JAPAN ANNUAL REVIEWS IN ELECTRONICS,
COMPUTERS & TELECOMMUNICATIONS
Vol. 7

COMPUTER SCIENCE & TECHNOLOGIES

Editor T. KITAGAWA

OHM • North-Holland

JAPAN ANNUAL REVIEWS IN ELECTRONICS, COMPUTERS & TELECOMMUNICATIONS

Vol. 7

COMPUTER SCIENCE & TECHNOLOGIES

1983

Editor T. KITAGAWA

OHM
Tokyo · Osaka · Kyoto

NORTH-HOLLAND
Amsterdam · New York · Oxford

OHMSHA, LTD. and N·H P·C NORTH-HOLLAND PUBLISHING COMPANY

ISSN 0167-5036

Co-published by
OHMSHA, LTD.
1-3 Kanda Nishiki-cho, Chiyoda-ku, Tokyo 101, Japan
Sole distributors for Japan
and
NORTH-HOLLAND PUBLISHING COMPANY
Molenwerf 1, 1014 AG, Amsterdam, The Netherlands
Sole distributors for outside Japan

Printed in Japan

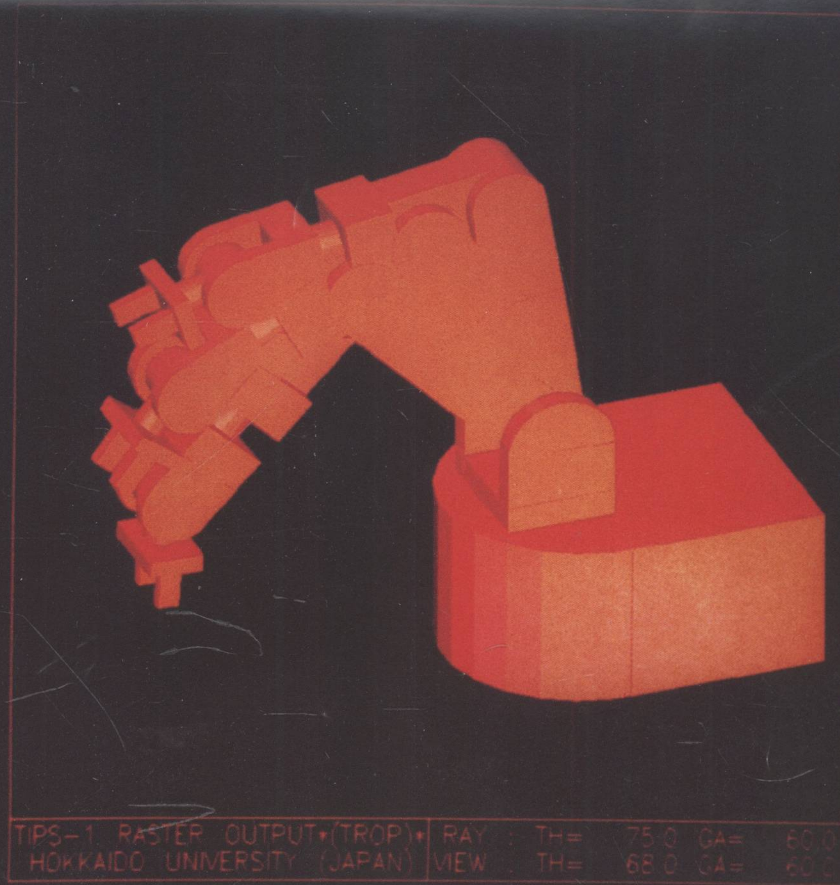
JAPAN ANNUAL REVIEWS IN ELECTRONICS, COMPUTERS & TELECOMMUNICATIONS

Vol. 7

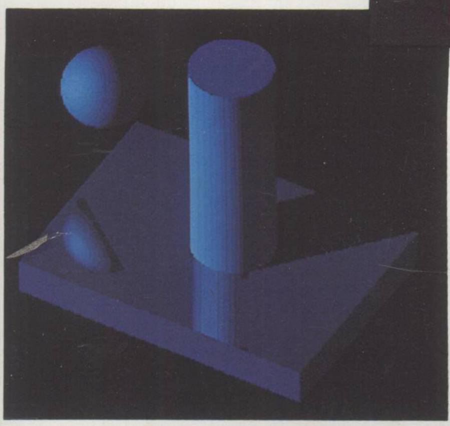
COMPUTER SCIENCE & TECHNOLOGIES

Examples of Graphic Output by Application Programs of 3-D CAD/CAM System; TIPS-1.

