COMPLEXITY OF MONOTONE NETWORKS FOR COMPUTING CONJUNCTIONS

by

Robert Endre Tarjan

STAN-CS-76-553
JUNE 1976

COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

Complexity of Monotone Networks for Computing Conjunctions

Robert Endre Tarjan */

Computer Science Department
Stanford University
Stanford, California 94305

Abstract. Let $F_1, F_2, \ldots, F_m$ be a set of Boolean functions of the form $F_i = \wedge \{x \in X_i\}$, where $\wedge$ denotes conjunction and each X_i is a subset of a set X of n Boolean variables. We study the size of monotone Boolean networks for computing such sets of functions. We exhibit anomalous sets of conjunctions whose smallest monotone networks contain disjunctions. We show that if $|F_i|$ is sufficiently small for all i, such anomalies cannot happen. We exhibit sets of m conjunctions in n unknowns which require $c_2 m\alpha(m,n)$ binary conjunctions, where $\alpha(m,n)$ is a very slowly growing function related to a functional inverse of Ackermann's function. This class of examples shows that an algorithm given in [12] for computing functions defined on paths in trees is optimum to within a constant factor.

Keywords: Ackermann's function, computational complexity, lower bound, monotone Boolean circuit, path compression, tree.

*/ Research partially supported by National Science Foundation grant MCS 75-22870 and Office of Naval Research contract N00014-76-C-0688. Reproduction in whole or in part is permitted for any purpose of the United States Government.

1. Introduction.

Let $X = \{x_1,\ldots,x_n\}$ be a set of Boolean variables. A Boolean network is a sequence of triples $(\theta_{n+1},a_{n+1},b_{n+1}),\ldots,(\theta_{n+k},a_{n+k},b_{n+k})$ where each θ_i is a binary Boolean operation and each a_i, b_i is an integer less than i. We associate with each integer i, $1 \le i \le n+k$, a Boolean function $f(i)$ given by $f(i) = x_i$ if $1 \le i \le n$, $f(i) = f(a_i)\,\theta_i\,f(b_i)$ if $n+1 \le i \le n+k$. If $F_1,F_2,\ldots,F_m$ are Boolean functions of $x_1,x_2,\ldots,x_n$, the network computes $F_1,F_2,\ldots,F_m$ if there is a function $\phi: \{1,2,\ldots,m\} \to \{1,2,\ldots,n+k\}$ such that $F_i \equiv f(\phi(i))$, where $\equiv$ denotes logical equivalence. The network is monotone if $\theta_i \in \{\wedge,\vee\}$ for all i, where $\wedge$ denotes conjunction and $\vee$ denotes disjunction.

In this paper we study the size of monotone networks for computing certain Boolean functions. Our interest in this problem stems from three sources. (1) Techniques for analyzing monotone network complexity may be useful in analyzing non-monotone network complexity, about which little is known. (2) Our lower bound results apply not only to monotone networks, but to algorithms for other kinds of computation. (3) Our main lower bound result implies that an almost-linear algorithm for computing functions defined on paths in trees [12] is optimum to within a constant factor.

We restrict our attention to Boolean functions F_i of the form $F_i = \wedge\{x \in X_i\}$, where $X_i \subseteq X$. That is, $F_1,F_2,\ldots,F_m$ is a set of conjunctions of various subsets of variables. In Section 2 we review previous results on such sets of functions. In Section 3 we prove a basic result which gives conditions under which we can afford to ignore disjunctions. In Section 4 we exhibit an anomalous set of conjunctions whose minimum-size monotone network contains a disjunction. We also use

the results of Section 3 to derive sufficient conditions for the non-existence of such anomalies. In Section 5 we prove a non-linear lower bound for sets of conjunctions which correspond to paths in trees, thus proving the optimality, to within a constant factor, of the main algorithm in [12].

2. Previous Results.

Several researchers, including Lamagna, Savage [5,10], and Nechiporuk [7], have studied the complexity of monotone networks for computing conjunctions. We summarize their main results here. See [4,8,9] for lower bounds on the size of monotone networks for computing other types of functions. The following two theorems are special cases of much more general results proved by Savage [10].

Theorem A. For $1 \leq i \leq m$, let $X_i \subset X$. Let $F_i = \wedge \{x \in X_i\}$. Then $F_1, F_2, \ldots, F_m$ can be computed by a monotone network using $2m\lceil n/\log m \rceil$ */ binary conjunctions and no additional operations.

The idea used in the proof of Theorem A is the same as used in the four Russians' algorithm for matrix multiplication [2]. For details, see [10].

Theorem B. For m and n polynomially related **/ and sufficiently large, almost all sets of m conjunctions in n unknowns require $c\, mn/\log m$ ***/ operations when computed by any Boolean network.

This theorem can be proved by a straightforward counting argument. See [10] for details and Moon and Moser [6] for a related result.

