Minimization of Memory Size for Heterogeneous MDDs

Shinobu NAGAYAMA [1]   Tsutomu SASAO [1,2]

[1] Department of Computer Science and Electronics, Kyushu Institute of Technology
[2] Center for Microelectronics Systems, Kyushu Institute of Technology

Abstract— In this paper, we propose exact and heuristic algorithms for minimizing the memory size for heterogeneous Multi-valued Decision Diagrams (MDDs). In a heterogeneous MDD, each multi-valued variable can take a different domain. To represent a binary logic function using a heterogeneous MDD, we partition the binary variables into groups, and treat the groups as multi-valued variables. Therefore, the memory size of a heterogeneous MDD depends on the partition of the binary variables. Our experimental results show that heterogeneous MDDs require smaller memory size than Reduced Ordered Binary Decision Diagrams (ROBDDs) and Free BDDs (FBDDs).

I. Introduction

Reduced Ordered Binary Decision Diagrams (ROBDDs) [2], Free BDDs (FBDDs) [5, 6, 19], and Multi-valued Decision Diagrams (MDDs) [9] are extensively used in logic synthesis [4], logic simulation [10], software synthesis [8], etc. Since the memory sizes for these applications depend on the size of the Decision Diagrams (DDs), minimization for DDs are very important. Particularly, in software synthesis, intensive minimization of DDs is required to generate a compact program code. Most minimization algorithms for DDs use variable reordering approaches [4, 5, 6, 7, 15, 19]. However, when MDDs are used to represent binary logic functions, we can use an additional minimization approach, which is a partition of binary variables [14]. To represent a binary logic function using an MDD, we partition the binary variables into groups, and treat each group as a multi-valued variable. In many cases, the groups have the same number of binary variables. In a heterogeneous MDD [14], the groups can have different numbers of binary variables. Even if the ordering of the input variables is fixed, heterogeneous MDDs can represent logic functions with smaller memory size than the ROBDDs and comparable memory size to the FBDDs by considering only the partition of input variables [14].

In this paper, we propose exact and heuristic algorithms for minimizing memory size that consider both partitions and orderings of binary variables. By experiment, we show that the memory sizes for heterogeneous MDDs are smaller than ROBDDs and FBDDs.

II. Definitions

A. Partitions of Binary Variables

Definition 2.1 *Let $f(X)$ be a two-valued logic function, where $X = (x_1, x_2, \ldots, x_n)$, and x_i $(i = 1, 2, \ldots, n)$ are binary variables. Let $\{X\}$ denote the set of variables in X. If $\{X\} = \{X_1\} \cup \{X_2\} \cup \ldots \cup \{X_u\}$ and $\{X_i\} \cap \{X_j\} = \phi (i \neq j)$, then $(X_1, X_2, \ldots, X_u)$ is a* **partition** *of X. X_i is treated as a* **multi-valued variable**. *If $|X_i| = k_i$ $(i = 1, 2, \ldots, u)$ and $k_1 + k_2 + \ldots + k_u = n$, then a two-valued logic function $f(X)$ can be represented by the mapping $f(X_1, X_2, \ldots, X_u)$: $P_1 \times P_2 \times P_3 \times \ldots \times P_u \to B$, where $P_i = \{0, 1, 2, \ldots, 2^{k_i} - 1\}$ and $B = \{0, 1\}$.*

Definition 2.2 *A* **fixed-order partition** *of $X = (x_1, x_2, \ldots, x_n)$ is a partition into $(X_1, X_2, \ldots, X_u)$, where $X_1 = (x_1, x_2, \ldots, x_{k_1})$, $X_2 = (x_{k_1+1}, x_{k_1+2}, \ldots, x_{k_1+k_2})$, $\cdots$, $X_u = (x_{k_1+k_2+\ldots+k_{u-1}+1}, x_{k_1+k_2+\ldots+k_{u-1}+2}, \ldots, x_{n-1}, x_n)$, and $|X_i| = k_i$. That is, in the fixed-order partition of X, the variable order of X is fixed.*

Definition 2.3 *When the variable order of $X = (x_1, x_2, \ldots, x_n)$ is not fixed, a partition of X is a* **non-fixed-order partition** *of X.*

We assume that the given logic function is completely specified and excludes redundant variables.

