UN ALGORITMO DE INTELIGENCIA ARTIFICIAL

Claudio Gutiérrez

INTRODUCCION

Inteligencia artificial es un campo nuevo de investigación en que confluye el interés de personas adiestradas en muy diversas disciplinas: se dan ahí cita los psicólogos, los conocedores de la cibernética, los lingüistas, los ingenieros de sistemas, los lógicos, los fisiólogos y, ¿por qué no? , los filósofos. Intenta ser un comienzo de respuesta a inquietantes preguntas planteadas por el advenimiento de los computadores, a saber: ¿cuál es la naturaleza y cuáles los límites del fenómeno similar a la inteligencia humana desplegado por el comportamiento de los computadores? ¿Ha dejado la inteligencia humana de ser un fenómeno único, en algún sentido definitivamente importante? ¿Será posible asimilar algún día los fenómenos intelectuales a fenómenos estrictamente físicos, presumiblemente electromagnéticos? ¿O por el contrario existe en la inteligencia humana un fondo irreductible a cualquier forma de mecanismo, lo que podríamos llamar la creatividad o potencia heurística de la razón? Cualquiera que sea la posición de cada uno sobre estas cuestiones radicales, todos los investigadores coinciden en reconocer que los procesos que se requieren para realizar incluso las más simples de las rutinas asociadas con la inteligencia humana son extremadamente complejos y que los esfuerzos necesarios para replicarlos con las máquinas exigen prodigiosas capacidades de almacenamiento de información y muy complicadas interconexiones de los elementos correspondientes. Pero coinciden igualmente en pensar que a pesar de lo difícil del problema, el sendero de búsqueda estará jalonado por muchos resultados secundarios que contribuyan a hacer valedero el intento, incluso en el caso de que las preguntas fundamentales debieran quedar sin contestación: una mejor comprensión de la naturaleza de la inteligencia humana, mejores vías de comunicación entre el hombre y las máquinas, diseño de máquinas más capaces y versátiles, mayor entendimiento de los fenómenos del lenguaje y de la percepción, etc.

La inteligencia artificial como campo científico de investigación es un acontecimiento reciente. Si dejamos aparte el trabajo de algunos pioneros, como McCulloch y Pitts (**11**), quizá podría situarse su fecha oficial de nacimiento en 1950, con la publicación del celebrado artículo de A.M. Turing en que propone una prueba para decidir si una máquina es o no inteligente (**22**). El campo no llega a poblarse de suficiente número de investigadores y proyectos sino en la década de los sesenta, con la aparición de los trabajos de Newell, Simon y Shaw (**14, 15**), Slagle (**20**), Robinson (**19**), Fogel, Owens y Wash (**5**), Amarel (**1**), y Minsky y Papert (**12**). No es sino en 1970 que comienza a publicarse una revista dedicada exclusivamente al tema, llamada precisamente *Inteligencia*

Artificial (1); con ello podemos dar el campo de investigación como definitivamente consolidado.

El campo de la inteligencia artificial comprende proyectos de investigación de índole diversa; podemos distinguir en particular proyectos que corresponden a alguno de los siguientes tópicos:

a) solución de problemas (subtópico: prueba de teoremas);
b) reconocimiento de formas (visuales o auditivas);
c) almacenamiento y readquisición de información;
d) manejo de lenguajes naturales u ordinarios.

El presente trabajo se referirá en lo que sigue exclusivamente al tópico *solución de problemas*; el proyecto de investigación que se describirá después corresponde al subtópico *prueba de teoremas*.

CONCEPTOS GENERALES

Para resolver un problema de cualquier clase, el primer paso es formularlo adecuadamente, o como se dice en matemáticas, *plantearlo*. Un buen planteo de un problema tiene que tomar en cuenta varias cosas. En primer lugar debe establecerse con claridad cuál es la situación original de que se parte, las llamadas codiciones iniciales (premisas, en el caso de prueba de teoremas). En segundo lugar, debe establecerse el tipo de situación a que se desea llegar, el estado final o solución del problema (conclusión, en el caso de prueba de teoremas). Pero además debe quedar claro cuál es el conjunto de medios, u operadores, mediante cuya aplicación recursiva a las condiciones iniciales el problema deberá ser resuelto. Si esas aclaraciones no pueden darse, diremos que el problema está mal formado, y en la situación actual de la investigación será considerado como exterior al campo de la inteligencia artificial: la transformación de un problema mal formado en uno bien formado parece ser todavía monopolio absoluto de la inteligencia natural (2).

Los tres elemetos mencionados, *situación inicial, situación final y operadores* para transformar situaciones, son la base para el planteo de un problema; no obstante, son necesarios otros dos elementos, de carácter más abstracto, para que un aparato lógico determinado pueda contribuir a la solución de problemas. En primer lugar, es necesario introducir el concepto de *espacio* de situaciones o estados posibles, entre los cuales figurarán la situación inicial y la final; en segundo lugar, precisamos también del concepto de *búsqueda* o exploración en ese espacio, que nos permita encontrar un camino que lleve a través de estados intermedios del estado inicial al final. En ese camino cada estado posterior estará determinado por un estado anterior y por la aplicación a éste de los operadores transformadores de estados. El espacio de estados puede representarse como un gráfico cuya raíz sea el estado inicial, de donde salgan distintos caminos (sucesión de estados), algunos de los cuales terminan en un estado final (si existe solución al problema). Solucionar un problema consistirá en mostrar una sección de ese gráfico que

---

(1) El *Journal of the Association for Computing Machinery* y el IBM *Journal of Research and Development* publican ocasionalmente material de inteligencia artificial.

(2) Nos abstenemos de entrar aquí en la cuestión filosófica de si antes de su planteo estricto una "situación incómoda" puede considerarse realmente un problema, o si toda problematización de una situación ya implica un cierto grado de formalización.

presente un camino (de preferencia el más corto o el de mínimo costo) del estado inicial a uno final. Desde otro punto de vista, la solución será una secuencia de operadores que transforme un estado inicial en un estado final (3).

La selección de una buena representación para los estados del espacio es parte del proceso de planteo, o sea, de la transformación de un problema mal formado en uno bien formado. Todavía no se conocen métodos automáticos para ayudar en ese trabajo, que por el momento, como dijimos, queda fuera del campo de inteligencia artificial. En cambio, la búsqueda dentro de los gráficos de espacio de estados, para encontrar un camino, o el camino más corto o económico, cae de lleno dentro de los esfuerzos actuales de los investigadores. Se pueden reconocer dos grandes métodos de búsqueda: el método de *anchura–primero* y el método de *profundidad–primero.* Si llamamos *desarrollar un estado* al acto de aplicarle los operadores para producir en el espacio de estados sus estados sucesores, diremos que el método anchura–primero desarrolla los estados en el orden en que estos van siendo generados; en tanto que el método profundidad–primero da preferencia para el desarrollo al estado no desarrollado más recientemente producido. En este último método es necesario establecer un *límite de profundidad* más allá del cual el algoritmo no explorará sino que se devolverá a buscar el próximo estado no desarrollado más recientemente producido que no caiga bajo ese límite (4). A continuación presentamos un algoritmo general con variantes en los dos métodos:

Resuelve–un–problema (5)

Coloca la descripción del estado inicial en la lista abierta
Repite aplica–operadores
hasta un estado sucesor es estado final.
Imprime el camino encontrado de estado inicial a estado final y termina

Aplica–operadores (variante de anchura–primero)

Convierte primera línea de lista abierta en última de lista cerrada; llama a esa línea N.

Desarrolla $estado_N$ generando todos sus posibles sucesores por aplicación de operadores y colócalos en orden de generación *al final* de la lista abierta; anota en cada

(3) (9), p.4.

