SYMmetric Complementation*

John H. Reif

TR-07-81

*This is a revised version of "O(log N) algorithms in sequential space and parallel time (for min spanning trees, k-connectivity, planarity and other symmetric games.)
**Title:** Symmetric Complementation  
**Type of Report & Period Covered:** Technical Report  
**Performing Organization Name and Address:** Harvard University, Cambridge, MA 02138  
**Controlling Office Name and Address:** Office of Naval Research, 800 North Quincy Street, Arlington, VA 22217  
**Security Classification:** Unlimited  

**Key Words:** parallel algorithm, planarity, connectivity, symmetric computation, probabilistic algorithm, randomized algorithm.  

**Abstract:** See reverse.
Summary. This paper introduces a class of 1 player games of perfect information, which we call complementing games; the player is allowed moves which complement the value of successive plays. A complementing game is symmetric if all noncomplement moves are reversible (i.e., form a symmetric relation). These games are naturally related to a class of machines we call symmetric complementing machines. Symmetric nondeterministic machines were studied in [Lewis and Papadimitriou, 80]; they are identical to our symmetric complementing machines with complement moves allowed only on termination. (A companion paper to appear will describe the computational complexity of symmetric complementing and alternating machines.) Of particular interest is the complexity class $\text{CSYMLOG}$, which contains the outcome problem of symmetric complementing games with constant complement bound, with game positions encoded in log space, and next move relations computable in log space. We show that the decision problem for a restricted quantified Boolean logic $\mathcal{DLBCP}$ is complete in $\text{CSYMLOG}$. We also show that $\text{CSYMLOG}$ contains many well-known and common combinatorial problems:

1. minimum spanning forests
2. $k$-connectivity and $k$-connected components

and also the recognition problems for many classes of graphs:

3. planar graphs of valence 3
4. chordal graphs
5. comparability graphs
6. interval graphs
7. split graphs
8. permutation graphs.

(cont. on next page)
We present a probabilistic algorithm (this is an algorithm which makes probabilistic choices [Rabin, 74], but with no assumptions about the probability distribution of the inputs) for recognizing the languages of $\Sigma^*_r$CSYMLOG within space $O(\log(n))$ and simultaneous time $n^{O(1)}$, with error probability $< \epsilon$ for any given $\epsilon, 0 < \epsilon < 1$. As a consequence, problems (1)-(8) can be done probabilistically in space $O(\log(n))$ and within simultaneous polynomial time. The best previous known algorithms for problems (1), (2) and (3) required deterministic space $\Omega(\log^2 n)$ [Ja'Ja' and Simon, 79], and algorithms for problems (4)-(8) previously required space $\Omega(n)$.

Also, we give a probabilistic parallel algorithm (which employs the Hardware Modification Machines of [Cook, 80], with probabilistic choice) for recognizing the languages of $\Sigma^*_r$CSYMLOG within parallel time $O(\log n)$ and error probability $< \epsilon$, for any given $\epsilon, 0 < \epsilon < 1$. Thus we also have parallel time $O(\log n)$ algorithms for problems (1)-(8). Our parallel algorithms seem practical since they require only a small polynomial number of processors. The best previously known parallel algorithms for problems (1)-(3) required parallel time $\Omega(\log^2 n)$ [Ja'Ja' and Simon, 80] and we know of no previous parallel algorithms for problems (4)-(8). Furthermore, we show (by a nonconstructive technique) that for each input length $n \geq 0$, the probabilistic choice can be eliminated in both our sequential and parallel algorithms. This does not affect the efficiency of the algorithms, but makes our algorithms nonuniform (i.e., we have a different algorithm for each input length).
SYMOMETRIC COMPLEMENTATION

by

John H. Reif*

Aiken Computation Laboratory
Harvard University
Cambridge, MA 02138

*This work was supported in part by the National Science Foundation Grant NSF-MCS79-21024, the Office of Naval Research Contract N00014-80-C-0647.
Summary. This paper introduces a class of 1 player games of perfect
information, which we call complementing games; the player is allowed
moves which complement the value of successive plays. A complementing
game is symmetric if all noncomplement moves are reversible (i.e., form a
symmetric relation). These games are naturally related to a class of
machines we call symmetric complementing machines. Symmetric nondeterministic
machines were studied in [Lewis and Papadimitriou, 80]; they are identical to
our symmetric complementing machines with complement mover allowed only on
termination. (A companion paper to appear will describe the computational
complexity of symmetric complementing and alternating machines.) Of
particular interest is the complexity class $\Sigma^+_E$CSYMLOG, which contains the
outcome problem of symmetric complementing games with constant
complement bound with game positions encoded in log space, and next move
relations computable in log space. We show that the decision problem for
a restricted quantified Boolean logic $\Sigma^+_E$QBF is complete in $\Sigma^+_E$CSYMLOG.
We also show that $\Sigma^+_E$CSYMLOG contains many well-known and common combinatorial problems:

1. minimum spanning forests
2. k-connectivity and k-connected components

and also the recognition problems for many classes of graphs:

3. planar graphs of valence 3
4. chordal graphs
5. comparability graphs
6. interval graphs
7. split graphs
8. permutation graphs.
We present a probabilistic algorithm (this is an algorithm which makes probabilistic choices [Rabin, 74], but with no assumptions about the probability distribution of the inputs) for recognizing the languages of $\Sigma_a$CSYMLOG within space $O(\log(n))$ and simultaneous time $n^{O(1)}$, with error probability $<\varepsilon$ for any given $\varepsilon, 0 < \varepsilon < 1$. As a consequence, problems (1)-(8) can be done probabilistically in space $O(\log(n))$ and within simultaneous polynomial time. The best previous known algorithms for problems (1), (2) and (3) required deterministic space $\Omega(\log^2 n)$ [Ja'Ja' and Simon, 79], and algorithms for problems (4)-(8) previously required space $\Omega(n)$.

Also, we give a probabilistic parallel algorithm (which employs the Hardware Modification Machines of [Cook, 80], with probabilistic choice) for recognizing the languages of $\Sigma_a$CSYMLOG within parallel time $O(\log n)$ and error probability $<\varepsilon$, for any given $\varepsilon, 0 < \varepsilon < 1$. Thus we also have parallel time $O(\log n)$ algorithms for problems (1)-(8). Our parallel algorithms seem practical since they require only a small polynomial number of processors. The best previously known parallel algorithms for problems (1)-(3) required parallel time $\Omega(\log^2 n)$ [Ja'Ja' and Simon, 80] and we know of no previous parallel algorithms for problems (4)-(8). Furthermore, we show (by a nonconstructive technique) that for each input length $n \geq 0$, the probabilistic choice can be eliminated in both our sequential and parallel algorithms. This does not affect the efficiency of the algorithms, but makes our algorithms nonuniform (i.e., we have a different algorithm for each input length).
1. INTRODUCTION

In the previous decade, considerable success has been made in the design of time optimal sequential algorithms for many combinatorial problems on graphs (such as spanning trees, $k$-connectivity for $k=1,2,$ and 3 [Hopcroft and Tarjan, 73b] and planarity testing [Hopcroft and Tarjan, 73a]) using the technique of depth-first search. Also, breadth-first search has been used for optimal algorithms for other graph problems. By applying well-known simulation results (for example, [Fortune and Wyllie, 78]), we can derive parallel space optimal algorithms from these sequential time optimal algorithms. Also, parallel time has been related to sequential space (by the simulation results of [Fortune and Wyllie, 78] and [Dymond and Cook, 80], for example). It is intriguing therefore to ask:

(i) Is there a general graph search technique which yields optimal sequential space algorithms and (either by simulation results or directly) also yields optimal parallel time algorithms?

We require that these algorithms be reasonable: that a sequential algorithm with space bound $S(n)$ uses no more than $2^{O(S(n))}$ sequential time (note that if $S(n) = O(\log n)$ then $n^{O(1)}$ sequential time will be used) and that a parallel algorithm with time bound $T(n)$ use no more than $2^{O(T(n))}$ processors (again, note that $T(n) = O(\log n)$ implies $n^{O(1)}$ processors will be used). (Note that certain probabilistic TMs have a time bound doubly exponential in these space bound and are there for not reasonable).

The depth-first search and breadth first search techniques appear not to be applicable to (i), due to their sequential nature. Another possible technique for solving a graph problem is to efficiently reduce the problem to boolean transitive closure, for which there is a known $\Omega(\log^2 n)$ parallel time algorithm [Csanky, 76]. [Ja'Ja' and Simon, 80] apply this
technique to do planarity testing and to solve connectivity problems in \( \Omega(\log^2 n) \) parallel time. (However, boolean transitive closure has no known algorithm with less than \( \Omega(\log^2 n) \) parallel time bound, and this bound seems very difficult to improve, whereas we show here the planarity testing and connectivity problems actually have \( O(\log n) \) probabilistic parallel time algorithms.)

A related problem is:

(ii) Is there a logic, in which a significant class of combinatorial problems may be succinctly expressed, and such that validity of sentences in the logic may be decided efficiently or even optimally? (with respect to sequential space or parallel time)

A logic satisfying the conditions of (ii) could be used as the kernel of a language for parallel programming, where programs may be "compiled" into time optimal code for parallel machines.

This paper proposes solutions to (i) and (ii). A space efficient sequential probabilistic search technique was first introduced by [Aleliunas, Karp, Lipton, Lővész, and Rackoff, 79] to test 1-connectivity. We generalize the probabilistic search technique to yield optimal algorithms (in sequential space and also parallel time) for the complexity class \( \Sigma_p \text{CSYMLOG} \), which contains a large class of many important combinatorial problems. Furthermore, we propose a restricted quantified boolean logic \( \Sigma_q \text{QBF}^0 \) as a solution to (ii).