B. Heterogeneous MDD

In this paper, we use the standard terminologies for BDDs, Reduced Ordered BDDs (ROBDDs or OBDDs) [2], Free BDDs (FBDDs) [5, 6, 19], MDDs, and Reduced Ordered MDDs (ROMDDs) [9].

Definition 2.4 *When $X = (x_1, x_2, \ldots, x_n)$ is partitioned into $(X_1, X_2, \ldots, X_u)$, an ROMDD representing a logic function $f(X)$ is a* **heterogeneous MDD**. *A heterogeneous MDD represents a mapping $f : P_1 \times P_2 \times \ldots \times P_u \to B$, where $P_i = \{0, 1, \ldots, 2^{k_i} - 1\}$ and $B = \{0, 1\}$. In a heterogeneous MDD, non-terminal nodes representing X_i have 2^{k_i} outgoing edges, where k_i denotes the number of binary variables in X_i.*

Definition 2.5 *In a Decision Diagram (DD), the* **number of nodes in the DD**, *denoted by $nodes(DD)$, includes only non-terminal nodes.*

Definition 2.6 *The* **width of the MDD with respect to** *X_i, denoted by $width(MDD, i)$, is the number of nodes in the MDD corresponding to the variable X_i. The number of nodes in the MDD for a function $f(X_1, X_2, \ldots, X_u)$ is given by*

$$nodes(MDD) = \sum_{i=1}^{u} width(MDD, i).$$

Example 2.1 *Consider the function $f = x_1x_2x_3 \vee x_2x_3x_4 \vee x_3x_4x_1 \vee x_4x_1x_2$. Fig. 1 represents two heterogeneous MDDs for f. In Fig. 1(a), the binary variables are partitioned into (X_1, X_2), where $X_1 = (x_1, x_2, x_3)$ and $X_2 = (x_4)$. In Fig. 1(b), $X_1 = (x_1)$ and $X_2 = (x_2, x_3, x_4)$. (End of Example)*

In this paper, we use Shared BDDs (SBDDs) [11] and Shared MDDs (SMDDs) to represent multiple-output functions $F = (f_0, f_1, \ldots, f_{m-1}) : B^n \to B^m$, where $B = \{1, 0\}$, and n and m denote the number of inputs and outputs, respectively. In the following, a BDD and an MDD mean an SBDD and an SMDD, respectively, unless stated otherwise.

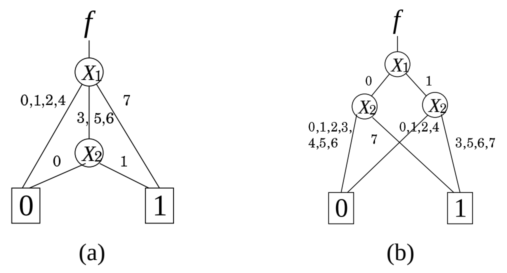

Fig. 1. Heterogeneous MDDs

TABLE I
THE NUMBERS OF DIFFERENT DDs FOR n-VARIABLE FUNCTION

n	OBDD: $n!$	MDD: $N_{non\text{-}fix}(n)$	FBDD: S_n
1	1	1	1
2	2	3	2
3	6	13	12
4	24	75	576
5	120	541	1658880
6	720	4683	16511297126400

III. MINIMIZATION OF HETEROGENEOUS MDDS

A. Number of Heterogeneous MDDs

Theorem 3.1 [1] *Let $N_{non\text{-}fix}(n)$ be the number of different non-fixed-order partitions of $X = (x_1, x_2, \ldots, x_n)$. Then,*

$$N_{non\text{-}fix}(n) = \sum_{r=1}^{n} \sum_{i=0}^{r} {}_rC_i (r-i)^n (-1)^i.$$

Therefore, the number of different heterogeneous MDDs for an n-variable logic function is given by $N_{non\text{-}fix}(n)$.

Table I compares the numbers of OBDDs, FBDDs, and heterogeneous MDDs for n-variable logic functions, where the number of FBDDs S_n is given by [19]

$$S_n = nS_{n-1}^2 = \prod_{k=1}^{n} k^{2^{n-k}}.$$

The number of heterogeneous MDDs increases with n more slowly than the number of FBDDs. Thus, optimum heterogeneous MDDs are easier to find than optimum FBDDs for $n \geq 4$. However, when n is large, finding optimum heterogeneous MDDs within a reasonable time is difficult.

