4. The Round-Robin Tournament Method

$$K \in V/F \Rightarrow K \in S'$$

where the set of strings K is complete in L with respect to an appropriate reducibility. The hypothesis tells us that K is of the form $S : h$ where S is a language in V and a bound on $|h(|x|)|$ is known. The proof that $K \in S'$ consists of giving an appropriate uniform algorithm to recognize K. The function $h(|x|)$ is not available to this uniform algorithm, but the algorithm can exploit the fact that $h(|x|)$ is consistent; i.e. for all strings y of the same length as x, $y \in K \Leftrightarrow h(|x|) \cdot y \in S$. The algorithm must somehow filter through all the strings that might be $h(|x|)$, and come up with the right decision about x. The method of doing so depends on the structure of K. The following section treats the case where K is a "game". Section 5 considers the case where K is self-reducible. Finally, Section 6 deals with the case where K has a simple recursive definition.

The main results of this paper are summarized in Figure 1. The rest of the paper is devoted to supplying proofs and additional comments on these main results. As promised in the introduction each result demonstrates that a nonuniform hypothesis can have uniform consequences.

4. The Round-Robin Tournament Method

Insight into the nature of a complexity class can often be gained by identifying "hardest" problems in the class, i.e., problems that are complete in the class with respect to an appropriate definition of reducibility. For complexity classes defined in terms of time and space on alternating Turing machines, these complete problems often take the form of games ([3, 4]). In this section we explain and apply a proof technique called "the round-robin tournament method", which enables us to relate the nonuniform complexity of a game to its uniform complexity. The specific complexity classes we consider are *PSPACE*, *P* and *EXPTIME* (alias *AP*, *ASPACE* (*log n*) and *APSPACE*, respectively ([3, 10])).

A *game G* is specified by

(i) a set $W \subseteq \{0, 1\}^*$ and

(ii) a pair of length-preserving functions F_0 and F_1, each mapping $\{0, 1\}^* - W$ into $\{0, 1\}^*$.

There is a straightforward interpretation of this structure as a game of perfect information. Each string $x \in \{0, 1\}^*$ is a possible position in the game. Starting in an initial position, the players move alternately until a position in W is reached. When a player is to move in position x, he may move either to $F_0(x)$ or to $F_1(x)$. When a position in W is reached, the player to move is declared the winner. Note that all the positions arising in a single play of the game have the same length.

We further require that our games be *terminating*; i.e.,

(iii) there is no sequence of moves leading from a position x back to itself.

Given a game G, let G denote the set of positions from which the first player can force a win. The set G is specified recursively by

$$\mathrm{G} = \mathrm{W} \cup \{x \mid F_0(x) \notin \mathrm{G}\} \cup \{x \mid F_1(x) \notin \mathrm{G}\} .$$

This specification of G suggests the following method of selecting an optimal move in any position $x \notin \mathrm{W}$: move to $F_0(x)$ if $F_0(x) \notin \mathrm{G}$; otherwise move to $F_1(x)$. If $x \in \mathrm{G}$, then this method of move selection will force a win against any choice of moves by the opponent.

Let us now apply nonuniform complexity to games. Suppose $\mathrm{G} = \mathrm{S} : h$, where $\mathrm{S} \subseteq \{0, 1\}^*$ and h is a function from N into $\{0, 1\}^*$. Then

$$x \in \mathrm{G} \Leftrightarrow h(|x|) \cdot x \in \mathrm{S} .$$

The optimal move selection rule can be restated as follows:

> in any position $x \notin \mathrm{W}$, move to $F_0(x)$ if $h(|x|) \cdot F_0(x) \notin \mathrm{S}$, and otherwise to $F_1(x)$.

We would like to consider situations in which $\mathrm{G} = \mathrm{S} : h$, but $h(|x|)$ is not known. If we guess that $h(|x|) = w$, then the following move selection rule is indicated:

> in any position $x \notin \mathrm{W}$,
>
> if $w \cdot x \notin \mathrm{S}$, then move to $F_0(x)$,
>
> else move to $F_1(x)$.
>
> Call this rule *Strat* (w).

Given strings w, w' and x, the predicate *Win* (w, w', x) is defined as follows: play out position x with the first player choosing his moves according to *Strat* (w), and the second player using *Strat* (w'); *Win* (w, w', x) is true if the first player wins.

The following easy lemma is the basis of the round-robin tournament proof technique.

LEMMA 4.1. Let G be a game, G, the associated set of strings, S a subset of $\{0, 1\}^*$ and h a function from N to $\{0, 1\}^*$, such that $G = S : h$. Let w and w' range over some set of strings $T(x)$ which includes $h(|x|)$. Then the following are equivalent:

(1) $x \in G$

(2) $\exists\, w \,\forall\, w'\ Win\ (w, w', x)$

(3) $\forall\, w' \,\exists w\ Win\ (w, w', x)$.