This paper is organized as follows:

Section 2 defines complementing games and machines, complexity notation. Section 3 provides a sequential decision algorithm for the problems of \( \Sigma_p \text{CSYMLOG} \), which runs in probabilistic space \( O(\log n) \), and can be modified to run in nonuniform deterministic space \( O(\log n) \).
Section 4 gives a probabilistic parallel algorithm which can simulate any sequential space \( S(n) \) bounded probabilistic computation within parallel time \( O(S(n)) \). We also show in Section 4 that we can eliminate probabilistic choices in our parallel algorithm, without degrading its running time, but introducing nonuniformity. Section 5 introduces our logic \( \Sigma_\circ \mathbb{QBF} \) and shows its invalidity problem is complete in \( \Sigma_\circ \text{CSYMLOG} \). We also show in Section 5 that various combinatorial problems (including minimum spanning trees, \( k \)-connectivity and graph recognition problems such as valence 3 planarity testing) are in \( \Sigma_\circ \text{CSYMLOG} \).

2. PRELIMINARY DEFINITIONS

2.1 Symmetric Relations

Let \( R \subseteq D \times D \) be a relation on domain \( D \). Let its inverse be \( R^\sim = \{(b,a) | a \in R b\} \). \( R \) is symmetric if \( R = R^\sim \). \( R \) is deterministic if for all \( a \in D \) there is at most one \( b \in D \) such that \( arb \).

2.2 Complementing Games

A (1 player) complementing game consists of a quadruple \( G = (P,W,\mathcal{R},\mathcal{R}_C) \) where:

(i) \( P \) is the set of positions; \( P \) is assumed to be a set of strings over a finite alphabet.

(ii) \( W \subseteq P \) are the winning positions,

(iii) \( \mathcal{R} \subseteq P \times P \) is the next move relation, and

(iv) \( \mathcal{R}_C \subseteq \mathcal{R} \) are the complement moves.
Fix an initial position $p_0 \in P$. A move is a pair $(p, p') \in t$. The move $(p, p')$ is an initial move (termination move, complement move) if $p = p_0$ (if $p'$ has no next move, and $(p, p') \in t_c$, respectively). We assume there is no next move from any winning position.

Let complement game $G$ be symmetric if $t_s = t - t_c$ is a symmetric relation (see 2.1). Let $t$ be log space if there is a deterministic log space next move transducer which given any $p \in P$, outputs $(p'|p\rightarrow p')$.

Let a play be a maximal length sequence of positions $P_0, P_1, ...$ where $P_0$ is the initial position, and $P_{i-1} \rightarrow P_i$ for $i = 1, 2, ...$ (we allow a trivial play $P_0$). For any finite $k > 0$, let $p_0$ have complement bound $k$ if any play from $P_0$ has $<k$ complements ignoring the initial move (ignoring the initial move allows us to maintain the duality between $\Sigma_k$ and $\Pi_k$).

Suppose $p_0$ has finite complement bound $k$. Let $\text{OUTCOME}(p_0) = \text{true}$ if there exists a finite play prefix $P_0, P_1, ..., P_j, j \geq 0$, with no complement moves and where either $P_j \in W$ or there exists at least one complement move from $P_j$ and $\text{OUTCOME}(P') = \text{false}$ for each complement move $(p_j, p') \in t_c$. Otherwise $\text{OUTCOME}(p_0) = \text{false}$. The outcome problem for $G$ is given any $p \in P$, next move transducer for $t$, and a recognizer for $W$, compute $\text{OUTCOME}(p)$.

2.2 Machine Definitions

Let a complementing (Turing) machine be a 9-tuple $M = (\Sigma, \Gamma, b, t, \Omega, q_0, Q_A, \delta, \delta_c)$ where

*See note at end of this paper on previous and equivalent machine definitions for symmetric alternating machines.
\[ \Sigma \] is the finite input alphabet
\[ \Gamma \] is the finite tape alphabet, with \[ \Sigma \subseteq \Gamma \]
\[ b \in \Gamma \] is the distinguished blank symbol,
\[ t \] is the number of tapes where tape 1 is the input tape
\[ Q \] is the finite state set
\[ q_0 \in Q \] is the initial state
\[ Q_A \subseteq Q \] are the accepting states
\[ \delta \subseteq (Q \times (\Sigma^2 \times \{ \text{left, right} \}) \times (\Sigma^2 \times \{ \text{left, right} \})^{t-1})^2 \] is the transition relation
\[ \delta_c = \delta \] are the complement transitions.

\( \delta \) has a slightly nonstandard definition so that we can syntactically define symmetric machines in a manner similar to [Lewis and Papadimitriou, 80].
Suppose transition \((q, a_1 b_{i_1} m_1, \ldots, a_t b_{t m_t}), (q', a_1' b_{i_1} m_1', \ldots, a_t' b_{t m_t}') \in \delta \) is taken. Then the previous state was \( q \) and the new state is \( q' \). Each tape \( i \in \{1, \ldots, t\} \) moves its head one cell in direction \( m_i \), and \( m_i \) is the reverse of direction \( m_i' \). If \( m_i = \text{right} \) then previously the head of tape \( i \) was scanning symbol \( a_i \) and "peeking" at symbol \( b_i \) located one cell to the right; in the new configuration these symbols \( a_i b_i \) replaced by symbols \( a_i' b_i' \) and the head is scanning symbol \( b_i' \). The case \( m_i = \text{left} \) is similar, except the head was previously scanning over symbol \( b_i \) while "peeking" at symbol \( a_i \) located one cell to the left; afterwards the head is scanning symbol \( a_i' \). We assume there is no transition from an accepting state. Let \( M \) be symmetric if noncomplement transitions \( \delta = \delta - \delta_c \) are a symmetric relation. Let \( \mathcal{I} \) be the configurations of \( M \), defined in the usual way for Turing machines. We may extend \( \delta \) in the usual way to the next move relation \( \vdash \subseteq \mathcal{I} \times \mathcal{I} \). Let \( \vdash_c \subseteq \vdash \) be the next moves which are complements. Note that if \( M \) is symmetric then the relation \( \vdash_s = \vdash - \vdash_c \).
is symmetric. Let $W = \mathcal{J}$ be the accepting configurations. $G_M = (\mathcal{S}, W, \rightarrow, c)$ is the computation complementing game of $M$.

Given an input string $w \in \Sigma^*$, we let the initial configuration $I_0(w)$ be the initial position of $G_M$. Computation sequences of $M$ are plays of $G_M$ from $I_0(w)$. Suppose $M$ has a finite complement bound from $I_0(w)$. $M$ accepts $w$ if $\text{OUTCOME}(I_0(w)) = \text{true}$. If $M$ has only complement moves which are terminations, than $M$ is a nondeterministic machine; furthermore, if $I$ is symmetric than $M$ is a symmetric nondeterministic machine as defined by [Lewis and Papadimitriou, 80]. Let $L(M) = \{w \in \Sigma^* | M \text{ accepts } w\}$.

2.3 Complexity Classes

Let complementing machine $M$ have space bound $S(n)$ (complement bound $K(n)$) if on any input of length $n > 0$, each computation sequence has no more than $S(n)$ nonblank cells on any work tape in each configuration (less than $K(n)$ complements on any computation sequence ignoring the initial moves).

Note that nondeterministic and co-nondeterministic machines have complement bound 1 (see 2.2). For notational simplicity, we define a complementing machine with complement bound 0 to be a deterministic TM. Let $\mathcal{M}$ be a class of complementing machines. Let $\mathcal{MSPACE}(S(n))$ be the languages accepted by those machines in $\mathcal{M}$ with space bound $S(n)$. Let $\mathcal{MSPACE}(S(n))$ (or $\mathcal{MSPACE}(S(n))$) be the languages accepted by those machines in $\mathcal{M}$ with space bound $S(n)$, complement bound $k$ and no complement moves (only complement moves, respectively) for the initial moves. Let $\Sigma^{\mathcal{MSPACE}(S(n))} = \bigcup_{k>0} \mathcal{MSPACE}(S(n))$; that is the machines operate in some constant number of complements, independent of the input length.
In the context of complexity classes, we let $D$ denote the class of deterministic TMs, let $N$ denote the nondeterministic TMs, and let $NSYM$ be the symmetric nondeterministic machines. Let $C$ be the complementing machines and let $CSYM$ be the symmetric complementing machines. For example, the complexity class $NSYMSPACE(S(n)) = \{L | L$ is accepted by a symmetric nondeterministic machine with space $S(n)\}$ previously investigated by [Lewis and Papadimitriou, 80]. The complexity class $\Sigma_k(n)CSYMSPACE(S(n)) = \{L | L$ is accepted by a symmetric complementing machine with space bound $S(n)$, complement bound $K(n)$, and no complement initial moves\} is of central importance to this paper. For notational simplicity, let $NSYMLOG = NSYMSPACE(\log(n))$, and $CSYMLOG = CSYMSPACE(\log(n))$.

Let $L_1 \leq_{\log} L_2$ denote that language $L_1$ can be many-one reduced in deterministic log-space to language $L_2$. Let $L_1$ be $\leq_{\log}$ equivalent to $L_2$ if $L_1 \leq_{\log} L_2$ and $L_2 \leq_{\log} L_1$. Let $L_2$ be complete in a family of languages $\mathcal{P}$ if $L_2 \in \mathcal{P}$ and $L_1 \leq_{\log} L_2$ for each $L_1 \in \mathcal{P}$. Note that if $S(n) \geq \log n$, $L_1 \leq_{\log} L_2$, and $L_2 \in CSYMSPACE(S(n))$, then $L_1 \in CSYMSPACE(S(n))$, by Proposition 2.3.

### 2.4 Preliminary Results for Symmetric Complementing Machines

It follows immediately from our definition of complementing machines that:

**Proposition 2.1.** For any $L \subseteq \Sigma^*$, $L \in \Sigma_k(n)CSYMSPACE(S(n))$ iff $\Sigma^* - L \in \Pi_k(n)CSYMSPACE(S(n))$.

Since any complementing machine has a finite tape alphabet, it is easy to show

**Proposition 2.2.** If complementing machine $M$ has space bound $S(n) \geq \log n$, then $M$ has complement bound $2^0(S(n))$. 