B. Memory Sizes for Heterogeneous MDDs

Definition 3.7 *The* **memory size of a DD** *is the number of words needed to represent the DD in memory, where we assume that a word is large enough to store a single data.*

In memory, a non-terminal node requires an index and a set of pointers that refer to the succeeding nodes. Since each non-terminal node in a BDD has two pointers, the memory size needed to represent a BDD is given by

$$(2+1) \times nodes(\text{BDD}). \qquad (1)$$

In a heterogeneous MDD, each non-terminal node has 2^{k_i} pointers, where k_i is the number of binary variables in X_i. Therefore, the memory size for heterogeneous MDD is calculated by

$$\sum_{i=1}^{u} (2^{k_i}+1) \times width(\text{heterogeneous MDD}, i).$$

[1]The proof is available in http://www.lsi-cad.com/Hetero-MDD/.

Algorithm 1

```
1:  exhaustive_search (BDD)
2:     min_memory = **minimize_memory** () ;
3:     for (all permutations of binary variables)
4:        Change the variable order for BDD ;
5:        if ($min\_memory < nodes(\text{BDD}) + 2$) continue ;
6:        current_memory = **minimize_memory** () ;
7:        if ($current\_memory < min\_memory$)
8:           min_memory = current_memory ;
9:           Record the variable order for the BDD ;
10:          Record the partition of binary variables ;
```

Fig. 2. Exact memory minimization algorithm.

Example 3.2 *The memory sizes of heterogeneous MDDs in Fig. 1 (a, b) are* 12 *and* 21*, respectively. (End of Example)*

Definition 3.8 *Given a logic function f and an order of its input variables, the* **fixed-order minimum heterogeneous MDD** *for f is the heterogeneous MDD with the minimum memory size among the fixed-order partitions of the variables.*

Definition 3.9 *Given a logic function f, the* **minimum heterogeneous MDD** *for f is the heterogeneous MDD with the minimum memory size among all possible non-fixed-order partitions of the variables.*

Property 3.1 *Consider a logic function $f(X)$. Let $Mem_{min}(f)$ be the memory size needed to represent a fixed-order minimum heterogeneous MDD for f. When f is decomposed into $f = g(h(X_1), X_2)$, let $Mem_{min}(g)$ and $Mem_{min}(h)$ be the memory sizes needed to represent fixed-order minimum heterogeneous MDDs for g and h, respectively. For many benchmark functions, the following relations hold:*

$$Mem_{min}(f) > Mem_{min}(g), \quad Mem_{min}(f) > Mem_{min}(h).$$

C. Minimization Algorithms

Since the memory size of a heterogeneous MDD depends on both the partitioning and the ordering of the binary variables X, we formulate the memory minimization problem for heterogeneous MDD as follows:

Problem 3.1 *Given a logic function $f(X)$, find a non-fixed-order partition of X that produces the minimum heterogeneous MDD.*

Example 3.3 *Fig. 1(a) shows the minimum heterogeneous MDD for the function f, while Fig. 1(b) shows the maximum heterogeneous MDD for f. (End of Example)*

Fig. 2 shows pseudo-code to solve Problem 3.1. In the 2nd and 6th lines in Fig. 2, **minimize_memory** [14] finds an optimum fixed-order partition. In the 5th line, a theorem in [13] is used to reduce the computation time. Algorithm 1 finds the minimum heterogeneous MDD by exhaustive search.

However, as described in Section III-A, when the number of binary variables is large, finding a minimum heterogeneous MDD within a reasonable time is difficult. Thus, we developed a heuristic minimization method for heterogeneous MDDs using the sifting algorithm [15] and the fixed-order partition algorithm [14]. The sifting algorithm consists of two steps:

1. Change the variable order.
2. Compute a cost.

Algorithm 2

```
sifting_memory (BDD)
    $cost$ = minimize_memory () ;
    do
        for ($\forall x_i \in X$)
            $best\_p$ = current position of $x_i$ ;
            for (all position $p$)
                Move $x_i$ to position $p$ ;
                $memory$ = minimize_memory () ;
                Compute $L_{mem}$ ;
                if ($cost \leq L_{mem}$) break ;
                if ($memory < cost$)
                    $cost = memory$ ;
                    $best\_p = p$ ;
                    Record the partition of binary variables ;
            Move $x_i$ to $best\_p$ ;
    while ($cost$ is reduced)
```

Fig. 3. Heuristic memory minimization algorithm.