(4) Para un amplio tratamiento de este tema véase (16), capítulo III.

(5) Se supone que todo estado tiene sucesores: si no existe solución, el algoritmo procederá indefinidamente. Con espacios que se agotan, debe proveerse una "salida con fracaso" para el caso de que la lista abierta llegue a estar vacía.

línea nueva de la lista una referencia a la línea N (6).

Aplica–operadores (variante de profundidad–primero)

Convierte primera línea de lista abierta en última de lista cerrada; llama a esa línea N.
Si : profundidad de estado $_N$ bajo límite,
recomienza esta secuencia;
: de otro modo,
continúa.

Desarrolla estado$_N$ generando todos sus posibles sucesores por aplicación de operadores y colócalos en orden de generación *al comienzo* de la lista abierta; anota en cada línea nueva de la lista una referencia a la línea N (6).

Parece conveniente introducir a esta altura de la exposición los conceptos de *admisibilidad y optimalidad* de un algoritmo de búsqueda de solución de problemas. Digamos que en general un algoritmo de búsqueda es admisible si, para cualquier gráfico de estados, el algoritmo termina descubriendo el camino más corto al estado–meta, siempre que ese camino exista (7). Si el camino se mide por el número de generaciones que separan el estado final del inicial, es fácil ver que un algoritmo de anchura–primera será admisible en este sentido, mientras que uno de profundidad–primero no lo será. Por otro lado, un algoritmo de búsqueda es óptimo si, para cualquier gráfico de estados, no sólo siempre termina descubriendo el camino más corto a un estado–meta, sino que su búsqueda, expresada en número de estados desarrollados, será la más corta en comparación con búsquedas de otros algoritmos admisibles dotados de la misma información (8).

Los dos métodos mencionados son métodos de búsqueda ciega; es decir, cada uno a su manera, ambos proceden a explorar todas las posibilidades dentro del espacio de estados hasta encontrar la solución, si ella existe. No es difícil darse cuenta de que tal búsqueda ciega, aunque infalible en su propósito, puede muchas veces, en presencia de problemas de cierta complejidad, ser impracticable por su alto costo, por ejemplo por la enormidad del tiempo que requiera. En este punto se sitúa la comparación más frecuentemente realizada, a propósito de la inteligencia artificial, entre el hombre y la máquina: la búsqueda ciega es infalible, a diferencia de la búsqueda intuitiva humana, porque dándole el tiempo necesario encuentra la solución, si ella existe; pero por otra parte, se argumenta, el inmenso consumo de tiempo y de otros recursos hace que esta aparente superioridad de la máquina se desvanezca en un completo ridículo. La búsqueda de la máquina, se dice, es algorítmica o repetitiva; mientras que la del hombre es heurística, creativa o ilustrada. No obstante, precisamente los más valiosos esfuerzos de los investigadores del campo de la inteligencia artificial están hoy concentrados en desarrollar las potencias creativas o heurísticas de la máquina, en realizar esa aparente paradoja de un *algoritmo heurístico.* Que lo que muchas mentes filosóficas tienden a

---

(6) Estas anotaciones permitirán, al encontrarse un estado final, retrazar el camino recorrido y proceder a su impresión como solución del problema.

(7) (16), p.59. El autor toma en cuenta aquí consideraciones de costo, diferentes de la simple distancia entre el estado inicial y el final, de las que prescindimos nosotros por no ser atinentes a nuestro tema.

(8) (16), p.61. El concepto de "misma información", aunque suficientemente claro para la intuición, es considerablemente difícil de definir. El intento de Nilson de definirlo, p.62, no parece enteramente satisfactorio.

considerar esencialmente imposible, algo así como la cuadratura del círculo, se haya ya producido en parte, es algo que nos debe enseñar a desconfiar de las "evidencias racionales" de los filósofos. En efecto, se ha demostrado que es posible adicionar a los métodos de búsqueda ciega con diversos procedimientos encamiados a ahorrar pasos y a dejar que, por así decirlo, la meta atraiga al algoritmo hacia el camino correcto. Uno de tales procedimientos sería el que evita que el algoritmo desarrolle todos los estados producidos, limitando su trabajo al de aquellos estados que ofrecen alguna promesa. Este procedimiento es aplicable tanto al método de profundidad–primero como al de anchura–primero. Además, ha sido posible reforzar el método de profundidad–primero intercalando en la secuencia aplica–operadores una instrucción inicial para reordenar la lista abierta de modo que los estados quedan en orden decreciente de la magnitud de alguna *función de evaluación* medida de la promesa heurística del estado. Tales funciones no son particularmente difíciles de concebir con cierto grado de generalidad aplicable a tipos de problemas, y su elaboración se basa en el conocimiento de la naturaleza de los problemas concomitante a la percepción misma de sus condiciones; dicho de otra manera, la información necesaria para establecer la función de evaluación no requiere ser idéntica con el conocimiento previo del camino del estado inicial al estado final, ni tampoco mayor que la que el hombre tiene de las circunstancias definitorias del problema en el momento de su planteo.

En general, ya se ha demostrado en multitud de casos la posibilidad de especificar procedimientos heurísticos que reducen substancialmente el esfuerzo de búsqueda sin que se sacrifique por ello la garantía de que se encontrará eventualmente el camino más corto a la meta. Por supuesto, existe también la posibilidad de incrementar grandemente la efectividad de búsqueda sacrificando aquella garantía, con lo que no se pondría a la máquina en desventaja frente al hombre, pues la falta de garantía es precisamente una de las características de los esfuerzos heurísticos humanos. En la mayor parte de los problemas prácticos que deseamos resolver con ayuda de la máquina, sin embargo, lo que nos interesa es minimizar cierta combinación de la extensión del camino entre los dos estados y el costo o esfuerzo de la búsqueda misma, es decir, encontrar una transacción entre los atributos de admisibilidad y optimalidad explicados antes. Por otra parte, probablemente lo que lo que más interese sea ante todo minimizar la combinación de esos atributos en el promedio de los casos de todos los problemas con que tendrá que enfrentarse el algoritmo (9).

Hasta ahora hemos considerado los métodos de búsqueda en relación con una forma determinada de representación del problema, la llamada *descripción en el espacio de estados.* Una representación alternativa, que implica una estrategia en cierta forma inversa a la descripción de estados, es la llamada *reducción de problemas.* Si la estrategia de descripción de estados parte de un estado inicial que se desarrolla hasta producir uno final, la reducción de problemas consiste en razonar hacia atrás y preguntarse cuál situación existiría inmediatamente antes de terminar de resolver el problema, cuál situación un paso antes de esa situación prefinal, y así sucesivamente, reduciéndose en último término el problema original a muchos subproblemas elementales, de solución obvia, encadenados entre sí. De manera más directa, definamos la reducción de problemas como el proceso por el cual un problema original es substituido por un conjunto de problemas, cuya solución es equivalente a la solución del problema original, de los cuales cada uno, o es un problema elemental (de solución obvia) o puede por ulteriores reducciones transformarse en problemas elementales. En este caso la situación inicial del proceso será una descripción del problema original; la situación final, un conjunto de

---

(9) (16), p.54.

descripciones de problemas elementales; y los operadores serán maneras prescritas de transformar una descripción de problema en varias descripciones de subproblemas, más sencillos que el que sirve de base a la aplicación de los operadores (10). Los métodos de anchura–primero y profundidad–primero son en principio tan aplicables a la búsqueda en un espacio de problemas como a la búsqueda en un espacio de estados; las dos estrategias, sin embargo, deben distinguirse. La estrategia de reducción de problemas es especialmente digna de mención en relación con la prueba de teoremas, pues ha sido el enfoque usado por los investigadores asociados al importante proyecto *General Problem Solver* (11).