[Lewis and Papadimitriou, 80] show $\text{DSPACE}(S(n)) \subseteq \text{NSYMSPACE}(S(n))$.

Also, since any symmetric complementing machine is a complementing machine:

**Proposition 2.3.** $\text{DSPACE}(S(n)) \subseteq \Sigma_{K(n)} \text{CSYMSPACE}(S(n)) \subseteq \Sigma_{K(n)} \text{CSPACE}(S(n))$.

(Note that space bounded complementing machines accept the same languages as space bounded alternating machines and that complementing machines without space bounds accept some not r.e. languages but that is not relevant to this paper. Further properties of symmetric complementing and alternating machines will appear in a companion paper, but are not required for the results of this paper.)

3. A SPACE EFFICIENT DECISION ALGORITHM FOR SYMMETRIC COMPLEMENTING MACHINES

We give a $O(S(n)K(n))$ space sequential algorithm for recognizing the languages of $\Sigma_{K(n)} \text{CSYMSPACE}(S(n))$. The algorithm is probabilistic (see 3.1 and 3.2), though we show it can be made deterministic by introducing nonuniformity (see 3.3).

3.1 Probabilistic Sequential Machines

We define a probabilistic TM to be a multitape deterministic Turing Machine PM with a special read-only, one-way tape (distinct from the input and work tapes) containing an infinite binary sequence. The contents of this "random bitvector" tape are chosen randomly on each execution of PM. Let $\Sigma$ be the input alphabet of PM and let $L \subseteq \Sigma^*$. For any $\epsilon(n)$, $0 \leq \epsilon(n) < 1$ say PM recognizes $L$ within error $\epsilon(n)$ if for all $w \in \Sigma^n$,
Cl: \( w \notin L \) implies \( \text{Prob}(PM \text{ accepts } w) > 1 - \epsilon(n) \).

C2: \( w \in L \) implies \( \text{Prob}(PM \text{ accepts } w) < \epsilon(n) \).

To justify this definition, we note that Adleman's [78] definition of acceptance of probabilistic machines is similar to ours, except \( \epsilon(n) = 1/2 \) in C1 and he strengthens condition C2 by requiring that \( w \notin L \) imply PM does not accept \( w \) on any probabilistic choice. Many probabilistic algorithms in number theory satisfy this more restrictive property, but it is too restrictive for many of the applications in this paper. On the other hand, Gill [77] defines acceptance of probabilistic machines with the max of the error of acceptance and rejection less than 1/2.

Note that a probabilistic machine may not be reasonable in the sense defined in the introduction (since Gill [77] gives a probabilistic machine with space bound \( S(n) \) and expected time bound \( 2^{O(S(n))} \)); however, the probabilistic machine implementing the PROB-SEARCH algorithm of Section 3.2 will be reasonable.

3.2 Probabilistic Simulation of Space Bounded Symmetric Complementing Machines

We shall show:

**THEOREM 3.1.** For any \( \epsilon(n) \), \( 0 < \epsilon(n) < 1 \), there is a probabilistic TM which recognizes \( L \in \Sigma_{K(n)}^{L} \text{CSYMSPACE}(S(n)) \) within given error \( \epsilon(n) \) and space \( O(K(n)(S(n) + \log d(n))) \) and time \( (d(n)2^{O(S(n))})^{K(n)} \), where \( d(n) = K(n)(O(S(n)) + \log(O(K(n)))) - \log \epsilon(n) \).

Note that if \( \epsilon(n) \) is constant, then by Proposition 2.2, we require space \( O(K(n)S(n)) \) and time \( 2^{O(K(n)S(n))} \). Thus:
COROLLARY 3.1. For each constant $\varepsilon, 0 < \varepsilon < 1$, and $L \in \Sigma^* \text{SYMLOG}$, there is a probabilistic TM which recognizes $L$ within error $\varepsilon$ and simultaneous space $O(\log n)$ and time $O(1)$.

Although Theorem 3.1 suffices for our applications, we also show its space bounds can be improved.

THEOREM 3.2. For each $L \in \Sigma^* \text{SYMSPACE}(S(n))$, there is a probabilistic TM which recognizes $L$ within error $\varepsilon(n)$ and space $O(K(n)(S(n) + \log(S(n) + \log d(n))))$.

Our probabilistic search technique will utilize the following result:

LEMMA 3.1. [Aleliunas, Karp, Lipton, Lovasz, and Rackoff, 79]. Let $G = (V, E)$ be any undirected, connected graph. Let $r$ be a random walk in $G$ from any vertex $v \in V$ be constructed from trivial path $v$ by repeatedly extending the front end of $r$ by adding a random edge of $E$ which is connected to the current front end vertex of $r$. Let $r$ be a random walk of length $2|E|(|V| - 1)$. Then $\text{Prob}(r \text{ visits all vertices in } V) > 1/2$.

[Lewis and Papadimitriou, 80] observe that this Lemma immediately implies a space $O(S(n))$ probabilistic algorithm for $\text{NSYMSPACE}(S(n))$. A generalized probabilistic search technique is used here to decide acceptance of symmetric complementing machines.

We now prove here Theorems 3.1 and 3.2. Let $M$ be a symmetric complementing machine as defined in Section 2.2. We assume $M$ has complement bound $K(n) > 1$ and constructible space bound $S(n) > \log n$ (otherwise, we use the standard technique of trying $S(n) = \log n, 1 + \log n, ...$ to the
construction given below). Let \( \mathcal{F} \) be the set of configuration of \( M \) and let \( \mathcal{W} \subseteq \mathcal{F} \) be the accepting configurations. Let \( \vdash \) be the next move relation of \( M \). Let \( \vdash_{c}, \vdash_{s} \subseteq \vdash \) be the complement and noncomplement moves of \( \vdash \), respectively. Fix some \( n > 0 \). Let \( \mathcal{J}' \subseteq \mathcal{F} \) be the configurations which have \( \leq S(n) \) nonblank cells on each work tape.

We define a recursive procedure which takes as input a configuration \( I \in \mathcal{J}' \). Also, the procedure has a global variable \( t \) (which determines the procedure's probability of success).

procedure PROB-SEARCH\(_{t}(I)\)

\begin{verbatim}
begin
    local integer i, set COMP
    i+0
    while i<t do
        begin
            if I is accepting then return true
            COMP = \{I'I \in \mathcal{J}' | I \vdash_{c} I'\}
            if COMP \neq \emptyset then
                if PROB-SEARCH(I') = false for all I' \in COMP then
                    return true
                choose a random I' from \{I'I \in \mathcal{J}' | I \vdash_{s} I'\}
                I+I'
                i+i+1
            end
        return false
    end
end
\end{verbatim}

For each \( k, 1 \leq k \leq K(n) \) let \( \mathcal{F}_{k} \subseteq \mathcal{F}' \) be the configurations which also have complement bound \( k \).

Let \( \varepsilon_{k,t} \) be the max probability that \( \text{PROB-SEARCH}_{t}(I) = \text{false} \) for any \( I \in \mathcal{J}_{k} \) such that \( \text{OUTCOME}(I) = \text{true} \). Let \( \tilde{\varepsilon}_{k,t} \) be the max probability
that \( \text{PROB-SEARCH}_t(I) = \text{true} \) for any \( I \in \mathcal{S}_k \) such that \( \text{OUTCOME}(I) = \text{false} \).

Thus, \( e_{k,t} \) and \( e_{k,t}^- \) are the worst case error probabilities for rejection and acceptance, respectively.

**Lemma 3.2.** There are constants \( b, c > 1 \), dependent only on \( M \), such that for any \( d > 1, k > 1 \),

\[
\max(e_{k,t}, e_{k,t}^-) \leq k^{-d} (tb)^{k-1}
\]

where \( t = 2dS(n) \).

**Proof.** Let \( k' = (\mathcal{S} \times \mathcal{S}) \cap \mathcal{S}_s \). Since \( M \) is symmetric \( (\mathcal{S}', k') \) is an undirected graph. There exists a constant \( b > 1 \) upper bounding the number of next configurations \( \{I'|I \rightarrow I'\} \) possible from any given configuration \( I \). Also there exists a constant \( c_1 > 1 \) such that

\[
|\mathcal{S}'| \leq c_1^S(n) .
\]

Thus there exists a constant \( c_2 > 0 \) such that \( |k'| \leq c_2^S(n) \).

Let \( c = c_1 \cdot c_2 \) so \( t \geq 2d|\mathcal{S}'| |k'| \).

For each pair of configurations \( I, I' \in \mathcal{S}_k \) for which there is a non-complementing computation sequence from \( I \) to \( I' \), by Lemma 3.1, we have

\[
\text{prob}(r \text{ visits } I') \leq 1 - 2^{-d}
\]

for a random walk \( r \) in \( (\mathcal{S}, k') \) starting at \( I \) and of length \( 2d|\mathcal{S}'| |k'| \).

Now we prove Lemma 3.2 by induction on \( k \). For \( k = 1 \) we show

\[
\max(e_{1,t}, e_{1,t}^-) \leq 2^{-d}
\]

The worst case error probability for rejection and acceptance from any \( I \in \mathcal{S}_1 \) with no complement next move is \( \leq 2^{-d} \) and \( 0 \), respectively. Thus the total worst case error probability for rejection and acceptance from any \( I \in \mathcal{S}_1 \) is \( \leq 2^{-d} \) and \( \leq 2^{-d} \), respectively.