Most sifting algorithms use the number of nodes in the BDD as the cost. In this paper, however, we use the memory size of the heterogeneous MDD as the cost. Fig. 3 shows pseudo-code for the heuristic minimization algorithm. The L_{mem} in the 9th line denotes the memory size of fixed-order minimum heterogeneous MDD for subfunction g or h obtained by functional decomposition $f(X) = g(h(X_1), X_2)$. When x_i moves down to the bottom of the BDD, we use h to compute L_{mem}, where X_1 contains the binary variables which are above the level of x_i in the variable order, and X_2 contains the remaining ones. If $cost \leq L_{mem}$, we stop the sifting of x_i to the bottom because sifting of x_i further down to the bottom seldom reduces the memory size due to Property 3.1. Similarly, when x_i moves up to the top of the BDD, we use g to compute L_{mem}, where X_2 contains the binary variables which are below the level of x_i in the variable order, and X_1 contains the remaining ones. This lower bound for the memory size is similar to the one introduced for the number of nodes during the classical sifting [3].

IV. Experimental Results

We used the following environment: CPU: Pentium4 Xeon 2.8GHz, L1 Cache: 32KB, L2 Cache: 512KB, Memory: 4GB, OS: redhat (Linux 7.3), and C-compiler: gcc -O2.

A. All Functions Up To Five Variables

We implemented Algorithm 1 and compared the minimum heterogeneous MDDs with the minimum OBDDs and the minimum FBDDs for all 4 and 5 variable logic functions. To compare them, we classified all the logic functions into NPN-equivalence classes [12, 17]. For the 4-variable case, $65,536$ functions are classified into 222 NPN-equivalence classes, and for the 5-variable case, $4,294,967,296$ functions are classified into $616,126$ NPN-equivalence classes. Table II compares minimum DD sizes for the 4-variable case. (The table for the 5-variable case is omitted due to the page limitation.) In Table II, 222 NPN-representative functions are grouped into 9 rows according to the memory size for the minimum OBDD. The column "Mem" denotes the memory size (words) for each DD. The columns "#class" and "#function" in Table II denote the number of NPN-equivalence classes and the number of functions included in the classes, respectively. The bottom row "Avg." denotes the arithmetic average of the relative memory sizes for all functions, where the memory size needed for OBDD is set to 1.00. In this experiment, no complemented edges are used in OBDDs, FBDDs, or heterogeneous MDDs.

For the 4-variable case, FBDDs are smaller than OBDDs for $5,568$ functions, 8.5% of all functions, while heterogeneous MDDs are smaller than OBDDs and FBDDs for all functions except for 10 degenerate functions (0, 1, x_i, and $\bar{x_i}$ where $i = 1, 2, 3, 4$). For these 10 functions, the memory sizes of OBDDs, FBDDs, and heterogeneous MDDs are equal. On average over all functions, minimum FBDDs require 99% of the memory size of minimum OBDDs, while minimum heterogeneous MDDs require 72% of the memory size for minimum OBDDs.

For the 5-variable case, FBDDs are smaller than OBDDs for $1,938,548,576$ functions, 45% of all functions, while heterogeneous MDDs are smaller than OBDDs for $4,294,967,284$ functions, 99% of all functions. Also, heterogeneous MDDs are smaller than FBDDs for $4,294,921,204$ functions, 99% of all functions, and for the others, heterogeneous MDDs are equal in size to FBDDs. There was no function whose FBDD is smaller than the heterogeneous MDD. On average over all functions, minimum FBDDs require 96% of the memory size for minimum OBDDs, while minimum heterogeneous MDDs require 67% of the memory size for minimum OBDDs.

Algorithm 1 could obtain exact minimum heterogeneous MDDs for the functions with up to 12 inputs within a reasonable computation time, while the exact FBDD minimization [5] can find the minimum one for the functions with up to 8 inputs.

