DE COMO CIERTOS CONJUNTOS NO PUEDEN SER NP-COMPLETOS

J. BALCÁZAR

En este artículo se presenta una visión general a los últimos resultados acontecidos en el area de la estructura de la clase NP-completa y su relación con el problema de si ------- P = NP o P ≠ NP.

1. INTRODUCCION

El estudio de la complejidad de los proble-- mas en términos de las cotas de tiempo que - precisan los algoritmos que los resuelven es una rama de la teoria de la calculabilidad - cuya importancia ha crecido de forma especta cular en los últimos años. Este análisis ha conducido a la definición de varias clases - de problemas, incluyendo una gran cantidad - de los propuestos por las sistintas ramas -- científicas relacionadas con la informática, y al estudio de las relaciones entre dichas clases, muchas de las cuales plantean aún - problemas abiertos.

En particular las clases correspondientes a las cotas de tiempo polinómicas han mostra- do ser de gran interés, por la gran cantidad de problemas comunes que se encuentran en -- ellas^ problemas de asignación, de transpor- te, de isomorfismos entre estructuras alge-- braicas, de teoría de números, de planifica- ción de actividades y otras muchas resultan ser resolubles por ciertos modelos de algo-- ritmos en tiempo polinómico. Así, más adelan te definiremos las conocidas clases *P* y *NP*, que incluyen un gran número de problemas de entre los antes citados.

Una de las subclases de la clase *NP*, la for- mada por los conjuntos *NP*-completos, será el objeto de nuestra atención. Un cuidadoso aná lisis de la estructura interna de tales con- juntos revela una fuerte similitud entre --- ellos; a pesar de sus diferentes proceden--- cias y de sus muy distintos enunciados, des- de un punto de vista técnico aparece una ca- racterística común a todos los problemas has ta el momento conocidos en dicha clase. A lo largo de este artículo nos proponemos demos- trar como problemas excesivamente diferentes a los *NP*-completos conocidos no pueden per- tenecer en ningún caso a esta misma clase; - para ello seguiremos los pasos marcados por los trabajos de L. Berman y J. Hartmanis --- (/3/, /11/), de P. Berman (/2/) y de S. Maha ney (/5/), todos ellos, y en especial este - último, de reciente publicación.

A lo largo de la sección 2 fijaremos las de- finiciones y hechos elementales a utilizar. La sección 3 estará dedicada a revisar los - trabajos de Berman y Hartmanis, y especial-- mente a su "conjetura de isomorfía". En las secciones 4 y 5 veremos algunos resultados - que avalan dicha conjetura: respectivamente

J.L. Balcazar de la Facultat d'Informatica de la Universitat Politécnica de Barcelona - Barcelona

Article rebut el Octubre de 1982.

los teorema de Berman y Mahaney. El segundo de ellos, del cual el primero es consecuencia, es el más fuerte obtenido hasta el momento en torno a la conjetura de isomorfía, y por tanto parece tener una cierta relevancia.

Este estudio incluye y amplía el realizado en el tercer capítulo de /1/.

2. DEFINICIONES BASICAS

2.0 Codificación y algoritmos.

Todas las muy diversas definiciones equivalentes del concepto de algoritmos tienen en común el hecho de consistir en manejo de símbolos exclusivamente, siguiendo un cierto tipo de transformaciones sintácticas definidas de forma muy estricta, y prescindiendo en todo momento del posible "significado" que se dé a los símbolos.

Aquí supondremos la existencia de un alfabeto (en ocasiones varios) formado por símbolos distintos; salvo indicación en contrario, se supondrá que se dispone de al menos dos símbolos. Las cadenas finitas o palabras, sobre un alfabeto Σ forman el conjunto Σ^* , y para cada natural n el conjunto Σ^{*n} es el formado por todas las palabras sobre Σ de longitud menor o igual a n. Denominaremos "longitud" de una palabra x, y lo denotaremos por $|x|$, al número de símbolos de que consta en x. La única palabra de longitud cero, llamada "palabra vacía", será denotada $\wedge$.

En algún momento nos podrá interesar tener numerado Σ^*. Una posible forma de numerarlo es considerar los símbolos de Σ asociados a los enteros 1...k, donde k=card(Σ), y las palabras asociadas al entero representado en notación k-ádica por los dígitos correspondientes a sus letras. Así, a la palabra $x= \sigma_0,\ldots,\sigma_n$ haremos corresponder

$$\nu(x) = \sum_{i=0}^{n} \nu(\sigma_i).k^{n-i}$$

donde $\nu(\sigma_i)$ es el entero entre 1 y k asociado al símbolo σ_i.

Asimismo supondremos definido un modelo formal de algoritmo, como por ejemplo las máquinas de Turing, cuya definición puede encontrarse en /10/ y en /13/; otros modelos, como las máquinas combinacionales ya descritas en QUESTIIO por J. Díaz (/8/), son válidas igualmente, al ser modelos equivalentes como recientemente se ha demostrado. Las máquinas elegidas admitirán como entradas oakabras sobre el alfabeto Σ, y su comportamiento sobre una entrada concreta consistirá en realizar un cierto cómputo, que consideraremos discretizado en una sucesión finita o no de pasos. Si la sucesión es finita, el estado final de la máquina deberá ser "aceptar" o "rechazar". Convendremos que una computación infinita equivale a rechazar la entrada de la que se parte.

Nos interesará también un cierto modelo de máquina cuyo comportamiento se aparta ligeramente del concepto usual de algoritmo: las máquinas no deterministas. Desde el punto de vista de las máquinas de Turing, una máquina no determinista tiene la posibilidad de optar en ciertos momentos entre varias formas de continuar su cómputo; por definición, una tal máquina acepta una palabra si alguno de sus posibles cómputos la acepta. Es decir, en cierto modo consideraremos que al presentarse una opción, la máquina "adivina" cual es el camino que conduce a la solución. Una forma equivalente de considerar el n-determinismo es la siguiente: inicialmente la máquina realiza una conjetura sobre el comportamiento que va a seguir; es decir, decide que opción va a tomar en cada una de las elecciones que se le presenten en la posterior etapa de cálculo. Una vez conjeturado el camino que lleva a la solución, se realiza el cómputo en forma determinista, siguiendo en cada elección la opción previamente decidida. Veremos más adelante un ejemplo de cómo funciona un algoritmo determinista, analizando las partes de conjetura y cálculo.

2.1. La clase *P*.

Un enfoque adecuadamente teórico para analizar el tiempo precisado por un algoritmo para funcionar puede venir dado por los propios pasos elementales de cómputo de una máquina de Turing. (Similar planteamiento se encuentra en /8/ para máquinas combinacionales). Es razonable pensar que un mismo algoritmo para datos más complicados tardará más tiempo en resolver un problema determinado;

el tiempo es función del "tamaño" de los datos.

Así, dada una familia T= $\{f_i\}$, $f_i:N\to N$, diremos que una máquina determinista funciona en tiempo T si existe una función $f\in T$ que acota el número de pasos que la máquina da en función del tamaño de la entrada; es decir, si $n\in N$ la máquina para antes de f(n) pasos para cualquier entrada que conste de menos de n símbolos.

Comporta especial interés la familia de cotas polinomiales $T=\{f_i \mid f_i(n)=n^i\}$; viene siendo comúnmente aceptado desde hace algunos años que un algoritmo cuyo tiempo de trabajo no está polinómicamente acotado es en la práctica irrealizable, ya que el tiempo preciso para resolver problemas para datos no elementales alcanza rápidamente valores del orden de siglos.

Por ello nos interesaremos en la clase *P* de los problemas resolubles polinómicamente; es decir, el conjunto formado por los conjuntos de palabras sobre Σ tales que existe una máquina determinista que los reconoce en tiempo $T=\{n^i\}$, es decir, en tiempo polinómico. Esta clase incluye todos los conjuntos finitos, y también muchos más de gran interés.

Desde el punto de vista práctico las máquinas de Turing como modelo de algoritmo se tornan inadecuados pronto, pues la extrema sencillez del modelo obliga a que cualquier algoritmo expresado en forma de máquina de Turing alcance gran tamaño y dificultad de comprensión; pero este paso no suele ser necesario, pues el análisis de un algoritmo expresado en un lenguaje de alto nivel en término de la cantidad de "operaciones elementales" que realiza conduce a resultados equivalentes módulo un polinomio.

2.2. La clase *NP*; otras clases.

Pasemos a continuación a establecer similar método de acotación del tiempo para máquinas no deterministas.

En tales máquinas el número de diferentes cómputos posibles puede resultar potencialmente infinito; pero, puesto que hasta que uno de ellos acepte la entrada para que ésta se considere aceptada por la máquina, parece asimismo razonable considerar que basta con que uno de ellos la acepte en tiempo t para dar la entrada como aceptada por la máquina en tiempo t. No se impondrá la cota, por tanto, a los cómputos que rechacen la entrada, y menos aún a los posibles cómputos infinitos.

Así pues, diremos que una máquina no determinista funciona en tiempo T, donde T es nuevamente una familia de funciones $f_i:N\to N$, si para toda entrada de longitud n aceptada por la máquina existen una $f_i\in T$ y un cómputo de la máquina que acepta la entrada en menos de $f_i(n)$ pasos.

Finalmente, la clase *NP* es la formada por los conjuntos de palabras decidibles mediante máquinas no deterministas en tiempo polinómico; es decir, acotando el funcionamiento de las máquinas por la familia $T=\{n^i\}$.

A modo de ejemplo, consideremos un problema que será útil más adelante: la satisfactibilidad de fórmulas booleanas proposicionales, que llamaremos SAT. Una fórmula proposicional se obtiene relacionando símbolos de variable, a los que se pueden asignar los valores "cierto" y "falso", mediante operadores de negación, disyunción y conjunción. Cuando en una fórmula todas las variables han tomado uno de los dos valores, la fórmula puede simplificarse utilizando las tablas de verdad de los operadores que en ella aparezcan. El problema que nos interesa es: dada una fórmula, ¿Existe alguna asignación de valores a las variables que aparecen en ella, de forma que al sustituir la fórmula simplifique a "cierto"? De una tal asignación diremos que "satisface" la fórmula.