Since there at most \( tb \) direct calls to \( \text{PROB-SEARCH} \) during a single execution of the body of the \( \text{PROB-SEARCH} \) procedure, for \( k > 1 \) we have:
By the induction hypothesis,

$$\max(\varepsilon_{k-1, t}, \varepsilon_{k-1, t'}) \leq (k-1, 2^{-d}(tb))^{k-2}$$

Hence

$$\max(\varepsilon_{k, t}, \varepsilon_{k, t'}) \leq 2^{-d} + tb\varepsilon_{k-1}$$

$$\leq 2^{-d} + (k-1)2^{-d}(tb)^{k-1}$$

$$\leq k2^{-d}(tb)^{k-1}$$

Let $L$ be the language accepted by $M$. Suppose we are given some error function $\varepsilon(n)$, $0 < \varepsilon(n) < 1$. Let $PM$ be the probabilistic TM which on input $w \in \Sigma^n$, computes $\text{PROB-SEARCH}_{\tau(n)}(I_0(w))$ and accepts iff the result is true, provided that $d(n) = K(n)\log(t(n)b) - \log \varepsilon(n)$ and $t(n) = 2d(n)c^{S(n)}$. (Note that both $d(n)$ and $t(n)$ are decreasing functions of $\varepsilon(n)$.)

By Lemma 3.2, $PM$ recognizes $L$ within error $\varepsilon_{K(n)} \leq \varepsilon(n)$. Furthermore, $PM$ has time bound $O(t(n))K(n) = (d(n)2^{O(S(n))})K(n)$. $PM$ has space bound $O(K(n)(S(n) + \log d(n)))$ since we must store $b = O(1)$ configurations of size $O(S(n))$, and a "time counter" requiring space $\log t(n) = O(S(n) + \log d(n))$, to implement each of the $K(n)$ recursive cells. Thus we have proved Theorem 3.1.

Although Theorem 3.1 is good enough for our applications to $\Sigma$, CSYMLOG, it is nevertheless interesting to observe that we may decrease the space bound by using a trick due to Gill [77]. To avoid storing the "time counter" in the procedure $\text{PROB-SEARCH}_c$, we instead sample a random bit on each iteration of the while statement. If at any time there have been
log $t$ consecutive $r$'s chosen, then we immediately exit the while
statement. This test replaces the test $(i \leq t)$ in the original text of
Prob-Search. To achieve error $\varepsilon_K(n) \leq \varepsilon(n)$, we must only increase $d(n)$
by a factor of $1/(1 - \log \exp(l))$. Only $O(K(n)(S(n) + \log \log t(n))) =
O(K(n)(S(n) + \log(S(n) + \log d(n))))$ space is required by this method (but
note that we no longer have a time bound). Thus we have proved
Theorem 3.2.

3.3 Eliminating Probabilistic Choices

Let a nonuniform deterministic TM be a deterministic TM augmented
with a special read-only tape, called the advice tape, whose contents are
fixed for all inputs of the same length $n$, but which may have different
contents for distinct input lengths $n$ and $n'$. (Neither the input tape
nor the advice tape is considered in the space bound of this machine.)
This nonuniform machine has advice bound $A(n)$ if on inputs of length $n$,
the advice tape has $A(n)$ cells (see [Karp and Lipton, 80]). We will
show:

THEOREM 3.3. Each $L \in \Sigma^* \subset \Sigma^*$ is accepted by a nonuni-
form deterministic TM within space bound $O(K(n)S(n))$, time bound
$2^{O(K(n)S(n))}$, and advice bound $2^{O(S(n))}$.

COROLLARY 3.3. Each $L \in \Sigma^* \subset \Sigma^*$ is accepted by a nonuniform deter-
ministic TM within simultaneous space bound $O(\log n)$, time bound $n^{O(1)}$
and advice bound $n^{O(1)}$.

We now prove Theorem 3.3. We require a technical graph theoretic
result.
Let $G = (V, E)$ be a undirected regular graph, with valence $b$. We assume $G$ has a fixed adjacency list representation, so for each vertex $v$ we have a list $\ell(v)$ of vertices adjacent to $v$. Given a string $U \in \{1, \ldots, b\}^*$, and a vertex $v$, let $U(G, v)$ be the path $v = v_0, \ldots, v_{|U|}$ such that $v_i$ is the $U(i)$ element of list $\ell(v_{i-1})$ for $i = 2, \ldots, |U|$. Let $\mathcal{G}_{n,b}$ be the class of all undirected, regular graphs with $n$ vertices and valence $b$. Let $U \in \{1, \ldots, b\}^*$ be $(n,b)$-universal if for each graph $G \in \mathcal{G}_{n,b}$ and each vertex $v$ of $G$, $U(G, v)$ visits all the vertices of $G$.

**Lemma 3.3.** [Aleliunas, Karp, Lipton, Lovasz, and Rackoff, 1979] For each $b \geq 1$, there is a $c(b)$ such that for each $n > 0$ there is a $(n,b)$-universal string $U_{n,b}$ of length $\ll c(b)n^3 \log n$.

Let $M$ be a symmetric complementing machine of Section 3.2 with complement bound $K(n)$, and space bound $S(n)$. Let $(\mathcal{F}', \mathcal{F}_s')$ be the undirected graph of valence $b$ defined in the Proof of Lemma 3.2. Clearly we can add redundant transitions so that $(\mathcal{F}', \mathcal{F}_s')$ is regular with valence $b$. Let NONUNIFORM-SEARCH$_c(I)$ be the deterministic procedure derived from PROB-SEARCH$_c(I)$ of Section 3.2 by using the $(|\mathcal{F}'|, b)$-universal string $U_{|\mathcal{F}'|, b}$ in place of probabilistic choice, for choosing the configurations to be explored in $(\mathcal{F}', \mathcal{F}_s')$.

By the Proof of Lemma 3.2, there exists some $c_1 \geq 1$ such that $|\mathcal{F}'| \leq c_1^S(n)$. Let $t(n) = c(b)(S(n) \log c_1)c_1^3S(n)$. Then Lemma 3.3 immediately implies

**Lemma 3.4.** For each input string $w \in \Sigma^n$,

$\text{NONUNIFORM-SEARCH}_{c(t(n))}(I_0(w)) = \text{true}$ iff $M$ accepts $w$. 

This procedure may be implemented by a nonuniform deterministic \( \mathcal{T} \mathcal{M} \) with space bound \( O(K(n)S(n)) \), time bound \( t(n) = O(K(n)S(n)) \), and advice bound \( t(n) = 2^{O(S(n))} \). Thus we have proved Theorem 3.3.

4. A PARALLEL ALGORITHM

4.1 The Hardware Modification Machine

Our parallel machine model is the Hardware Modification Machine (HMM) of [Dymond and Cook, 80] (through we consider probabilistic and nonuniform variants of it below). The HMM was invented as the parallel analog of the storage modification machine of [Schönhage, 79]. The HMM seems to be the simplest possible parallel machine with modifiable storage structure, and the HMM can be simulated within real time, with the same number of processors, by many other such parallel machines, including the P-RAM of [Fortune and Wyllie, 78] (this P-RAM model was assumed for the parallel graph algorithms of [Ja'Ja' and Simon, 80]), the PRAM of [Savitch and Stimson, 79], and the SIMDAGs of [Goldschlager, 78].

Intuitively, a HMM consists of a finite collection of deterministic finite state machines which we call processors. The state transition function of these processors are identical. Each processor also contains the same fixed, finite number of input and output connections for transmission of values, from a finite alphabet, between processors. On each step (the state transitions of the processors are synchronous) a processor will read the values of its input connections which were set by its neighboring processors on the last step, and write new values on each of its output connections, (only one process is associated with each output
connection), and enter a new state. In addition, a processor may reconnect any input connection to any machine which can be reached by a path of length $\leq 2$ from the previous input connection. Also, a processor may reconnect an input connection to a new processor (with the same finite state control, initialized in some given state and with input connections directed to its creator).

Given an input string $\omega \in \Sigma^n$, we assume the initial configuration of the HMM consists of a chain of $n+1$ identical processors $PP_0, PP_1, \ldots, PP_n$ each in the same initial state, and each with input connections connected back to itself, except each $PP_{i-1}$ for $0 < i \leq n$, has a distinguished input connection to $PP_i$ where the value output by $PP_i$ is the $i$-th symbol of the input string $\omega$. (This initialization scheme is somewhat simpler than that defined by [Dymond and Cook, 80], but yields the same technical results of interest here.) The HMM accepts $\omega$ if $PP_0$ ever enters a distinguished accepting state $q_A$.

The time bound $T(n)$ (processor bound $P(n)$) of the HMM is the maximum number of steps (processors, respectively) taken on any accepting computation for any input of length $n$. Generally we assume the HMM is uniform: the processors have the same finite state transition function for all input strings. However, we consider in Section 4.4 nonuniform HMMs which must only have the same finite state transitions for all input strings of the same length. The advice bound $A(n)$ of a nonuniform HMM is the number of tuples defining the processor state transition function for input of length $n$. 
4.2 The Probabilistic HMM

In addition to the above of a uniform HMM, suppose we allow each processor $PP_i$ probabilistic choice by providing a special read-only register $r_i$ which is set randomly to 0 or 1 each step. Let $PPM$ be the resulting probabilistic HMM. $PPM$ recognizes language $L \subseteq \Sigma^*$ within error $\epsilon(n)$, $0 < \epsilon(n) < 1$, if for all $\omega \in \Sigma^n$,

\begin{align*}
C1: & \quad \omega \in L \implies \text{Prob}(PPM \text{ accepts } \omega) \geq 1 - \epsilon(n) \\
C2: & \quad \omega \notin L \implies \text{Prob}(PPM \text{ accepts } \omega) < \epsilon(n)
\end{align*}

(Note that the conditions $C1$, $C2$ for probabilistic recognition are identical to those given in 3.1. [Reif, 81] gives complexity bounds for various other probabilistic parallel machines and for both the [Adleman, 78] and also the [Gill, 77] definitions of probabilistic acceptance; if Adleman's definition of acceptance is used, then we can eliminate probabilistic choice in our parallel machines by introducing nonuniformity without any increase in parallel time; on the other hand if Gill's definition of probabilistic acceptance is used, then we show that probabilistic parallel space $S(n)$ contains parallel space $S(n)$ with nondeterministic choice.)