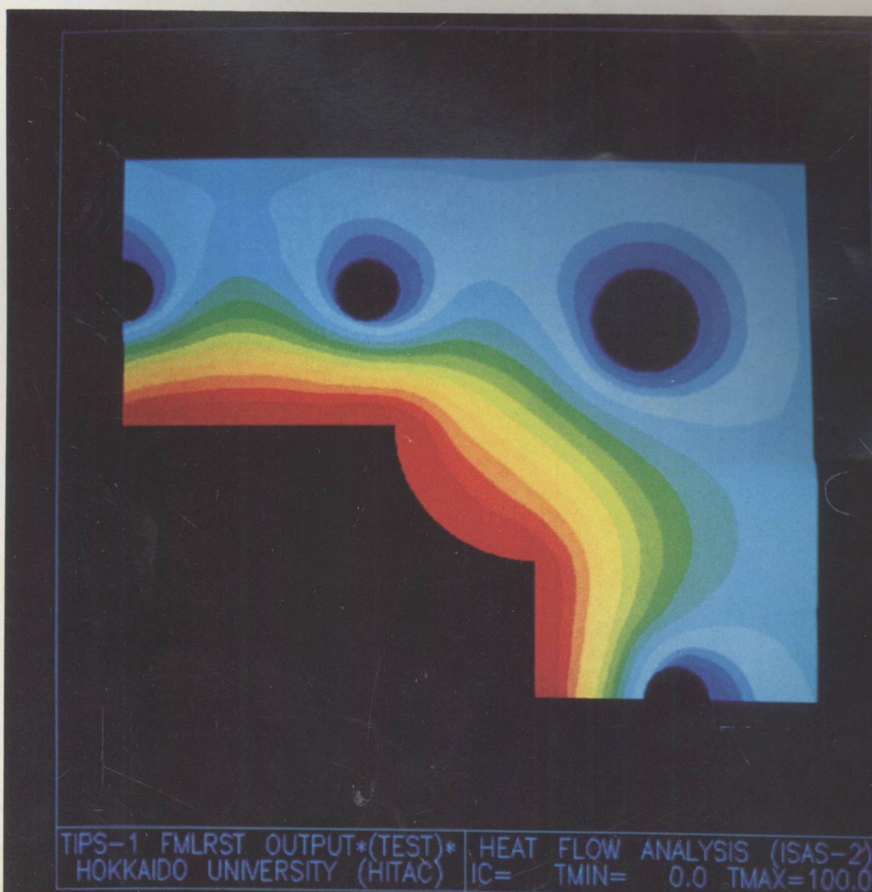
(See Section 3.2)



Overlap-drawing of ▶ Robot-arm Motions by Robot Simulator.



▲Shading and Reflection of 3-D Solid Models by Probe Detector.



Temperature Distribution ▶ of Injection Molding by Boundary Element Methods.

▲ Portable Japanese Word Processor (Toshiba JW-1) (See Section 2.3)

▼ Japanese Voice Input Word Processor (NEC VWP-100 Series) (See Section 2.7)

Preface

This is the 1983 edition of Computer Science & Technologies of the Japan Annual Reviews in Electronics, Computers & Telecommunications. It aims to provide international readers with up-to-date information and help them understand the most recent research trends and representative achievements in Japan in the field of computer science and technology.

In view of the rapid developments in this field and the urgent demand for more detailed information about Japanese activities and achievements, the Editorial Board has decided to concentrate only on a few selected topics in each volume. The topics will alternate each year so as to obtain adequate coverage of the whole range of activities within each period of a few years.

The present edition (1983 JARECT Vol. 7) chooses four topics: (I) computing theory, (II) Japanese language processing, (III) CAD system, and (IV) database and information systems. The 1984 edition is scheduled to focus on two topics: software engineering—overview, tools and methods, and software engineering—systems; and intelligent robots and their applications.