Cuando una fórmula se obtiene de otra sustituyendo parte de las variables mediante una asignación parcial, diremos que la así obtenida es una"instancia" de la otra. Si una instancia de una fórmula es satisfactible, la fórmula lo es: basta completar la asignación que satisface la instancia con la asignación que la produjo.

Presentamos a continuación un algoritmo recursivo determinista para decidir si una fórmula es o no satisfactible. Su tiempo de trabajo es exponencial. Se ignora si existe un

algoritmo polinómico para decidir SAT, pues no se sabe si SAT esta en *P* o no. El segundo algoritmo muestra cómo la introducción del no determinismo permite decidir la satisfactibilidad de una fórmula en tiempo polinómico:
SATϵ*NP*.

```
Algoritmo para analizar una fórmula es SAT
   si la fórmula no tiene variables
     entonces si simplifica a "cierto"
              entonces responder "es satisfactible"
              si no responder "no es satisfactible"
              fin del si
     si no dar los valores "cierto" y "falso" a una de
            las variables, obteniendo dos instancias;
           analizar ambas instancias;
           si una de ellas al menos es satisfactible
             entonces responder "es satisfactible"
             si no responder "no es satisfactible"
             fin del si
     fin del si
fin del analizar.

Algoritmo no determinista para analizar una fórmula es
    conjeturar un valor "cierto" o "falso" para cada
               variable de la fórmula;
     sustituir los valores en la fórmula;
     simplificar la fórmula:
     si el resultado es "cierto"
       entonces responder "es satisfactible"
       fin del si
fin del algoritmo.
```

Además de las clases ya mencionadas, *P* y *NP*, existen otras clases que tienen asimismo interés. La primera es *CO-NP*, formada por los complementarios Σ^*-L de los conjuntos L es *NP* . Se ignora si *CO-NP=NP* o no.

Definiremos ahora las correspondientes a la mínima cota exponencial respectivamente para determinismo y para no determinismo.

La clase *DEXT* es la fórmula por aquellos subconjuntos de Σ^* que pueden ser reconocidos en tiempo acotado por la familia de funciones $T=\{f_i | f_i(n)=2^{i.n}\}$ por máquinas deterministas.

Similarmente, la clase *NEXT* es la formada por subconjuntos reconocibles por máquinas no deterministas bajo la misma cota $T=\{2^{i.n}\}$.

Nos preguntamos ahora: ¿añade el no determinismo potencia al determinismo? Es claro que *P*$\subset$*NP* , pero ¿es estricta esta inclusión? Similarmente, ¿está *DEXT* contenido extrictamente en *NEXT*?

Esta pregunta es una más de toda una gama de preguntas similares, planteables para todos los tipos de autómatas que se conocen: ¿En qué casos el no determinismo es estrictamente más potente que el determinismo? Así, se sabe que es así para máquinas de Turing no acotadas, donde las máquinas no deterministas reconocen los conjuntos recursivamente enumerables. Sin embargo, para los autómatas finitos ocurre lo contrario: es sabido que para todo autómata finito no determinista existe otro determinista que reconoce el mismo conjunto.

En principio la imposibilidad de encontrar algoritmos polinómicos para *NP* llevó a conje

turar que $P \neq NP$. Pero no ha sido posible - por el momento construir un ejemplo de con-- junto en *NP* que no esté en *P* . Actualmente - muchos matemáticos se inclinan por la inde-- pendencia de dicha conjetura respecto a la - axiomática.

2.3. Reducibilidad y complejidad.

Las herramientas fundamentales de la teoría de la recursividad, las reducibilidades, tie nen también un análogo polinómico.

Diremos que un conjunto $L_1 \subset \Sigma_1^*$ es m-reducible polinomicamente a $L_2 \subset \Sigma_2^*$, y lo denotaremos -- $L_1 \propto L_2$, si existe una función $f:\Sigma_1^* \to \Sigma_2^*$ calcula ble en tiempo polinómico, tal que $x \in \Sigma_1^*$ ---- $x \in L_1 \Leftrightarrow f(x) \in L_2$. Esta definición se diferencia únicamente de su análogo en la recursivi dad en la condición de calculabilidad polinó mica impuesta a la función de reducción; en adelante, y salvo que expresamente se indi-- que lo contrario, siempre que se cite la reducibilidad deberá entenderse m-reducibili-- dad polinómica.

Intuitivamente, el hecho de que L_1 sea reducible a L_2 formaliza la idea de que L_1 es -- "más sencillo" que L_2: si podemos resolver - en tiempo polinómico L_2, también podemos resolver L_1; basta, sobre una entrada x, calcular f(x) y decidir si $f(x) \in L_2$ para decidir - si $x \in L_1$. Otra reducibilidad, la T-reducibili dad, es también útil en el campo polinómico. En /14/ puede encontrarse un cuidadoso análi sis comparativo de las varias reducibilida-- des polinómicas.

Aún debemos definir un concepto de importancia, que formaliza la idea de "conjunto más difícil": un conjunto en *NP* es *NP*-completo si todo conjunto en *NP* es reducible a él.

Naturalmente, esta definición carece de inte rés si *P=NP*, pues en tal caso todo conjunto en *P* es *NP*-completo. Pero si *P≠NP*, esta no-- ción tendría importantes consecuencias.

Los siguientes hechos son inmediatos a par-- tir de la definición.

Proposición. *Sea $L \subset \Sigma^*$ un conjunto NP-completo. Entonces, $L \in P \Leftrightarrow P=NP$.*

□ *Proposición.*

Proposición. *Sea $L \subset \Sigma^*$ un conjunto NP-completo. Entonces para todo L', L' es NP-completo si y sólo si $L' \in NP$ y $L \propto L'$.*

□ *Proposición.*

En /7/ se demuestra que el funcionamiento de una máquina no determinista puede expresarse mediante fórmulas proposicionales, y se ex-- trae la siguiente conclusión:

Proposición. *SAT es NP-completo.*

□ *Proposición.*

A partir de SAT, y reduciéndolo a otros con-- juntos en *NP*, la clase *NP*-completa ha crecido rápidamente. En /10/ pueden encontrarse -- una elevada cantidad de problemas, todos ---- ellos *NP*-completos, provenientes de las más - diversas ramas de la ciencia.

Analogamente a *NP*, un conjunto es *CO-NP*-completo si está en *CO-NP* y todos los conjuntos en *CO-NP* son reducibles a él. El siguiente -- hecho es asimismo inmediato:

Proposición. *L es NP-completo si y sólo si -- $\bar{L}=\Sigma^*-L$ es CO-NP-completo.*

□ *Proposición.*

Finalmente, dos conjuntos L_1 y L_2 son equivalentes si $L_1 \propto L_2$ y $L_2 \propto L_1$. Evidentemente, todos los conjuntos *NP*-completos son equivalentes - entre sí.

Todas estas definiciones se orientan hacia la siguiente idea: para demostrar constructiva-- mente que *P≠NP* , parece razonable intentarlo con los "más difíciles" en la clase *NP*: los *NP*-completos. Viceversa, para demostrar que *P=NP* bastaría con encontrar un problema ---- *NP*-completo que admitiera resolución mediante un algoritmo determinista polinómico. Y aun-- que ambos objetivos han desaparecido del hori zonte de objetivos inmediatos, pues se piensa en otro tratamiento para la cuestión *P≠NP*, re ducibilidad y completitud han demostrado so-- bradamente ser conceptos de suficiente inte-- rés y riqueza como para merecer un estudio de tallado.

3. LA ISOMORFIA Y SUS CONSECUENCIAS.

3.0. Problemas distintos que en el fondo no lo son.

El grado de abstracción que impone el estudio de la complejidad desde nuestra óptica, cuyo principal exponente es la codificación de problemas en términos de conjuntos de palabras sobre alfabetos y su manipulación mediante tan precarios mecanismos como son las máquinas de Turing, comporta una interesante consecuencia: el olvidar el origen de cada problema, haciendo abstracción de sus características peculiares, y conservando únicamente aquellas propiedades que permanecen invariantes al codificar el problema; y, puesto que parece razonalbe considerar que el problema es el mismo si únicamente variamos de forma no sustancial la codificación, podremos incluso considerar tan sólo propiedades invariantes bajo cualquier codificación, siempre y cuando la codificación no modifique artificiosamente la complejidad del problema. Formalizaremos estos últimos conceptos, sin perder de vista el hecho de que nos interesan principalmente los grados polinómicos de complejidad.

Para ello partiremos del siguiente hecho: dadas dos codificaciones del mismo problema, relacionadas entre sí mediante una biyección que se comporta como reducción polinómica en ambos sentidos, no sólo podremos afirmar que son polinómicamente equivalentes, sino que en cierto modo conservan una misma estructura (hecho éste último que procede de la existencia de una reducción mutua biyectiva).

Se puede considerar en este punto una analogía con la teoría de la recursividad, en la que aparecen similares consideraciones: además de las m-equivalencias y de las T-equivalencias dadas por sus respectivos preórdenes - las reducciones -, la definición de isomorfía recursiva plantea similares exigencias: una biyección que actúa como reducción en ambos sentidos. Las relaciones entre equivalencias e isomorfía recursiva, cuyo principal resultado en la recursividad es el teorema de Myhill, pueden entonces replantearse bajo la cota polinómica; en la próxima subsección demostraremos el análogo polinomial, ligeramente debilitado, al teorema de Myhill.

Así pues, definimos: dos conjuntos de palabras $L_1 \subset \Sigma_1^*$ y $L_2 \subset \Sigma_2^*$ son polinómicamente isomorfos (abreviado p-isomorfos) si existe una función biyectiva $f:\Sigma_1^* \to \Sigma_2^*$ tal que tanto f como f^{-1} son calculables en tiempo polinómico y actúan como reducciones de $L_1 \propto L_2$ mediante f y $L_2 \propto L_1$ mediante f^{-1}.

Evidentemente, en tal caso L_1 y L_2 no sólo son polinómicamente equivalentes, sino que además la complejidad en su reconocimiento procede de las mismas causas; su estructura es suficientemente similar como para permitir trazar entre ellas un paralelismo fácil de calcular: computable en tiempo polinómico.

