On Parallel Versions of Jumping Finite Automata

Radim Kocman
Department of Information Systems
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno Czech Republic
ikocman@fit.vutbr.cz

Alexander Meduna
Department of Information Systems
Faculty of Information Technology
Brno University of Technology
Božetěchova 2, Brno Czech Republic
meduna@fit.vutbr.cz

***Abstract*—The present paper proposes a new investigation area in automata theory — *n-parallel jumping finite automata*. These automata further extend recently presented jumping finite automata that are focused on discontinuous reading. The proposed modification uses multiple reading heads that work in parallel and can discontinuously read from the input in several places at once. We also define the more restricted version of these automata which only allows jumping to the right. This restricted version is then further studied, compared with n-parallel right linear grammars, and several of its properties are derived.**

***Index Terms*—Jumping finite automata, n-parallel right linear grammars, discontinuous tape reading, parallel tape reading.**

I. Introduction

In the previous century, most formal models were designed for continuous information processing. This, however, does not often reflects the requirements of modern information methods. Therefore, there is currently active research around formal models that process information in a discontinuous way. Most notably, there are newly invented *jumping finite automata* (see [1]) that are completely focused on discontinuous reading. These automata go so far that they cannot even define some quite simple languages (e.g. a^*b^*) because they cannot guarantee any specific reading order between their jumps.

The present paper proposes the modification of these automata — *n-parallel jumping finite automata*. This modification presents a concept where the input is divided into several arbitrary parts and these parts are then separately processed with distinct synchronized heads. A quite similar concept was thoroughly studied in terms of formal grammars, where several nonterminals are being synchronously rewritten at once; for example, simple matrix grammars (see [2]) and n-parallel grammars (see [3], [4], [5], [6], [7]). However, to the best of our knowledge, no such research was done in terms of automata, where several heads synchronously read from distinct parts on the single tape. When this concept is combined with the mechanics of jumping finite automata, each part can be read discontinuously, but the overall order between parts is preserved; such automaton then can handle additional languages (e.g. a^*b^*). Therefore, this modification represents the combined model of discontinuous and continuous reading.

The unrestricted version of jumping finite automata handles a quite unique language family, which has not yet been sufficiently studied and which had no counterparts in grammars; until jumping grammars were introduced (see [8]). Therefore, we decided to base our initial research on the restricted version of these automata, which use only right jumps. Such restricted jumping finite automata define the same language family as classical finite automata. When these restricted automata are combined with the previously described concept, we get a model which is very similar to n-parallel grammars. Such automata utilize jumping only during the initialization, when heads jump to their start positions. After that, all heads read their parts of the input continuously in a left-to-right way. The paper compares these automata with n-parallel right linear grammars and shows that these models actually represent the same language families. Consequently, several properties of these automata are derived from the previous results.

II. Preliminaries

This paper assumes that the reader is familiar with the theory of automata and formal languages (see [9], [10]). Let $\mathbb{N}$ denote the set of all positive integers. For a set Q, $card(Q)$ denotes the cardinality of Q. For an alphabet (finite nonempty set) V, V^* represents the free monoid generated by V under the operation of concatenation. The unit of V^* is denoted by ε. For $x \in V^*$, $|x|$ denotes the length of x, and $alph(x)$ denotes the set of all symbols occurring in x; for instance, $alph(0010) = \{0, 1\}$. For $a \in V$, $|x|_a$ denotes the number of occurrences of a in x. Let $x = a_1a_2 \dots a_n$, where $a_i \in V$ for all $i = 1, \dots, n$, for some $n \geq 0$ ($x = \varepsilon$ if and only if $n = 0$).