Un último enfoque que debe mencionarse a propósito específicamente de la prueba de teoremas es el de *resolución semántica*. Dicho enfoque se basa en el concepto de universo de Herbrand (12) y en la aplicación que de ese concepto hicieron Prawitz, Robinson y Kowalsky–Hayes a una lógica orientada al trabajo de máquina (13). En esencia, esta prueba de teoremas usa el procedimiento de interpretar, de manera totalmente general, el conjunto formado por las premisas y la negación de la conclusión, que son formas de enunciados, con el objeto de mostrar que no puede satisfacerse: ninguna interpretación da lugar a premisas verdaderas y conclusión falsa. El concepto de resolución entra en juego en cuanto que, en los gráficos de refutación que origina la aplicación del procedimiento interpretativo, dos cláusulas contradictorias entre sí se anulan recíprocamente y sirven para inferir una cláusula derivada, su resolvente. Sucesivas resoluciones sirven para hacer desaparecer progresivamente todo el gráfico, demostrando que la clase de interpretaciones posibles es la clase vacía–en el caso de que el teorema sea válido. Es importante hacer notar que este enfoque es esencialmente semántico, no sintáctico; se liga a la *interpretación* de las formas de enunciado, que les da un contenido informativo determinado; en tanto que el enfoque sintáctico consiste en una *axiomatización* que liga las formas de enunciado unas con otras sin darles contenido, por reglas de transformación estrictamente formales. Para tratar de que las máquinas emulen el razonamiento humano pareciera más apropiado usar el enfoque sintáctico, tradicionalmente empleado por los lógicos y matemáticos, aun cuando en la práctica probablemente el enfoque semántico haya ya demostrado ser bastante más eficiente (14).

EL PROYECTO

Durante el año 1969, y estimulado por el apoyo que para ello le ofrecieron el Director del Departamento de Filosofía de aquel entonces, Lic. Víctor Brenes, y la Directora del Centrao de Cálculo, Ing. Clara Zomer, en la Universidad de Costa Rica, este investigador elaboró un algoritmo de búsqueda ciega, tipo anchura–primero, para probar teoremas de la lógica de cuantificación uniforme de primer orden. Tal algoritmo poseía, no obstante su carácter básicamente no heurístico, dos mecanismos para evitar desarrollos innecesarios: el primero consistía en una subrutina para eliminar repeticiones que hacía al algoritmo abstenerse de producir una línea adicional en la lista de teoremas demostrados si una fórmula idéntica ya estaba en la lista con la misma fuerza de verdad, es decir ubicada en el mismo nivel hipotético (15); el segundo consistía en una subrutina para

---

(10) Para una explicación detallada de este enfoque puede consultarse (16), capítulo IV.

(11) (2), (3), (4) y (15).

(12) (9) y (16), capítulo VI.

(13) (10), (17) y (19).

(14) Aplicaciones recientes de este enfoque se pueden encontrar en (21) y (23).

(15) Más adelante se aclara este concepto.

limitar los teoremas producidos a sólo aquellos que fueran estrictamente atinentes, según los linderos prescritos por el texto de las premisas y la conclusión. Es de notar que el algoritmo usaba un tipo de planteamiento que no correspondía estrictamente ni a la descripción de estados ni a la reducción de problemas, pero que podríamos asimilar a una descripción de estados convenientemente simplificada para adaptarla a las necesidades de la prueba de teoremas. En efecto, si se adoptara el planteamiento de descripción de estados para este tipo de función inteligente, cada estado debería incluir una formulación completa de todas las formas de enunciado, premisas o conclusiones intermedias, que en ese momento estuvieran "sobre el pizarrón", con indicación exacta de su status lógico (verdad incondicional, hipótesis de primer nivel, o hipótesis de algún nivel más alto). Cada línea de la lista de estados, desarrollados o por desarrollar, debería incluir todas las formas de enunciado vigentes entonces, cuyo número iría creciendo, con cada aplicación de los operadores, en proporción geométrica. De ahí que optáramos, sin abandonar completamente el enfoque de descripción de estados, por incluir en cada línea no el inventario completo de toda la situación sino únicamente la fórmula nueva producida en la última aplicación de algún operador, con lo que la lista de teoremas venía a contener una abreviatura de las descripciones de estados en el espacio lógico.

En aquella época formulamos los objetivos del proyecto de investigación de la siguiente manera: "El proyecto tiene por objeto enseñar al computador IBM 1620 a probar teoremas de lógica de cuantificación uniforme, incluyendo desde luego su base la lógica de proposiciones. Tal enseñanza a su vez tiene por propósitos-- (a) investigar por analogía en los procesos del razonamiento humano; (b) investigar las potencialidades del computador para el cálculo no numérico; (c) proveer eventualmente una herramienta para la investigación lógica y un instrumento de demostración para la enseñanza de la lógica; y (d) contribuir eventualmente en la inmensa empresa de mejorar las posibilidades de comunicación entre el hombre y las máquinas pensantes". Al evaluar los resultados de aquella etapa del proyecto, que duró aproximadamente seis meses y consistió en la programación y corrida del algoritmo en lenguaje FORTRAN, el investigador señalaba la facilidad con que el programa probaba los teoremas de lógica de proposiciones, la mediana facilidad con que probaba teoremas simples de cuantificación uniforme (silogismos, por ejemplo) y la imposibilidad práctica de terminar la prueba de un problema difícil, el considerado por Quine una empresa de vacaciones (*holyday venture*) (16). Con respecto a este último problema, el programa estuvo corriendo el 31 de diciembre de 1969 por espacio de doce horas, y hubo de ser detenido al comenzar el año 1970 cuando había explorado sólo una pequeñísima parte del inmenso espacio de estados a que daban lugar las hipótesis consideradas atinentes para la prueba; se calculó que el programa, sin ayuda humana, duraría quinientas horas para encontrar una solución; con ayuda humana—que hiciera arrancar el programa en una etapa evolucionada del desarrollo de las combinaciones de hipótesis—ese tiempo podía tal vez reducirse a una décima parte.

Frente a esa dificultad, el investigador manifestaba al rendir su informe de labores haber pensado en dos posibles medios que, sin imponer prejuicios heurísticos al computador, permitieran seleccionar ciertas combinaciones de hipótesis de entre la inmensa multitud de posibilidades, como las más prometedoras: un método probabilístico y un método constructivo. Con respecto al primero, escribía: "Existe la posibilidad de capacitar al computador para que 'aprenda por experiencia', usando un viejo principio de Hume, la asociación de ideas. Consistiría en hacerle apuntar los pares ordenados de hipótesis que resulten fecundos en refutaciones, cuando la búsqueda trabaje con dos

---

(16) (18), p.175.

niveles de hipótesis; cuando la búsqueda trabaje con tres niveles, seleccionar tríos ordenados que resulten fecundos y así sucesivamente. Este procedimiento consumiría mucha memoria y, lo que es más grave, mucho tiempo en el cálculo de la función heurística (anotar pares ordenados, tríos ordenados, etc.)". Con respecto al segundo método, consistiría en poner al computador a desarrollar todas las contradicciones posibles atinentes a las premisas; después, mediante un procedimiento que habría que idear, levantar las contradicciones halladas a diversos niveles de hipótesis para construir de tal modo el árbol de prueba. Este último método tendría un carácter eminentemente analítico, racional o a priori, mientras que el primero tendría un carácter eminentemente empírico, a posteriori o experimental. En lo que sigue llamaremos al primer método *variante de orientación empírica,* ya que el algoritmo se orientará en la búsqueda por los resultados de su propia experiencia; y al segundo, *variante de análisis racional,* por no evaluar resultados empíricos sino descansar totalmente en el examen del planteo y en la aplicación de técnicas estrictamente deductivas.

