A Problem That Is Complete for
PS (Polynomial Space): QBF

1. PS-Completeness.
2. QBF: (Quantified Boolean Formulas).
3. Evaluating QBF (Quantified Boolean Formulas).
4. PS-Completeness of the QBF problem.

by:
Saleh Al-shomrani

1. PS-Completeness:

- A *Polynomial-space-bounded TM* means: there is some polynomial \( p(n) \) such that when given input \( w \) of length \( n \), the TM never visits more than \( p(n) \) cells of its tape.
- We define a class of languages *PS* (*Polynomial Space*) to include all and only the languages that are \( L(M) \) for some Polynomial-space-bounded, deterministic TM \( M \).
- We define a problem \( P \) to be *complete* for *PS* (*PS-Complete*) if:
  1. \( P \) is in \( PS \).
  2. All Languages \( L \) in \( PS \) are polynomial-time reducible to \( P \).

*Theorem 11.6:* Suppose \( P \) is in PS-Complete problem. Then:
  a) If \( P \) is in \( P \), then \( P = PS \).
  a) If \( P \) is in \( NP \), then \( NP = PS \).
2. QBF: (Quantified Boolean Formulas):

- **Roughly**, a “Quantified Boolean Formula” is a boolean expression with the addition of the operators $\forall$ (“for all”) and $\exists$ (“there exists”).
- For simplicity, we will assume that no QBF contains two or more quantifications ($\forall$ or $\exists$) of the same variable $x$. This restriction corresponds roughly to disallow two different functions in a program from using the same local variable.
- **Formally**, the Quantified Boolean Formulas are defined as follows:
  1. 0 (false), 1 (true), and any variable are QBF’s.
  2. If $E$ and $F$ are QBF’s then so are $(E)$, $\neg(E)$, $(E) \land (F)$, and $(E) \lor (F)$.
     - It uses usual precedence rules: NOT, then AND, then OR (lowest).
     - We will use the “arithmetic” style of representing AND and OR, AND is represented by juxtaposition (no operator) and OR is represented by $+$.  

3. If $F$ is QBF that does not include a quantification of the variable $x$, then $(\forall x)(E)$ and $(\exists x)(E)$ are QBF’s. We say that the scope of $x$ is the expression $E$.

**Example1:**

Here is an example of QBF:

$(\forall x)((\exists y)(xy) + (\forall z)(\neg x + z))$.
- Starting with the variable $x$ and $y$, we connect them with AND and then apply the quantifier $(\exists y)$ to the subexpression $(\exists y)(xy)$. Similarly for the other subexpressions.

**3. Evaluating QBF (Quantified Boolean Formulas):**

- If a variable $x$ is in the scope of some quantifier of $x$, then that use of $x$ is said to be *bound*, Otherwise, an occurrence of $x$ is *free*.

**Q:** What are the types of variables in the QBF of example1 above?

**Example2:**

Evaluate the QBF of example1 above for $E0(x=0)$ and for $E1(x=1)$? Note the QBF is of the form $(\forall x)(E)$.
4. PS-Completeness of the QBF problem:

- Now, we can define the Quantified Boolean Formula problem: Given a QBF with no free variables, does it have the value 1?

* Theorem 11.10: QBF is in $P\Sigma$.

**Proof**

We use the recursive process for evaluating a QBF $F$. We can implement this algorithm using a stack, which we may store on the tape of the TM. Suppose $F$ is of length $n$. Then we create a record of length $O(n)$ for $F$ that includes $F$ itself and space for notation about which subexpression of $F$ we are working on. This example among the six possible forms of $F$ will make the evaluation process clear:

- Suppose $F = F_1 + F_2$?

The basis case, were $F$ is a constant, requires us to return that constant, and no further records are created on the tape.

In any case, we note that to the right of the record for an expression of length $m$ will be a record for an expression of length less than $m$. Note that even though we often have to evaluate two different subexpressions, we do so one-at-a-time. Thus, in our example above, there are never records for both $F_1$ or any of its subexpressions and $F_2$ or any of its subexpressions on the tape at the same time.

Therefore, if we start with an expression of length $n$, there can never be more than $n$ records on the stack. Also, each record is $O(n)$ in length. Thus, the entire tape never grows longer than $O(n^2)$. We now have a construction for a polynomial-space-bounded TM that accepts QBF; its space bound is quadratic. Note that this algorithm will typically take time that is exponential in $n$, so it is not polynomial-space-bounded.

- Now we turn to the reduction from an arbitrary language $L$ in $P\Sigma$ to the problem QBF.

* Theorem 11.11: The problem QBF is $PS$-Complete.

**Proof**

- Too long!!