B. MCNC Benchmark Functions

Table III compares heterogeneous MDDs with OBDDs and FBDDs for selected MCNC benchmark functions. The OBDDs are obtained by the best known variable orders [18], and the numbers of nodes for FBDDs are taken from [5, 6]. The memory sizes for OBDDs and FBDDs are calculated by the formula (1) in Section III-B. The OBDDs and FBDDs may not be the exact minimum. The columns "#in" and "#out" in Table III denote the number of inputs and outputs for each benchmark function, respectively. "MDD" denotes the heterogeneous MDD obtained by Algorithm 2, where Algorithm 2 uses the OBDDs [18] as the initial one. The column "Time" denotes the CPU time for Algorithm 2, in seconds. The bottom row "Average of ratios" denotes the arithmetic average of the relative memory size, where the memory size needed for OBDD is set to 1.00. In this experiment, OBDDs, FBDDs, and heterogeneous MDDs use complemented edges. Heterogeneous MDDs require smaller memory size than FBDDs for 14 out of 21 benchmark functions in Table III. Especially, for *C499*, *dalu*, and *vda*, heterogeneous MDDs require at most 80% of the memory sizes for the FBDDs. And, the computation time for Algorithm 2 is short.

V. Conclusion

In this paper, we have proposed exact and heuristic algorithms for minimizing the memory size for heterogeneous MDD. Our experimental results show that: 1) Heterogeneous MDDs represent logic functions more compactly than ROBDDs and Free BDDs. Especially, for all 4-variable and 5-variable logic functions, the minimum heterogeneous MDDs require 72% and 67% of the memory sizes for the minimum OBDDs, on average, respectively. For MCNC benchmark functions, the heterogeneous MDDs require 87% of the memory sizes for the OBDDs, on average. 2) Algorithm 1 can find

TABLE II
MEMORY SIZES OF OBDDS, FBDDS, AND HETEROGENEOUS MDDS FOR ALL 4-VARIABLE LOGIC FUNCTIONS

Group	OBDD			FBDD			Heterogeneous MDD		
No.	Mem	#class	#function	Mem	#class	#function	Mem	#class	#function
0	0	1	2	0	1	2	0	1	2
1	3	1	8	3	1	8	3	1	8
2	6	1	48	6	1	48	5	1	48
3	9	4	364	9	4	364	5	1	12
							8	3	352
4	12	14	3168	12	14	3168	8	3	320
							9	1	96
							10	6	1216
							11	4	1536
5	15	38	12440	15	38	12440	9	3	104
							10	7	1056
							11	13	4400
							12	12	6528
							14	3	352
6	18	70	22488	18	70	22488	10	3	168
							12	41	12064
							14	13	4928
							15	13	5328
7	21	68	20346	18	3	1536	12	11	3520
				21	65	18810	15	57	16826
8	24	25	6672	21	10	4032	15	25	6672
				24	15	2640			
Avg.	1.00	–	–	0.99	–	–	0.72	–	–

TABLE III
MEMORY SIZES FOR OBDDS, FBDDS, AND HETEROGENEOUS MDDS FOR MCNC BENCHMARK FUNCTIONS

			Memory Size			Time
Function	#in	#out	OBDD	FBDD	MDD	[sec]
C432	36	7	3189	3171	2824	0.23
C499	41	32	77595	77595	59739	5.12
C880	60	26	12156	8394	11812	1.23
C1908	33	25	16575	15141	13493	0.67
C2670	233	64	5313	3186	4649	1.99
C3540	50	22	71481	62997	65029	36.96
C5315	178	123	5154	4434	4582	1.31
C7552	207	107	6477	4782	6119	4.76
alu4	14	8	1047	900	855	0.02
apex1	45	45	3735	3531	3016	0.29
apex6	135	99	1470	1365	1414	0.33
cps	24	102	2910	2706	2533	0.12
dalu	75	16	2064	1947	1548	0.25
des	256	245	8832	8706	7288	3.59
frg2	143	139	2883	2760	2671	0.89
i3	132	6	396	396	330	0.23
i8	133	81	3825	3570	3662	0.59
i10	257	224	61977	56439	55766	69.27
k2	45	45	3735	3408	3018	0.29
too_large	38	3	954	858	857	0.07
vda	17	39	1431	1401	1088	0.01
Average of ratios			1.00	0.90	0.87	–

exact minimum heterogeneous MDDs for the functions with up to 12 inputs in a reasonable computation time. 3) Algorithm 2 can reduce heterogeneous MDDs as fast as the classical sifting algorithm [15].