Es de notar que las dos posibles avenidas de solución en que pensó el investigador al evaluar los resultados desfavorables del algoritmo de búsqueda ciega fueron ideas sueltas o fortuitas, las dos únicas que se le ocurrieran como practicables en ese momento; no deja de ser interesante, entonces, la coincidencia de que ellas correspondan a dos campos de reflexión epistemológica tan definidos y característicos como son la investigación empírica y la reflexión racional. Además, es fácil darse cuenta de que las dos posibilidades agotan por completo las alternativas de lo que un hombre puede hacer creativamente para enfrentarse a un problema, dado su carácter totalmente general: o bien tratar de encontrar su solución por ensayo y error, llevando cuenta de los puntos de acierto en los tanteos fracasados para orientar los intentos futuros; o bien tratar de encontrar la solución mediante un desarrollo teórico del alcance del problema y un análisis subsecuente que por eliminación de posibilidades conduzca deductivamente a una respuesta correcta. Si podíamos, entonces, hacer a la máquina pensar con alguna de estas dos estrategias, o con las dos en combinación, estaríamos produciendo una emulación electromecánica de los procesos heurísticos humanos más característicos: el razonamiento por inducción y el razonamiento deductivo. Esa era la situación del proyecto a principios de 1970, cuando la investigación fue interrumpida, fundamentalmente por haberse agotado la capacidad del computador que se tenía disponible (17).

No fue sino durante el primer semestre de 1973 que la investigación pudo ser continuada, esta vez en la ciudad de Las Cruces, New Mexico, usando las instalaciones de la New Mexico State University donde el investigador se encontraba en calidad de Profesor Visitante, y gracias a la colaboración de sus autoridades, muy en especial del Director del Departamento de Ciencias de la Computación Prof. J. Mack Adams (él mismo investigador en el campo de la inteligencia artificial). Se contó entonces con una máquina IMB 360–65 y se programó en lenguaje ASSEMBLER. Gracias a la inspiración del Profesor Adams, fue posible identificar el procedimiento para "levantar las contradicciones" a diversos niveles de hipótesis que era necesario para concretar la variante analítica del algoritmo; dicho procedimiento, se decidió, vendría a ser una aplicación del principio de reducción de problemas explicado antes. La producción de contradicciones se haría por el simple expediente de aplicar las reglas de transformación a

---

(17) En aquel entonces la IBM 1620 propiedad de la Universidad de Costa Rica contaba sólo con 40.000 posiciones de memoria, sin el recurso de memoria adicional de acceso libre que se le agregó después.

las premisas activas en el correspondiente nivel; la progresión en los niveles de hipótesis ocurriría, a partir del primer nivel, cada vez que en uno de los dos campos en que la prueba estuviera bifurcada surgiera una contradicción, ramificándose a su vez en ese momento la prueba en el otro campo. Este proceso equivale a una reducción del problema anterior a dos problemas más sencillos: uno, el subproblema de encontrar una contradicción en una de las dos ramas de la prueba (problema elemental, pues encontrada la contradicción queda automáticamente refutado ese campo); y dos, el subproblema de refutar la otra rama de la prueba (presumiblemente por el procedimiento de bifurcarla de nuevo y repetir el proceso); eventualmente se encontrará una contradicción en ambos lados de la prueba, con lo que el programa habrá llegado a la meta (demostrar que las premisas y la negación de la conclusión son incompatibles, pues producen contradicciones en todos los campos del árbol de prueba). Por supuesto, si la contradicción puede producirse sin bifucarción alguna, en el campo principal o nivel hipotético cero, entonces el problema queda reducido a un problema elemental y solucionado trivialmente. La variante del algoritmo, *variante de análisis racional,* fue programada y corrida, con los resultados que al final se dirán. Por lo demás, y para efectos de completitud, el algoritmo de búsqueda ciega fue refinado para adaptarse a la posibilidad de variantes heurísticas, y se diseñó también una *variante de orientación empírica* que resultara más practicable que la concebida originalmente (muy dispendiosa de tiempo y memoria, como se explicó). Se explica esa variante en seguida.

La idea, inspirada en Hume, de anotar las asociaciones de hipótesis fructíferas, no es en principio impracticable si se cuenta con una máquina de gran velocidad e inmensa memoria. No obstante, postulamos la conjetura de que un resultado igualmente orientador puede lograrse, sin necesidad de anotar asociaciones, por el procedimiento de simple calificación de hipótesis posicionadas y reordenamiento de la lista de hipótesis cada vez que comienza una vuelta hipotética. Estos significa que el algoritmo trabajará sin orientación alguna en la primera vuelta, durante la cual se prueban todas las hipótesis atinentes, una por una, en el nivel uno de bifurcación. Durante esa primera vuelta el algoritmo anotará cuáles de las hipótesis son fructíferas en contradicciones y cuáles no lo son. En la segunda vuelta, cuando para cada hipótesis probada en primer nivel bifurcaciones sucesivas trabajan con todas las hipótesis en el segundo nivel, serán tanteadas primero en el primer nivel las hipótesis ya demostradamente fecundas en ese nivel; a su vez, las hipótesis serán de nuevo calificadas, en listas separadas para cada uno de los dos niveles, de modo que se tome esencialmente en cuenta la posición de la hipótesis en el momento de engendrarse una contradicción. Dicho de otro modo, lo que se califique no será una hipótesis en abstracto, sino una hipótesis en una determinada posición en la constelación particular que dé origen a una refutación de campo; se toma en cuenta el orden de las hipótesis sin necesidad de anotar los pares, tríos, etc..., con la consiguiente economía de memoria y tiempo; además el algoritmo comienza a actuar heurísticamente (reordenar hipótesis) ya antes de la segunda vuelta, y no después de esta como en la versión original. La variante descrita está definida en diseño, como puede verse en el algoritmo presentado al final de este trabajo, pero no ha sido programada ni corrida en la máquina; volveremos sobre ello al final.

Previamente a que presentemos formalmente el algoritmo en sus tres variantes, las de búsqueda ciega, búsqueda heurística con orientación empírica y búsqueda heurística por análisis racional, debemos decir algunas palabras sobre el sistema de lógica de cuantificación uniforme en que serán planteados los problemas de prueba de teoremas.

EL SISTEMA LOGICO

El sistema lógico que usaremos es una variante del sistema basado en las relaciones de dualidad cuyas versiones principales han sido expuestas en **(6)**, **(7)** y **(8)**. La presente variante está restringida al cálculo de cuantificación uniforme, es decir, de una sola variable; además, aunque sus reglas son suficientemente generales, la aplicación que de ellas hace el algoritmo está limitada a ramificaciones de prueba hipotética en una sola dirección (18) para efectos de simplificación de los cálculos. Esta limitación condena la variante a la incompletitud; sin embargo, el poder demostrativo de la misma se mantiene suficientemente grande y el algoritmo es capaz, a pesar de esta conveniente simplificación, de probar todos los teoremas tradicionalmente interesantes de esta rama de la lógica.