Theorem B shows that the bound in Theorem A is tight for almost all sets of m conjunctions in n unknowns, if m and n are polynomially related. However, it seems very hard to explicitly exhibit sets of conjunctions which require as many operations as indicated by Theorem B.

---

*/ All logarithms in this paper are base two; $\lceil x \rceil$ denotes the smallest integer not less than x.

**/ We say m and n are polynomially related if there is some polynomial p such that $m \leq p(n)$ and $n \leq p(m)$.

***/ Throughout this paper, c denotes a suitable positive constant.

Nechiporuk [7] and Lamagna and Savage [5] have exhibited sets of n conjunctions in n unknowns which require $c\, n^{3/2}$ binary conjunctions for their monotone computation. Their constructions use the same ideas, which we shall review in Section 5. To the author's knowledge, no harder sets of conjunctions have been explicitly exhibited.

3. Properties of Minimum-Length Monotone Networks.

In order to bound the number of binary conjunctions required by monotone networks for computing specific sets of conjunctions, we need a result which allows us to ignore the effect of disjunction::. In thissectionweshowthat disjunctions can be ignored, provided we allow certain subconjunctions of. previously computed conjunctions to be computed for free. We accomplish this by showing how to transform a monotone network for computing functions $F_i = A\{x \in X_i\}$ into a straight-line computation of the sets X_i from the singleton sets $\{x_j\}$, using the operations of set union and arbitrary subset.

Let $(\Theta_{n+1},a_{n+1},b_{n+1}),\ldots,(\Theta_{n+k},a_{n+k},b_{n+k})$ be a monotone network for computing $A\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$. Let $f(1),f(2),\ldots,f(n+k)$ be the associated Boolean functions and let $\phi: \{1,2,\ldots,m\} \to \{1,2,\ldots,n+k\}$ be such that $f(\phi(i)) = A\{x \in X_i\}$. For $1 \leq i \leq n+k$, let $\bigvee_j A\{x \in Y_j(i)\} \equiv f(i)$ be a disjunctive normal form for $f(i)$. This form is unique up to adding conjunctions $A\{x \in Y_{j'}(i)\}$ such that $Y_j(i) \subseteq Y_{j'}(i)$ for some j.

Let $Z(i) = \bigcap_j Y_j(i)$. $Z(i)$ is independent of the disjunctive normal form chosen to represent $f(i)$. $Z(i) = \{x_i\}$ for $1 \leq i \leq n$ and $Z(\phi(j)) = X_j$ for $1 \leq j \leq m$. If $\Theta_i = \vee$, $Z(a_i) \cap Z(b_i) = \left(\bigcap_j Y_j(a_i)\right) \cap \left(\bigcap_{j'} Y_{j'}(b_i)\right) = Z(i)$, since $\left(\bigvee_j A\{x \in Y_j(a_i)\}\right) \vee \left(\bigvee_{j'} \wedge\{x \in Y_{j'}(b_i)\}\right)$ is a disjunctive normal form for $f(i)$. If $\Theta_i = \wedge$, $Z(a_i) \cup Z(b_i) = \left(\bigcap_j Y_j(a_i)\right) \cup \left(\bigcap_{j'} Y_{j'}(b_i)\right) = \bigcap_j \bigcap_{j'} \left(Y_j(a_i) \cup Y_{j'}(b_i)\right) = Z(i)$, since $\bigvee_j \bigvee_{j'} A\{x \in Y_j(a_i) \cup Y_{j'}(b_i)\} \equiv \left(\bigvee_j A\{x \in Y_j(a_i)\}\right) \wedge \left(\bigvee_{j'} \wedge\{x \in Y_{j'}(b_i)\}\right)$ is a disjunctive normal form for $f(i)$.

Let $1 \leq j,j' \leq n+k$. We say j depends on j' in the network $(\theta_{n+1},a_{n+1},b_{n+1}),\ldots,(\theta_{n+k},a_{n+k},b_{n+k})$ if there is a sequence $j = j(1),j(2),\ldots,j(t) = j'$ such that $j(i+1) \in \{a_{j(i)},b_{j(i)}\}$ for $1 \leq i \leq \ell-1$.

Theorem 1. Let $1 < i < n+k$. Let G be any Boolean function of $x_1,\ldots,x_n$ such that $G \supset A\{x \in Z(i)\}$, and if $\wedge\{x \in X_j\} \supset f(i)$ and $\phi(j)$ depends on i then $A\{x \in X_j\} \supset G$. For $1 \leq j \leq n+k$, let g(j) be the Boolean function defined by $g(j) = x_j$ if $1 \leq j \leq n$ and $j \neq i$, $g(j) = G$ if $j = i$, $g(j) = g(a_j)\,\theta_j\,(b_j)$ if $n+1 \leq j \leq n+k$ and $j \neq i$. Then $g(\phi(j)) = f(\phi(j))$ for $1 \leq j \leq m$.