El teorema de Myhill afirma: todos los conjuntos Σ_1-completos son recursivamente isomorfos (/16/). Podemos preguntarnos, descendiendo a las regiones polinomiales: ¿Son isomorfos entre sí todos los conjuntos *NP*-completos? Si así fuese, un conjunto finito no podría ser *NP*-completo, luego $P \neq NP$. Inversamente, Berman y Hartmanis conjeturan que bajo la hipótesis $P \neq NP$, la respuesta es afirmativa; es la llamada "conjetura de isomorfía", que ocupa posición privilegiada entre los problemas abiertos en la teoría de la complejidad.

Sí que es posible dar una condición suficiente para que dos problemas *NP*-completos sean isomorfos, estableciendo un teorema cuya demostración sigue los pasos del teorema de Myhill. En su virtud, y mediante algunos lemas técnicos de sencilla aplicación, Berman y Hartmanis han comprobado que los conjuntos *NP*-completos conocidos son p-isomorfos entre sí (/11/, /12/): la condición suficiente del teorema es de tal generalidad que aún no ha aparecido ningún conjunto *NP*-completo que no la cumpla.

Dado que la exhibición de un contraejemplo no parece, a la vista de los resultados, la mejor forma de intentar refutar la conjetura de isomorfía, el esfuerzo se ha concentrado en ciertas familias de conjuntos que más adelante definiremos. Los resultados de este estudio, si bien no demuestran la conjetura, la avalan, en el sentido de que los resultados que se demuestran coinciden con las consecuencias que tendría la conjetura de isomorfía.

3.1. El teorema de Berman y Hartmanis.

La pregunta que da origen al teorema de Berman y Hartmanis es sencilla: ¿Bajo qué hipótesis puede asegurarse la posibilidad de construir un p-isomorfismo entre dos conjuntos a partir de reducciones entre ellos en ambos sentidos?

El equivalente en teoría de la recursividad se conforma con la hipótesis, bastante razonable por otra parte, de inyectividad. La obligación adicional de que el isomorfismo sea calculable en tiempo polinómico impondrá una nueva hipótesis que permita controlar el tiempo de cálculo del isomorfismo, así como la posibilidad de calcular en tiempo polinómico las funciones inversas.

Una función de reducción polinómica f recibirá el calificativo de "invertible" si es inyectiva y además su inversa f^{-1} es calculable en tiempo polinómico. Una función de reducción polinómica f es "creciente en longitud" si la longitud de f(w) es estrictamente mayor que la de w para cualquier posible argumento w de f; es decir, $|f(w)|>|w|$.

En este punto podemos enunciar y demostrar el siguiente:

Teorema. (/3/) *Sea $A\propto B$ mediante p, y $B\propto A$ mediante q, donde p y q son funciones de reducción invertibles y crecientes en longitud. Entonces, A y B son p-isomorfos.*

Demostración. Sea s(n) un polinomio que acote simultáneamente el tiempo necesario para calcular p, q, p^{-1} y q^{-1}.

Su existencia garantiza que el que p^{-1} o q^{-1} estén indefinidas sea decidible en tiempo polinómico: basta dar s(n)+1 pasos y contestar "indefinido" si en este tiempo no se ha hallado un resultado.

El isomorfismo que definiremos coincidirá con p en ciertos puntos y con q^{-1} en otros; ambas funciones cumplen que $x\in A \Leftrightarrow p(x)\in B$ (respect. $x\in A \Leftrightarrow q^{-1}(x)\in B$), con el inconveniente de que la primera no agota todo B (no necesariamente es suprayectiva) y de que la segunda no siempre está definida; sin embargo, podemos utilizar p en unos casos y q^{-1} en otros, de forma que q^{-1} se utilice suficientemente como para asegurar que se abarca todo B; más aún, la decisión de si utilizar p o q puede tomarse en tiempo polinómico. Veamos cómo.

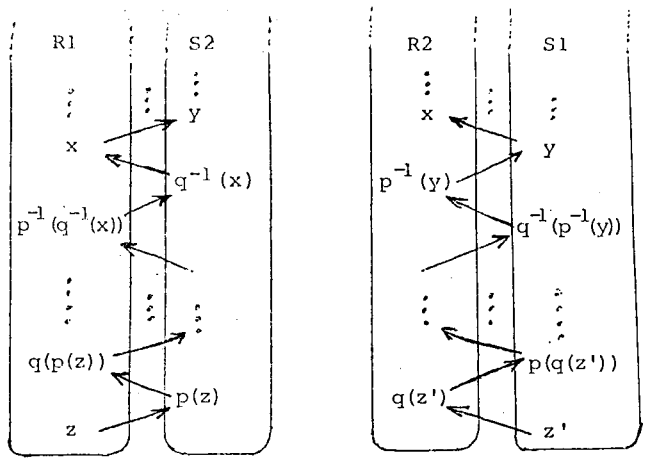
Sea Σ el alfabeto en que codificamos A y Γ el de B: $A\subset\Sigma^*$, $B\subset\Gamma^*$. Dividiremos Σ^* (y Γ^*) en dos partes disjuntas de la siguiente forma: para cada $x\in\Sigma^*$, consideramos sus sucesivas imágenes inversas, en la siguiente sucesión:

$x, q^{-1}(x), p^{-1}(q^{-1}(x)),\ldots,$

$p^{-1}(q^{-1}\ldots(p^{-1}(p^{-1}(x)\ldots),\ldots$

Similar construcción puede hacerse para $y\in\Gamma^*$, aplicando en forma alterna p^{-1} y q^{-1}: y, $p^{-1}(y)$, $q^{-1}(p^{-1}(y)),\ldots$ Ahora bien, siendo p y q crecientes en longitud, $|x|>|a^{-1}(x)|>\ldots$, lo que implica que tales sucesiones son finitas; es decir, que en algún momento no será posible aplicar p^{-1} o q^{-1} por estar indefinidas. Es más, las sucesiones tienen a lo más $|x|+1$ elementos, pues cada uno tiene una longitud inferior al menos en 1 a la del anterior.

Definimos ahora R_1 como el conjunto de aquellos $x\in\Sigma^*$ tales que la sucesión así construida termina en un elemento de Σ^*, es decir, llega a un punto en que no es aplicable q^{-1}. Su complementario $R_2=\Sigma^*-R_1$ será por tanto el conjunto de los $x\in\Sigma^*$ cuya sucesión termina en un elemento de Γ^*, es decir, llega a un punto en que no es aplicable p^{-1}. Análogamente, S_1 será el conjunto de las $y\in\Gamma^*$ tal que su sucesión asociada termine en Γ^* por no poderse aplicar p^{-1}, y $S_2=\Gamma^*-S_1$ estará formado por palabras cuya sucesión termina en Σ^* por no estar definida q^{-1}. La figura siguiente ilustra la definición:



$z \in \Sigma^*,\ z \notin q(\Gamma^*) \Rightarrow q^{-1}(z)$ indef. $\quad z \in \Gamma^*,\ z \notin p(\Sigma^*) \Rightarrow p^{-1}(z')$ indef.

Puesto que para cada $x \epsilon \Sigma^*$ (respect. $y \epsilon \Gamma^*$) la sucesión a la que pertenece está definida -- unívocamente, $R_1 \cap R_2 = \emptyset$ (respect. $S_1 \cap S_2 = \emptyset$). -- Más aún, podemos decidir si $x \epsilon R_1$ o $x \epsilon R_2$ en - tiempo polinómico, pues reconstruir hacia -- atrás la sucesión precisa únicamente $|x|+1$ - evaluaciones de p^{-1} y q^{-1}, cada una de las - cuales precisa menos de $s(|x|)$ pasos, y la - comprobación de estar al final de la suce--- sión (p^{-1} o q^{-1} indefinidos) puede hacerse - nuevamente en $s(|x|)$ pasos. Luego la deci--- sión $x \epsilon R_1$ ó $x \epsilon R_2$ puede realizarse en $(n\ 2)s(n)$ pasos.

Definamos ahora

$$\phi(x) = \begin{cases} p(x) & \text{si } x \in R_1 \\ q^{-1}(x) & \text{si } x \in R_2 \end{cases}$$

Puesto que en todo R_2, por construcción, está definida q^{-1}, ϕ es una función total. Su inyectividad se deduce de la de p y q, te--- niendo en cuenta que $R_1 \cap R_2 = \emptyset$, $S_1 \cap S_2 = \emptyset$, ----- $\phi(R_1) = S_2$ y $\phi(R_2) = S_1$. La sobreyectividad de ϕ se deduce del hecho de que su inversa es:

$$\phi^{-1}(y) = \begin{cases} p^{-1}(y) & \text{si } y \in S_2 \\ q(y) & \text{si } y \in S_1 \end{cases}$$

que es total puesto que si $y \epsilon S_2$ entonces --- $p^{-1}(y)$ está definido por construcción.

Que Φ es calculable en tiempo polinómico se deduce de su definición, así como que Φ^{-1} -- también lo es. Finalmente, para cualquier -- $x \epsilon \Sigma^*$, si $x \epsilon R_1$ entonces

$$x \epsilon A \Leftrightarrow \Phi(x) = p(x) \epsilon B$$

por ser p una reducción de A a B, y si $x \epsilon R_2$, entonces

$$x \epsilon A \Leftrightarrow \phi(x) = q^{-1}(x) \epsilon B$$

por ser q una reducción de B a A, de donde ϕ es reducción de A a B; es inmediato por análogos motivos que ϕ^{-1} es una reducción de B a A, con lo cual Φ es una biyección que funciona como reducción en ambos sentidos: por tanto, A y B son p-isomorfos.

□ *Teorema*

3.2. Y su modo de empleo.

Está claro que el anterior teorema por sí so lo no comporta consecuencias de interés de - cara a la conjetura de isomorfía sin un análisis de las condiciones que exige, y espe-- cialmente de la hipótesis de crecimiento en longitud. Podría darse el caso de que esta - condición fuera excesivamente restrictiva e impidiera la aplicación del teorema con un - mínimo de generalidad. En esta sección daremos algunos lemas, quizá excesivamente técnicos, pero que facilitan la comprobación de - condiciones suficientes para que se cumplan las hipótesis del teorema de Berman y Hartmanis. Estos lemas se encuentran en el mencionado artículo junto al teorema principal.