El sistema consiste de las siguientes reglas:

Reglas estructurales

1—Regla de premisas

La negación de la conclusión y las premisas originales si las hubiere constituyen la población inicial del campo principal de prueba o campo cero; representan las formas de enunciado que van a ser examinadas desde el punto de vista de su compatibilidad; si de su desarrollo resulta una contradicción, la conclusión puede válidamente inferirse del subconjunto formado por las premisas (19). Todo campo de prueba puede bifucarse, mediante el expediente de crear una alternativa formada por cualquier fórmula bien formada y su negación, puestas cada una como población única de uno de los campos secundarios abiertos por la bifurcación de la prueba. En tal caso, toda la población del campo antecedente es activa dentro de cada una de las ramas de la bifuración; pero desde luego, la fórmula creada en uno de los campos gemelos no será activa en la otra rama de la bifurcación, ni tampoco en el campo subordinante de ambas. Que una fórmula sea activa querrá decir que se puede contar con ella a la hora de aplicar las reglas de transformación; una fórmula A será activa sobre una fórmula B si y sólo si A y B se encuentran dentro de una misma rama de prueba, siendo el nivel de B no inferior al nivel de A (es decir, ambas fórmulas están en el mismo campo o B está en un campo subordinado al campo de A).

2—Regla de refutación

Si dos fórmulas contradictorias entre sí llegan a coexistir en el mismo campo, en el sentido de que ambas son activas en él, ese campo se dirá refutado; si dos campos gemelos están ambos refutados, refutan por ese mismo hecho el campo antecedente inmediato (el campo que se bifurcó para darles origen). Si el campo principal o campo cero de la prueba es refutado, la incompatibilidad entre la negación de la conclusión y las premisas originales ha sido hallada y por lo tanto el teorema ha quedado demostrado (la conclusión puede válidamente inferirse de las premisas originales).

---

(18) Cada vez que se bifurca una prueba se abren dos líneas de prueba secundarias dependientes del campo de prueba que se bifurca; de ese par de líneas, originadas por la apertura de dos campos nuevos de prueba de nivel hipotético mayor, solamente una se podrá bifurcar a su vez. El algoritmo reconoce al campo bifurcable como *campo positivo* y al no bifurcable como *campo negativo*.

(19) Ese conjunto puede estar vacío, si lo que se prueba es una tautología.

Reglas de transformación

3–Regla de introducción de functor

Se puede introducir un functor diádico (la conectiva de conjunción), con la función de unir dos fórmulas preexistentes activas entre sí; o bien monádico (el cuantificador universal), con la función de generalizar una fórmula preexistente. El resultado de la introducción del functor diádico será una fórmula nueva conjuntiva que comenzará a existir en el campo de la fórmula base de la aplicación. La regla de introducción del functor monádico sólo puede aplicarse en una línea continua de prueba; es decir, si se ha aplicado en una rama de la prueba no puede aplicarse también en la rama divergente.

Regla de eliminación de functor

Se puede eliminar un fuctor diádico (la conjunción); la regla permite reproducir las fórmulas que el functor une, ahora como fórmulas independientes. Se puede eliminar también un fuctor monádico (el cuantificador universal); la regla permite reproducir el radical de la fórmula base como fórmula independiente (desprovista de cuantificador). El resultado de la aplicación de esta regla será una o dos fórmulas nuevas que comenzarán a existir en el campo de la fórmula base de la aplicación.

El sistema emplea la siguiente notación:

Libertad de paréntesis

Hemos adoptado la llamada notación polaca, en que las fórmulas son filas de símbolos, diádicos, monádicos o básicos; el símbolo diádico, al aparecer en una fila de izquierda a derecha, "anuncia" que necesita completarse con dos subfórmulas completas inmediatas y consecutivas; el símbolo monádico, "anuncia" que necesita completarse con una subfórmula completa que le siga inmediatamente; el símbolo básico es una fórmula completa y satisface las necesidades de los functores diádicos monádicos que le anteceden, si alguno lo hace. Recursivamente, y de derecha a izquierda, si una fórmula es completa puede hacer completa a otra fórmula más larga y que la incluye inicializada por un functor (diádico o monádico). Hemos escogido como símbolo de la conectiva de disyunción la letra A (por Alternativa); los símbolos P, Q, R, etc... como representación de formas proposicionales no analizadas; los símbolos F, G, H, etc... como representación de funciones proposicionales de una variable (que queda implícita en la notación por razones de brevedad); y el símbolo E (por Existencia) como representación del cuantificador existencial (con la variable también implícita). Para explicar la manera en que se representa el cuantificador universal y la conectiva de conjunción necesitamos enunciar primero el modo en que vamos a representar la negación. El condicional se representará por la técnica definitoria común de reducirlo a negación y disyunción.

Negación digital

Tradicionalmente la idea de negación se ha expresado en lógica simbólica mediante un prefijo, la letra N o el signo menos por ejemplo, aplicado a la totalidad de lo que se desea negar. No obstante, creemos haber descubierto una alternativa mejor: la prefijación del signo de negación a cada uno de los elementos de lo que se desea negar; hemos bautizado a esta forma de negación "negación digital' por la obvia razón de que el signo de la negación afecta a cada uno de los dígitos de la fila de símbolos en que consiste la

base de la negación (20). Complemento muy importante de esta forma de notación es la exclusión en principio de la doble negación, de modo que la negación de –P sea simplemente P, y no––P (21).

La negación digital tiene la virtud de incorporar las relaciones de dualidad lógica dentro de la notación misma, siendo entonces innecesario expresarlas en forma de leyes; así por ejemplo las leyes de DeMorgan resultan inexpresables, lo que no es una pérdida sino una economía, pues se requieren menos principios lógicos para operar demostrativamente en la práctica. Además, toda una serie de pares de leyes se reducen cada uno a una sola ley, con la consiguiente simplificación de procedimientos: si tomamos el cuidado de integrar en una sola fórmula todas las premisas de un razonamiento, por el procedimiento de unirlas con el signo de conjunción, el par compuesto por la conclusión y la única premisa puede ser "reflejado en el espejo" es decir complementado a través de la negación digital, y se producirá otro razonamiento igualmente válido con premisa y conclusión contrapuestas. Por lo demás, la negación digital se presta soberanamente bien para la formalización en lenguaje de computación, dado el carácter último binario de la representación de información en la memoria de la máquina.

Tomando en cuenta el procedimiento de negación digital, no hace falta ningún signo nuevo para la conjunción y el cuantificador universal, pues estos conceptos son duales de la disyunción y el cuantificador existencial respectivamente. Así, la fórmula +A +P +Q puede considerarse (de conformidad con las leyes de DeMorgan) como la negación de la conjunción de –P y –Q, conjunción que podremos expresar como –A–P–Q. De aquí que optemos por representar el functor diádico de conjunción como –A. Igualmente sucede con la fórmula +E +F, que puede considerarse como la negación de la cuantificación universal de –F, cuantificación que podremos expresar como –E–F. De donde que optemos por representar el functor monádico de cuantificación universal como –E. En resumen, todos nuestros signos elementales, functores o argumentos, tendrán un signo positivo o negativo delante; la interpretación del mismo con respecto a los argumentos es obvia: +P es equivalente a "es el caso que P"; –P a "no es el caso que P"; +F a "es así que F se dice de x"; –F a "no es así que F se dice de x". En cuanto a los functores, +A será el signo de disyunción, +E el signo de cuantificador existencial, –A el signo de conjunción y –E el signo de cuantificador universal Los functores negativos, –A y –E, serán los únicos que puedan eliminarse o introducirse de acuerdo con las reglas 3 y 4.

EL ALGORITMO

El algoritmo mismo lo presentaremos en secuencias de instrucciones a la máquina sucesivamente más detalladas; el algoritmo completo aparecerá al principio bajo el título de secuencia *prueba–un–teorema*, que podríamos considerar como el nombre del algoritmo. Tal secuencia cubre las tres variantes del algoritmo, pues las diferencias entre los distintos métodos de búsqueda sólo se hacen presentes en un nivel de detalle mucho

(20) La negación digital tiene un modelo físico muy ilustrativo y de gran valor pedagógico: la reflexión en el espejo según el eje horizontal de las fórmulas. Consúltese al respecto (6), (7) y (8).