A *general jumping finite automaton* (see [1]), a *GJFA* for short, is a quintuple $M = (Q, \Sigma, R, s, F)$, where Q is a finite set of states, Σ is an input alphabet, $Q \cap \Sigma = \emptyset$, $R \subseteq Q \times \Sigma^* \times Q$ is finite, $s \in Q$ is the start state, and F is a set of final states. Members of R are referred to as rules of M and instead of $(p, y, q) \in R$, we write $py \to q \in R$. A *configuration* of M is any string in $\Sigma^*Q\Sigma^*$. The binary *jumping relation*, symbolically denoted by $\curvearrowright$, over $\Sigma^*Q\Sigma^*$, is defined as follows. Let $x, z, x', z' \in \Sigma^*$ such that $xz = x'z'$ and $py \to q \in R$; then, M makes a *jump* from $xpyz$ to $x'qz'$, symbolically written as $xpyz \curvearrowright x'qz'$. In the standard manner, we extend $\curvearrowright$ to $\curvearrowright^m$, where $m \geq 0$. Let $\curvearrowright^+$ and $\curvearrowright^*$ denote the transitive closure of $\curvearrowright$ and the transitive-reflexive closure of $\curvearrowright$, respectively. The language accepted by M, denoted by $L(M)$, is defined as $L(M) = \{uv \mid u, v \in \Sigma^*, usv \curvearrowright^* f, f \in F\}$. We also define the special case of the jumping relation. Let $w, x, y, z \in \Sigma^*$, and $py \rightarrow q \in R$; then, M makes a *right jump* from $wpyxz$ to $wxqz$, written as $wpyxz\ {}_r\!\curvearrowright\ wxqz$. We extend ${}_r\!\curvearrowright$ to ${}_r\!\curvearrowright^m$, ${}_r\!\curvearrowright^*$, and ${}_r\!\curvearrowright^+$, where $m \geq 0$, by analogy with extending the corresponding notations for $\curvearrowright$. The language accepted by M using only right jumps, denoted by ${}_rL(M)$, is defined as ${}_rL(M) = \{uv \mid u, v \in \Sigma^*, usv\ {}_r\!\curvearrowright^*\ f, f \in F\}$. Let $w \in \Sigma^*$. We say that M accepts w if and only if $w \in L(M)$. M rejects w if and only if $w \in \Sigma^* - L(M)$. Two GJFAs M and M' are said to be equal if and only if $L(M) = L(M')$.

Let $n \in \mathbb{N}$. An *n-parallel right linear grammar* (see [3], [4], [5], [6], [7]), an *n-PRLG* for short, is an $(n+3)$-tuple $G = (N_1, \dots, N_n, T, S, P)$, where N_i, $1 \leq i \leq n$, are mutually disjoint nonterminal alphabets, T is a terminal alphabet, S is the sentence symbol, S not in $N_1 \cup \dots \cup N_n \cup T$, and P is a finite set of pairs. Members of P are referred as rules of G and instead of $(X, x) \in P$, we write $X \rightarrow x \in P$. Each rule in P has one of the following forms: (1) $S \rightarrow X_1 \dots X_n$, $X_i \in N_i$, $1 \leq i \leq n$, (2) $X_i \rightarrow a_i Y_i$, $X_i, Y_i \in N_i$, $a_i \in T^*$, $1 \leq i \leq n$, (3) $X_i \rightarrow a_i$, $X_i \in N_i$, $a_i \in T^*$, $1 \leq i \leq n$. The binary *yield operation*, symbolically denoted by $\Rightarrow$, is defined as follows. Let $x, y \in (N_1 \cup \dots \cup N_n \cup \{S\} \cup T)^*$ then $x \Rightarrow y$ iff either $x = S$ and $S \rightarrow y \in P$ or $x = a_1 X_1 \dots a_n X_n$, $y = a_1 x_1 \dots a_n x_n$ and $a_i \in T^*$, $X_i \in N_i$, $X_i \rightarrow x_i \in P$, $1 \leq i \leq n$. In the standard manner, we extend $\Rightarrow$ to $\Rightarrow^m$, where $m \geq 0$. Let $\Rightarrow^+$ and $\Rightarrow^*$ denote the transitive closure of $\Rightarrow$ and the transitive-reflexive closure of $\Rightarrow$, respectively. The language generated by G, denoted by $L(G)$, is defined as $L(G) = \{x \mid S \Rightarrow^* x,\ x \in T^*\}$.

III. Definitions and Examples

In this section, we define the modification of jumping finite automata — n-parallel jumping finite automata — which read input words discontinuously with multiple synchronized heads. Consequently, we also define the more restricted version of these autamata which uses only right jumps.