El primero de estos lemas da una condición - suficiente para asegurar la existencia de reducciones crecientes en longitud e inverti-- bles a partir de reducciones invertibles sin más: Más adelante veremos que SAT cumple tales condiciones, lo que asegura que la hipótesis de crecimiento en longitud es muy poco restrictiva.

El segundo lema permite asegurar la existencia de funciones invertibles a partir de reducciones cualesquiera bajo condiciones que como también veremos son asimismo poco restrictivas.

La combinación de ambos lemas conduce a un teorema que da condiciones suficientes para poder aplicar al teorema de Berman y Hartmanis. Tales condiciones resultan ser de sencilla comprobación en otros problemas *NP*-completos, y en /3/, /11/, /12/ puede encontrarse un análisis exhaustivo de los principales conjuntos *NP*-completos que conduce al hecho de que todos ellos son p-isomorfos a SAT (y por tanto p-isomorfos entre sí).

Previamente precisamos definir el concepto de "función de relleno". Dado un conjunto $A \subset \Sigma^*$, Z es una función de relleno para A si es inyectiva y verifica que $Z(x) \in A \iff x \in A$.

Podemos demostrar ahora el siguiente

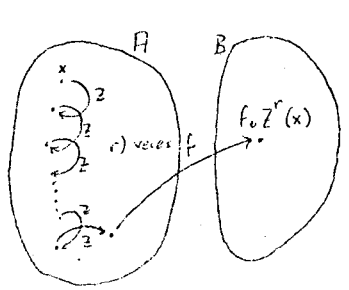
Lema. *Si A $\propto$ B mediante una función invertible f, y o bien A o bien B posee una función de relleno Z tal que tanto ella como su inversa Z^{-1} son calculables en tiempo polinómico, y que cumple que $|Z(y)| > |y|^2+1$ para cualquier y, entonces A $\propto$ B mediante una función f' invertible y creciente en longitud.*

Demostración. Sea q(n) un polinomio que acota el tiempo de cálculo de f y de f^{-1}, y supongamos que es A quien posee la función de relleno Z. La condición impuesta a Z implica que aplicáncola repetidas veces podremos conseguir que $|Z^r(x)| > q(|x|)$ para todo x y un cierto r fijo. Por otra parte, el ser de relleno implica que a partir de $x \in A$, $Z^r(x)$ queda en A, y a partir de $x \notin A$, $Z^r(x)$ no entra en A; por tanto, su composición con la reducción f, sea $f' = f \; Z$, es una nueva reducción. Veamos que el hecho de que Z^r incremente la longitud al menos según q asegura de f' es creciente en longitud.

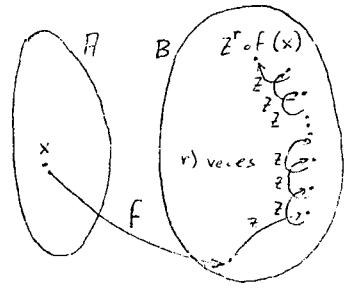
Para ello veamos que f no decrementa la longitud en más de q; la idea es que si tal fuese, f^{-1} no podría calcularse en tiempo q. En efecto, f^{-1} no puede escribir más de $q(|x|)$ símbolos, pues sólo en escribir el resultado se tardaría más tiempo que el que permite la cota q. (Este razonamiento será nuevamente utilizado más adelante en varias ocasiones.) Así, si $|f_0 Z'(x)| \leq |x|$, entonces $|f^{-1}{}_0 f_0 Z^r(x)| \leq q(|x|)$; pero $f^{-1}{}_0 f_0 Z^r(x) = Z^r(x)$, y $|Z^r(x)| \leq q(|x|)$, contra la definición de r. Luego $f' = f_0 Z^r$ es reducción creciente en longitúd, y tanto ella como su inversa $(Z^{-1})^r{}_0 f^{-1}$ son calculables en tiempo polinómico.

Queda por analizar el caso, relativamente similar en que quien posee la función de relleno Z es B. Por la razón antes expresada, f no decrementa la longitud en una relación mayor de q; es decir, $q(|f(x)|) \geq |x|$. Si así no fuese, puesto que f^{-1} es calculable en tiempo q, $|x| = |f^{-1}{}_0 f(x)| \leq q(|f(x)|) < |x|$ es una clara contradicción. Ahora bien, fijado r tal que $|Z^r(x)| > q(|x|)$, la función $f' = Z^r{}_0 f$ verifica $|f'(x)| = |Z^r{}_0 f(x)| = |Z^r(f(x))| > q(|f(x)|) \geq |x|$, luego es creciente en longitud; análogamente al caso anterior, hereda de f y Z^r calculabilidad polinómica, invertibilidad y ser reducción.

La figura adjunta da una idea intuitiva de cómo se utiliza la función Z para conseguir una reducción creciente en longitud, según sea el caso de que Z es una función de relleno para A o para B.



Z es relleno en A



Z es relleno en B.

□ *Lema*

Este lema nos permitirá construir reducciones sin preocuparnos de si son creciente en longitud. Veamos ahora un lema de similar estilo que asegura bajo ciertas condiciones la inversibilidad de una reducción construida a partir de cualquier otra reducción. De hecho, lo que el lema demuestra es que dicha inversibilidad depende, en cierto modo exclusivamente, de la estructura interna del conjunto analizado.

Lema. *Si un conjunto A admite dos funciones* $D:\Sigma^*\to\Sigma^*$, $S:\Sigma^* \times \Sigma^*\to\Sigma^*$, *computables en tiempo polinómico, que cumplen las condiciones siguientes:*

i) $\forall x \quad \forall y, \quad S(x,y)\in A \iff x\in A$

ii) $\forall x \quad \forall y, \quad D(S(x,y)) = y$

entonces para todo conjunto C $\subset$ *A y toda reducción f de C de A, la función* $f'(y) = S(f(y),y)$ *es una reducción invertible de C a A.*

Demostración. En primer lugar, nótese que S deberá ser inyectiva respecto de su segundo argumento: si $S(x,y) = S(x,y')$, aplicando D, $y=D(S(x,y))=D(S(x,y'))=y'$. Por tanto, la f' definida es nyectiva. La condición i) asegura que si f es una reducción, f' también lo es: $y\in C \Leftrightarrow f(y)\in A \Leftrightarrow S(f(y), y)\in A$. Finalmente, la inversa $(f')^{-1}$ está definida sobre aquellos z para los que existe y con $S(f(y),y)=z$, para los que se cumple por tanto $D(z)=D(S(f(y),y))=y$; sustituyendo, son los z que cumplen $S(f(D(z)),D(z))=z$. Por tanto $(f')^{-1}$ viene definida por

$$(f')^{-1}(z) = \begin{cases} D(z) \text{ si } z = S(f(D(z)),D(z)) \\ \text{indefinido si no} \end{cases}$$

que claramente es calculable en tiempo polinómico por serlo f,S y D.

□ *Lema.*

De hecho, lo que el anterior lema exige al conjunto A es, en cierto modo, ser suficientemente rico en elementos como para poder permitirse el lujo de incluir, para cada uno de sus elementos $x\in A$, una "copia" entera, calculable en tiempo polinómico, de la cantidad exponencial de palabras que forma todo Σ^* ("copia" que es $S(x,\Sigma^*)\subset A$ para $x\in A$; recuérdese que S es inyectiva), pero sin irse demasiado lejos" a buscar palabras para $S(x,\Sigma^*)$, pues la información correspondiente al segundo argumento de S(x,y) debe poderse obtener en tiempo polinómico (mediante la función $D(S(x,y))=y$). Es más, obsérvese que el complementario de A, $\bar{A}$, debe gozar de parecida riqueza, pues asimismo debe albergar una copia de Σ^* para cada uno de sus elementos (pues $x\notin A \quad S(x,\Sigma^*)\subset\bar{A}$). Como veremos, esto obliga al conjunto a tener una "densidad" mínima, por debajo de la cual el conjunto no tendría suficientes elementos. En la sección 3 se detallarán estas afirmaciones.

Así, si el conjunto admite funciones S y D, mientras la función S permite convertir en inyectiva una reducción, la función D permite asegurar que en dicha conversión se gana además la posibilidad de calcular la inversa en tiempo polinómico, por lo cual a partir de una reducción cualquiera obtendremos una reducción inversible.

Un lema más, que concierne específicamente a SAT, será el último escalón hacia la versión directamente aplicable del teorema de Berman-Hartmanis.

Lema. *SAT admite una función de relleno Z y funciones S y D que satisfacen los anteriores lemas.*

Demostración. Unicamente daremos las definiciones de S, D y Z; que las hipótesis de los lemas se satisfacen se reduce a una mera comprobación.

La función S(w,y) se calcula como sigue: supongamos que
$\Sigma = \{0,1\}$:

Compruébese que w es una fórmula bien formada;

Si lo es, entonces sea r el mayor índice de variable usado en w;
si no, r:=0;

Defínase entonces

$$S(w,y)=w\wedge(x_{r+1}\vee\neg x_{r+1})\wedge z_1 \wedge \cdots\cdots \wedge z_n$$

donde $n=|y|$, y

$$z_i = \begin{cases} x_{r+1+i} & \text{si la i-ésima letra de y es 1,} \\ \neg x_{r+1+i} & \text{si es 0;} \end{cases}$$

Tal cálculo precisa únicamente tiempo polinómico, y también basta tiempo polinómico para comprobar que una fórmula dada tiene una terminación de la forma

$$(x_s \vee \neg x_s) \wedge z_1 \wedge \ldots \wedge z_m$$

y reconstruir a partir de tal terminación la palabra y; con lo que D también es calculable en tiempo polinómico.

Finalmente, la función $Z(w), 0^{(|w|^2+1)}$ es una adecuada función de relleno, por lo cual el lema queda demostrado.

□ *Lema*

Una vez en este punto, el siguiente teorema es casi inmediato.