Proof. Let $1 \leq j \leq m$. If #(j) does not depend on i , then obviously $g(\phi(j)) = f(\phi(j))$. Suppose $\phi(j)$ does depend on i . Let $\bar{y} = (y_1,\ldots,y_n)$ be the Boolean vector such that $y_\ell = 1$ iff x_ℓ [illegible] j . Then $f(\phi(j))(\bar{y}) = 1$. If $f(i)(y) = 1$, then $A\{x \in X_j\} \supset f(i)$ since f(i) is monotone, and $A\{x \in X_j\} \supset G$ by hypothesis. Thus $G(\bar{y}) = 1$, and $g(\phi(j))(\bar{y}) = f(\phi(j))(\bar{y}) = 1$. If $f(i)(y) = 0$, then $f(i)\,(\bar{y}) \leq g(i)\,(\bar{y})$, and $1 = f(\phi(j))(\bar{y}) \leq g(\phi(j))(\bar{y})$. In either case, since g is monotone, $g(\phi(j))\,(\bar{z}) = 1$ for any Boolean vector $\bar{z} = (z_1,z_2,\ldots,z_n)$ such that $f(\phi(j))(\bar{z}) = 1$.

Let $\bar{z}$ be such that $f(\phi(j))(\bar{z}) = 0$. Let ℓ be such that $z_\ell =$ [illegible] . If $x_\ell \in Z(i)$, then $G(\bar{z}) = 0 \leq f(i)(z)$ and $g(\phi(z))(\bar{z}) \leq f(\phi(j))(\bar{z}) = 0$. If $x_\ell \notin Z(i)$, let $Y_{,,}(i)$ be such that $x_\ell \notin Y_{\ell'}(i)$. Let $\bar{y} = (y_1,\ldots,y_n)$ be the Boolean vector such that $y_{\ell''} = 1$ if $z_{\ell''} = 1$ or $x_{\ell''} \in Y_{\ell'}(i)$. Then $f(\phi(j))(\bar{y}) = o$ since $y_\ell = z_\ell =$ [illegible] . But $f(i)(y) = 1 \geq G(\bar{y})$. Thus

$g(\phi(j))(\bar{z}) \leq g(\phi(j))(\bar{y}) \leq f(\phi(j))(\bar{y}) = 0$. Hence $g(\phi(j))(\bar{z}) = 0$ whenever $f(\phi(j))(\bar{z}) = 0$. □

Theorem 2. Let $n+1 \leq i \leq n+k$. If $Y_i(i) \notin X_j$ for all j such that $\phi(j)$ depends on i and all ℓ , then there is a monotone network for computing $A\{x \in X_1\}, \ldots, A\{x \in X_m\}$ of length shorter than k and with fewer binary conjunctions than in $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$.

Proof. Let i be such that $Y_i(i) \notin X_j$ for all j such that $\phi(j)$ depends on i and all ℓ . Then $G = 0$ satisfies the hypotheses of Theorem 1. For $1 \leq j \leq n+k$, let g(j) be the Boolean function defined by $g(j) = x_j$ if $1 \leq j \leq n$, $g(j) = 0$ if $j = i$, and $g(j) = g(a_j)\ \theta_j\ g(b_j)$ if $n+1 \leq j \leq n+k$ and $j \neq i$. By Theorem 1 a network which computes $g(1), \ldots, g(n+k)$ will compute $\wedge\{x \in X_1\}, \ldots, \wedge\{x \in X_n\}$. We can thus simplify $(\theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\theta_{n+k}, a_{n+k}, b_{n+k})$ by deleting all triples (θ_j, a_j, b_j) such that $g(j) = 0$ and modifying other a_j , b_j values appropriately. □

By Theorem 2, any monotone network that uses a minimum number of conjunctions to compute $A\{x \in X_1\}, \ldots, \wedge\{x \in X_m\}$ must satisfy

(*) For all i , $Z(i) \subseteq Y_i(i) \subseteq X_j$ for some j such that $\phi(j)$ depends upon i and some ℓ .

A set network for computing $X_1, X_2, \ldots, X_m$ is a sequence of ordered pairs $(W_1, \tau(1))$, $(W_2, \tau(2)), \ldots, (W_k, \tau(k))$ satisfying

(1) $W_i \subseteq X_{\tau(i)}$ for all i .

(2) For all j , $X_j = W_i$ for some i such that $\tau(i) \leq j$.

(3) For all i , either

(a) $W_i = \{x_j\}$ for some j , or

(b) $W_i \subseteq W_{i'}$ and $\tau(i') \leq \tau(i)$ for some $i' < i$, or

(c) $W_i = W_{i'} \cup W_{i''}$ and $\tau(j'), \tau(i'') \leq t(i)$ for some i', i'' .

Note that this definition depends upon the order of the sets X_j .

Theorem 3. Let $(\Theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\Theta_{n+k}, a_{n+k}, b_{n+k})$ be a monotone network for computing $\wedge\{x \in X_1\}$, ... $\wedge\{x \in X_m\}$. There exists a set network for computing $X_1, \ldots, X_m$ which has no more set unions than the monotone network has binary conjunctions, and no more subset operations than the monotone network has binary disjunctions.