Definition 1. *Let $n \in \mathbb{N}$. An* n-parallel general jumping finite automaton*, an* n-PGJFA *for short, is a quintuple*

$$M = (Q, \Sigma, R, S, F),$$

where Q is a finite set of states, Σ is an input alphabet, $Q \cap \Sigma = \emptyset$, $R \subseteq Q \times \Sigma^* \times Q$ is finite, $S \subseteq Q^n$ is a set of start state strings, and F is a set of final states. Members of R are referred to as rules of M and instead of $(p, y, q) \in R$, we write $py \rightarrow q \in R$.

A configuration *of M is any string in $\Sigma^* Q \Sigma^*$. Let X denote the set of all configurations over M. The binary* jumping relation*, symbolically denoted by $\curvearrowright$, over X, is defined as follows. Let $x, z, x', z' \in \Sigma^*$ such that $xz = x'z'$ and $py \rightarrow q \in R$; then, M makes a* jump *from $xpyz$ to $x'qz'$, symbolically written as*

$$xpyz \curvearrowright x'qz'.$$

Let \$ be a special symbol, $\$ \notin Q \cup \Sigma$. An n-configuration *of M is any string in $(X\{\$\})^n$. Let ${}_nX$ denote the set of all n-configurations over M. The binary* n-jumping relation*, symbolically denoted by ${}_n\!\curvearrowright$, over ${}_nX$, is defined as follows. Let $\zeta_1\$ \dots \zeta_n\$, \vartheta_1\$ \dots \vartheta_n\$ \in {}_nX$, so $\zeta_i, \vartheta_i \in X$, $1 \leq i \leq n$; then, M makes an* n-jump *from $\zeta_1\$ \dots \zeta_n\$$ to $\vartheta_1\$ \dots \vartheta_n\$$, symbolically written as*

$$\zeta_1\$ \dots \zeta_n\$\ {}_n\!\curvearrowright\ \vartheta_1\$ \dots \vartheta_n\$$$

iff $\zeta_i \curvearrowright \vartheta_i$ for all $1 \leq i \leq n$. In the standard manner we extend ${}_n\!\curvearrowright$ to ${}_n\!\curvearrowright^m$, where $m \geq 0$. Let ${}_n\!\curvearrowright^+$ and ${}_n\!\curvearrowright^*$ denote the transitive closure of ${}_n\!\curvearrowright$ and transitive-reflexive closure of ${}_n\!\curvearrowright$, respectively.

The language accepted by M, denoted by $L(M, n)$, is defined as $L(M, n) = \{u_1 v_1 \dots u_n v_n \mid s_1 \dots s_n \in S,\ u_i, v_i \in \Sigma^*,\ u_1 s_1 v_1\$ \dots u_n s_n v_n\$\ {}_n\!\curvearrowright^* f_1\$ \dots f_n\$,\ f_i \in F,\ 1 \leq i \leq n\}$. Let $w \in \Sigma^*$. We say that M accepts w if and only if $w \in L(M, n)$. M rejects w if and only if $w \in \Sigma^* - L(M, n)$.

Definition 2. *Let $M = (Q, \Sigma, R, S, F)$ be an n-PGJFA, and let X denote the set of all configurations over M. The binary right jumping relation, symbolically denoted by ${}_r\!\curvearrowright$, over X, is defined as follows. Let $w, x, y, z \in \Sigma^*$, and $py \rightarrow q \in R$; then, M makes a* right jump *from $wpyxz$ to $wxqz$, symbolically written as*

$$wpyxz\ {}_r\!\curvearrowright\ wxqz.$$

Let ${}_nX$ denote the set of all n-configurations over M. The binary right n-jumping relation, symbolically denoted by ${}_{n-r}\!\curvearrowright$, over ${}_nX$, is defined as follows. Let $\zeta_1\$ \dots \zeta_n\$, \vartheta_1\$ \dots \vartheta_n\$ \in {}_nX$, so $\zeta_i, \vartheta_i \in X$, $1 \leq i \leq n$; then, M makes a right n-jump *from $\zeta_1\$ \dots \zeta_n\$$ to $\vartheta_1\$ \dots \vartheta_n\$$, symbolically written as*

$$\zeta_1\$ \dots \zeta_n\$\ {}_{n-r}\!\curvearrowright\ \vartheta_1\$ \dots \vartheta_n\$$$

iff $\zeta_i\ {}_r\!\curvearrowright \vartheta_i$ for all $1 \leq i \leq n$.