Teorema. *Un conjunto NP-completo B es p-isomorfo a SAT si y sólo si existen funciones* S_B *y* D_B *verificando*

i) $\forall x \quad \forall y, \quad S_B(x,y) \in B \Leftrightarrow x \in B$

ii) $\forall x \quad \forall y, \quad D_B(S_B(x,y)) = y.$

Demostración. Por ser B un conjunto *NP*-completo existen reducciones polinómicas SAT B, sea mediante f y B SAT, sea mediante g. Puesto que SAT posee funciones S,D que satisfacen el correspondiente lema, g puede transformarse en otra reducción g' invertible; por análoga razón, S_B y D_B permiten construir una f' invertible. Ahora bien, en virtud de la función Z de SAT, componiendo una cierta z^r por delante con f' y una cierta $z^{r'}$ por detrás con g', obtendremos f'' y g'' invertibles y crecientes en longitud. Por el teorema de la subsección anterior, B será p-isomorfo a SAT.

La implicación recíproca es sencilla: sean B y SAT p-isomorfos, y sea ϕ el isomorfo: las funciones

$$S_B = \phi^{-1} \circ S$$

$$D_B = D \circ \phi$$

verifican las condiciones exigidas, como un cuidadoso análisis muestra.

□ *Teorema*

Así, basta pues demostrar que un conjunto *NP*-completo B admita funciones S_B y D_B como las expresadas para que se siga de inmediato su isomorfismo con SAT; como queda dicho, en /3/, /11/ y /12/ pueden verse ejemplos de tales funciones para una inmensa variedad de conjuntos *NP*-completos. Si bien no queda demostrado para todos, por lo que la conjetura de isomorfía se queda en conjetura, sí se deduce que la vía del contraejemplo directo no será la mejor para refutarla.

Sin embargo, pasamos a estudiar nuevas vías que no resultan, en tal sentido, mejores, pero que conducen a interesantes resultados.

3.3. Ciertos conjuntos parecen útiles.

Renunciando a buscar un conjunto *NP*-completo que no sea isomorfo a SAT, se puede intentar refutar la conjetura de isomorfía de la manera dual: buscando conjuntos no isomorfos a SAT que sean *NP*-completos. Definiremos dos familias de conjuntos, una contenida en la otra, cuyos elementos en ningún caso pueden ser isomorfos a SAT. Nuestra pregunta será entonces: alguno de ellos puede ser *NP*-completo?

La primera familia, que viene estudiándose desde 1974, es la de los conjuntos monoliterales ("tally languages" en terminología inglesa). Se puede demostrar (/8/) que las clases NEXT y DEXT definidas en 2.3 coinciden si y solamente si todos los conjuntos monoliterales en *NP* están en *P* ; si hubiese un conjunto monoliteral en *NP-P*, podríamos construir un conjunto en *NEXT-DEXT* a partir de él. Al respecto, se deduciría de la conjetura de isomorfía que no habría conjuntos monoliterales en la clase *NP*-completa, como veremos más adelante. Luego la existencia de conjuntos monoliterales *NP*-completos implicaría simultáneamente la falsedad de la conjetura de isomorfía y la no coincidencia de las clases *NEXT* y *DEXT* .

Al analizar el porqué de la imposibilidad de que un conjunto monoliteral sea isomorfo a SAT se encuentra que la causa radica en el hecho de que un conjunto monoliteral no contiene suficientes palabras como para admitir una función inyectiva como la función S de la anterior subsección; pero esto es generalizable a toda una familia de conjuntos cuyo tamaño no crece adecuadamente para albergar copias distintas de Σ^*. para precisar estos conceptos, definimos la función "censo".

Definición. Dado $A \in \Sigma^*$, la función $C_A: N \to N$, - llamado "censo de A", es

$$C_A(n) = \text{card}\,(A \cap \Sigma^{*n}).$$

Es decir, $C_A(n)$ es el número de palabras en A de longitud menor o igual a n. Al acotar - el censo, impediremos que el conjunto posea la función inyectiva S adecuada.

Definición. Un conjunto T es esparso si existe un polinomio p que acota el censo de T; - es decir,

$$C_T(n) \leq p(n) \qquad n \in N.$$

Ahora bien, SAT no es esparso. Más aún:

Proposición. *Ningún conjunto no vacío que posea funciones S,D cumpliendo el lema de la - subsección 3.2 puede ser esparso.*

Demostración. Supongamos que T es esparso, y sea p un polinomio que acota su censo. Suponemos además que T admite una función S calculable en tiempo polinómico y tal que cumple la hipótesis $\forall x\, \forall y$, $S(x,y) \in T \iff x \in T$, y -- sea $x_0 \in T$. El que $S(x_0,y)$ sea calculable en - tiempo polinómico, $q(y)$, implica que

$$|S(x_0,y)| \leq q(|y|)$$

pues de lo contrario no habría tiempo suficiente ni siquiera para escribir el resultado. Luego, teniendo en cuenta que x_0 T,

$$S(x_0,\Sigma^{*n}) \subset \Sigma^{*q(n)} \quad , \quad S(x_0,\Sigma^{*n}) \subset T,$$

luego

$$S(x_0,\Sigma^{*n}) \subset \Sigma^{*q(n)} \cap T.$$

Por tanto,

$$\text{card}\,(S(x_0,\Sigma^{*n})) \leq \text{card}(\Sigma^{*q(n)} \cap T) \leq p(q(n)).$$

de donde se deduce que S aplica la cantidad exponencial de palabras en Σ^{*n} en sólo una cantidad polinómica de imágenes $p(q(n))$: no puede ser, por tanto, inyectiva, lo que impide la existencia de una función D que a partir de S(x,y) recupere la segunda componente. Si el conjunto admite funciones S y D, no -- puede crecer sólo polinómicamente, no puede ser esparso. La proposición queda demostrada.

□ *Proposición*

De la proposición anterior y el teorema final de la subsección 3.2 se deduce de inmediato el resultado que sigue:

Corolario. *Ningún conjunto esparso puede -- ser isomorfo a SAT.*

□ *Corolario.*

Puesto que todo conjunto monoliteral es esparso, ya que su censo está acotado por la identidad, ningún conjunto monoliteral puede ser isomorfo a SAT.

Nos preguntamos ahora: ¿Existen conjuntos monoliterales *NP*-completos? ¿Existen, al menos conjuntos esparsos *NP*-completos?

Piotr Berman respondió negativamente a la -- primera cuestión en 1978 (/2/); Steve Mahaney respondió negativamente a la segunda en 1980 (/15/). Estudiaremos el resultado de P. Berman en la sección 4, y el de Mahaney en - la sección 5.

4. CONJUNTOS MONOLITERALES.

4.0. Un teorema y su corolario.

El hecho de que si *P=NP*, entonces todo conjunto en *P*, finitos, monoliterales y esparsos inclusive, es *NP*-completo, hecho que -- fué visto en la sección 1, conlleva el planteamiento de su recíproca. Evidentemente, la existencia de un conjunto *NP*-completo finito implicaría *P=NP*, pues todo conjunto finito - está en *P* (ver proposición en la sección 2.1) ¿Tiene la misma consecuencia la existencia de conjuntos monoliterales, o esparsos, en la -- clase *NP*-completa? La conjetura de isomorfía, como hemos visto, implicaría una respuesta -- afirmativa.

El principal teorema en la presente sección. debido al polaco Piotr Berman, afirma que no hay conjuntos monoliterales *NP*-completos salvo que *P = NP* . Lo obtendremos como corolario de un teorema ligeramente más técnico.

Consideremos previamente la siguiente definición: una función $f: \Sigma^* \to \Sigma^*$ será "esparsa" si existe un polinomio p tal que ---------- $\text{card}\,(f(\Sigma^{*n})) \leq p(n)$ $\forall n \in N$. Es decir, la función f "comprime" la cantidad exponencial de palabras en Σ^{*n} en sólo una cantidad polinómica - de imágenes.

Los ejemplos que siguen muestran una función esparsa y una que no lo es.

Ejemplo. La función f definida por

$f(\wedge) = \wedge$
$f(0) = f(1) = 0$
$f(00) = f(01) = f(10) = f(11) = 1$
$f(000) = f(001) = \ldots\ldots = f(111) = 00$
$f(0000) = \ldots\ldots\ldots = f(1111) = 01$

es esparsa : $\text{card}(f(\Sigma^{*n})) = n+1$.

Ejemplo. La función f' definida por:

$f(\wedge) = \wedge$
$f(0) = 1$
$f(1) = 11$
$f(00) = 111$
$f(01) = 1111$
$f(10) = 11111$

no es esparsa: una función inyectiva nunca - esparsa, pues $\text{card}(f(\Sigma^{*n})) = \text{card}(\Sigma^{*n})$.

Una sencilla proposición relaciona la definición de función esparsa con la de conjunto -- esparso.

Proposición. *Toda función de rango esparso f calculable en tiempo polinómico es esparsa.*

Demostración. Sean p, q polinomios que acotan respectivamente el censo del rango y el tiempo preciso para calcular f. Por el razonamiento, ya utilizado en otros puntos, de que los pasos de escritura están acotados, --- $|f(x)| \leq q(|x|)$, por lo cual $f(\Sigma^{*n}) \subset \Sigma^{*q(n)}$. --- Luego

$$\text{card}(f(\Sigma^{*n})) \leq \text{card}(f(\Sigma^{*}) \cap \Sigma^{*q(n)}) \leq p(q(n))$$

ya que $f(\Sigma^*)$ es esparso.

□ *Proposición.*

La condición de calculabilidad en tiempo polinómico es necesaria, como muestra el segundo de los anteriores ejemplos; a su vez el primero muestra que el recíproco no es cierto.

El teorema que precisamos, cuya demostración se verá en la próxima subsección, se enuncia como sigue:

Teorema. *Si SAT es reducible a cualquier -- conjunto L mediante una reducción esparsa f, entonces P = NP.*

□ *Teorema*

Veamos como este resultado conduce al que -- ahora nos interesa.

Corolario. *Si P≠NP, entonces no existen conjuntos monoliterales NP-completos.*

Demostración. Sea S un conjunto monoliteral, y $a \in \Sigma$ tal que $S \subset a^*$. Supongamos que S es --- *NP*-completo, y consideremos una función f - que reduzca SAT a S.

Definamos la función

$$f'(x) = \begin{cases} f(x) & \text{si } f(x) \in a^* \\ b & \text{si no, donde } b \in \Sigma,\ b \neq a. \end{cases}$$

Por ser f calculable en tiempo polinómico, - también lo es f'; en efecto, la longitud de f(x) es polinómica en la de x, y basta recorrer f(x) para comprobar que sólo contiene - el símbolo a.