Proof. If the monotone network does not satisfy (*), simplify it applying Theorem 2 until it does satisfy (*). Assume without loss of generality that $\bigcup_{j=1}^{m} X_j = X$. (That is, each variable occurs in some conjunction.) For $1 \leq i \leq n+k$, let $\tau(i)$ be the minimum j such that $\phi(j)$ depends upon i and $Y_\ell(i) \subseteq X_j$ for some ℓ . We claim $(Z(1), \tau(1)), \ldots, (Z(n+k), t(n+k))$ satisfies (1), (2), (3).

Condition (1) is immediate. Condition (2) follows from $X_j = Z(\phi(j))$. For $1 \leq i \leq n$, $Z(i) = \{x_i\}$ and (3a) holds. For $n+1 \leq i \leq n+k$ with $\Theta_i = \vee$, $Z(i) = Z(a_i) \cap Z(b_i)$. Let ℓ be such that $Y_\ell(i) \subseteq X_{\tau(i)}$ and $Y_{\ell'}(i) \not\subseteq Y_\ell(i)$ for $\ell' \neq \ell$. Then either

$Y,(i) = Y_{\ell'}(a_i)$ for some ℓ', in which case $\iota(a_i) \leq \iota(i)$ and $(\exists b)$ holds with $i' = a_i$, or $Y,(i) = Y_{\ell'}(b_i)$ for some ℓ', in which case $\iota(b_i) \leq \iota(i)$ and $(\exists b)$ holds with $i' = b_i$. For $m+1 \leq i \leq m+k$ with $\Theta_i = A$, $Z(i) = Z(a_i) \cup Z(b_i)$. Let ℓ be such that $Y,(i) \subseteq X_{\iota(i)}$ and $Y,,(i) \# Y,(i)$ for $\ell' \neq \ell$. Then $Y,(i) = Y_{\ell'}(a_i) \cup Y_{\ell''}(b_i)$ for some ℓ', ℓ''. It follows that $(\exists c)$ holds with $i' = a_i$, $i'' = b_i$. □

Theorem 3 is powerful enough to allow us to derive lower bounds on the number of binary conjunctions required to compute some interesting sets of conjunctions, as we shall see in Section 5.

4. The Power of Disjunctions.

We might conjecture that any set of conjunctions can be computed in a minimum number of operations by using only conjunctions. The following example shows that this is not true.

Let $X = \{p,q,r,s,u,w,x_1,x_2,x_3,y,z\}$ and consider the following fourteen conjunctions (we use juxtaposition in place of $\wedge$).

$PY = c(1)$	$x_1y = c(5)$	$x_1z = c(8)$	$pux_1x_2x_3y = F(1)$
$qz = c(2)$	$x_1x_2y = c(6)$	$x_1x_2z = c(9)$	$qux_1x_2x_3z = F(2)$
$ry = c(3)$	$x_1x_2x_3y = c(7)$	$x_1x_2x_3z = c(10)$	$rwx_1x_2x_3y = F(3)$
$sz = c(4)$			$swx_1x_2x_3y = F(4)$

We can compute these conjunctions using sixteen binary conjunctions and one disjunction by the following method:

$PY \equiv p \wedge y$	$x_1y \equiv x_1 \wedge y$	$x_1z \equiv x_1 \wedge z$
$qz \equiv q \wedge z$	$x_1x_2y \equiv x_2 \wedge x_1y$	$x_1x_2z \equiv x_2 \wedge x_1z$
$ry \equiv r \wedge y$	$x_1x_2x_3y \equiv x_3 \wedge x_1x_2y$	$x_1x_2x_3 \equiv x_3 \wedge x_1x_2z$
$sz \equiv s \wedge z$		

$$x_1x_2x_3(y \vee z) \equiv x_1x_2x_3y \vee x_1x_2x_3z$$

$$ux_1x_2x_3(y \vee z) \equiv u \wedge x_1x_2x_3(y \vee z)$$

$$wx_1x_2x_3(y \vee z) \equiv w \wedge x_1x_2x_3(y \vee z)$$

$$pux_1x_2x_3y \equiv py \wedge ux_1x_2x_3(y \vee z)$$

$$qux_1x_2x_3z \equiv qz \wedge ux_1x_2x_3(y \vee z)$$

$$rwx_1x_2x_3y \equiv ry \wedge wx_1x_2x_3(y \vee z)$$

$$swx_1x_2x_3z \equiv sz \wedge wx_1x_2x_3(y \vee z)$$

However, at least eighteen binary conjunctions are necessary if no disjunctions -are used. To see this, note that py , qz , ry , sz , x_1y , x_1x_2y , $x_1x_2x_3y$, x_1z , x_1x_2z , $x_1x_2x_3z$ each require a separate binary conjunction, for a total of ten. Each F(i) requires at least two additional conjunctions. To beat eighteen, at least one conjunction must contribute to the construction of two of the F(i) 's. Such a conjunction must construct a subconjunction of either $ux_1x_2x_3$ or $wx_1x_2x_3$.