Extend ${}_{n-r}\!\curvearrowright$ to ${}_{n-r}\!\curvearrowright^m$, ${}_{n-r}\!\curvearrowright^+$, and ${}_{n-r}\!\curvearrowright^*$, where $m \geq 0$, by analogy with extending the corresponding notations for ${}_n\!\curvearrowright$. Let $L(M, n{-}r)$ denote the language accepted by M using only right n-jumps.

Next, we illustrate the previous definitions by two examples.

Example 3. *Consider the 2-PGJFA*

$$M = (\{s, r, p, q\}, \Sigma, R, \{sr\}, \{s, r\}),$$

where $\Sigma = \{a, b, c, d\}$ and R consists of the rules

$$sa \rightarrow p,\ \ pb \rightarrow s,\ \ rc \rightarrow q,\ \ qd \rightarrow r.$$

Starting from sr, M has to read some a, and some b with the first head and some c, and some d with the second head, entering again the start (and also the final) states sr. Therefore, the accepted language is

$$L(M, 2) = \{uv \mid u \in \{a, b\}^*,\ v \in \{c, d\}^*,\ |u|_a = |u|_b = |v|_c = |v|_d\}.$$

It can be easily shown that such a language cannot be defined by any original jumping finite automaton.

Example 4. *Consider the 2-PGJFA*

$$M = (\{s, r, t\}, \Sigma, R, \{ss\}, \{s\}),$$

where $\Sigma = \{a, b, c\}$ *and* R *consists of the rules*

$$sa \to r, \ \ rb \to t, \ \ tc \to s.$$

Starting from ss, M has to read some a, some b, and some c with both heads. If we work with unbound jumps, each head can read a, b, and c in an arbitrary order. However, if we work only with right jumps, each head must read input symbols in the original order; or the automaton will eventually get stuck. Therefore, the accepted languages are

$$L(M, 2) = \{uv \mid u, v \in \{a, b, c\}^*, \\ |u|_a = |u|_b = |u|_c = |v|_a = |v|_b = |v|_c\},$$

$$L(M, 2{-}r) = \{uu \mid u \in \{abc\}^*\}.$$

Denotation of language families

Throughout the rest of this paper, the language families under discussion are denoted in the following way. **REG**, **CF**, and **CS** denote the families of regular languages, context-free languages, and context-sensitive languages, respectively. $_r$**GJFA**, $_r n$**-PGJFA**, and n**-PRLG** denote the families of languages accepted or generated by GJFAs using only right jumps, n-PGJFAs using only right n-jumps, and n-PRLGs, respectively.

IV. Conversions

In this section, we prove that n-PGJFAs with right n-jumps and n-PRLGs define the same language families.

Theorem 5. *For every* n-PRLG $G = (N_1, \dots, N_n, T, S1, P)$, *there is an* n-PGJFA *using only* right n-jumps $M = (Q, \Sigma, R, S2, F)$, *such that* $L(M, n{-}r) = L(G)$.

Proof. Let $G = (N_1, \dots, N_n, T, S1, P)$ be an n-PRLG. Without a loss of generality, assume that $f \notin N_1 \cup \dots \cup N_n \cup T$. Keep the same n and define the n-PGJFA with right n-jumps

$$M = (\{f\} \cup N_1 \cup \dots \cup N_n, T, R, S2, \{f\}),$$

where R and $S2$ are constructed in the following way:

(1) For each rule in the form $S1 \to X_1 \dots X_n$, $X_i \in N_i$, $1 \leq i \leq n$, add the start state string $X_1 \dots X_n$ to $S2$.
(2) For each rule in the form $X_i \to a_i Y_i$, $X_i, Y_i \in N_i$, $a_i \in T^*$, $1 \leq i \leq n$, add the rule $X_i a_i \to Y_i$ to R.
(3) For each rule in the form $X_i \to a_i$, $X_i \in N_i$, $a_i \in T^*$, $1 \leq i \leq n$, add the rule $X_i a_i \to f$ to R.

The constructed n-PGJFA with right n-jumps M simulates the n-PRLG G in such a way that its heads read symbols in the same fashion as the nonterminals of G generate them.