(21) En el modelo físico esta convención resulta obvia: la imagen en el espejo tiene la forma del objeto original.

más fino. La secuencia global *prueba–un–teorema* se desglosa en otras secuencias que se presentan subsiguientemente; así, en la secuencia global aparecen dos rubros secuenciales, es decir títulos de procesos programados más refinados, a saber *lee–una–premisa* y *trabaja–con–las premisas*. Ambas secuencias son explicitadas después, bajo los títulos correspondientes; ahí en esas secuencias parciales aparecerán nuevos rubros secuenciales de todavía mayor detalle, y así sucesivamente hasta llegar a un nivel en que todas las operaciones de la secuencia secundaria se describen directamente. Esperamos que esta manera de presentar las cosas ayude a comprender las ideas centrales sin que los pormenores hagan perder de vista el panorama del conjunto. Hemos agregado notas al pie para aclarar el sentido de algunos términos o la naturaleza de ciertos procesos.

El algoritmo es el siguiente:

Prueba–un–teorema

```
Repite      Lee–una–premisa
hasta       no más premisas.
Repite      trabaja–con–las–premisas
hasta       campo cero refutado.
Imprime la prueba y da por terminado el trabajo.
```

Lee–una–premisa

```
Lee una tarjeta (22).
Imprime lo ahora leído
Repite  verifica–carácter
hasta   no más caracteres.
   Si : fórmula mal formada.
        abandona protestando error;
      : de otro modo,
        continúa.
   Si : primera vez que pasas por aquí
        complementa FORMULA; (23)
           : en todo caso,
             continúa.
Inscribe–en–la–lista.
Da por terminada esta secuencia.
```

---

(22) La primera contiene la conclusión; las siguientes, cada una una premisa; la última un aviso de que no hay más premisas.

(23) La negación de la conclusión se toma como primera premisa. Complementar consiste en sustituir cada símbolo interno por otro que representa la misma letra pero con signo contrario. FORMULA es el lugar de la memoria donde se inscribe provisionalmente el resultado de la aplicación de cada regla: la secuencia verifica–carácter ha puesto allí la representación simbólica interna de lo que se leyó de la primera tarjeta.

Verifica--carácter

```
Si   :  carácter ilegal,
        abandona protestando error;
     :  de otro modo,
        Si :  esta pasada es vez impar,
              toma nota de signo y da por terminada esta secuencia;
           :  de otro modo,
              continúa.
```

Identifica functor diádico, functor monádico o argumento (24).
Traduce signo y letra por símbolo interno y escríbelo en FORMULA.
Da por terminada esta secuencia.

Inscribe–en–la–lista

Incrementa NUMERO (25) de fórmula.
Replica FORMULA, NUMERO, REFERENCIA–N(26), REFERENCIA–M(26) y nivel y signo de CAMPO (26) en la próxima línea vacía de la lista. (27).
Da por terminada esta secuencia.

Trabaja–con–las–premisas

```
Repite  aplica–reglas
hasta   lista abierta vacía.
        Si   :   primera vez que pasas por aquí,
                 identifica–hipótesis–atinentes;
             :   en todo caso,
                 crea–nuevas–premisas.
```

Da por terminada esta secuencia.

Aplica–reglas

Convierte primera línea de lista abierta en última línea de lista cerrada y llámala N (28).
Replica el número de fórmula de N en REFERENCIA–N.
Replica el nivel y signo de campo de N en CAMPO.
Llama M a la primera línea de la lista cerrada.

```
Repite  aplica–reglas–diádicas
hasta   lista cerrada agotada.
```

Da por terminada esta secuencia.

---

(24) Para discriminar más tarde entre fórmula bien o mal formada.

(25) Valor inicial es cero.

(26) Valor inicial cero. Sin progenitores en campo 0.

(27) Valor inicial es la dirección de la primera línea de la lista.

(28) Al comienzo la lista cerrada está vacía y la abierta contiene sólo las premisas originales, incluyendo la negación de la conclusión que va primera.

Aplica–reglas–diádicas

Replica el número de fórmula de M en REFERENCIA–M.

Si : M y N son la misma línea,
  Si : línea M contiene fórmula (29),
    si : fórmula comienza por functor negativo,
      quita–functor;
    : en todo caso,
      agrega–functor;
      engendra–fórmulas–compuestas;
  : de otro modo,
    si : nivel de línea es cero,
      da por terminado el ciclo repetitivo de la
      secuencia trabaja–con–las premisas puesto que
      el campo cero ha sido ya refutado;
    : de otro modo,
      da por terminada la secuencia aplica–reglas;
: de otro modo,
  si : ambas líneas contienen fórmula y M es activa (30) sobre N,
    si : fórmula $_M$ es contradictoria de fórmula $_N$,
      engendra–línea–de–campo;
    : en todo caso,
      engendra–fórmulas–compuestas;
  : de otro modo,
    si : líneas contienen anotaciones de campos refutados y esos campos
      son complementarios, replica el nivel de las líneas en CAMPO y
      disminuye CAMPO en 1;
      engendra–línea–de–campo;
    : de otro modo,
      continúa.
Incrementa el valor de M en una línea y da por terminada esta secuencia.

Quita–functor

Identifica primera fórmula interna gobernada por el functor y replícala en FORMULA.
Si : teorema nuevo (31),
  inscribe–en–la–lista;
: en todo caso,
  continúa.
Si : functor inicial diádico,
  continúa;
: de otro modo,
  da por terminada la secuencia.
Identifica segunda fórmula interna gobernada por el functor y replícala en FORMULA.
Si : teorema nuevo (31),

---

(29) Podría contener la anotación "campo refutado"; a una línea con tal contenido la llamaremos en lo que sigue una línea de campo.

(30) Ambos campos situados en una rama continua de prueba.

(31) El teorema resultado de la aplicación de esta regla es nuevo si no hay en la lista total de teoremas una fórmula idéntica que sea activa sobre N.

inscribe–en–la–lista;
: en todo caso,
da por terminada esta secuencia.

Agrega–functor

Si : No hay nota de generalización en campo no activo sobre M,
escribe en FORMULA la cuantificación universal de fórmula $_M$;
si : teorema nuevo (31) y atinente (32),
inscribe–en–la–lista y toma nota de signo y nivel de campo$_M$ (33).
: en todo caso,
continúa;
: en todo caso,
da por terminada esta secuencia.

Engendra–línea–de–campo

Escribe en FORMULA la anotación "campo refutado".
Si : teorema nuevo (34),
inscribe–en–la–lista;
: en todo caso,
toma–nota–de–contradicción.
Da por terminada la secuencia.

Engendra–fórmulas–compuestas

Escribe en FORMULA la Conjunción de fórmula$_M$ y fórmula$_N$.
Si : teorema nuevo (31) y atinente (32),
inscribe–en–la–lista;
: en todo caso,
continúa.
Escribe en FORMULA la conjunción de fórmula$_N$ y fórmula$_M$.
Si : teorema nuevo (31) y atinente (32),
inscribe–en–la–lista;
: en todo caso,
da por terminada esta secuencia.

Identifica–hipótesis–atinentes
(variantes de búsqueda ciega y de orientación empírica)

Guarda dirección próxima línea vacía de la lista (35).

---

(32) El teorema es atinente si una réplica suya o de su negación es parte de alguna de las premisas originales (incluyendo la negación de la conclusión).

(33) Para poder aplicar restricción con que comienza esta secuencia en ocasiones futuras.

(34) El teorema es nuevo si refutación del mismo campo no está ya anotada en la lista.