Ahora bien, si $F \in SAT$, $f(F) \in S \subset a^*$, por lo cual $f(F) = f'(F)$ y $f'(F) \in S$. Asimismo, si ------- $f'(F) \in S \subset a^*$, entonces $f'(F) = f(F)$, y de $f(F) \in S$ deducimos que $F \in SAT$. Luego f' es también una reducción de SAT a S.

El rango de f' está contenido en $a^* \cup \{b\}$, luego es esparso; f' es calculable en tiempo polinómico; en virtud de la anterior proposición, es esparsa. Nos encontramos en las hipótesis del teorema, y deducimos de él que - *P = NP*, en contradicción con nuestra hipótesis.

□ *Corolario.*

4.1. La demostración pendiente.

Queda por demostrar el teorema de la anterior subsección. Para ello, consideremos el siguiente algoritmo recursivo:

```
Analizar una fórmula es
   si la fórmula es "cierto"
      entonces responder "es satisfactible"
   si no
      aplicar a la fórmula la función f;
      asociar a la fórmula el valor obtenido;
      si la fórmula es "falso"
      o el valor asociado coincide con el de
           otra fórmula ya analizada y marcada
           como insatisfactible
       entonces
           marcarlo como insatisfactible;
           responder "es insatisfactible"
       si no
           dar los valores "cierto" y "falso" a una
           de las variables obteniendo dos nuevas
           fórmulas;
           analizar ambas fórmulas;
           si alguna de ellas es satisfactible
             entonces responder "es satisfactible"
           si no
             marcarla como insatisfactible;
             responder "es insatisfactible"
           fin del si
      fin del si
   fin del si
fin de analizar.
```

La corrección de este algoritmo para decidir la satisfactibilidad de una fórmula viene asegurada por el siguiente hecho:

Proposición. *El algoritmo anterior sólo marca fórmulas insatisfactibles.*

Demostración. Sólo se marcan aquellas fórmulas que se encuentran en al menos una de las siguientes condiciones:

i) La fórmula no tiene variables y es "falso" ;

ii) La substitución de ambos valores "cierto" y "falso" en alguna de sus variables ha dado como resultado dos fórmulas marcadas;

iii) Su valor asociado coincide con el de alguna otra fórmula marcada.

Tengamos en cuenta que si $f(F) = f(F')$, entonces $F \in SAT \iff f(F) \in L \iff f(F') \in L \iff F' \in SAT$ puesto que F es una reducción; consideremos ahora una inducción sobre el orden en que las fórmulas son marcadas. La primera fórmula marcada deberá serlo por i), luego es insatisfactible. Si en un momento dado todas las fórmulas marcadas son insatisfactibles, la siguiente en ser marcada, sea por i), ii), será asimismo insatisfactible. Luego solo se marcan fórmulas insatisfactibles.

□ *Proposición*

La proposición que a continuación se enuncia establece que el algoritmo marca fórmulas cada cierto tiempo. Este hecho será esencial para demostrar que es realizable en tiempo polinómico.

Proposición. *Si el algoritmo empieza el análisis de las dos instancias de una fórmula con k variables, entonces antes de terminar el análisis de las siguientes k+1 fórmulas o bien se encuentra una asignación que satisface la fórmula, o bien se encuentra un nuevo valor asociado a marcar.*

Demostración. Induscción sobre k. Adviértase que si se analizan las dos instancias de una fórmula es que su valor asociado no estaba marcado.

Para k = 1 : en las instancias no quedan variables; si una es "cierto", se ha encontrado la asignación; si ambas son "falso", la fórmula desarrollada se marca.

Para k=n 1 : o bien ambas instancias son -- marcadas de inmediato, y entonces se marca la fórmula, o bien una de ellas, con a lo - más n variables, es desarrollada; se aplica la hipótesis de inducción.

□ *Proposición.*

Podemos ya demostrar que sólo se analiza una cantidad polinómica de fórmulas.

Proposición. *Sobre una entrada de longitud n, el anterior algoritmo analiza a lo más -- (m+1).p(m) fórmulas, donde p es un polinomio que acota el cardinal de* $f(\Sigma^{*m})$.

Demostración. Puesto que f es esparsa, ----- $card(f(\Sigma^{*m})) \leq p(m)$, únicamente disponemos de a lo más p(m) valores que asociar a las fórmulas. Dado que cada (m+1) fórmulas, a lo -- más, se debe marcar un nuevo valor, al terminar de analizar (m+1).p(m) fórmulas se debe haber encontrado una asignación, de haberla, y se deben haber marcado como insatisfactibles todos los valores asociados, incluído - el de la fórmula inicial, caso de no ser satisfactible.

□ *Proposición.*

Asi, caso de disponer de una reducción esparsa de SAT, mediante el anterior algoritmo podríamos decidir SAT en tiempo polinómico, de donde se deduciría $P = NP$.

4.2. Un bonito ejemplo.

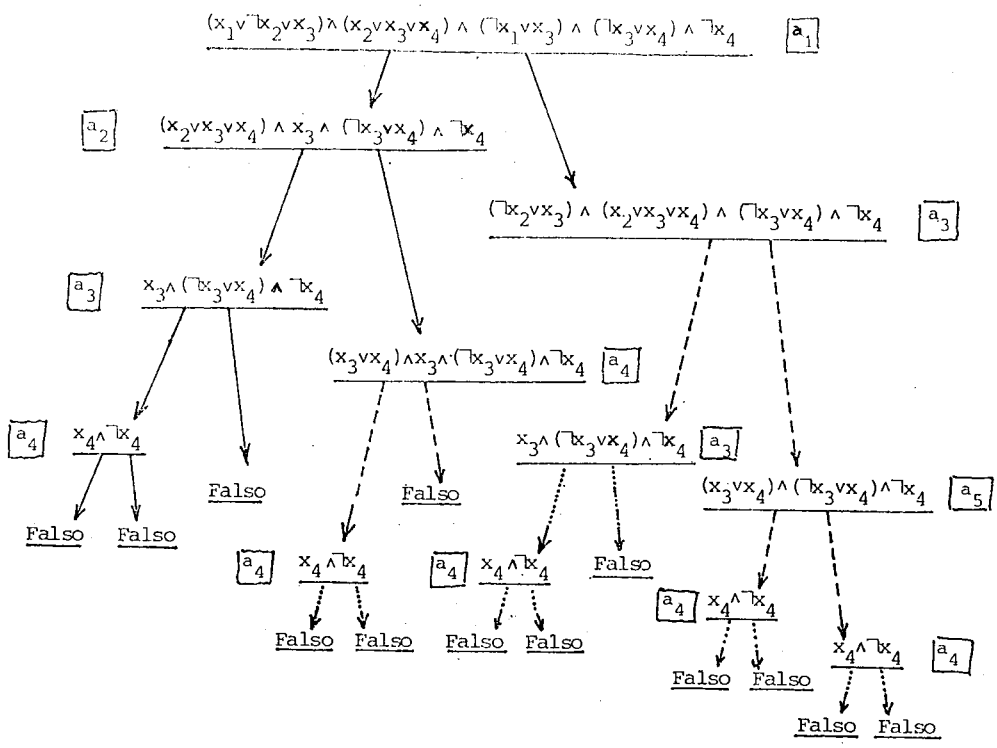
Con el fin de ilustrar cómo el algoritmo de la sección anterior consigue "podar" el árbol de instancias de una fórmula, reduciendo su exploración a un tamaño polinomial, proponemos el ejemplo de la fórmula (en forma normal conjuntiva por razones de claridad) que sigue. Hemos representado el árbol completo de instancias que exigiría una exploración - de tipo exponencial; parte del árbol está representado en trazo discontínuo, y una tercera parte en trazo punteado.

Junto a cada fórmula expresamos lo que podría ser el valor asociado a cada fórmula. Hemos - simulado astutamente el hecho de que el conjunto de posibles valores asociados es esparso permitiendo sólo cinco valores distintos que se repiten. Por otra parte, como es lógico, a fórmulas iguales corresponde el mismo valor.

El primer hecho que resaltamos es el siguiente: cuando la fórmula $x_4 \wedge \neg x_4$ aparece por segunda vez en la exploración del árbol (que - el algoritmo recorre en profundidad), podemos ahorrar su desarrollo, puesto que ya en su anterior exploración se demostró insatisfactible. Similar razonamiento respecto a - sucesivas apariciones de $x_4 \wedge \neg x_4$, así como -- de $x_3 \wedge (\neg x_3 \vee x_4) \wedge \neg x_4$, permite evitar la exploración de la parte del árbol que aparece en línea de puntos.

Generalicemos ahora esta observación al caso en que lo que se repite es, no ya la fórmula, sino su valor asociado. Puesto que dos fórmulas con el mismo valor asociado son simultaneamente satisfactibles o simultáneamente insatisfactibles (según el valor asociado esté o no en L), una vez que se ha demostrado que la primera aparición de un cierto valor corresponde a una fórmula insatisfactible, podremos asegurar que toda nueva aparición del mismo valor corresponde a fórmulas asimismo insatisfactibles.

Asi, cuando encontramos que el valor asociado a $(x_3 \vee x_4) \wedge x_3 \wedge (\neg x_3 \vee x_4) \wedge \neg x_4$ coincide con - el asociado a $x_4 \wedge \neg x_4$, ya probado no satisfactible, podremos suspender el desarrollo, pues de antemano sabemos que no será satisfactible. Por similares razones, será innecesario desarrollar $(\neg x_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\neg x_3 \vee x_4) \wedge x_4$, cuyo valor coincide con el de $x_3 \wedge (\neg x_3 \vee x_4) \wedge \neg x_4$, ya probada insatisfactible. Toda la parte del árbol en trazo discontínuo es "podada", pues su exploración es innecesaria.



En este ejemplo queda claro que el número de fórmulas analizadas es pequeño en relación - al número total de instancias en el árbol; - las proposiciones de la anterior sección demuestran que es función polinómica del número de variables, como se requiere, en todos los casos.

4.3. Aquí no acaba la cuestión.

Al margen de la pregunta sobre conjuntos esparsos *NP*-completos, que estudiaremos en la siguiente sección, podemos generalizar la -- pregunta sobre monoliterales en la siguiente forma: ¿Existen conjuntos monoliterales en - la clase *NP-P*?