Any sentence $w \in L(G)$ can be divided into $w = u_1 \dots u_n$, where u_i represents the part of the sentence which can be generated from the nonterminal X_i of a rule $S1 \to X_1 \dots X_n$, $X_i \in N_i$, $1 \leq i \leq n$. In the same way, M can start from an n-configuration $X_1 u_1 \$ \dots X_n u_n \$$, where all its heads with the states X_i need to read u_i. Therefore part (1), where we convert the rules $S1 \to X_1 \dots X_n$ into the start state strings. The selection of a start state string thus covers the first derivation step of the grammar.

Any consecutive non-ending derivation step of the grammar then rewrites all n nonterminals in the sentential form with the rules $X_i \to a_i Y_i$, $X_i, Y_i \in N_i$, $a_i \in T^*$, $1 \leq i \leq n$. Therefore part (2), where we convert the grammar rules $X_i \to a_i Y_i$ into the automaton rules $X_i a_i \to Y_i$. The automaton M always works with all its heads simultaneously and thus the equivalent effect of these steps should be obvious.

In the last derivation step of the grammar, every nonterminal is rewritten with a rule $X_i \to a_i$, $X_i \in N_i$, $a_i \in T^*$, $1 \leq i \leq n$. We can simulate the same behavior in the automaton if we end up in a final state for which there are no ongoing rules. Therefore part (3), where we convert the grammar rules $X_i \to a_i$ into the automaton rules $X_i a_i \to f$, where f is the sole final state. All heads of the automaton must also simultaneously end up in the final state or the automaton will get stuck; there are no ongoing rules from f and all heads must make a move during every step.

The automaton M can also start from an n-configuration where the input is divided into such parts that they cannot be generated from the nonterminals X_i of the rules $S1 \to X_1 \dots X_n$, $X_i \in N_i$, $1 \leq i \leq n$. However, such an attempt will eventually get the automaton stuck because the automaton simulates only derivation steps of the grammar. □

Theorem 6. *For every* n-PGJFA *using only* right n-jumps $M = (Q, \Sigma, R, S2, F)$, *there is an* n-PRLG $G = (N_1, \dots, N_n, T, S1, P)$, *such that* $L(G) = L(M, n{-}r)$.

Proof. Let $M = (Q, \Sigma, R, S2, F)$ be an n-PGJFA with right n-jumps. Keep the same n and define the n-PRLG

$$G = (N_1, \dots, N_n, \Sigma, S1, P),$$

where $N_1, \dots, N_n$, and P are constructed in the following way:

(1) For each state $p \in Q$, add the nonterminal p_i to N_i for all $1 \leq i \leq n$.
(2) For each start state string $p_1 \dots p_n \in S2$, $p_i \in Q$, $1 \leq i \leq n$, add the start rule $S1 \to p_{1_1} \dots p_{n_n}$ to P.
(3) For each rule $py \to q$, $p, q \in Q$, $y \in \Sigma^*$, add the rule $p_i \to y q_i$ to P for all $1 \leq i \leq n$.
(4) For each state $p \in F$, add the rule $p_i \to \varepsilon$ to P for all $1 \leq i \leq n$.

The constructed n-PRLG G simulates the n-PGJFA with right n-jumps M in such a way that its nonterminals generate terminals in the same fashion as the heads of M read them.

The definition of n-PRLGs requires that $N_1, \dots, N_n$ are mutually disjoint nonterminal alphabets. However, the states of n-PGJFAs do not have such a restriction. Therefore, we use a new index in each converted occurrence of a state, this creates a separate item for every nonterminal position. The index is represented by i and is used in all conversion steps.

Any sentence $w \in L(M, n{-}r)$ can be divided into $w = u_1 \dots u_n$, where u_i represents the part of the sentence which can be accepted by the head of M with a start state p_i from a start n-configuration $p_1 u_1\$ \dots p_n u_n\$$, where $p_1 \dots p_n \in S2$, $1 \leq i \leq n$. In the grammar, we can simulate the start n-configurations with the start rules $S1 \rightarrow p_{1_1} \dots p_{n_n}$, where the nonterminals p_{i_i} must be able to generate u_i. Therefore part (2), where we convert the start state strings into the rules.

During every step of the automaton all heads simultaneously make a move. Likewise, during every non-start step of the grammar all non-terminals are simultaneously rewritten. Therefore part (3), where we convert the automaton rules $py \rightarrow q$ into the grammar rules $p_i \rightarrow yq_i$. The equivalent effect of these steps should be obvious.