No subconjunction of ux_ix_j or of wx_ix_j for $i,j \in \{1,2,3\}$ allows the computation of any F(i) in one step. Thus some subconjunction of $ux_1x_2x_3$ or $wx_1x_2x_3$ which contains $x_1x_2x_3$ must be constructed. Without loss of generality we can assume $x_1x_2x_3$ is constructed, using two binary conjunctions. But no single additional conjunction will allow the construction of F(1) , F(2) , F(3) , F(4) in one step each. Thus at least five more binary conjunctions are required, for a total of eighteen.

This example has the undesirable property that certain required conjunctions are subconjunctions of other required conjunctions. We can eliminate this property by adding, for each of the ten short conjunctions

$C(i)$, a new set of variables $\{v_1(i), v_2(i), \ldots, v_{18}(i)\}$, and replacing $C(i)$ by the set of conjunctions $C(i) \wedge v_1(i)$, $C(i) \wedge v_2(i)$, ... $C(i) \wedge v_{18}(i)$. The entire set of conjunctions $\{(C(i) \wedge v_j(i) \mid 1 \leq i \leq 10 , 1 \leq j \leq 18\} \cup \{F(1), F(2), F(3), F(4)\}$ can be computed in $10 \cdot 18 + 16 = 196$ binary conjunctions and one disjunction; if the computation is carried out using only binary conjunctions and some $C(i)$ is not computed, at least $11 \cdot 18 \quad 198$ binary conjunctions are necessary; if each $C(i)$ is computed, $10 \cdot 18 + 18 = 198$ binary conjunctions are required.

This example can be generalized to show that for any n there is a set of n conjunctions in n variables whose computation is faster by a constant factor if disjunctions are used. The author does not know whether the use of disjunctions can speed up such computations by more than a constant factor.

By using the results in Section 3, we can show that if $|X_i|$ is suffiently small for all i , there are minimum-length monotone networks which use only conjunctions to compute the functions $F_1 = A\{x \in X_i\}$.

Theorem 4. Let $(\Theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\Theta_{n+k}, a_{n+k}, b_{n+k})$ be any minimum-length monotone network for computing $A\{x \in X_1\}, \ldots . A\{x \in X_m\}$. Suppose $|X_i| < \ell$ for $1 \leq i \leq m$. For $n+1 \leq i \leq n+k$, if $\Theta_i = \vee$, then $2 \leq |Z(i)| < R-2$.

Proof. Let i be such that $\Theta_i = \vee$. If $Z(i) = \emptyset$, then by Theorem 1 any use of the function $f(i)$ can be replaced by use of the constant function 1 , and the triple (Θ_i, a_i, b_i) is unnecessary. If $Z(i) = \{x_j\}$, then by Theorem 1 any use of the function $f(i)$ can be replaced by use of x_j , and the triple (Θ , a , b) is unnecessary. Suppose $|Z(i)| = \ell$. By (*) $Z(i) \subseteq V,(i) \subseteq X_J$ for some j and since $|X_j| \leq I = |Z(i)|$, $f(i) = \wedge\{x \in X_j\}$. But $Z(i) = Z(a_i) \cap Z(b_i)$, and it follows that $f(a_i) = f(b_i) = A\{x \in X_j\}$. Thus the triple (Θ_i, a_i, b_i) is unnecessary.

Suppose $|Z(i)| = \ell-1$. Choose the minimum i such that $\Theta_i = \vee$ and $|Z(i)| = R-1$. Then $Z(i) = Z(a_{-1}) \cap Z(b_{-1})$. By (*), there must be

some j such that $\phi(j)$ depends on i and $Y_\ell(i) \subseteq X$ for some ℓ. Choose ℓ such that $Y_{\ell'}(i) \not\subset Y_\ell(i)$ for all $\ell' \neq \ell$. Then either $Y_\ell(i) = Y_{\ell''}(a_i)$ for some ℓ'' or $Y_\ell(i) = Y_{\ell''}(b_i)$ for some ℓ''. Suppose without loss of generality that the former is the case. If $|Z(a_i)| = \ell$, then $Z(a_i) = X_j$, $f(a_i) = A\{x \in X_j\}$, and the triple $(\Theta_{\phi(j)}, a_{\phi(j)}, b_{\phi(j)})$ is unnecessary. If $|Z(a_i)| = \ell\text{-}1$, say $X_j - Z(a_i) = \{x_{i'}\}$, we can replace the triple $(\Theta_{\phi(j)}, a_{\phi(j)}, b_{\phi(j)})$ by $(\wedge, a_i, i')$. Repeating this construction for each j such that $\#(j)$ depends on i and $Y_\ell(i) \subseteq X_j$ for some ℓ, we eventually create a network which violates (*) and which can be simplified by Theorem 2. □

Theorem 5. If $|X_i| \leq 4$ for $1 \leq i \leq m$, then any minimum-length monotone network for computing $A\{x \in X_1\}, \ldots A\{x \in X_m\}$ uses only conjunctions.