(35) Esta secuencia va a usar la lista de teoremas para desarrollar las hipótesis atinentes, pero debe recordar el punto dode comienza la sublista de hipótesis que será luego borrada, al inscribirse sobre ella premisas hipotéticas y sus consecuencias lógicas.

Restringe criterio de teorema nuevo para secuencia quita–functor (36).
Llama M a la primera línea de la lista cerrada.
Repite aplica–seudo–regla
hasta lista total agotada
Restablece criterio de teorema nuevo (31).
Determina número de líneas nuevas engendradas por esta secuencia y asigna ese valor a L.
Asigna a H el valor 2L.
Si : H es cero,
abandona lamentando fracaso;
: de otro modo,
continúa.
Construye L listas iguales de las hipótesis identificadas y sus complementos; llámalas TIPOs, subindiciadas de 1 a L (37); alista L punteros para señalar hipótesis, cada uno con función dentro de un TIPO; llámalos FLECHAs, subindiciados de 1 a L (38); restablece la dirección de la próxima línea vacía de la lista en el valor que tenía al comienzo de esta secuencia. Da por terminada la secuencia.

Identifica–hipótesis–atinentes
(variante de análisis racional)

Guarda dirección próxima línea vacía de la lista (35).
Restringe criterio de teorema nuevo para secuencia quita–functor (36).
Llama M a la primera línea de la lista cerrada.
Repite aplica–seudo–regla
hasta lista total agotada.
Restablece criterio de teorema nuevo (31).
Determina número de líneas nuevas engendradas por esta secuencia y asigna ese valor a L.
Asigna a H el valor L.
Si : H es cero,
abandona lamentando fracaso;
: de otro modo,
continúa.
Construye L listas iguales de las hipótesis identificadas; llámalas TIPOs, subindiciadas de 1 a L (37); alista L punteros para señalar hipótesis, cada uno con función dentro de un TIPO; llámalos FLECHAs, subindiciados de 1 a L (38); restablece la dirección de la próxima línea vacía de la lista en el valor que tenía al comienzo de esta secuencia. Da por terminada la secuencia.

---

(36) Se considerará nuevo únicamente el teorema que no tenga fórmula idéntica *ni contradictoria* activa sobre N.

(37) El algoritmo se referirá a cada lista de hipótesis como $TIPO_J$, donde J podrá tener cualquier valor de 1 a L.

(38) FLECHA se usará como segundo subíndice de TIPO; el algoritmo podrá referirse a cada una de las H hipótesis de las L listas como $TIPO_J$, $FLECHA_J$, donde $FLECHA_J$ podrá tener cualquier valor de 1 a H.

Aplica–seudo–regla

Si : fórmula$_M$ comienza por functor,
quita–functor;
: en todo caso,
incrementa el valor de M en una línea.
Da por terminada esta secuencia.

Toma–nota–de–contradicción
(variante de orientación empírica)
Asigna un punto de fecundidad a cada TIPO$_K$, FLECHA$_K$ $K < J$.
Da por terminada la secuencia.

Toma–nota–de–contradicción
(variante de análisis racional)

Toma nota de que hay contradicción, guardando el signo de campo$_N$.
Da por terminada la secuencia.

Crea–nuevas–premisas
(variantes de búsqueda ciega y de
orientación empírica)

Si : esta no es la primera vez que pasas por aquí,
disminuye J en 1 y borra–teoremas;
: en todo caso,
si : FLECHA$_J$ (40) < H,
incrementa FLECHA$_J$ en 1;
: de otro modo,
prepara–lista;
si : J = 1,
si : LIMITE (41) $\not<$ L,
abandona lamentando fracaso;
: de otro modo,
incrementa LIMITE en 1; asigna a FLECHA$_J$ el valor cero y
recomienza secuencia como si fuera primera vez;
: de otro modo
asigna a FLECHA$_J$ el valor cero y recomienza la secuencia.

Si : FLECHA$_J$ = FLECHA $_K$ para algún K < J,
recomienza secuencia como si fuera la primera vez;
: de otro modo,
continúa.
Asigna a REFERENCIA–M y REFERENCIA–N el valor cero y replica J en CAMPO.
Replica TIPO$_J$, FLECHA$_J$ en FORMULA.

---

(39) Tomada en la secuencia agrega–functor;

(40) Valor inicial de FLECHAs = O. Valor inicial de J = 1.

(41) Valor inicial de LIMITE = 1.

Si : teorema nuevo en sentido restringuido (36).
inscribe–en–la–lista; complementa CAMPO y FORMULA;
inscribe–en–la–lista;
: de otro modo,
recomienza la secuencia como si fuera la primera vez.
Incrementa J en 1.
Si : J ⊱ LIMITE (41),
recomienza la secuencia como si fuera la primera vez;
: de otro modo,
da por terminada esta secuencia.

Borra–teoremas
(variantes de búsqueda ciega y de
orientación empírica)

Si : hay en lista campos refutados hijos de campos refutados complementarios,
imprime su prueba;
: en todo caso,
borra teoremas de nivel J y nota de generalización de nivel J (39); cierra
grietas en la lista de teoremas. Avisa que borraste.
Da por terminada esta secuencia.

Borra–teoremas
(variante de análisis racional)

Borra teoremas de nivel J y nota de generalización de nivel J (39).
Da por terminada esta secuencia.

Prepara–lista
(variante de búsqueda ciega)

Da por terminada este secuencia.

Prepara–lista
(variante de orientación empírica)

Reordena $TIPO_J$ de mayor a menor cantidad de puntos (42). Da por terminada secuencia.

---

(42) Puntos de fecundidad asignados en la secuencia toma nota de contradicción.

Crea–nuevas–premisas
(variante de análisis racional)

Si : esta no es la primera vez que pasas por aquí,
si : J > H,
borra nota de contradicción (43);
: en todo caso,
si : no hay nota de contradicción,
disminuye J en 1; borra–teoremas;
: de otro modo,
si : contradicción en campo positivo,
complementa nota de generalización y todo campo de nivel J–1 en la lista total de teoremas (44);
: en todo caso,
borra nota de contradicción;
: en todo caso,
si : $FLECHA_J$ 40< H,
incrementa $FLECHA_J$ en 1;
: de otro modo,
si : J = 1,
abandona lamentando fracaso;
: de otro modo,
asigna valor cero a $FLECHA_J$ y
recomienza la secuencia.
Si : $FLECHA_J = FLECHA_K$ para algún K < J,
recomienza la secuencia como si fuera la primera vez;
: de otro modo,
continúa.

Asigna a REFERENCIA–M y REFERENCIA–N el valor cero y replica J en CAMPO.
Replica $TIPO_J$, $FLECHA_J$ en FORMULA.
Si : teorema nuevo en sentido restringido (36),
inscribe–en–la–lista; complementa CAMPO y FORMULA;
inscribe–en–la–lista;
: de otro modo,
recomienza como si fuera la primera vez.
Incrementa J en 1 y da por terminada esta secuencia.

---

(43) Para preparar abandono por agotamiento de hipótesis.

(44) A efecto de que la contradicción quede en campo negativo y la prueba pueda bifurcarse en el campo positivo.