Proof. If $x \in G$ then the sentence $\forall\, w'\ (Win\ (h(|x|), w', x)$ is true. Hence (2) and (3) are true. If $x \notin G$ then, for all w, $Win\ (w, h(|x|), x)$ is false; hence (2) and (3) are false. ∎

Lemma 4.1 suggests how to decide if $x \in G$ when $h(|x|)$ is not known but a set $T(x)$ containing $h(|x|)$ is known. Simply play a round-robin tournament among the strategies associated with all the strings in $T(x)$, starting each game in position x. Then $x \in G$ if and only if some strategy emerges undefeated. A subtle point is that the round-robin tournament method determines whether $x \in G$ without necessarily identifying $h(|x|)$.

To prepare for the applications of the round-robin tournament method, we assert the existence of games with certain properties.

Fact 1. There is a game G such that the associated set G is complete in *EXPTIME* with respect to many-one polynomial-time reducibility. Moreover, the set W is in P, and the functions F_0 and F_1 are computable in polynomial time.

Fact 2. There is a game G such that G is complete in *PSPACE* with respect to many-one polynomial-time reducibility. For this game, the set W is in P, and the functions F_0 and F_1 are computable in polynomial time. Moreover, there is a polynomial $p(\cdot)$ such that, for every position x, every play of G starting at x terminates within $p(|x|)$ moves.

Fact 3. There is a game G such that G is complete in P with respect to many-one logspace reducibility. For this game W is in logspace and the functions F_0 and F_1 are computable in logspace. Moreover, each position $x \notin W$ consists of the concatenation of a *fixed part* x_1 with a *variable part* x_2, such that $F_0(x)$ and $F_1(x)$ have the same fixed part as x does. Also, if $|x_1| = n$, then $|x_2| = f(n)$, where $f(n)$ is a nondecreasing function which is $\leqslant 3 \log_2 n$.

Facts 1 and 3 may be derived by simple modifications and encodings of games described in [3]. One of the modifications is to encode into each non-terminal position a "clock" which is decremented at each move; this is done to ensure termination. Similarly, a game of the type referred to in Fact 2 can be derived from any of several *PSPACE*-complete games derived in [17].

We are now ready to give the main theorems of this section.

THEOREM 4.2. If $PSPACE \subseteq P/poly$ then $PSPACE = \sum_2^p \cap \prod_2^p$.

Proof. Since $\sum_2^p \cap \prod_2^p \subseteq PSPACE$, it suffices to prove

$$PSPACE \subseteq P/poly \Rightarrow PSPACE \subseteq \sum_2^p \cap \prod_2^p .$$

For this it is sufficient to show

$$G \in P/poly \Rightarrow G \in \sum_2^p \cap \prod_2^p$$

where G is the *PSPACE*-complete set described in Fact 2. Suppose $G \in P/poly$. Then there is a set $S \in P$, a positive constant k, and a function $h: N \to \{0, 1\}^*$ such that $|h(n)| \leqslant k + n^k$, so that $G = S:h$. By lemma 4.1,

$$x \in G \Leftrightarrow \exists w \, \forall \, w' \; Win \; (w, w', x) .$$

Here each of w and w' ranges over all strings of length $\leqslant k + |x|^k$. Since F_0 and F_1 are polynomial-time computable, W is polynomial-time recognizable and play from x terminates within $p(|x|)$ moves, the predicate $Win\ (w, w', x)$ is computable in polynomial-time. Thus $G \in \sum_2^p$. Similarly, since

$$x \in G \Leftrightarrow \forall w' \, \exists \, w \; Win \; (w, w', x) .$$

it follows that $G \in \prod_2^p$. ∎

THEOREM 4.3. $PSPACE \subseteq P/log \Leftrightarrow PSPACE = P$.

Proof. Since $P \subseteq PSPACE$, and since $PSPACE = P$ implies $PSPACE \subseteq P/log$, it suffices to prove $PSPACE \subseteq P/log \Rightarrow PSPACE \subseteq P$. For this it suffices to show $G \in P/log \Rightarrow G \in P$, where G is the *PSPACE*-complete set described in Fact 2. Again, the round-robin tournament method yields the proof. Suppose $G \in P/log$. Then there is a set $S \in P$, a positive constant k, and a function $h: N \to \{0, 1\}^*$ such that $h(n) \leqslant k \log_2 n$ so that $G = S:h$. Then $x \in G \Leftrightarrow \exists w \, \forall w' \; Win \; (w, w', x)$, where w and w' range over the $0\,(|x|)^k$ strings of length $\leqslant k \log_2 |x|$. Since $Win\ (w, w', x)$

can be computed in polynomial time, we can decide in polynomial time whether $x \in G$ by actually enumerating the polynomially-many pairs (w, w'), computing $Win\ (w, w', x)$ for each pair, and determining whether some strategy $Strat\ (w)$ indeed wins from x against all the competing strategies. Thus $G \in P$. ■

THEOREM 4.4. $EXPTIME \subseteq PSPACE/poly \Leftrightarrow EXPTIME = PSPACE$.