4.3 Parallel Simulation of Probabilistic Sequential Computations

[Dymond, and Cook, 80] prove that

THEOREM 4.1. If $L \in \text{DSPACE}(S(n))$ for $S(n) \gg \log n$, then $L$ is recognized by a (deterministic) HMM in simultaneous parallel time bound $O(S(n))$ and processor bound $2^{O(S(n))}$.

We generalize their results to probabilistic computations.
THEOREM 4.2. Let PM be a probabilistic TM with space bound \( S(n) \geq \log n \) and time bound \( T(n) \). Suppose for some \( \varepsilon(n), 0 < \varepsilon(n) < 1 \), \( L \subseteq \Sigma^* \) is recognized by PM within error \( \varepsilon(n) \). Then there is a probabilistic HMM which recognizes \( L \) within error \( \varepsilon(n) \) and with \( O(S(n) + \log T(n)) \) parallel time and utilizes \( T(n) \cdot 2^O(S(n)) \) processors. Furthermore, this HMM is uniform.

Proof. Fix some input string \( \omega \in \Sigma^n \) and let \( I_0(\omega) \) be the initial configuration of PM. Let \( \mathcal{F}' \) be the configurations of PM using \( \leq S(n) \) tape cells. Clearly there exists a constant \( c > 0 \) such that \( |\mathcal{F}'| \leq c^{S(n)} \).

We assume \( S(n) \) and \( T(n) \), are constructible (otherwise we must in parallel use a diagonalization of \( S(n) = 0,1,\ldots \), and \( T(n) = 0,1,\ldots \)).

Our simulating probabilistic HMM, which we call PPM, will utilize a processor \( P_{I,t} \) for each \( t, 0 \leq t < T(n) \) and \( I \in \mathcal{F}' \). These processors can be created in binary tree fashion within \( O(\log(T(n)|\mathcal{F}'|)) \) time.

Each processor \( P_{I,t} \) chooses a configuration \( I' \) randomly from those allowed from configuration \( I \) by PM. \( P_{I,t} \) then makes a distinguished jump connection to processor \( P_{I',t+1} \). These connections can be made in time \( O(\log(\mathcal{F}'|)) \), again using binary trees for indexing. Thereafter, each process \( P_{I,t} \) repeatedly connects its jump connection to that which was its jump connection of distance 2 in the previous step. These steps are executed synchronously by all the processes, and the HMM is allowed to halt and accept only when process \( P_{I_0(\omega),0} \) has a jump connection to a process \( P_{I,t} \) where \( I \) is an accepting configuration of PM.

Suppose \( I_0, I_1,\ldots \) is an execution sequence of M, with particular probabilistic choices \( r \). Suppose also that the RAMs of PPM make particular probabilistic choices \( r' \), such that \( P_{I,t} \) initially sets its
jump connection to process $PP_{I_{t+1}, t+1}^{t+1}$ for $t=0,1,...,T(n)-1$. Then it is easy to verify that $PPM$ accepts $\omega$ (when making probabilistic choices $r'$) iff $PM$ accepts $\omega$ (when making probabilistic choices $r$). Since $r$ and $r'$ are chosen randomly, it follows that

$$\text{Prob}(PM \text{ accepts } \omega) = \text{Prob}(PPM \text{ accepts } \omega).$$

Furthermore, if $PPM$ accepts $\omega$, then there is a path $PP_{I_{t}, t}^{I_{t+1}, t+1}$ induced by the initial jump connections such that $I_{0}(\omega) = I_{0}', I_{1}', ..., I_{t}$ is an accepting computation of $M$, and $t \leq T(n)$. On each iteration, this path's length decreases by a factor of $1/2$. Thus, $PPM$ accepts within parallel time

$$O(\log(T(n) \mid \mathcal{F} \mid)) = O(\log(T(n))2^{O(S(n))}) = O(S(n) + \log T(n)).$$

Combining Theorems 3.1 and 3.2, we have:

**THEOREM 4.3.** For any $S(n) > \log n$, $K(n) > 1$, and for each $\epsilon(n)$, $0 < \epsilon(n) < 1$ and $L \in \Sigma_{K(n)}^{CSYMSPACE}(S(n))$, there is a probabilistic HMM which recognizes $L$ within error $\epsilon(n)$, with parallel time bound $O(K(n)(S(n) + \log d(n)))$ and processor bound $2^{O(S(n))}$, where $d(n)$ is defined as in Theorem 3.1.

Note that if $\epsilon(n)$ is constant, then HMM has parallel time bound $O(K(n)S(n))$ and processor bound $2^{O(K(n)S(n))}$. Thus:

**COROLLARY 4.3.** For each constant $c$, $0 < c < 1$, and $L \in \Sigma_{c}^{CSYMLOG}$, there is a probabilistic HMM which recognizes $L$ within error $\epsilon$ and with parallel time $O(\log n)$ and $n^{O(1)}$ processor bound.
4.4 A Nonuniform Parallel Algorithm

By Theorems 4.1 and 3.3., we can eliminate probabilistic choice in our parallel algorithm, by introducing nonuniformity.

THEOREM 4.4. For each \( L \in \Sigma^K(n) \text{CSYMSPACE}(S(n)) \) with \( S(n) \gg \log n \), \( L \) is accepted by a nonuniform HMM with parallel time bound \( O(K(n)S(n)) \), processor bound \( 2^{O(K(n)S(n))} \) and advice bound \( 2^{O(S(n))} \).

COROLLARY 4.4. Each \( L \in \Sigma^* \text{CSYMLOG} \) is accepted by a nonuniform HMM within simultaneous parallel time \( O(\log n) \), processor bound \( n^{O(1)} \) and advice bound \( n^{O(1)} \).

5. COMPUTATIONAL PROBLEMS IN \( \Sigma^* \text{CSYMLOG} \)

5.1 Symmetric Complementing Games

Let \( \text{CSYMAMES}(S(n),K(n)) \) be the outcome problem for symmetric complementing games with positions of plays (starting at an initial position of size \( n \)) of length \( \ll S(n) \) complement bound \( K(n) \), log space next move relation and log space recognizer for winning positions. By our definition of complementing machines, we have:

THEOREM 5.1. For \( S(n) \gg \log n \), \( \text{CSYMAMES}(S(n),K(n)) \) is complete for the languages accepted by symmetric complementing machines with space bound \( S(n) \), and complement bound \( K(n) \).

COROLLARY 5.1. The outcome problems

\( \bigcup_{K \gg 0} \text{CSYMAMES}(\log n,k) \) are complete for \( \Sigma^* \text{CSYMLOG} \).
5.2 QBF

Given a set $X$ of boolean variables, let $\text{liters}(X) = X \cup \{-x \mid x \in X\} \cup \{\text{true}, \text{false}\}$. Let $\text{CNF} \oplus$ be the set of boolean formulas consisting of a conjunction of clauses, each clause consisting of the exclusive-or $l \oplus l'$ of two literals $l, l'$. Note that $l \oplus l'$ is equivalent to $(\neg l) \oplus (\neg l')$.

[Jones, Lien, and Laaser, 76] and [Lewis and Papadimitriou, 80] show $\text{CNF} \oplus$ unsatisfiability is complete in NSYMLOG. Let $\Sigma_0 \text{QBF} \oplus$ and $\Pi_0 \text{QBF} \oplus$ be the truth values {true, false}. Inductively, let $\Sigma_k \text{QBF} \oplus$ be the set of quantified boolean formulas $F$ of the form $(\exists X)C_1 \land \ldots \land C_m$ where $X$ is a set of boolean variables and each clause $C_i$ is of either form $l \oplus l'$ or of the form $l \lor F'$ where $l, l' \in \text{liters}(X)$ and $F'$ is a formula of $\Pi_{k-1} \text{QBF} \oplus$. Also, let $\Pi_k \text{QBF} \oplus$ be the set of quantified boolean formulas $F$ of the form $(\forall X)C_1 \lor \ldots \lor C_m$ where each clause $C_i$ is of the form $l \oplus l'$ or of the form $l \land F'$ where $l, l' \in \text{liters}(X)$ and formula $F'$ must be in $\Sigma_{k-1} \text{QBF} \oplus$. Let $\Sigma_k \text{QBF} \oplus = \bigcup_{k \geq 0} \Sigma_k \text{QBF} \oplus$.

Note that all variables are bound in $\text{QBF} \oplus$ formulas.

**THEOREM 5.2.** For all $k \geq 0$, the invalidity problem for $\Sigma_k \text{QBF} \oplus$ is complete in $\Sigma_k \text{CSYMLOG}$, and the invalidity problem for $\Pi_k \text{QBF} \oplus$ is complete in $\Pi_k \text{CSYMLOG}$.

We prove here Theorem 5.2. Our proof requires a technical Lemma. This Lemma is an easy generalization of a result of [Jones, Lien, and Laaser, 76] which characterized decidability of $\text{CNF} \oplus$ formulas.

**LEMMA 5.1.** Let $F$ be a formula of $\Sigma_k \text{QBF} \oplus$. $F$ is invalid iff there exists a sequence of literals $l_0', \ldots, l_j$ such that $l_0' \oplus \neg l_1', \ldots, l_{j-1} \oplus \neg l_j'$ are equivalent to clauses of $F$ and $l_0' = \neg l_j'$ or both (1) and (2) hold.
(1) $l_0 = \text{true}$ or $l_0 \lor F'$ is a clause of $F$ where $F'$ is an invalid formula in $\Pi_{k-1}^{\text{QBF}}$.

(2) $l_j = \text{false}$ or $\neg l_j \lor F''$ is a clause of $F$ where $F''$ is an invalid formula in $\Pi_{k-1}^{\text{QBF}}$.

It will also be useful to note that

PROPOSITION 5.1. If $F$ is a formula of $\Pi_k^{\text{QBF}}$, then $\neg F$ is equivalent to a formula $\hat{F}$ of $\Sigma_k^{\text{QBF}}$, where $\hat{F}$ is formed by switching the quantification symbols $\land, \lor$ and also switching the logical connectives $\land, \lor$ in $F$. So $F$ is invalid iff $\hat{F}$ is valid.