ACKNOWLEDGMENTS

This research is partly supported by a Grant in Aid for Scientific Research from the Japan Society for the Promotion of Science (JSPS), and funds from MEXT via the Kitakyushu innovative cluster project. Prof. D. M. Miller improved presentation.

REFERENCES

[1] F. Brglez and H. Fujiwara, "Neutral netlist of ten combinational benchmark circuits and a target translator in FORTRAN," *Special session on ATPG and fault simulation, Proc. IEEE Int. Symp. Circuits and Systems*, June 1985, pp. 663-698.

[2] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Trans. Comput.*, Vol. C-35, No. 8, pp. 677–691, Aug. 1986.

[3] R. Drechsler, W. Günther, and F. Somenzi, "Using lower bounds during dynamic BDD minimization," *IEEE Trans. CAD*, Vol. 20 (1), pp. 51–57, Jan. 2001.

[4] M. Fujita, Y. Matsunaga, and T. Kakuda, "On variable ordering of binary decision diagrams for the application of multi-level logic synthesis,“ *EDAC*, pp. 50–54, Mar. 1991.

[5] W. Günther and R. Drechsler, "Minimization of free BDDs,“ *Asia and South Pacific Design Automation Conference (ASP-DAC'99)*, pp.323-326, Jan. 18-21, 1999, Wanchai, Hong Kong.

[6] W. Günther, "Minimization of free BDDs using evolutionary techniques,“ *International Workshop on Logic Synthesis (IWLS-2000)*, pp. 167-172, May 31-June 2, 2000, Loguna Cliffs Marriott, Dana Point, CA.

[7] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables,“ *ICCAD*, pp. 472–475, Nov. 1991.

[8] Y. Jiang and B. K. Brayton, "Software synthesis from synchronous specifications using logic simulation techniques," *Design Automation Conference*, pp. 319-324, New Orleans, LA, U.S.A, June 10-14, 2002.

[9] T. Kam, T. Villa, R. K. Brayton, and A. L. Sagiovanni-Vincentelli, "Multi-valued decision diagrams: Theory and Applications," *Multiple-Valued Logic*, 1988, Vol. 4, No. 1-2, pp. 9–62, 1998.

[10] P. C. McGeer, K. L. McMillan, A. Saldanha, A. L. Sangiovanni-Vincentelli, and P. Scaglia, "Fast discrete function evaluation using decision diagrams," *ICCAD'95*, pp. 402–407, Nov. 1995.

[11] S. Minato, N. Ishiura, and S. Yajima, "Shared binary decision diagram with attributed edges for efficient Boolean function manipulation," *Proc. 27th ACM/IEEE Design Automation Conf.*, pp. 52–57, June 1990.

[12] S. Muroga, *Logic Design and Switching Theory*, Wiley-Interscience Publication, 1979.

[13] S. Nagayama and T. Sasao, "Compact representations of logic functions using heterogeneous MDDs," *33rd International Symposium on Multiple-Valued Logic*, pp. 247-252, Tokyo, Japan, May 15-18, 2003.

[14] S. Nagayama and T. Sasao, "Compact representations of logic functions using heterogeneous MDDs," *IEICE Trans. on fundamentals*, (accepted for publication).

[15] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," *ICCAD'93*, pp. 42–47.

[16] T. Sasao and M. Fujita (ed.), *Representations of Discrete Functions*, Kluwer Academic Publishers 1996.

[17] T. Sasao, *Switching Theory for Logic Synthesis*, Kluwer Academic Publishers, 1999.

[18] F. Somenzi, "CUDD: CU Decision Diagram Package Release 2.3.1," University of Colorado at Boulder, 2001.

[19] K. Takagi, H. Hatakeda, S. Kimura, and K. Watanabe, "Exact minimization of Free BDDs and its application to pass-transistor logic optimization," *IEICE Trans. on fundamentals*, Vol. E82-A, No. 11, pp. 2407-2413, Nov., 1999.

[20] S. Yang, *Logic synthesis and optimization benchmark user guide version 3.0*, MCNC, Jan. 1991.