It should be noted that we are always ready to learn from international reviews and individual comments regarding any improvements we could take to obtain a more adequate editing procedure. In this way, we hope to achieve the aim of fulfilling our international duties by promoting intimate cooperation among those working in this increasing important field.

Tosio KITAGAWA

Musashino, Tokyo, Japan
April 1983

Contents

CHAPTER 1

COMPUTING THEORY

JARECT Vol.7, Computer Science & Technologies (1983), T. Kitagawa (ed.)
OHMSHA, LTD. and North-Holland Publishing Co.

1.1 Historical Review of Mathematical Studies in Japanese Computer Science

Satoru TAKASU*

As an introductory section for this chapter, we give a brief survey on the outset of our mathematical studies in Japanese computer science.

1.1.1 Language and Automata

The eariest work in Japan, if we include the switching theory, was the study of sequential networks started by Goto in 1948.[2)] His group had made a concerted effect until the completion of the design for the automatic relay computer ETL MARK II in 1956[4)] to which many theoretic results were applied.

In 1958, finite automata on events and reduction theorems were first studied by Huzino.[7,8)] Together with Yamada, McNaughton studied the relationship of regular expressions and state graphs in 1960.[16)]

In 1959, the International Conference on Information Processing, organized by UNESCO, was held in Paris. This conference was later referred to as the first IFIP Congress. At this conference, (1) Takahashi and Goto presented a paper, Application of Error Correcting Codes for Multi-way Switching,[24)] (2) Muroga talked on The Principle of Majority Decision Elements and the Complexity of their Circuits,[17)] which was the start of threshold logic study flourished in the 60's, and (3) Goto coordinated a symposium on switching algebra in which the above mentioned results by Goto were discussed.

In 1962, Yamada published the paper "Real-time Computations and Recursive Functions Not Real-Time Computable", which is the origin of automata theoretic studies on the complexities of algorithms.[28)] In 1963, Nagao wrote the paper "Syntax Analysis for Phrase Structure Grammar"[18)] in which he reported on the grammar and parsing algorithm similar to the procedence grammar.

Other works will be reviewed in the language and automata section.

1.1.2 Semantics of Programming Languages

Program schemata

Igarashi proposed in 1962 an improvement of Yanov's schemata, i.e., the so-called logical schemata of algorithms and he studied the equivalence problem.[9)] In 1972, Hirose and Oya developed a general theory on flowcharts, in which an algorithm to transform while-

* Research Institute for Mathematical Sciences, Kyoto University, Kyoto 606.

programs into single loop while-programs was obtained.[6] Recently, this has been reviewed by Harrel.[5] Kasai obtained a necessary and sufficient condition for the translatability of flowcharts into while-programs in 1974.[15] Tokura and Kasami studied the equivalence problems for program schemata with various data structures in 1974.[26,27]

Semantics of programming languages

Igarashi, in his dissertation (1964), designed a formal system in which the equivalence of programs can be proved. His dissertation was published in Refs. 10) and 11). His results were surveyed by de Bakker.[1] The paper[10] was first to introduce the meta-mathematical approach to the study of programming. Igarashi[13] contributed to form the verification system concept together with London and Luckham in 1975. He also studied the admissibility of fixed point induction in first-order logic of typed theories.[12]

Semantics of concurrent programs

Nishimura[20] addressed a formalization of concurrent programs at IFIP 77. It is too early to review other works, since the topic is still being studied and there is not a conclusion as yet.

1.1.3 Theorem Proving

In 1959, a large scale cooperative research plan was initiated by the Japanese Government, in which a research group for programming was formed for the first time in Japan. Within this research group, the late Professor Shigekatsu Kuroda advocated, in June 1959, to make a "theorem proving program" using Gentzen's sequent calculus. Then there was an attempt to construct two theorem proving programs. One was by Takasu at Electrical Communication Laboratory and the other by logicians Simauti, Nishimura, Iwamura, Maehara, Seki, Fujikawa, Hiramoto, and Kondo. The results were reported at the first Programming Symposium at Oiso, January 1960.[22,25] One credit goes to Simauti, who found a method to reduce the number of cases for variable substitutions in predicate calculus without function symbols. Since then several theorem proving systems have been developed.[14,19,21,29]