Proof. Let $(\Theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\Theta_{n+k}, a_{n+k}, b_{n+k})$ be any minimum-length monotone network for computing $A\{x \in X_1\}, \ldots, A\{x \in X_m\}$. Choose the minimum i such that $\Theta_i = \vee$ if there is any such i. By Theorem 4, $|Z(i)| = 2$. Suppose $Z_i = \{x_j, x_{j'}\}$. If $Z(i) = Z(a_i)$ or $Z(i) = Z(b_i)$, then by Theorem 1 (Θ_i, a_i, b_i) and one of the triples $(\Theta_\ell, a_\ell, b_\ell)$ for $\ell \in \{a_i, b_i\}$ can be replaced by the triple $(\wedge, j, j')$. If $Z(i) \subset Z(a_i)$ and $Z(i) \subset Z(b_i)$, then $|Z(a_i)| \geq \ell\text{-}1$ and $|Z(b_i)| \geq \ell\text{-}1$. An argument like that in Theorem 4 for the case $|Z(i)| = \ell\text{-}1$ shows that the network can be simplified. Thus any network containing a disjunction is not of minimum length. □

Theorem 6. If $|X_i| \leq 5$ for $1 < i < m$, then some minimum-length monotone network for computing $A\{x \in X_1\}, \ldots, A\{x \in X_m\}$ uses only conjunctions.

Proof. Let $(\Theta_{n+1}, a_{n+1}, b_{n+1}), \ldots, (\Theta_{n+k}, a_{n+k}, b_{n+k})$ be any minimum-length monotone network for computing $A\{x \in X_1\}, \ldots, A\{x \in X_m\}$. Choose the minimum i such that $\Theta_i = V$, if there is any such i. By Theorem 4, $|Z(i)| \in \{2,3\}$. If $Z(i) = \{x_j, x_{j'}\}$, then by Theorem 1 the triple (Θ_i, a_i, b_i) can be replaced by the triple $(\wedge, j, j')$. If $|Z(i)| = 3$, an argument like that in Theorem 5 shows that part of the network, including the disjunction Θ_i, can be replaced by conjunctions without increasing the length of the network. By repeating the construction for each disjunction, a minimum-length network without disjunctions is obtained. □

The example previously considered shows that the bound of five in Theorem 6 cannot be improved, and a similar example shows that the bound of four in Theorem 5 cannot be improved.

5. Lower Bounds for Explicit Sets of Conjunctions.

In this section we review the $cn^{3/2}$ lower bound result of [5] and [7] and provide another non-linear lower bound, tight to within a constant factor, on the monotone complexity of a family of sets of conjunctions.

Theorem 7[5]. Let $X_i \subset X$ for $1 \leq i \leq m$ be subsets of variables such that $|X_i \cap X_j| \leq 1$ for all i, j. Then any monotone network for computing $A\{x \in X_i\}$ for $1 \leq i \leq m$ has $\sum_{i=1}^{m} |X_i| - m$ binary conjunctions.

Proof. Consider any set network for computing $X_1, X_2, \ldots, X_m$. For any particular X_i, at least $|X_i| - 1$ unions are required to combine the elements of X_i into a single set. Each union used to combine elements of X_i produces a set containing at least two elements of X_i. Since any pair of elements is contained in a unique set $_i$, each union used to combine elements of X_i produces a set contained in X_i but in no $X_j \neq X_i$. It follows that the $\sum_{i=1}^{m} |X_i| - m$ unions required to combine elements in $X_1, \ldots, X_m$ are all distinct. The theorem follows from Theorem 3. This proof is due to Lamagna and Savage [5], except that they use a less general result than Theorem 3 as an intermediate step. ❑

Lamagna and Savage [5] and Nechiporuk [7] have exhibited families of functions which satisfy Theorem 7 and have $\sum_{l-1}^{m} |X_i| \geq cn^{3/2}$. Here is another family of such functions.

Let $X = \{x_1, x_2, \ldots, x_n\}$, where $n = k^2+k+1$. Let $X_i \subset X$, for $1 \leq i \leq n$, be the n lines of a projective plane [3] whose set of points is X. A projective plane has the property that each pair of points is contained in exactly one line, and each line contains exactly $k+1$ points. Projective planes exist for all prime powers $k = p^r$ [3].

Theorem 8. Any monotone network for computing $\wedge \{x \in X_i\}$ for $1 \leq i \leq n$ requires nk binary conjunctions.

Proof. Immediate from Theorem 7. □

The set of conjunctions defined by the lines of a projective plane thus provides a simple, explicit example of a monotone Boolean function which requires a non-linear number of operations when computed by any monotone network. Note that the bound in Theorem7 is obviously tight and implies that any minimum-length monotone network for such sets of conjunctions uses no disjunctions. It is not hard to show that the largest lower bound that can possibly be proved using Theorem7 is $O(n^{3/2})$.