Proof of Theorem 5.2 by induction on $k$. $\Pi_0^{\text{QBF}}$ and $\Sigma_0^{\text{QBF}}$ can easily be shown complete in $\Pi_0^{\text{CSYMLOG}} = \Sigma_0^{\text{CSYMLOG}} = \text{DSPACE}(\log n)$.

Suppose for some $k > 1$ the Theorem holds for all $k' < k$. Let $F$ be a $\Sigma_k^{\text{QBF}}$ formula of length $n$. To decide $F$, we play a symmetric complementing game. Let the player begin by choosing a sequence of literals $l_0, \ldots, l_j$ such that $l_0 \land l_1, \ldots, l_{j-1} \land l_j$ are equivalent to clauses of $F$. Note that only the first literal and last literal need be stored, and this requires $O(\log n)$ space. This choice sequence is reversible since $(l_{j-1} \land l_j) \equiv (l_j \land l_{j-1})$. The player enters the accepting state (and thus wins) if either $l_0 = l_j$ on both cases (1), (2) of Lemma 5.2 holds. This may require deciding formulas $F'$, $F''$ of $\Pi_{k-1}^{\text{QBF}}$. To do this, we allow the player two simultaneous complement moves from the current position. In these complement moves, we let the player test if both $F'$ and $F''$ are invalid. By the induction hypothesis, these tests are in $\Pi_{k-1}^{\text{CSYMLOG}}$. Thus the symmetric complementing game can be implemented by a symmetric complementing machine with complement bound $k$ and space bound $O(\log n)$. By Lemma 5.2, the player wins iff $F$ is invalid. We have thus shown that $\Sigma_k^{\text{QBF}}$ invalidity is in $\Sigma_k^{\text{CSYMLOG}}$. 
Now let $M$ be a symmetric complementing machine with complement bound $k$ and space bound $\log n$. Let $t_s$ be the noncomplement moves of $M$. For each $k'$, $1 \leq k' \leq k$, let $S_{k'}$ be the set of configurations of $M$ with $\ll \log n$ nonblank cells per work tape and for which there is a complement bound of $k'$. Let $S_{k'}$ be the configurations of $S_{k'}$ which have a complement as a next move. By the induction hypothesis we can assume for each $I \in S_{k-1}$ a formula $F'(I)$ of $\Pi_{k-1} \Sigma \Omega$ such that $F'(I)$ is invalid iff $\text{OUTCOME}(I') = \text{false}$ for each complement move $(I, I')$ from $I$.

For each $I \in S_k$ we assume a distinct variable $x_I$. Let $X = \{x_I \mid I \in S_k\}$. Let $W'$ be configurations of $M$ which are accepting and have $\ll \log n$ nonblank cells per work tape. For each $I \in W'$, let $g_I$ be the formula $x_I \oplus \text{true}$. Thus $g_I$ is true iff $x_I$ = false. For each $I \in S_{k-1}$, let $g_I$ be the formula $(\neg x_I) \lor F'(I)$. For each $I \in S_k$ and $J \in W' \cup S_{k-1}$, let $f_{I,J} = (\text{true} \oplus x_I) \land (\land_{I \in S_k, I \neq I} x_I \oplus x_{I'}) \land g_J$ where $t_s = t_s \cap (S_k \times S_k)$. Thus $(\exists x)f_{I,J}$ is invalid iff there exists a computation sequence $I_1, \ldots, I_j$ such that $I_1 = I$, $I_j = J$ and $\text{OUTCOME}(J) = \text{true}$.

Now for each $I \in S_k$ and $J \in W' \cup S_{k-1}$ let $x_{I,J}'$ be a new distinct variable, and let $f'_{I,J}$ be derived from $f_{I,J}$ by substituting $x_{I,J}$ for each instance of variable $x_I$, for each $I \in S_k$. Let $X' = \{x_{I,J}' \mid I \in S_k$ and $J \in W' \cup S_{k-1}\}$. For each $I \in S_k$, let $F(I) = (\exists x') \land_{J \in W' \cup S_{k-1}} f'_{I,J}$. Clearly $F(I)$ is in $\Sigma_k \Sigma \Omega$ and furthermore, $F(I)$ is invalid iff $\text{OUTCOME}(I) = \text{true}$. Hence $F(I_0(w))$ is invalid iff $M$ accepts $w$.
We have thus shown that the invalidity problem for $\Sigma_k^{QBF}$ is complete in $\Sigma_k^{CSYMLOG}$. By Propositions 5.1 and 2.1, the invalidity problem for $\Pi_k^{QBF}$ is complete in $\Pi_k^{CSYMLOG}$. 

5.3 k-Connectivity

Given a graph $G = (V,E)$ and vertices $u, v \in V$, let $k$-PATHS$(G,u,v)$ be the problem: are there $k$ vertex disjoint paths from $u$ to $v$? The problem 1-PATHS is commonly called the UGAP problem.

Theorem 5.3. UGAP is complete in NSYMLOG.

Proof [Lewis and Papadimitriou, 80]. Given an undirected graph $G$ of $n$ vertices with distinguished vertices $u, v$, we nondeterministically traverse a path in $G$ from $u$, and accept if the vertex $v$ is reached. This can easily be done by a nondeterministic machine in space $O(\log n)$ to store the currently visited vertex. But this nondeterministic machine can be made symmetric since any edge can be traversed in both directions.

On the other hand, suppose $M$ is a nondeterministic symmetric machine with $\log n$ space boundedly and input string $w \in \Sigma^*$. Let $\mathcal{S}$ be the configurations of $M$, with space $< \log n$, let $T_s \subseteq \mathcal{S} \times \mathcal{S}$ be the nondeterministic moves of $M$, and let $W \subseteq \mathcal{S}$ be the accepting configurations.

We construct a undirected graph with vertices $V = \mathcal{S} \cup \{I_f\}$ where $I_f \notin \mathcal{S}$, and edges $E = \{(I,I') | (I,I') \in T_s \text{ and } I' \text{ is accepting}\}$. Then $M$ accepts $w$ if there is a path in $(V,E)$ from $I_o(w)$ to $I_f$.

By Theorems 2.1 and 5.3,

Corollary 5.3. The complement of the UGAP problem is complete in $\Pi_1^{CSYMLOG}$. 

...
THEOREM 5.4. For each $k \geq 1$, $k$-PATHS is complete in NSYMLOG.

We also show that for each $k \geq 0$, the $k$-connected components recognition problem for an undirected graph is in NSYMLOG.

By Menger's Theorem [Bondy and Murty, 77], for any graph $G = (V,E)$ and vertices $u, v \in V$, $k$-PATHS$(G, u, v) \leftrightarrow (\forall x_1, \ldots, x_{k-1} \in V - \{u, v\}) \text{UGAP}(G', u, v)$

where $G'$ is derived from $G$ by deleting vertices $x_1, \ldots, x_{k-1}$ and all edges connected to these vertices. Thus we have a deterministic log space reduction to UGAP, which by Theorem 5.3 is in NSYMLOG. Theorem 5.4 is thus proved.

Let a graph $G = (V,E)$ be $k$-connected if for all distinct vertices $u, v \in V$, there exists $k$ vertex disjoint paths from $u$ to $v$. We define a $k$-connected component of $G$ to be a maximal $k$-connected subgraph of $G$. (Note: to facilitate planarity testing, [McLane, 37] and others define "tri-connected" components somewhat differently. However McLane's components are homeomorphic (i.e., derived by replacing paths, with internal vertices of valence 2, by edges) to the 3-connected components of our definition.)

By Menger's Theorem, any two $k$-connected components intersect at no more than $k-1$ vertices. Thus some $k$ vertices suffice to uniquely determine any $k$-connected component of $G$. Let $k$-CC$(G, x; \{v_1, \ldots, v_k\})$ be the problem: is vertex $x$ in the $k$-connected component of $G$ determined by $\{v_1, \ldots, v_k\}$?

COROLLARY 5.4. $k$-CC is complete in NSYMLOG.

Proof. By Menger's Theorem,

$k$-CC$(G, x; \{v_1, \ldots, v_k\}) \leftrightarrow \bigwedge_{1 \leq i \leq k} k$-PATHS$(G, x, v_i)$

Thus we may apply Theorem 5.4 \hfill $\square$
5.4 Minimum Spanning Forests

Here we show the $\Pi_1$-CSYMLOG contains the problem of recognizing an edge of a (unique) minimum spanning forest.

Let $G = (V,E)$ be an undirected graph with a mapping $W : E \rightarrow \mathbb{N}^+$ labeling the edges with distinct positive integers. Consider the following well known greedy algorithm for constructing a minimum (weight) spanning forest of $G$:

Input graph $G = (V,E)$ and edge weighting $W$.

1. sort the edges $E = \{e_1, \ldots, e_m\}$ so that $W(e_i) < W(e_{i+1})$ for $i = 1, \ldots, m-1$.

2. $SF \leftarrow \emptyset$

3. for $i = 1$ to $m$ do
   
   if $SF \cup \{e_i\}$ contains no cycles then $SF \leftarrow SF \cup \{e_i\}$

Return $(v, SF)$.

Note that the minimum spanning forest output by this algorithm is unique for a fixed $W$ (even though in general there may exist many spanning forests of a given graph).

Let $\text{SPANNING-EDGE}(G, W, e)$ be true if $e \in SF$ and false otherwise.

THEOREM 5.5. $\text{SPANNING-EDGE}$ is complete in $\Pi_1$-CSYMLOG

Proof. Let $e = \{u,v\}$ be an edge of $G = (V,E)$ and let $G_e = (V, \{e' \in E \mid W(e') < W(e)\})$. Then $\text{SPANNING-EDGE}(G, W, e) \leftrightarrow \text{-UGAP}(G_e, u, v)$. The result then follows from Corollary 5.3.
5.5. Other Graph Recognition Problems Contained in \( \Pi_1 \)SYMLOG

Here we note that the recognition problems for many interesting and commonly found classes of graphs (including chordal graphs, comparability graphs, interval graphs, split graphs, and permutation graphs) are contained in \( \Pi_1 \)ASYMLOG. Our proofs use known characterization Lemmas.