References

1) de Bakker, J.W.: Semantics of programming languages, in "Advances in Information Science", Vol. 2, ed. by J.T. Tou (Plenum Press, New York, 1969).
2) Goto, M.: The solutions of logical equations and their applications, The Collection of Memorial Treatises from Electrotechnical Laboratory (Tokyo, 1948).
3) Goto, M. and Komamiya, Y.: Theory of synthesis of sequential networks by means of the solution of logical functional equations and logical algebraic equations, in "Information Processing 1959" (UNESCO, Paris, 1959) 426.
4) Goto, M. et al.: Theory and Structure of the Automatic Relay Computer E.T.L. II (Electrotechnical Laboratory, Tokyo, 1956).
5) Harrel, D.: Commun. ACM, *23* (1980) 379.
6) Hirose, K. and Oya, M.: Proc. of the 1st USA-Japan Comput. Conf. (Tokyo, 1972) 367.
7) Huzino, S.: Mem. Fac. Sci. Kyushu Univ.

Ser., *A12* (1958) 159.

8) Huzino, S.: Mem. Fac. Sci. Kyushu Univ. Ser., *12* (1958) 159.

9) Igarashi, S.: Information Processing in Japan, *3* (1963) 12; originally published in "Johoshori", *3* (1962) 66 [in Japanese].

10) Igarashi, S.: Ph. D. Thesis, Univ. of Tokyo (1964); reprinted in "Report of the Computer Center", Univ. of Tokyo, *1* (1968) 1.

11) Igarashi, S.: Report of the Computer Center, Univ. of Tokyo, *1* (1968) 103.

12) Igarashi, S.: Admissibility of fixed point induction in first-order logic of typed theories, Memo AIM-168 (Stanford Univ., 1972).

13) Igarashi, S. et al.: Acta Informatica, *4* (1975) 145.

14) Iwamura, Y. et al.: Comment. Math. Univ. St. Pauli, XXIII-1 (1974) 45.

15) Kasai, T.: JCSS, *9* (1974) 177.

16) McNaughton, R. and Yamada, H.: IEEE Trans. Electron. Comput., *EC-9* (1960) 39.

17) Muroga, S.: The principle of majority decision logical elements and the complexity of their circuits, in "Information Processing 1959" (UNECSO, Paris, 1959) 400.

18) Nagao, M.: Johoshori, *4* (1963) 186 [in Japanese].

19) Nakanishi, M. et al.: Sci. Report of the Tokyo Kyoiku Daigaku, *13* (1975) 82.

20) Nishimura, T.: IFIP 77 (North-Holland, 1977) 929.

21) Nishimura, T.: Comment. Math. Univ. St. Pauli, XXIII-1 (1974) 37.

22) Simauti, T.: Theorem proving in LK, Comment. Math. Univ. St. Pauli, XXIII-1, E.5 (1974) [in Japanese].

23) Simauti, T.: Sugaku, *15* (1964) 48 [in Japanese].

24) Takahashi, H. and Goto, E.: Application of error correcting codes to multi-way switching, in "Information Processing 1959" (UNESCO Paris, 1959) 396.

25) Takasu, S.: Proc. 1st Programming Sympo., *E3* (Oiso, 1960) [in Japanese].

26) Tokura, N. and Kasami, T.: Syst. Comp. Contro., *5* (1974) 81.

27) Tokura, N. et al.: "Ianov schemas augmented by a pushdown memory", Proc. 15th Ann. Symp. on Switching and Automata Theory (1974).

28) Yamada, H.: IEEE Trans. Electron. Comput., *EC-11* (1962) 753.

29) Yonezawa, A.: Johoshori, *14* (1973) 106 [in Japanese].

JARECT Vol.7, Computer Science & Technologies (1983), T. Kitagawa (ed.)
OHMSHA, LTD. and North-Holland Publishing Co.

1.2 Complexity

Akihiro NOZAKI*[1], Akeo ADACHI*[2], and Takumi KASAI*[3]

Abstract

This section summarizes the works of three authors in the field of computational complexity. The first half, 1.2.1 and the first three subsections of 1.2.2 concern the joint work of Akeo Adachi and Takumi Kasai. The remaining part, the fourth and the fifth subsections of 1.2.2 and 1.2.3, deals with Akihiro Nozaki's work.