Recordemos que las clases *NEXT* y *DEXT* son -- las aceptadas por máquinas no deterministas y deterministas, respectivamente, en tiempo $2^{c.n}$ para alguna constante c. Los teoremas - que a continuación se expresan, y cuyas de-- mostraciones se limitan a esbozar la idea -- principal, dan una idea del interés que tiene la pregunta arriba expuesta. En las referencias bibliográficas pueden encontrarse -- las demostraciones completas.

Teorema. *(/4/). Si P = NP, entonces, DEXT = = NEXT.*

Demostración. La idea consiste en numerar biyectivamente todas las palabras de Σ^*, de -- forma que el número n(w) asociado a $w \in \Sigma^*$ pueda calcularse en tiempo polinómico. Definiendo

$$M(L) = \{1^{n(w)} \mid w \in L\}$$

para cada L, y la operación inversa

$$D(L) = \{w \mid 1^{n(w)} \in L\}$$

puede demostrarse fácilmente que

$$L \in NEXT\text{-}DEXT \iff M(L) \in NP\text{-}P$$

y que para todo conjunto monoliteral L',

$$L' \in NP\text{-}P \iff D(L') \in NEXT\text{-}DEXT$$

de donde se deduce el teorema.

□ *Teorema.*

Así pues, combinando los dos resultados anteriores, podemos afirmar que:

Teorema. *Si bajo la hipótesis P ≠ NP puede demostrarse la existencia de conjuntos monoliterales en NP-P, entonces P = NP si y sólo si DEXT=NEXT.*

Demostración. Que *P = NP ⇒ DEXT = NEXT* está ya demostrado. Pero si *DEXT=NEXT*, entonces - por el anterior teorema no hay conjuntos mo-

noliterales en *NP-P*, por lo cual la hipótesis $P \neq NP$ conduciría a contradicción. Luego se tendría $P = NP$.

□ *Teorema*.

Estos últimos resultados permiten hacerse una idea de la importancia de los conjuntos monoliterales en la clase *NP* , aunque se sepa que no pueden ser completos.

5. CONJUNTOS ESPARSOS.

5.0. Qué le falta y qué le sobra al teorema de P. Berman.

Atacaremos a continuación la pregunta relativa a la existencia de conjuntos esparsos en la clase *NP*-completa. Para ello, es preciso un cuidadoso análisis del teorema de Berman enunciado en 4.0 y de su demostración.

En efecto, a falta de un tal análisis la primera impresión puede ser que el problema está ya resuelto; sin embargo no es así. La hipótesis "SAT es reducible a un conjunto esparso" es distinta a "SAT es reducible mediante una reducción esparsa", y el hecho de que SAT sea reducible a conjuntos esparsos no implica que las reducciones que aparezcan sean esparsas. El teorema de Berman, tal como lo hemos enunciado y demostrado, no tiene por consecuencia aún la inexistencia de conjuntos esparsos --- *NP*-completos (salvo *P=NP*). Es preciso debilitar la hipótesis, de forma que baste con saber que la reducción es a un conjunto esparso sin necesidad de afirmar hechos sobre la propia reducción. Veremos que esta debilitación de la hipótesis es posible, aunque laboriosa.

El primer hecho a resaltar es una primera debilitación de la hipótesis que no obliga a modificar la demostración. En el algoritmo de exploración de fórmulas desarrollado, el hecho que nos garantiza la terminación en tiempo polinómico es el de disponer únicamente de una cantidad polinómica de valores asociables a las fórmulas; dado que el algoritmo encuentra fórmulas insatisfactibles con valores asociados aún no marcados a intervalos lineales de tiempo, hemos conlcuido una cota para el tiempo del algoritmo, (m+1) x p(n) fórmulas.

Pero para esta conclusión es innecesario acotar el total de valores asociables: basta con acotar los valores disponibles para fórmulas insatisfactibles, que serán las únicas que marcaremos; la cota de valores disponibles para fórmulas satisfactibles es inútil, pues en ningún momento se hace uso de ella. Por ello, la hipótesis de que la reducción f de SAT a L es esparsa puede sustituirse por la siguiente más débil:

$$\text{card}(f(\overline{SAT} \cap \Sigma^{*n})) \leq p(n)$$

Ya a partir de este nuevo enunciado pueden ser obtenidos los primeros resultados.

5.1. S. Fortune nos echa una mano.

La anterior observación, puesta de manifiesto por Steve Fortune, conduce a un resultado que difiere del que buscamos en menos de lo que podría parecer.

Teorema (/q/). *Salvo que P = NP, no existen conjuntos esparsos CO-NP-completos.*

Demostración. Sea $\overline{SAT} \propto S$ via f, donde f es computable en tiempo p(n), y sea q(n) una cota polinómica del censo de S. Se tiene:

$$\text{card}(f(\Sigma^{n} \cap \overline{SAT})) \leq \text{card}(\Sigma^{*p(n)} \cap S) \leq q(p(n)),$$

es decir, puesto que las fórmulas insatisfactibles en $\overline{SAT}$ son aplicables sobre elementos de S, hay únicamente una cantidad polinómica de valores asociables a las fórmulas insatisfactibles. En virtud de la observación de la anterior subsección, el algoritmo de la sección 3 reconoce SAT en tiempo polinómico, por lo que $P = NP$.

□ *Teorema.*

En este punto no es difícil obtener una versión débil del teorema que buscamos, como preliminar al teorema final. Lo deduciremos del siguiente

Lema. *Si existe un conjunto esparso NP-completo, S, cuyo censo es computable en tiempo polinómico, entonces NP=CO-NP.*

Demostración. Basta demostrar que un conjunto *CO-NP*-completo, como es $\overline{S}$, está en *NP*. Pero bajo nuestras hipótesis es posible decidir $\overline{S}$ de forma no determinista, del siguiente modo:

```
Algoritmo para decidir si w∈ S̄ es
    Calcular  k:= C_s(|w|);
    Conjeturar k palabras distintas w_i ∈ Σ* tales
    que |w_i| ≤ |w|;
    Comprobar mediante el algoritmo no determinista
    que reconoce S que w_i ∈ S  ∀i;
    Comprobar  que w ≠ w_i  ∀i;
    si todas las comprobaciones resultan válidas
      entonces responder "w pertenece a S̄"
      fin del si
fin del algoritmo.
```

De hecho el algoritmo se limita a conjeturar todas las palabras de S de longitud menor o igual que la dada; si acierta con ellas, basta que ninguna de ellas coincida con la dada para que ésta no esté en S.

□ *Lema.*

Combinando los últimos resultados, obtenemos el siguiente

Teorema. *Si existe un conjunto esparso ----- NP-completo, S, cuyo censo es calculable en tiempo polinómico, entonces P = NP.*

Demostración. Bajo nuestras hipótesis, por el anterior lema, se tiene que *NP = CO-NP*. Pero entonces todo conjunto *NP*-completo es tambien *CO-NP*-completo, luego S estaría en las hipótesis del anterior teorema, de donde *P = NP*.

□ *Teorema*

Hasta aquí cuanto puede obtenerse del teorema de Berman. Nos encontramos cerca del resultado de Mahaney, pero la eliminación en el anterior teorema de la hipótesis de calculabilidad polinómica del censo exigiría modificar la propia estructura del algoritmo utilizado para demostrar el teorema de Berman.

5.2. El teorema de Mahaney.

Nos sería preciso para nuestro resultado, siguiendo la línea del algoritmo del teorema de Berman, una función que nos asegure, primero, que dos fórmulas con el mismo valor asociado son equisatisfactibles, por un lado y tal que el conjunto de valores asociados a fórmulas insatisfactibles esté polinómicamente acotada, por otro. En el teorema inicial, así como en el de la anterior subsección, el primer hecho quedaba garantizado por utilizar una reducción, y el segundo gracias, en un caso, a ser una función esparsa, y en el otro caso, a conseguir que la función redujera $\overline{SAT}$ a un conjunto esparso. En este último caso, esta reducción se obtenía a partir de una reducción de SAT a S, y por tanto de $\overline{SAT}$ a $\overline{S}$, y consiguiendo que $\overline{S}$ fuera reducible al conjunto esparso S.

Este último paso es el que no podremos dar sin la hipótesis adicional de computabilidad del censo. La idea para la demostración es que, aún sin conocer el censo exacto, el que está acotado polinómicamente nos permite "ir probando" con todos los valores posibles del censo, con la garantía de que la vez que utilicemos el censo real el algoritmo va a funcionar correctamente.

Así, podremos construir una máquina polinomial no determinista M que con el valor exacto del censo como dato reconoce justamente $\overline{S}$, y utilizarla para construir a través de L(M) una función de $\overline{SAT}$ a S para cada valor posible del censo, en la misma forma en que en la anterior subsección la función se construia a través de $\overline{S}$. Estas funciones no podrán ser reducciones, pues en principio es posible que $\overline{SAT} \in NP$. Pero nos conformaremos con que cumplan la hipótesis de equisatisfactibilidad de fórmulas con el mismo valor asociado, para fórmulas "pequeñitas", menores que el dato inicial; entre éstas, todas las instancias a explorar.

Formalizaremos la construcción de estas funciones en un lema previo. Sean:

p, un polinomio que acota el censo de S: $C_s(n) \leq p(n) \quad \forall n$;

M_S, una máquina no determinista que reconoce S en tiempo polinómico;

f, una función de reducción SAT $\propto$ S;

r, un polinomio no decreciente que acota el tiempo necesario para calcular f.

Enunciemos un lema cuya demostración se verá más adelante.

Lema. *Existe una familia de funciones* $L_{n,k}: \Sigma^* \to \Sigma^*$, *todas ellas calculables en tiempo polinómico, y un polinomio no decreciente q, tales que para todo m, y para toda fórmula F de longitud menor o igual que m, la función* $L_{n,k}$ *correspondiente a los valores* $n=r(m)$ *y* $k=C_s(n)$ *verifica:*

i) Para cada fórmula F' de longitud menor o igual que m, $F' \in \overline{SAT} \iff L_{n,k}(F') \in S$;

ii) $card(L_{n,k}(\Sigma^{*n} \cap \overline{SAT})) \leq p(q(3r(m)))$.