We shall now use Theorem 3 to show that the algorithm of [12] for computing functions defined on paths in trees is optimum to within a constant factor. For this purpose we need a few definitions from graph theory. A directed graph. $G = (V,E)$ consists of a finite set V of vertices and a set E of ordered pairs (v,w) of distinct vertices, called edges. A path of length k from v to w in G is a sequence of edges $p = (v_1,v_2),(v_2,v_3),\ldots,(v_k,v_{k+1})$ with $v_1 = v$ and $v_{k+1} = w$. A (directed, rooted) tree (T, r) is a directed graph T with a distinguished vertex r such that there is a unique path from r to any other vertex. We say v and w are related (v is an ancestor of w and w is a descendant of v) if there is a path from v to w in T .

Let T be a directed, rooted tree with n+1 vertices (and n edges). For each edge (v,w) in T , let $x(v,w)$ be an associated Boolean variable. For related vertices v, w in T , let $f(v,w)$ be the Boolean function $f(v,w) = \wedge \{x(y,z) \mid (y,z) \text{ on } p(v,w)\}$, where $p(v,w)$ is the path from v to w in T . Consider the problem of computing $f(v_j,w_j)$, for m related pairs of vertices (v_j,w_j) , using a monotone network. An $O(m\ \alpha(m,n))$ operation method for solving a more general version of this problem (where conjunction is replaced by any associative operation) is presented in [12]. Here we show that no better method is possible.

Given m related pairs (v_j,w_j) , order the pairs so that if (v_j,w_j) precedes $(v_{j'},w_{j'})$ in the ordering and $v_j \neq v_{j'}$, then v_j is not an ancestor of $v_{j'}$ in T . (This is always possible; see [12].) Now associate with T and with the pairs (v_j,w_j) a directed graph G^* and a cost C as follows. Initialize G^* to $G^* = T$. Process the

pairs (v_j,w_j) in the order defined above. To process a pair (v_j,w_j), let $p(v_j,w_j) = (y_1,y_2), (y_2,y_3), \ldots, (y_k,y_{k+1})$. Add to G^* each edge $(y_i,y_{i'})$ with $i < i'$ which is not already present in G^*. Let the cost of pair (v_j,w_j) be $\ell_j - 1$; where I_j is the length of the shortest path from y_j to w_j in G^* (before the new edges for (v_j,w_j) are added). Let the cost' C be the total cost of all pairs (v_j,w_j).

Theorem 9. The cost C is a lower bound on the number of unions in any set network which computes $X_1,X_2,\ldots,X_m$, where $X_j = \{x(y,z) \mid (y,z) \text{ is on } p(v_j,w_j)\}$.

Proof. Consider any set network $(W_1,z(1)), \ldots, (W_k,z(k))$ for computing $X_1,X_2,\ldots$. We claim that, for any j, the number of values i, such that $z(i) = j$ and W_i satisfies (3c) but not (3a) or (3b), is at least as great as the cost of (v_i,w_i).

Consider any set X_j. By (2), there must be some i such that $X_j = W_i$ and $z(i) \leq j$. Furthermore, W_i must be constructed from singleton sets using the subset and union operations of (3b) and (3c). It follows that there are indices $i_1,i_2,\ldots,i_\ell \leq i$ such that

(i) $X_j \subseteq W(i_1) \cup W(i_2) \cup \cdots \cup W(i_\ell)$;

(ii) each $W(i_{j'})$ satisfies either (3a) or $z(i_{j'}) < z(i)$; and

(iii) the number of sets $W(i')$ satisfying neither (3a) nor (3b) but such that $z(i') = i$ is at least $\ell - 1$.

(We can produce this set of indices $i_1,i_2,\ldots,i_\ell$ with a backward trace through the set network from index i.)

Order $W(i_1),\ldots,W(i_\ell)$ so that $W(i_1)$ contains the variable for the first edge on $p(v_j,w_j)$, $W(i_{j'})$ for $2 \le j' \le h$ contains the variable for the edge on $p(v_j,w_j)$ following the last edge whose variable is in $W(i_{j'-1})$, and $W(i_h)$ contains the variable for the last edge on $p(v_j,w_j)$. (Note that the contents of the sets $W(j_{h+1}),\ldots,W(j_\ell)$ are unimportant.)