The automaton can successfully end if all its heads are in the final states. We can simulate this step in the grammar if we rewrite every nonterminal with ε. Therefore part (4), where we create new empty rules for all final states. These rules can be used only once during the last derivation step of the grammar; otherwise, the grammar will get stuck. □

Theorem 7. $_r n$-***PGJFA*** $\subset n$-***PRLG***.

Proof. This theorem directly follows from Theorem 6. □

Theorem 8. n-***PRLG*** $\subset {}_r n$-***PGJFA***.

Proof. This theorem directly follows from Theorem 5. □

Corollary 9. $_r n$-***PGJFA*** $= n$-***PRLG***. □

V. CHARACTERIZATION

Theorem 10. *For all* $n \in \mathbb{N}$, $_r n$-***PGJFA*** $\subset {}_r(n+1)$-***PGJFA***.

Proof. This theorem directly follows from n-**PRLG** $\subset (n+1)$-**PRLG** (see [3]). □

Theorem 11. *For all* $n \in \mathbb{N}$, $_r n$-***PGJFA*** *is closed under union, finite substitution, homomorphism, reflection, and intersection with a regular set.*

Proof. This theorem directly follows from the same results for n-**PRLG** (see [3]). □

Theorem 12. *For all* $n > 1$, $_r n$-***PGJFA*** *is not closed under intersection or complement.*

Proof. This theorem directly follows from the same results for n-**PRLG** (see [3]). □

Theorem 13. $_r 1$-***PGJFA*** $= {}_r$***GJFA*** $=$ ***REG***.

Proof. This theorem directly follows from 1-**PRLG** $=$ **REG** (see [3]), and from $_r$**GJFA** $=$ **REG** (see [1]). □

Theorem 14. $_r 2$-***PGJFA*** $\subset$ ***CF***.

Proof. This theorem directly follows from 2-**PRLG** $\subset$ **CF** (see [3]). □

Theorem 15. $_r n$-***PGJFA*** $\subset$ ***CS*** *and there exist non-context-free languages in* $_r n$-***PGJFA*** *for all* $n > 2$.

Proof. This theorem directly follows from the same results for n-**PRLG** (see [3]). □

VI. REMARKS AND CONCLUSION

The presented results show that the concept of parallel jumping positively affects the model of jumping finite automata. The most significant result is that every additional head increases the power of these automata, which creates an infinite hierarchy of language families. Furthermore, due to the very simple conversions and the similar concepts, n-parallel jumping finite automata using only right n-jumps can be seen as a direct counterpart to n-parallel right linear grammars.

ACKNOWLEDGMENT

This work was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070), the TAČR grant TE01020415, and the BUT grant FIT-S-14-2299.

REFERENCES

[1] A. Meduna and P. Zemek, "Jumping finite automata," *International Journal of Foundations of Computer Science*, vol. 23, no. 7, pp. 1555–1578, 2012.

[2] O. H. Ibarra, "Simple matrix languages," *Information and control*, vol. 17, pp. 359–394, 1970.

[3] R. D. Rosebrugh and D. Wood, "Restricted parallelism and right linear grammars," *Utilitas Mathematica*, vol. 7, pp. 151–186, 1975.

[4] D. Wood, "n-linear simple matrix languages and n-parallel linear languages," *Rev. Roum. de Math. Pures et Appl.*, pp. 408–412, 1977.

[5] ——, "Properties of n-parallel finite state languages," *Utilitas Mathematica*, vol. 4, pp. 103–113, 1973.

[6] R. D. Rosebrugh and D. Wood, "A characterization theorem for n-parallel right linear languages," *Journal of Computer and System Sciences*, vol. 7, pp. 579–582, 1973.

[7] ——, "Image theorem for simple matrix languages and n-parallel languages," *Mathematical Systems Theory*, vol. 8, no. 2, 1974.

[8] A. Meduna and K. Zbyněk, "Jumping grammars," *International Journal of Foundations of Computer Science*, vol. 26, no. 6, pp. 709–731, 2015.

[9] A. Meduna, *Automata and Languages: Theory and Applications*. London: Springer, 2000.

[10] D. Wood, *Theory of Computation: A Primer*. Boston: Addison-Wesley, 1987.