Es decir, el "acertar" con el valor auténtico del censo $k=C_s(n)$, la función $L_{n,k}$ cumple cuanto necesitamos: equisatisfactibilidad de fórmulas con el mismo valor asociado, por i), y cota polinómica de la cantidad de valores asociables a fórmulas insatisfactibles, por ii). Esto sólo vale para fórmulas de longitud menor o igual que m, pero no es preciso más: las instancias de la fórmula inicial son así.

En base a este lema, el teorema de Mahaney se sigue con facilidad.

Teorema. *Si existe un conjunto esparso S que sea NP-completo, entonces P=NP.*

Demostración del teorema usando el lema. Basta considerar una iteración del algoritmo de la sección 4.1 para los diversos valores de k:

```
Algoritmo para decidir si F es satisfactible es
    (Sea m la longitud de F)
    Calcular n:=r(m);
    Para cada k entre 0 y p(r(m))
        iterar
          analizar F según el algoritmo 3.1,
           utilizando L_{n,k} para calcular los
           valores asociados, hasta analizar
           no más de (m+1).p(q(3r(m))) instan
           cias.
          salir si se ha encontrado una asigna
           ción que satisface F;
           de lo contrario, pasar al siguiente k;
          fin del iterar;
    si se ha encontrado asignación
          entonces responder "F es satisfactible"
          si no responder "F es insatisfactible"
    fin del si
fin del algoritmo.
```

La máquina M precisará como entrada 3-tuplas de la forma ($\$^n$, s, k), donde n es un entero expresado sobre el alfabeto monoliteral {\$}, k es un entero expresado sobre Σ (por ejemplo en binario, utilizando del orden de log k bits), y s una palabra de Σ^*. Sobre tal entrada realizará el siguiente algoritmo no determinista:

```
Algoritmo M es
    Calcular p(n);
    si k>p(n) o |s|>n
        entonces responder "no"
        si no
          conjeturar k palabras distintas, s_1...s_k,
            de Σ*, tales que |s_i| ≤ |s|   i;
          comprobar que s_i ∈ S  ∀i, mediante M_s;
          comprobar que s ≠ s_i  ∀i;
          si todo ello se cumple
             entonces responder "si"
          fin del si
    fin del si
fin del algoritmo.
```

Este algoritmo no determinista, claramente polinómico, acepta toda entrada con $k<C_s(n)$, rechaza toda entrada con $k>C_s(n)$ y con $k=C_s(n)$ acepta si y sólo si $s \notin S$. Es decir, si en los datos se acierta con el censo, se reconoce $\overline{S}$; pero puesto que la máquina M funciona polinómicamente, $L(M) \in NP$. En realidad, L(M) contiene toda la información sobre $\overline{S}$ que admite reconocimiento polinómico. Estamos en condiciones de pasar a la demostración del lema. Puesto que $L(M) \in NP$, y S es *NP*-completo, $L(M) \propto S$. Sea g la reducción, y sea q un polinomio no decreciente que acota el tiempo de cálculo de g.

Consideremos las funciones

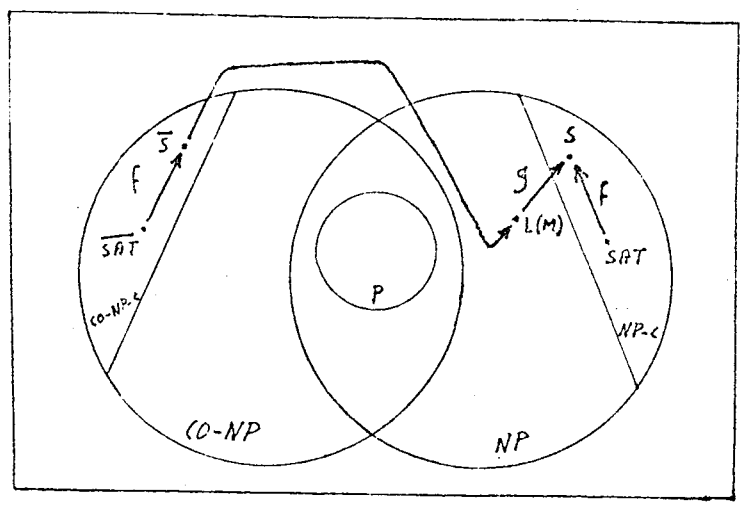
$$L_{n,k}(F) = g(\$^n, f(F), k)$$

Para el valor adecuado de k, el algoritmo se comporta exactamente como el de 4.1, por lo que de existir una asignación que satisface F debe ser encontrada con $(m\ 1).p(q(3r(m)))$ análisis de instancias, a lo más. Luego si la fórmula es satisfactible, este algoritmo así lo indica. Es obvio que funciona en tiempo polinómico, pues se limita a repetir una cantidad polinómica de veces un buble polinómico.

Con este algoritmo podríamos decidir SAT, y así *P=NP*.

□ *Teorema.*

Tan sólo queda pendiente la demostración del anterior lema. Definiremos una máquina no determinista, M, que nos servirá como escalón intermedio para construir las funciones $L_{n,k}$ que serán definidas siguiendo un esquema del siguiente tipo :



claramente calculables en tiempo polinómico. Se tiene para $|F| \leq m$, $n = r(m)$, $k = C_S(n)$:

$$F \in \overline{SAT} \Leftrightarrow f(F) \notin S \Leftrightarrow M_S \text{ rechaza } f(F) \Leftrightarrow$$

$$\Leftrightarrow M \text{ acepta } (\$^n, f(F), k) \Leftrightarrow g(\$^n, f(F), k) \in S,$$

luego $F \in \overline{SAT} \Leftrightarrow L_{n,k}(F) \in S$. Más aún, puesto que $|L_{n,k}(F)| \leq q(3n)$, ya que las tuplas $(\$^n, f(F), k)$ pueden escribirse en 3n símbolos, y siempre - teniendo en cuenta que q y r son decrecientes,

$$\text{card}(L_{n,k}(\overline{SAT} \cap \Sigma^{*n})) \leq \text{card}(S \cap \Sigma^{*q(3n)}) \leq p(q(3n)) =$$
$$= p(q(3r(m))),$$

ya que por hipótesis S es esparso.

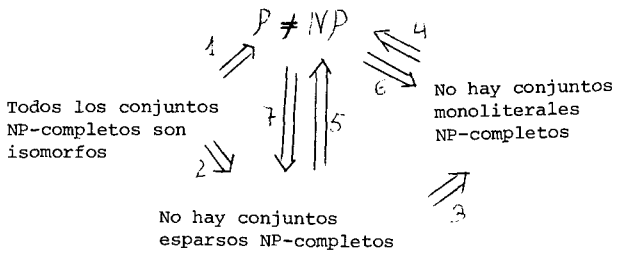
Por tanto, las funciones $L_{n,k}$ cumplen las --- aserciones del lema, con lo cual éste queda - demostrado.

□ *Lema*.

Todo ello completa la demostración del teorema de Mahney. Así pues, salvo que $p = NP$, no existen conjuntos esparsos *NP*-completos.

5.3. A modo de resumen.

Las implicaciones aquí estudiadas pueden resumirse, finalmente, en la siguiente figura:



Las implicaciones no obvias de las aquí ilustradas han sido demostradas en el texto: 1), 1) en 2.0; 2), en 2.3; 3) es obvia, así como 4) y 5); 6) es el teorema de P. Berman en 3.0; y 7) es el de Mahaney en 4.2. La implicación recíproca a 1) es la conjetura de isomorfía - de Berman y Hartmanis.

Esta conjetura es hoy por hoy uno de los problemas abiertos de más interés en la teoría - de complejidad; hemos expuesto aquí algunas - de las consecuencias a que se ha dado lugar - el estudio de dicha conjetura. No cabe duda - de que aún deberá en un futuro próximo motivar nuevos resultados que quizás resulten fundamentales para comprender con detalle la estructura interna de los conjuntos *NP*-completos.

6. BIBLIOGRAFIA.

/1/ BALCAZAR, J.L.: "Clases elementales de complejidad polinomial", RT 82/04. Fac. d'Informática de Barcelona.

/2/ BERMAN, P.:"Relationship between density and deterministic complexity of NP-complete languages", ICALP'78.

/3/ BERMAN, L.; HARTMANIS, J.: "On isomorphisms and density of NP and other complete sets", SIAM J. Comp. 6, 305-322.

/4/ BOOK, R.V.: "Comparing Complexity Classes", J.Comp. Syst. Sci. 9, 213-229.

/5/ BOOK, R.V.: "Tally languages and complexity classes". Inf. and Control, 26, -- 186-193.

/6/ BOOK, R.V.; WRATHALL, C.; SELMAN, A.; -- DOBKIN: "Inclusion complete tally languages and the HB conjecture".

/7/ COOK, S.D.: "The complexity of theorem - proving procedures", 3rd ACM Symp. on th. of Comp., 1971.

/8/ DIAZ, J.: "Complejidad de problemas combinatorios". QUESTIIO v.2, n.1, 35-48.

/9/ FORTUNE. S.: "A note on sparse complete sets", SIAM J.Comp., 1979, 431-433.

/10/ GAREY; JOHNSON : "Computers and Intractability", Freeman S. Francisco 1979.

/11/ HARTMANIS, J.; BERMAN, L.: "On polyno--- mial time isomorphisms of complete sets" Theor. Comp. Sci. 3rd GI Conference; --- Lect. Notes in Comp. Sci., Vol.48, ----- Springer Verlag, 1-15.

/12/ HARTMANIS, J.; BERMAN, L.: "On polyno--- mial time isomorphisme of some new complete sets", J. Comp. Syst. Sci., 16. -- 418-422.

/13/ HOPCROFT; ULLMAN: "Introduction to automata theory, languages, and computation" Addison-Wesley, 1979.

/14/ LADNER, R.E.; LYNCH, N.; SELMAN, A."Comparison of polynomial time reducibilities", 6th ACM Symp., 1974, 110-120.

/15/ MAHANEY, S.R.: "Sparse complete sets for NP; solution of a conjecture by Berman and Hartmanis", Dpt. of Comp. Sci. Cornell University, TR 80-417.

/16/ ROGERS, H.: "Theory of recursive functions and effective computability", McGraw-Hill, N.York, 1967.