuration in C at time j $(0 \leq j \leq T(n))$ is expressed by the sentential form $\$A_1^j(A_1^jA_2^j)A_2^j(\cdots(A_{l-1}^jA_l^j)A_l^j(A_l^jA_{l+1}^j)\cdots A_{n-1}^j(A_{n-1}^jA_n^j)A_n^j\$$ in G, in which the state of the l-th cell $(1 < l < m)$ is represented by a substring $A_l^j)A_l^j(A_l^j$ (the first and the last cells are $\$A_1^j(A_1^j$ and $A_n^j)A_n^j\$$). After that, the rules in (iv) are reversely applied in parallel, and then rules in (ii) and (iii). It is easy to verify that such two steps of maximum parallel reduction exactly simulates the state transition of each cell in C from j-th to $j+1$-st configuration. Since $a_1^0a_2^0\cdots a_n^0 \in L$, the rightmost cell $a_n^{T(n)}$ in the $T(n)$-th configuration in C enters an accepting state. This recognition process is properly simulated by a maximum parallel reduction in $2T(n)$ steps in G. Then the rules in (i) can be used to reduce the sentential form to $\$S\$$ in $2n$ steps. Hence $a_1^0a_2^0\cdots a_n^0 \in L(G)$ is obtained. Thus, we have $L \subseteq L(G)$.

Conversely, consider an arbitrary string $a_1^0a_2^0\cdots a_n^0$ in $L(G)$. Then by using rules in (v) and (iv), we have

$$
\begin{aligned}
& \$a_1^0a_2^0\cdots a_{j-1}^0a_j^0a_{j+1}^0\cdots a_{n-1}^0a_n^0\$ \\
\Leftarrow\ & \$A_1^0)A_1^0(A_1^0A_2^0)\cdots(A_{j-1}^0A_j^0)A_j^0(A_j^0A_{j+1}^0)\cdots A_{n-1}^0(A_{n-1}^0A_n^0)A_n^0(A_n^0\$
\end{aligned}
$$

which exactly represents the initial configuration in C with the same input. After that, the rules in (ii), (iii) and (iv) can be used in the subsequent maximum parallel reduction steps. We can see this parallel reduction process exactly simulates the language recognition process in C. Moreover, because $\$a_1^0a_2^0\cdots a_n^0\$ \overset{*}{\Leftarrow} \$S\$$, there must be some $\hat{t} > 0$ such that $\$a_1^0a_2^0\cdots a_n^0\$ \overset{*}{\Leftarrow} \$A_1^{\hat{t}}A_2^{\hat{t}}\cdots A_n^{\hat{t}}\$$ and $a_n^{\hat{t}} \in Q_f$. Thus the string $a_1^0a_2^0\cdots a_n^0$ is accepted by C (and by the assumption, $\hat{t} = T(n)$ holds). Hence we have $L \supseteq L(G)$.

Combining the above arguments, we obtain $L = L(G)$, and each string $w \in L$ is parsed in $2(T(|w|) + |w|)$ steps by a maximum parallel reduction in G. Thus we have the theorem. □

4 Concluding Remarks

In this paper, we showed the parallel language recognition process in a one-dimensional cellular automaton (CA) can be simply simulated by a parallel reduction in an equivalent uniquely parallel parsable grammar (UPPG). The main result of our research is that for a CA that can recognize a language in time $T(n)$, there exists a UPPG that generates the same language, where each string of length n can be parsed by a maximum parallel reduction in $kT(n)$ steps ($k > 0$). In our previous researches, we have shown that there are non-context-free languages that can be generated by UPPGs, and they are parsed by a maximum parallel reduction in sublinear steps of the length of the input [7]. However, such efficiency in language recognition can not be achieved by any CA, since at least n steps are needed to recognize a nontrivial language in CA.

References

[1] W. Bucher and K. Culik II, On real time and linear time cellular automata, *R.A.I.R.O. Information Theory*, Vol. 18, pp. 307-325, 1984.

[2] C. Choffrut and K. Culik II, On real-time cellular automata and trellis automata, *Acta Informatica*, Vol. 21, pp. 393-407, 1984.

[3] C. Dyer, One-way bounded cellular automata, *Information and Control*, Vol. 44, pp. 54-69, 1980.

[4] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, Massachusetts, 1979.

[5] O.H. Ibarra, M.P. Palis, and S.M. Kim, Fast parallel language recognition by cellular automata, *Theoret. Computer Science*, Vol. 41, pp. 231-246, 1985.

[6] T. Kasami and M. Fujii, Some results on capabilities of one-demensional iterative logical networks, *Electrical and Communication*, Vol. 51-C, pp. 167-176, 1968.

[7] J. Lee and K. Morita, Generation and parsing of the Fibonacci language by uniquely parallel parsable grammars, *Proc. of LA Summer Symposium*, pp. 10.1-10.10, 2000.

[8] J. Lee and K. Morita, Uniquely parallel parsable unification grammars, *IEICE Trans. on Information and Systems*, Vol. E84-D, No. 1, pp. 21-27, 2001.

[9] J. Lee, K. Morita, H. Asou, and K. Imai, Uniquely parsable unification grammars and their parser implemented in Prolog, *Grammars* Vol. 3, No. 1, pp. 63-81, 2000.

[10] K. Morita, N. Nishihara, Y. Yamamoto, and Z. Zhang, A hierarchy of uniquely parsable grammar classes and deterministic acceptors, *Acta Informatica*, Vol. 34, pp. 389–410, 1997.

[11] G. Rozenberg and A. Salomaa, (eds.), *Handbook of Formal Languages*, Vols. 1–3, Springer-Verlag, Berlin, 1997.

[12] A.R. Smith III, Cellular automata and formal languages, *Proc. 11th IEEE Annual Symp. on Switching and Automata Theory*, pp. 216-224, 1970.

[13] A.R. Smith III, Real-time language recognition by one-dimensional cellular automata, *J. Computer and System Sciences*, Vol. 6, pp. 233-253, 1972.

[14] V. Terrier, On real time one-way cellular array, *Theoret. Computer Science*, Vol. 141, pp. 331-335, 1995.

[15] H. Umeo, K. Morita, and K. Sugata, Deterministic one-way simulation of two-way real-time cellular automata and its related problems, *Information Process. Lett.*, Vol. 14, pp. 159-161, 1982.