Many contributions from other Japanese researchers, e.g., important ones on Network Algorithms, are not included here; however, some of them will be reviewed in Section 1.3.

1.2.1 Complexity Based on Turing Machines

A problem is said to be computable iff there exist algorithms for the problem. There have been found many problems not to be computable in principle. Even if a problem is computable in principle, however, it is not always computable in considerable time or space. In trying to solve problems by computer, it is necessary to solve them in a limited amount of time or space. The complexity theory is the theory to consider computability in a practical sense. Many problems have been shown to be intractable in the above sense.

In particular, a great number of problems have been shown to be NP-complete. There is general agreement that if a problem is shown to be NP-complete (or PSPACE-complete), then the problem is intractable. By diagonalization argument, we know that some problems in class P (the class of problems which can be solved in polynomial time) have no efficient solution. However compelling the circumstantial evidence may be, no one has yet been able to find a problem in P, NP or PSPACE that can be shown to actually be intractable. From this point of view, it is relevant to study class P more precisely. That is, low level lower bound studies are required to know if a problem is computable in a practical sense, which have scarcely been discussed. Here we consider some combinatorial game problems and show that these "concrete" problems have nontrivial lower bounds.

We introduce here pebble games.[39] A pebble game involves moving pebbles according to certain rules. The goal of the game is to put a pebble on a particular place. It has been shown that when the game is played by two persons the problem to determine whether there is a winning strategy is complete in exponential time, (hence, this problem cannot be solved in polynomial time,) and when played by one person, the problem is to determine whether one can put a pebble on a particular place is PSPACE-complete. We can get various classes of

*1 Division of Natural Sciences, College of Liberal Arts, International Christian University, Tokyo 181.
*2 Science Institute, IBM Japan, Ltd., Roppongi, Minato-ku, Tokyo 106.
*3 Department of Computer Science, The University of Electro-Communications, Tokyo 182.

complete problems by restricting the pebble games. In particular, we consider the case that the number of pebbles is fixed to k, which is called k-pebble games. A k-pebble game problem is a problem to determine whether the first player has a forced win in the game when the game is played by two persons with only k pebbles. It has been shown that a $(2k+1)$-pebble game problem requires $\Omega(n^k)$ time.

Pebble games are applied to show that certain problems cannot be solved in polynomial time and also can be applied to analyze the complexity of synchronization problems in parallel computation.

Turing machines and complexity classes

Most of the computational problem can be formulated as language recognition problem. Let Σ and Δ be two alphabets, Σ^* and Δ^* the sets of words over Σ and Δ, respectively. A language (problem) is a subset of Σ^* for some Σ. Here we use Turing machines as a model of computations. By a Turing machine, we mean a machine with a finite state control, a two-way read-write work tape and a one-way write-only output tape.

Let $T:N \to N$ and $S:N \to N$ be functions, where N denotes the set of non-negative integers. A Turing machine M is said to be $T(n)$-time-bounded ($S(n)$-space-bounded) iff for any input x, the computation time (space) does not exceed $T(|x|)(S(|x|))$. Let $A \subseteq \Sigma^*$ be a problem. Then A is said to be $T(n)$-time-computable ($S(n)$-space-computable) iff there exists a $T(n)$-time-bounded ($S(n)$-space-bounded) Turing machine which accepts A. A total function $f: \Sigma^* \to \Delta^*$ is said to be $T(n)$-time-computable ($S(n)$-space-computable) iff there exists a $T(n)$-time-bounded ($S(n)$-space-bounded) Turing machine M such that for any input x, M halts with $f(x)$ on its output tape. Let $A \subseteq \Sigma^*$ and $B \subseteq \Delta^*$ be problems. If $f: \Sigma^* \to \Delta^*$ is a function such that $x \in A$ iff $f(x) \in B$ for all $x \in \Sigma^*$, then A is reducible to B under f, and we write $A <_f B$. Let $\mathscr{C}$ be a class of problems and F be a class of functions. Then a problem B is $\mathscr{C}$-hard under f iff for any $A \in \mathscr{C}$, then there exists $f \in \mathscr{F}$ such that $A <_f B$. B is $\mathscr{C}$-complete under $\mathscr{F}$ iff $B \in \mathscr{C}$ and B is $\mathscr{C}$-hard under $\mathscr{F}$. We simply say that a problem B is $\mathscr{C}$-complete if B is $\mathscr{C}$-complete under the class of functions computable in log-space. Let $T:N \to N$ and $S:N \to N$ be functions, and n be the size of problems (input to Turing machines). DTIME $(T(n))$ is defined to be the class of problems computable within $T(n)$ time. NTIME $(T(n))$ is defined analogously for nondeterministic Turing machines. Similarly, DSPACE $(S(n))$ and NSPACE $(S(n))$ are defined to be the class of problems computable in $S(n)$ space by deterministic and nondeterministic Turing machines, respectively. Now, let