Let \( G = (V,E) \) be an undirected graph. Let its complement be \( \overline{G} = (V,\{u,v\} \notin E \mid u,v \in V) \). We define here some graphs commonly found in the literature. Each has a characterization Lemma which immediately implies by Corollary 5.3 its recognition problem is in \( \Pi_1 \)CSYMLOG (by a deterministic logspace reduction to the complement of the UGAP problem). \( G \) is a **chordal graph** if every cycle \( C \) of length \( >3 \) contains a chord \( \{x,y\} \) (an edge connecting two nonconsecutive vertices of \( C \)).

**Lemma 5.2.** \( G \) is chordal iff for every vertex \( v \in V \) and cycle \( C \) of length \( >3 \), if \( C \) contains \( v \) then \( C \) has a chord \( \{x,y\} \) such that both \( x \) and \( y \) are of distance \( \leq 2 \) from \( v \).

**Proof.** Repeatedly apply the chordal graph definition. \( \square \)

\( G \) is a **comparability graph** if its edges may be transitively directed.

**Lemma 5.3.** [Gilmore and Hoffman, 64] \( G \) is a comparability graph iff for every cycle \( C \) of \( G \), if \( \{x,y\} \notin E \) for every pair of vertices \( x,y \) of distance 2 in \( C \), then \( C \) has an even number of edges.

\( G \) is an **interval graph** if its vertices can be put into 1-1 correspondence with a set of intervals on the real line, such that two vertices are connected by an edge of \( G \) iff their corresponding intervals have nonempty intersection.
LEMMA 5.4. [Gilmore and Hoffman, 64] G is an interval graph iff
G is a chordal graph and G is a comparability graph.

G is a split graph if its vertex set V can be partitioned into sets
V_1, V_2 such that E(V_1) = ∅ and (V_2, E(V_2)) is a complete graph.

LEMMA 5.5. [Földes and Hammer, 1977] G is a split graph iff G and
G are chordal graphs.

G = (V, E) is a permutation graph if V = {v_1, ..., v_n} and there is a permutation
σ of {1, ..., n} such that {v_i, v_j} ∈ E iff (i - j)(σ^{-1}(i) - σ^{-1}(j)) < 0.

LEMMA 5.6. [Pruei, Lempel, and Even, 71] G is a permutation graph
iff both G and G are comparability graphs.

By the above Lemmas and Corollary 5.3,

THEOREMS 5.7-11. The recognition problem for each of the graph classes:
chordal graphs, comparability graphs, interval graphs, split graphs, permutation
graphs are in \Pi_1 CSYMLOG.

G is bipartite if the vertex set V may be partitioned into disjoint
sets V_1, V_2 such that E ⊆ \{(u, v) | u ∈ V_1, v ∈ V_2}\.

LEMMA 5.7. G is bipartite iff G has no cycle of odd length.

By using this characterization Lemma, [Jones, Lien, and Laaser, 76]
show the recognition problem for nonbipartite graphs is \leq \log equivalent
to the complement of UGAP. Thus, by Corollary 5.3,
THEOREM 5.12. The bipartite graph recognition problem is complete in \( \Pi_1 \text{SYMLOG} \).

Also, [Jones, Lien, and Laaser, 76] give restricted cases of the NP-complete problems CHROMATIC NUMBER, CLIQUE COVER, EXACT COVER, HITTING SET, and show their restricted problems are \( \leq_{\text{log}} \) equivalent to UGAP, and thus complete in NSYMLOG.

5.6 Valence 3 Planarity Testing is in \( \Pi_3 \text{SYMLOG} \)

5.6.a. Embedding Rotations

Let \( G = (V, E) \) be a undirected graph with vertex set \( V \) and undirected edge set \( E \subseteq \{ (u, v) | \text{distinct } u, v \in V \} \). Let \( D(E) = \{ (u, v) | (u, v) \in E \} \cup \{ (v, u) | (u, v) \in E \} \) be the set obtained by directing edges of \( E \). Following [Edmonds, 60] (also see [White, 73]) we define an embedding onto an oriented surface purely combinatorially; let an embedding rotation be a set \( \theta = \{ \theta_v | v \in V \} \) where \( \theta_v \) is a cyclic permutation of the directed edges \( D_v(E) = \{ (x, y) \in D(E) | x = v \} \) of \( D(E) \) departing from vertex \( v \). Intuitively, \( \theta_v \) gives the clockwise rotation of edges as they are embedded around vertex \( v \), in a graph with a planar embedding.

Let \( R(\theta) \subseteq D(E) \times D(E) \) be the relation such that \( e_1 R(\theta) e_2 \) iff directed edge \( e_1 \) departs from the same vertex \( v \) that directed edge \( e_2 \) departs from, and \( e_2 \) appears immediately after \( e_1 \) in \( \theta_v \) (where \( e_1 \) is the reverse of edge \( e_1 \)). See Figure 1.
R(θ) is easily shown to be partitioned into a set of cyclic permutations. Let c be the number of 1-connected components of G. Let λ(θ) be the number of orbits of R(θ). By Euler's formula, the embedding rotation θ is planar if λ(θ) - |E| + |V| = 2c. Intuitively, if θ is planar and c = 1 then the orbits of R(θ) are in 1-1 correspondence with the connected regions of the embedding, with the borders of the regions oriented counterclockwise. If θ is planar and c > 1, then c orbits of R will be associated with the exterior regions of the embedding.

By Edmonds' [60] characterization, the graph G is planar iff it has a planar embedding rotation.

5.6.b. The Basis Cycles

Let us impose an arbitrary, fixed numbering of the edges E from 1, ..., |E|. We consider this numbering to be an edge weighting of the 2-connected graph G = (V,E). The greedy algorithm of Section 5.4 constructs a unique minimum spanning forest (V,SF_G). For each directed edge (u,v),
where \((u,v) \in E - SF_G\), there is a unique directed simple cycle \(C_{(u,v)}\) containing \((u,v)\) and directed edges derived from \(SF_G\). We call \(C_{(u,v)}\) a basis cycle. Let \(\mathcal{G}\) be the set of all basis cycles. By applying Theorem 5.5 we have:

**Lemma 5.8.** \(\Pi_1\text{SYMLOG}\) contains the test: Is edge sequence \(C\) in \(\mathcal{G}\)?

### 5.6.c. The Bridges of \(G\)

Let \(C\) be a basis cycle in \(\mathcal{G}\) of graph \(G = (V,E)\). Let a bridge of \(C\) be a maximal edge set \(B \subseteq E - \{(u,v) | (u,v) \in C\}\) such that \(\forall e_1, e_2 \in B\), there is a path \(p\) in \(B\) containing both \(e_1\) and \(e_2\), but visiting vertices of \(C\) only at the endpoints of \(p\). Given \(C\), we can construct in deterministic log-space a graph (this graph is derived from \(G\) by substituting a distinct new vertex \(v_e\) and edge \((u,v_e)\) for each edge \(e \in (u,v)\) of \(E\) such that a vertex \(v\) appears in \(C\)) whose \(1\)-connected components are in \(1\)-\(1\) correspondence with the bridges of \(C\). Thus, by Lemma 5.8 and Corollary 5.4, \n
**Lemma 5.9.** The bridges of all the cycles of the basic \(\mathcal{G}\) may be recognized in \(\Sigma_2\text{SYMLOG}\).

### 5.6.d. Embedding Formulas for Graphs of Valence 3

Suppose we have an oracle for \(\Sigma_2\text{SYMLOG}\). Given this oracle, we provide a deterministic log space construction of a \(\text{CNF}\oplus\) formula \(F_G\) which will encode embedding rotations of graph \(G = (V,E)\) of valence 3.

For each vertex \(v \in V\) and directed edges \(e_1, e_2, e_3 \in D_v(E)\) departing from \(v\), we have a distinct boolean variable \(O_v(e_1, e_2, e_3)\). Let \(h_1\) be the \(\text{CNF}\oplus\) formula:
Note that $h_1$ holds just for truth assignments to $O_v$ where there exists an embedding rotation $\theta = \{\theta_v \mid v \in V\}$ where $\forall v \in V$ and $e_1, e_2, e_3 \in D_v(E)$ departing from $v$, $O_v(e_1, e_2, e_3) = \text{true}$ iff $e_2$ is cyclically ordered between $e_1$ and $e_3$ in $\theta_v$.

Also, for each undirected edge $e \in E$ and basis cycle $C \in \mathcal{C}$ not containing $e$, we have a boolean variable $\text{OUT}(C, e)$. (Intuitively, this variable will be true iff $e$ is embedded into the exterior of the closed region of the sphere defined by $C$.)

Let $h_2$ be the CNF formula:

$$\bigwedge_{C \in \mathcal{C}} (O_v((v, u_1), (v, u_2), (v, u_3) \oplus \gamma \text{OUT}(C, \{v, u_3\})$$

$$\text{with } (u_1, v), (v, u_2) \in C$$

$$\text{and } \{v, u_3\} \in E$$

$$\text{distinct } v, u_1, u_2, u_3 \in V$$

Note that $h_1 \land h_2$ holds just when an embedding rotation $\theta$ induced such that for each $v \in V$ and basis cycle $C \in \mathcal{C}$ containing vertex $v$, if $(u_1, v), (v, u_2)$ are directed edges of $C$ and $\{v, u_3\}$ is any other edge containing $v$, then $(v, u_3)$ is embedded to the exterior of the region defined by $C$ if $\theta_v$ cyclically orders $(v, u_2)$ between $(v, u_1)$ and $(v, u_3)$ (see Figure 2.)