Proof. The proof is almost a carbon copy of the proof of theorem 4.3. It suffices to show that

$$G \in PSPACE / poly \Rightarrow G \in PSPACE\ ,$$

where G is the game referred to in Fact 1. Suppose $G \in PSPACE/poly$. Then $G = S : h$, where $S \in PSPACE$ and $\left| h\ (|x|) \right| \leqslant k + \left| x \right|^k$, for some k. Then

$$x \in G \Leftrightarrow \exists w\, \forall w'\ Win\ (w, w', x)$$

where w and w' range over all strings of length $\leqslant k + \left| x \right|^k$. Since W, F_0 and F_1 are computable in polynomial space, it suffices to play out the game from x, alternately using $Strat\ (w)$ and $Strat\ (w')$ for move selection; this simulation requires repeated calls on the polynomial-space recognizer for S. Thus the truth of the formula

$$\exists w\, \forall w'\ Win\ (w, w', x)$$

can be decided in polynomial space by simply running through the pairs (w, w'), and evaluating $Win\ (w, w', x)$ for each pair. It follows that $G \in PSPACE$. ■

The last in our clone of four theorems proved by the round-robin tournament method is the following.

THEOREM 4.5. For any positive integer l,

$$P \subseteq DSPACE\ ((log\ n)^l) / log\ n \Leftrightarrow P \subseteq DSPACE\ ((log\ n)^l)\ .$$

Proof. It suffices to prove

$$G \in DSPACE\ ((log\ n)^l (/log) \Rightarrow G \in DSPACE\ ((log\ n)^l)\ ,$$

where G is the set described in Fact 3. Suppose

$$G \in DSPACE\ ((log\ n))^l / log\ .$$

Then $G = S : h$ where $S \in DSPACE((\log n)^l)$ and $|h(x)| \leqslant k \log_2 |x|$, for some k. Then $x \in G \Leftrightarrow \exists w \, \forall w' \, Win(w, w', x)$, where w and w' range over all strings of length $\leqslant k \log_2 |x|$. Clearly space $0((\log n)^l)$ suffices to deterministically enumerate all pairs (w, w') and, for each, to play out $Strat(w)$ against $Strat(w')$ from position x, with the help of repeated calls on a deterministic space $(\log n)^l$ recognizer for S. It follows that

$$G \in DSPACE((\log n)^l) . \qquad \blacksquare$$

5. The Self-Reducibility Method

The "hardest" problems in complexity classes defined by bounds on nondeterministic time or space often possess a structural property called *self-reducibility*. Various formal definitions of self-reducibility can be found in the literature ([12, 18, 20]). Here is one version of the idea. Let K be a subset of $\{0, 1\}^*$. A *self-reducibility structure* for K is specified by a partial ordering $<$ of $\{0, 1\}^*$ such that

(i) A, the set of minimal elements in $<$, is recursive and

(ii) $A \cap K$ is recursive

together with a pair of computable functions G_0 and G_1 mapping $\{0, 1\}^* - A$ into $\{0, 1\}^*$, such that, for all $x \in \{0, 1\}^* - A$,

(iii) $G_0(x) < x$, $G_1(x) < x$, $|G_0(x)| = |G_1(x)| = |x|$ and $x \in K \Leftrightarrow G_0(x) \in K$ *or* $G_1(x) \in K$.

If K has a self-reducibility structure, then K is called *self-reducible.*

To illustrate the concept, we give self-reducibility structures for two important examples. The first example is the satisfiability problem for propositional formulas, encoded so that the following property holds: Let $F(t_1, t_2, ..., t_n)$ be a formula in which the variables $t_1, t_1, ..., t_n$ appear, and let $F(a, t_2, ..., t_n)$ be the same formula with the Boolean constant a substituted for t_1. Let $< F(t_1, t_2, ..., t_n) >$ and $< F(a, t_2, ..., t_n) >$ denote the encodings of these two formulas as strings. Then

$$| < F(t_1, t_2, ..., t_n) > | = | < F(a, t_2, ..., t_n) > | .$$

Let SAT denote this version of the satisfiability problem. The set SAT has a self-reducibility structure in which A is the set of propositional formulas containing no variables,