$$\text{NLOGSPACE} = \bigcup_{i \geq 0} \text{NSPACE}\ (i \cdot \log n)$$

$$\text{P} = \bigcup_{i \geq 0} \text{DTIME}(n^i)$$

$$\text{NP} = \bigcup_{i \geq 0} \text{NTIME}(n^i)$$

$$\text{PSPACE} = \bigcup_{i \geq 0} \text{DSPACE}(n^i) = \bigcup_{i \geq 0} \text{NSPACE}(n^i)$$

$$\text{EXP} = \bigcup_{i \geq 0} \text{DTIME}(2^{ni})$$

Pebble games

If a problem B is $\mathscr{C}$-complete, then intuitively the problem B is thought to be most difficult one in the class, that is, B is a representative of the problems in class $\mathscr{C}$ in difficulty. Thus, we can know the characteristics of class $\mathscr{C}$ by studying problem B. We show a couple of complete problems obtained by adding some restictions to pebble games.

Definition 1.2.1 A pebble game is a quadruple $G=(X,R,S,t)$ where:

(1) X is a finite set of nodes; the number of nodes is called the order of G.

(2) $R\subseteq\{(x, y, z)\,|\,x,y,z \in X,\ x\neq y,\ y\neq z,\ z\neq x\}$ is called a set of rules. For A, $B\subseteq X$, we write $A\vdash B$ if $(x,y,z)\in R$, $x,y\in A$, $z\notin A$, and $B=(A-\{x\})\cup\{z\}$. We say the move $A\vdash B$ is made by the rule (x,y,z). The symbol $\overset{*}{\vdash}$ denotes the reflexive and transitive closure of $\vdash$.

(3) S is a subset of X; the number of nodes in S is called the rank of G.

(4) t is a node in X, called the terminal node.

A pebble game is said to be solvable if there exists $A\subseteq X$ such that $S\overset{*}{\vdash}A$ and $t\in A$.

At the beginning of a pebble game, pebbles are placed on all nodes of S. If $(x,y,z)\in R$ and pebbles are placed on x, y but not on z, then we can move a pebble from x to z. The game is solvable if we can place a pebble on the terminal node t by moving pebbles according to rules.

A pebble game played by two persons is a game between two players, P_1 and P_2, who alternatively move pebbles on the pebble game, with P_1 playing first. The winner is the first player who can put a pebble on the terminal node, or who can make the other player unable to move.

By the term "one-person pebble game problem," we mean the problem to determine for a given pebble game G, whether G is solvable. By "two-person pebble game problem," we mean the problem when a pebble game is played by two persons to determine whether the first player has a winning strategy, i.e., a way to win the game.

Definition 1.2.2 A pebble game $G=(X,R,S,t)$ is acyclic if the digraph (X,E) is acyclic, where

$$E=\{(x,z),(y,z)\,|\,(x,y,z)\in R\}.$$

Definition 1.2.3 Let $G=(X,R,S,t)$ be a pebble game. G is called a pebble game of fixed rank if the number of nodes in S is fixed.

Theorem 1.2.1

	Solvability problem (played by one person)	Winning strategy problem (played by two persons)
Pebble game of fixed rank	NLOGSPACE-complete	P-complete
Acyclic pebble game	NP-complete	PSPACE-complete
Pebble game	PSPACE-complete	EXP-complete

EXP-complete problems

Since the early 1970s, there have been trials to find a polynomial algorithm to solve a