Introduction

The class IP consists of every language L for which there exists a polynomial-time verifier which can be convinced, through a dialog with a computationally unbounded prover, to accept any member of L with overwhelming probability. The verifier always accepts any $x \in L$ if the prover honestly follows the proof system's protocol. The argument is "convincing" because the protocol is designed such that the prover, despite having infinite computational power, has only a negligible chance of being able to give a false proof.

This paper summarizes the results of Adi Shamir's 1992 article which proves that IP and PSPACE are equivalent complexity classes. By providing a reduction which constructs an IP system for a PSPACE complete problem, Shamir demonstrates that PSPACE $\subseteq$ IP. This means that any language accepted by a polynomial space turing machine can be accepted with high probability by a verifier in an IP system. With IP $\subseteq$ PSPACE granted as an known containment, the conclusion is that IP = PSPACE.

Overview

The proof is a reduction from TQBF, the PSPACE complete problem of deciding the truth of a quantified boolean formula. Since any problem in PSPACE can be reduced to TQBF, this reduction proves that any PSPACE problem transitively reduces to IP, meaning that PSPACE is contained in IP.

TQBF is first rephrased as an arithmetic decision problem. The QBF is quickly transformed into an arithmetic expression whose value A is 0 iff the boolean formula is unsatisfiable. Although the magnitude of A is $O(2^{2^n})$, too large for a polynomial machine to compute, an unbounded prover can convince a verifier that the expression's evaluation is nonzero by reducing A modulo a polynomially long prime p such that $A \neq 0 \pmod{p}$.

The protocol then specifies a method of repeatedly shortening A to A' and recalculating the value of $A' \pmod{p}$. On each step, the number of variables in A' decreases by one, and the verifier chooses a random number for the prover to use in its computation. The verifier's participation is what

keeps the prover honest, and the chance of catching a false proof expands with each iteration. At the end of the process, A' is a constant expression, and the verifier can easily check whether this value is correct.

QBF Arithmetization

TQBF is transformed into an arithmetic decision problem by expressing a quantified boolean formula *B* as an arithmetic expression *A* such that B is true if and only if *A* is nonzero. This is done using a simple replacement strategy, wherein each boolean variable x_i is represented by an integer z_i, and a value of 0 or 1 for z_i is equivalent to a false or true assignment to x_i.

The variables in the formula are transformed by replacing x_i with z_i. A negated variable $\overline{x_i}$ is replaced with $1 - z_i$, which evaluates to 0 if x_i is true, and to 1 if x_i is false.

Operators are replaced with arithmetic operations. The evaluation of the arithmetic form of $(x_1 \wedge x_2)$ should be nonzero iff both x_1 and x_1 are true (if either z_1 or z_2 is 0, then the arithmetic expression should evaluate to 0). This behavior is achieved by replacing the boolean $\wedge$ operator with integer multiplication. Likewise, $\vee$ is replaced with integer addition, so that the arithmetization of $(x_1 \vee x_2)$ is nonzero iff either z_1 or z_2 is nonzero.

Quantifiers are similarly equated to multiplication and addition. A universal quantifier $\forall x_i$ is represented by $\prod_{z_i \in \{0,1\}}$ so that a nonzero result requires the subexpression to be nonzero for both possible assignments to x_i. An existential quantifier $\exists x_i$ is represented by $\sum_{z_i \in \{\cdot, 1\}}$, so a nonzero evaluation for either x_i = true or x_i = false is sufficient to yield a nonzero evaluation.

Since the structure of this arithmetization is recursive, its correctness can be proved inductively.

The benefit of giving TQBF this structure is that the arithmetic form can be evaluated to some constant. Once an all-powerful prover calculates this number, it can convince the verifier that it is nonzero.

Magnitude of Arithmetic Form

In order to show that there exists a polynomially long prime p which is not a factor of A, the evaluation of an arithmetized QBF B, it is first necessary to bound the size of A. This is done by recursively examining subexpressions B' of B, and considering their maximum value $v(B')$ over any unquantified variables. If B' is fully quantified, v(B') is equal to the evaluation of its arithmetic form. For open variables, the value of $v(B')$ is, as determined by the arithmetization process:

١	if B' is x_i or $\overline{x_i}$
$v(B'')+v(B''')$	if B' is $B''\vee B'''$
$v(B'')\cdot v(B''')$	if B' is $B''\wedge B'''$
٢v(B'')	if B' is $\exists x_i B''$
$v(B'')^2$	if B' is $\forall x_i B''$

At worst, the value is squared by universal quantifiers. Successive squaring over n nested subexpressions gives v$(B) = A$ an upper bound of $O(2^{٢^n})$. Under this constaint, it can be proven that there exists a polynomial length p such that $A \neq 0 \pmod p$ iff A is nonzero, which can be given by the prover to a polynomial time verifier.

The Interactive Proof

The protocol for the interactive proof, for an arithmetized QBF A, begins with the prover sending $a = A \pmod p$. The prover then converts A into a *functional form* by removing the first $\sum_{z_i\in\{٠,١\}}$ or $\prod_{z_i\in\{٠,١\}}$ and creating a polynomial function $q(z_i)$. The infinitely powerful prover can simplify this polynomial and send it to the verifier, making the assertion:

$q(0)+q(1)=a$	if a $\sum$ symbol was removed, or
$q(0)\cdot q(1)=a$	if a $\prod$ symbol was removed.

After the verifier checks this, it picks a random prime value r for z_i, sends it to the prover, and replaces the value of a with $q(r) \pmod p$. This process is repeated, while preserving the invariant that $a = A \pmod p$. Each iteration removes one variable from A. When A is simplified to a constant expression, the verifier can verify that $A = a$.

If B is false, a malicious prover must chose an incorrect a accompanied by an incorrect $q(z_i)$. Allowing the verifier to pick a random r at each step is very likely to reveal the prover's dishonesty, because if the given function was incorrect, the probability that the resulting $q(r)$ is equal to A (mod p) on every iteration is low. This probability is reduced a rate which is exponential to the number of iterations.

Simple QBF Form

One technical note regarding the polynomial functional form $q(z_i)$ is that its degree can grow exponentially with the number of variables in B, and this cannot be handled by a polynomially bound verifier. To handle this, Shamir defines a *simple* QBF, a form in which each variable is separated from its point of quantification by at most one universal quantifer. This bounds the degree of $q(z_i)$ because the only operator that can increase this degree is multiplication, which replaces a universal quantifier in the arithmetization process. The simple form ensures that this cannot occur more than once.

Although this appears to be a restrictive form, it can be shown that simple QBFs are sufficient for this proof. Any QBF of size n can be converted to a simple QBF whose size is polynomial in n, and deciding the truth of a simple QBF is also PSPACE complete.

Conclusion

The interactive proof system constructed in this paper provides a protocol by which a prover can convince a polynomially time bound verifier that a quantified boolean formula is true. An honest prover can always convince the verifier, and a dishonest prover can convince the verifier only with negligible probability. TQBF is PSPACE complete, so any problem in PSPACE can be verified by a polynomial time machine in an interactive proof. Since IP $\subseteq$ PSPACE is known, the containment of PSPACE in IP proves that these classes are identical.