Problem

Recall, in our discussion of the Church–Turing thesis, that we introduced the language D = { $\langle p \rangle$ | p is a polynomial in several variables having an integral root}. We stated, but didn't prove, that D is undecidable. In this problem, you are to prove a different property of D—namely, that D is NP-hard. A problem is ***NP-hard*** if all problems in NP are polynomial time reducible to it, even though it may not be in NP itself. So you must show that all problems in NP are polynomial time reducible to D.

Step-by-step solution

Step 1 of 3

Consider a language D (as discussed in **Church-Turing thesis**), which is defined as:

$$D = \{\langle p \rangle \mid p \text{ is a polynomial in several variable having an integral root}\}$$

Here, the given language D can be shown as $NP-hard$. This can be obtained by using the fact of $NP-completeness$ property of the language D.

Comment

Step 2 of 3

Now, consider the way to proof the $NP-completeness$ property of the given language. The language D can be shown as $NP-complete$ by using a reduction from $3SAT$.

• An expression L in $3SAT$ can be converted into L' by using the following facts:

$\sim a = (1-a)$ (Where the sign (~) is defined as a **NOT logic** operator)

$a \text{ and } b = (ab)$

$a \text{ or } b = (a+b-ab)$

• Now, for simplicity, a function $f(a,b,c)$ could be defined for each of the eight possibilities for all the clauses in $3SAT$. Here, eight possibilities will be taken because each variable $a, b \text{ and } c$ will acquire a value $1/0$.

Comment

Step 3 of 3

An algorithm can be used which replace every clause with one of the replacements and also used for putting the symbol multiplication between them. In other word, the description of above algorithm can be given as:

$$f(a \text{ or } b \text{ or} \sim c) = \left[(a+b-ab)+(1-c)-(a+b-ab)(1-c)\right]$$

The running time of the above given algorithm is linear.

• The result L' shows a polynomial behavior that consist **an integral root if and only if** L **is in** SAT^*. Therefore, the language D can be said as $NP-complete$.

• It is already known to be **un-decidable**, thus it will **not exist in** NP**.** Therefore from the above explanation, it can be said that **the language** D **is** $NP-hard$.

Comment