Since $W(i_{j'}) \subseteq X_{z(i_{j'})}$ for $1 \le j' \le \ell(0)$, there is a sequence of edges $(u_1,u_2), (u_2,u_3), \ldots, (u_h, u_{h+1})$ in G^* such that $u_1 = v_j$; $u_{h+1} = w_j$; and, for $1 < j' < h$, edge (u_j,u_{j+1}) is added to G^* before or during the time $(v_{z(i_{j'})},w_{z(i_{j'})})$ is processed. Since $z(i_{j'}) < z(i) \le j$, $(u_{j'},u_{j'+1})$ is present in G^* before (v_j,w_j) is processed. Hence h-l, and also R-l, is an upper bound on the cost of (v_j,w_j). This proves the claim, and the lemma follows by summing over all j. □

Now we apply the very general lower bound result of [11], which states:

Theorem C [11]. There is a positive constant c such that, for all m and n with $m \ge n$, there is a tree T of n vertices and a sequence of m pairs (v_j,w_j) of related vertices for which the total cost C is at least $c\,m\,\alpha(m,n)$. */

*/ We define $\alpha(m,n)$ as follows. Let A(i,x) be defined on non-negative integers by $A(0,x) = 2x$ for $x \ge 0$, $A(i,0) = 0$ for $i \ge 1$, $A(i,1) = 2$ for $i \ge 1$, and $A(i,x) = A(i-1, A(i,x-1))$ for $i \ge 1$, $x \ge 2$. A(i,x) is a slight variant of Ackermann's function [1]. Let $\alpha(m,n) = \min\{z \ge 1 \mid A(z, 4\lceil m/n\rceil) > \log n\}$.

We have, from Theorems 3, 9, and C:

Theorem 10. For any $m \geq n$, there is a tree T and a set of m pairs (v_j, w_j) of related vertices such that any monotone network for computing $f(v_j, w_j)$ for each pair uses at least $c\, m\alpha(m,n)$ binary conjunctions.

This result implies that the algorithm of [12] for computing $f(v_j, w_j)$ for such pairs is optimum to within a constant factor, among straight-line algorithms.

6. Remarks.

The lower bounds in Theorems 8 and 10 apply not only to monotone networks, but also to straight-line algorithms which use more general operations to combine the appropriate subsets of inputs. For instance, the lower bounds hold if $\wedge$ is replaced by maximum over real numbers and $\vee$ is replaced by minimum over real numbers.

For operations over certain domains, the lower bounds of Theorems 8 and 10 apply not only to straight-line algorithms, but to all algorithms which can compute values only by applying various operations to the input values. Such domains include sets (replacing $\wedge$ by set union and $\vee$ by set intersection), strings (replacing $\wedge$ by string concatenation and omitting $\vee$), and function spaces (replacing $\wedge$ by function composition and omitting $\vee$).

Apparently, no one has yet explicitly exhibited a family of monotone Boolean functions $f_n: \{0,1\}^n \to \{0,1\}$ such that f_n has a monotone circuit of size polynomial in n but no monotone circuit of size linear in n. The family of functions

$$f_n(x_1, \ldots, x_n) = \bigvee_{i=1}^{n} \wedge \{x \in X_{in}\} ,$$

where the sets $X_{1n}, X_{2n}, \ldots, X_{nn}$ are the lines of a projective plane whose points are $\{x_1, \ldots, x_n\}$, is a possibility for such a family. Another possibility is the family of functions

$$f_n(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{m} \wedge \{x \in X_{in}\} ,$$

where $x_{1n}, x_{2n}, \ldots, x_{mn}$ correspond to appropriate path:; in a tree of n edges. Perhaps the proofs of Theorems 8 and10 can be extended to give lower bounds in these cases. Another open problem is to determine by how much disjunction helps in computing sets of conjunctions.

Acknowledgments. Extensive discussions with Mike Paterson helped to clarify the ideas in this paper. John Savage provided several important references.--.

References

[1] W. Ackermann, "Zum Hilbertshen Aufbau der reelen Zahlen," Math. Ann. 99 (1928), 118-133.

[2] A. V. Aho, J. E. Hopcroft, J. D. Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley Publishing Co., Reading, Mass. (1974), 243-247.

[3] M. Hall, Jr., Combinatorial Theory, Blaisdell Publishing Co., Waltham, Mass. (1962), 167-188.

[4] E. A. Lamagna and J. E. Savage, "On the logical complexity of symmetric switching functions in monotone and complete bases," Technical Report, Center for Computer and Information Sciences, Brown University (1973).

[5] E. A. Lamagna and J. E. Savage, "Combinational complexity of some monotone functions," Fifteenth Annual Symp. on Switching and Automata Theory (1974), 140-144.

[6] J. W. Moon and L. Moser, "A matrix reduction problem," Mathematics of Computation 20 (1966), 328-330.

[7] É. I. Nechiporuk, "On a Boolean matrix," Systems Theory Research 21 (1971), 236-239.

[8] M. S. Paterson, "Complexity of monotone networks for Boolean matrix product," Theoretical Computer Science 1 (1975), 13-20.

[9] V. R. Pratt, "The power of negative thinking in multiplying Boolean matrices," Proceedings of Sixth Annual ACM Symp. on Theory of Computing (1974), 80-83.

[10] J. E. Savage, "An algorithm for the computation of linear forms," SIAM J. Comput. 3 (1974), 150-158.

[11] R. E. Tarjan, "Efficiency of a good but not linear set union algorithm," J.ACM 22 (1975), 215-225.

[12] R. E. Tarjan, "Applications of path compression on balanced trees," STAN-CS-75-512, Computer Science Department, Stanford University (1975).