---

\[ \land_{\nu \in V} (O_v(e_1, e_2, e_3) \oplus O_v(e_3, e_2, e_1)) \land (O_v(e_1, e_2, e_3) \oplus O_v(e_2, e_3, e_1)) \\
\land_{e_1, e_2, e_3 \in D_v(E)} \text{CNF formula:} \\
O_v((v, u_1), (v, u_2), (v, u_3) \oplus \text{OUT}(C, \{v, u_3\}))$
By Lemma 5.8, \( h_2 \) can be constructed in deterministic log-space, given a \( \Pi_1 \text{CSYMLOG} \) oracle. Finally, let \( h_3 \) be the formula

\[
\wedge_{C \in \mathcal{C}} \bigwedge_{e_1, e_2 \in \mathcal{E}_E} \bigwedge_{e_1, e_2 \in \mathcal{E}_E} \text{OUT}(C, e_1) \oplus \text{OUT}(C, e_2)
\]

\( e_1, e_2 \) are in the same bridge of \( C \)

Note that \( h_3 \) holds just when for each bridge \( B \) of any cycle \( C \in \mathcal{C} \), either all edges of \( B \) are embedded interior to the region defined by \( C \), or all edges of \( B \) are embedded exterior to the region defined by \( C \). By Lemmas 5.8 and 5.9, \( h_3 \) may be constructed in deterministic log-space, given an \( \Sigma_2 \text{CSYMLOG} \) oracle.

Let \( H_G = h_1 \wedge h_2 \wedge h_3 \).

**Lemma 5.10.** A graph \( G \) of valence 3 is planar iff \( H_G \) is satisfiable.

**Proof.** Suppose \( G \) is planar. Then by Edmond's characterization, \( G \) has some planar embedding orientation \( \theta \). We use \( \theta \) to define the truth values for the \( O_v \) and \( \text{OUT} \) variables. They clearly satisfy \( h_1 \), \( h_2 \), and \( h_3 \). Thus we can satisfy \( H_G \).

On the other hand, suppose \( H_G \) is satisfiable. Let \( \theta \) be the embedding rotation induced by the \( O_v \) variables. Let \( \lambda(\theta) \) be the number of orbits of \( R(\theta) \). Now we must show \( \lambda(\theta) - |E| + |V| = 2 \) implying by Euler's formula \( \theta \) is planar. Fix the spanning forest \( SF = SF_G \). Suppose we delete
an edge $e \in E - SF$ from $G$, so that the resulting graph is $G' = (V, E')$ with $E' = E - \{e\}$. We claim that if $\theta'$ is the resulting embedding rotation, then $\lambda(\theta) - |E| + |V| = \lambda(\theta') - |E'| + |V|$. Clearly $|E'| = |E| - 1$ so we must show $\lambda(\theta') = \lambda(\theta) - 1$.

Let $e = \{u, v\}$ be the undirected edge to be deleted from $E - SF$. Let $\pi_1, \pi_2$ be the orbits of $R(\theta)$ containing directed edges $d_1 = (u, v)$, $d_2 = (v, u)$ respectively, so $d_1^- = d_2$ and $d_2^- = d_1$. We claim now that $\pi_1 \neq \pi_2$.

For each $i \in \{1, 2\}$, let $C_{d_i} \in \mathcal{C}$ be the unique basis cycle which contains directed edge $d_i$. If $\pi_i$ is a cyclic permutation of $C_{d_i}$ then $C_{d_i}$ and $\pi_i$ do not contain the reverse edge $d_i^-$, so $\pi_1 \neq \pi_2$.

Otherwise, for each $i = 1, 2$ let $(x_i, y_i)$ be the first edge of $\pi_i$ following $d_i$ such that $(x_i, y_i)$ is not in $C_{d_i}$. Let $f_i$ be the directed edge of $\pi_i$ immediately preceding $(x_i, y_i)$. By definition of orbits, $(x_i, y_i)$ immediately follows $f_i$ in $\theta x_i$. Note also that $f_i$ must be a directed edge in $C_{d_i}$. Let $g_i$ be the directed edge of $\pi_i$ immediately following $f_i$. Then $(x_i, y_i)$ is cyclicly ordered between $f_i$ and $g_i$ in $\theta x_i$. Since $h_2$ is satisfied, $\text{OUT}(C_{d_i}, \{x_i, y_i\}) = \text{false}$ for $i = 1, 2$. But since $C_{d_2}$ is the reverse of directed cycle $C_{d_1}$, $g_2^-$ appears just before $f_2^-$ in $C_{d_1}$ and so $\text{OUT}(C_{d_1}, \{x_2, y_2\}) = \text{true}$.

However, we claim $\text{OUT}(C_{d_1}, \{x, y\})$ is false for all $(x, y) \in E$ such that $(x, y)$ is contained in $\pi_1$ but not $C_{d_1}$. This has already been proved for $x = x_1$ and $y = y_1$. For any subsequence $\pi'$ of $\pi_1$ contained on only a single bridge of $C_{d_1}$, since $h_3$ is satisfied we have $\text{OUT}(C_{d_1}, \{x, y\}) = \text{OUT}(C_{d_1}, \{x', y'\})$ for all $(x, y)$ and $(x', y')$ in $\pi'$.

Also, if $(x, y)\pi''(x', y')$ is a subsequence of $\pi_1$ where $\pi''$ is a
is a subsequence of $C_{d_1}$ but $(x,y)$ and $(x',y')$ do not appear in $C_{d_1}$, then since $h_2$ is satisfied we have $\text{OUT}(C_{d_1}, \{x,y\}) = \text{OUT}(C_{d_1}, \{x',y'\})$.

If $(x_2, y_2)$ is in $\pi_1$, then we have just shown $\text{OUT}(C_{d_1}, \{x_2, y_2\}) = \text{false}$, contradicting our previous proof then $\text{OUT}(C_{d_1}, \{x_2, y_2\}) = \text{true}$. Hence $(x_2, y_2)$ is not in $\pi_1$ but is by assumption in $\pi_2$, so $\pi_1 \neq \pi_2$.

Without loss of generality, let $\pi_1 = d_1 \pi_1'$ and $\pi_2 = d_2 \pi_2'$ where neither $\pi_1'$ nor $\pi_2'$ contain $d_1$ or $d_2$. Hence when $e$ is deleted from $E - SF$, the orbits $\pi_1$ and $\pi_2$ of $R(\theta)$ are merged into a single new orbit $\pi_1' \pi_2'$ of $R(\theta')$, and no other orbits are modified.

Thus $\lambda(\theta') = \lambda(\theta) - 1$, and we have shown $\lambda(\theta) - |E| + |V|$ remains invariant. We repeat this process (deleting edges not in $SF$) until we have only $E' = SF$. Let $c$ be the number of maximal trees in the spanning forest $SF$, $c$ is also a number of 1-connected components of $SF$. Then $\lambda(\theta') = c$, $|SF| = n - c$ and $|V| = n$. Hence our invariant is $\lambda(\theta) - |E| + |V| = \lambda(\theta') - |SF| + |V| = 2c$, which implies by Euler’s formula the embedding orientation $\theta$ is planar. Thus, by Edmonds characterization, $G$ is planar.

We have by applying Theorem 5.2 and Lemma 5.10,

THEOREM 5.13. Planarity testing of valence 3 graphs is in $\Pi_3 \text{SYMLOG}$. 


6. CONCLUSION

It may be significant that the symmetric complementing machine introduced in this paper has applications to many combinatorial problems found in practice, such as spanning trees, k-connectivity and planarity testing. In this case the theoretical study of a new machine type led us to the discovery of new techniques for practical combinatorial algorithm design. For example, applying our probabilistic decision algorithm for symmetric games to our proof that valence 3 planarity testing is in $\Pi_3^{CSYMLOG}$, yields a new probabilistic algorithm for planarity testing; this algorithm has a quite different structure than any previously known deterministic planarity testing algorithm such as that of [Hopcroft and Tarjan, 74]. This indicates to us that the field of combinatorial algorithms, as well as the field of abstract computational complexity would benefit by further study of unusual machine types and their decision algorithms.

Preliminary Work and Acknowledgments

In a preliminary draft of this paper, we utilized a restricted type of alternating machine whose nonalternation next moves are a symmetric relation. Dexter Kozen pointed out to us that such symmetric alternation machines are too restricted for our intended applications (in particular, thy do not satisfy a complementation property such as Proposition 2.1).

A succeeding draft of this paper generalized this machine to allow nonalternation moves to be essentially a cross-product of a symmetric and a deterministic relation. Joe Halpern informed us of certain difficulties of this generalization.
The symmetric complementing machine described in this current draft satisfies the required complementation property of Proposition 2.1 and also has efficient decision algorithms if the machine is both space and complementation bounded. Michael Sipser has also suggested an equivalent machine; this is a symmetric alternating machine \( M \) as we originally defined (with symmetric nonalternation moves), but with a modified definition of acceptance:

\[ M \text{ accepts (rejects, respectively) from an existential (universal, respectively) configuration } I \text{ if there exists a finite computation sequence } I = I_1, \ldots, I_{j-1}, I_j \text{ where } I_1, \ldots, I_{j-1} \text{ are existential (universal, respectively) and } I \text{ contains an accepting (rejecting, respectively) state or } I_j \text{ is universal (existential, respectively) and } M \text{ accepts (rejects, respectively) from } I_j. \text{ Also, } M \text{ rejects (accepts, respectively) from existential (universal, respectively) configuration } I \text{ iff } M \text{ does not accept (does not reject) from } I. \text{ Thus Sipser's definitions for acceptance and rejection of these machines are duals. This is the same as the standard definition of acceptance of an alternating machine from an existential configuration, but differs from the standard definition of acceptance from a universal configuration so as to allow for complementations of languages.} \]

The author wishes to thank Larry Denenberg, Vassos Hadzilacos, Joe Halpern, Harry Lewis, A. Prasad, Michael Sipser, and Paul Spirakis for a careful reading and many useful comments on preliminary drafts of this paper.
REFERENCES