CONCLUSIONES

¿A qué métodos de los explicados al principio corresponden las tres variantes del algoritmo? Ante todo, ya hemos señalado que el algoritmo trabaja con descripción de estados, aunque la descripción de cada situación no es sino una abreviatura pues anota únicamente la información nueva que añade después de cada aplicación de los operadores. Por otra parte, a la descripción de estados se superpone una descripción de problemas: el enunciado de la conclusión seguida por las premisas puede concebirse como tal descripción en su estado inicial; el problema consistirá en desarrollar esas formas proposicionales para tratar de producir una contradicción. Pero si el problema no se resuelve trivialmente, puede reducirse a otros dos, cada uno presuntamente más simple: el de encontrar una contradicción en cada uno de dos campos gemelos, una vez que se bifurque la prueba; los dos campos gemelos se parecerán en que en ambos son activas las premisas anteriores, y diferirán en que cada uno tendrá una premisa nueva, siendo el par contradictorio entre sí. De este modo, pues, el espacio de la búsqueda puede considerarse de dos maneras diferentes: para cada juego de premisas, su desarrollo forma estados de un espacio lógico; para el problema en su totalidad, su reducción produce un espacio de subproblemas y no de estados. En el algoritmo hemos optado por mantener el desarrollo de las premisas, dentro de cada subproblema, como una búsqueda ciega por anchura–primero, simplificada por los expedientes mencionados de eliminación de redundancias y de inatinencias. Tal búsqueda es demostrablemente admisible, es decir, si una contradicción es producible a partir de las premisas sin añadir ninguna más por creación hipotética, el algoritmo la originará en el mínimo número de generaciones, entendiendo por generación el paso de una fórmula progenitor a una fórmula hija.

En cuanto a la búsqueda en el espacio de subproblemas, es decir de combinacions posibles de hipótesis de distintos niveles, las variantes de algoritmo difieren en el enfoque usado. La variante de búsqueda ciega aplica el método anchura–primero; por lo tanto, es también demostrablemente admisible. Si existe una solución, el algoritmo la encontrará en el mínimo número de pasos, entendiendo aquí como pasos el número de niveles de hipótesis que la prueba tendría que crear. La variante heurística de orientación empírica es también admisible, puesto que su única diferencia con la de búsqueda ciega es el orden en que tantea las hipótesis atinentes dentro de cada nivel; su manera de penetrar en un nivel más alto, sin embargo, es la misma que en la búsqueda ciega: no sube al nivel siguiente sino cuando se ha agotado la anchura del nivel antecedente. Garantiza, entonces, que encontrará la prueba más corta, en número de niveles de hipótesis, si una prueba existe. En cuanto a la variante de análisis racional, por tratarse de un algoritmo de profundidad–primero, no garantiza de por sí que haya de encontrar la prueba más corta posible; no es por lo tanto admisible en el sentido explicado.

Al correr el programa correspondiente a la variante de análisis racional del algoritmo, los resultados fueron sumamente halagadores, en contraste con los resultados del algoritmo de búsqueda ciega obtenidos en 1969. Con teoremas simples, la solución se encontró casi en forma insttantánea; por ejemplo, los silogismos existencial y universal

que en la variante de búsqueda ciega tomaban cinco minutos de corrida de programa, esta vez corrieron hasta su solución en solamente dieciocho segundos. Más espectacular el caso del *holyday venture*, que no habiendo podido ser resuelto en doce horas de corrida por búsqueda ciega ahora llegó a una solución en solamente 230.8 segundos. Durante toda la operación el programa actuó heurísticamente y sin ayuda humana, eligiendo los caminos que consideró más prometedores por aplicación del método de resolución de problemas, exactamente como un matemático o un lógico habría actuado frente a la misma situación; en ese trabajo realizó 1.463.482 operaciones básicas asociadas con alguna instrucción elemental del lenguaje de máquina. El resultado fue el descubrimiento de una prueba inédita del teorema, que se eleva a través de siete niveles de hipótesis para luego, al encontrar refutaciones en ambas ramas de la bifurcación del nivel sétimo, descender por refutaciones sucesivas de los campos inferiores hasta llegar finalmente a refutar el campo cero o campo principal de la prueba. Esta delicada demostración, cuyo listado presentamos como Apéndice B de este trabajo, resultó ser más larga que otras posibles, en particular que una desarrollada manualmente en poco más de una hora de análisis por el propio investigador la cual se elevó solamente al quinto nivel hipotético. Es presumible que, ordenadas las premisas de otra manera, el programa pudiera producir esa prueba del investigador, o quizá alguna otra muy diversa; lamentablemente el término de estadía en Las Cruces llegó a su final antes de que pudiéramos explorar esas posibilidades.

Con la producción del presente algoritmo creemos haber mostrado la posibilidad de un enfrentamiento electromecánico de tipo heurístico a problemas difíciles de la lógica de cuantificación uniforme que emule los enfoques empirista y racionalista de la mente humana. También con la programación de la variante heurística de análisis racional, hemos demostrado que el algoritmo es eficiente en la prueba razonablemente rápida de un teorema complicado; mediante la comparación con el tiempo de prueba consumido por la variante de búsqueda ciega, hemos demostrado el enorme significado que tiene la incorporación de recursos heurísticos al comportamiento de las máquinas. Finalmente, por la naturaleza absolutamente general, no prejuiciada en favor de una línea de solución (es decir, desprovista de trucos *ad hoc*), del enfoque heurístico empleado, creemos haber demostrado que no es necesario el sometimiento servil de las máquinas a guías humanas taxativas para el desenvolvimiento creativo e intelectualmente productivo de su pensamiento. Queda como avenida inexplorada de confirmación de esta conclusión la programación y ulterior prueba del algoritmo presentado en su variante de orientación empírica. Si su desempeño en términos de tiempo de corrida fuera por lo menos tan bueno como el de la variante "racionalista", este modelo "empirista" del algoritmo tendría la enorme ventaja de ser admisible, es decir, de garantizar que las pruebas producidas por él serían las más cortas posibles.

APENDICE A

Impresión de resultados de un teorema simple: silogismo existencial

(el programa usado se basa en algoritmo
heurístico de análisis racional)

Conclusión a probar: +E–A+F–G
Premisas: +E–A+F+H
–E+A–H–G

NUMERO	PROGENITORES		CAMPO	REGLA	FORMA DE ENUNCIADO
001	000	000	0	1	–E+A–F+G
002	000	000	0	1	+E–A+F+H
003	000	000	0	1	–E+A–H–G
004	001	001	0	4	+A–F+G
005	003	003	0	4	+A–H–G
010	000	000	–1	1	+A–F–H
011	000	000	+1	1	–A+F+H
012	010	010	–1	3	–E+A–F–H
013	011	011	+1	4	+F
014	011	011	+1	4	+H
015	012	002	–1	2	CAMPO REFUTADO
016	000	000	+2	1	–G
017	000	000	–2	1	+G
018	016	013	+2	3	–A+F–G
019	017	014	–2	3	–A+H+G
020	018	004	+2	2	CAMPO REFUTADO
021	019	005	–2	2	CAMPO REFUTADO
022	021	020	+1	2	CAMPO REFUTADO
023	022	015	0	2	CAMPO REFUTADO

Equipo: IBM 360–65 Lenguaje: Single Pass Assembler
Tiempo de corrida: 18 segundos

APENDICE B

Impresión de resultados de un teorema complicado: holyday venture (el programa usado se basa en algoritmo heurístico de análisis racional)

Conclusión a probar: +A+E–A–A+F+H–G+E–A+F–A–H+G
Premisas: +A+E–A–A+F+G–H+E–A+F–G
+A–E+A–F+G–E+A–F+H

NUMERO	PROGENITORES		CAMPO	REGLA	FORMA DE ENUNCIADO
001	000	000	0	1	–A–E+A+A–F–H+G–E+A
002	000	000	0	1	+A+E–A–A+F+G–H
003	000	000	0	1	+A–E+A–F+G–E+A–F+H
004	001	001	0	4	–E+A+A–F–H+G
005	001	001	0	4	–E+A–F+A+H–G
006	004	004	0	4	+A+A–F–H+G
007	005	005	0	4	+A–F+A+H–G
039	000	000	+1	1	+A+A–F–G+H
040	000	000	–1	1	–A–A+F+G–H
041	039	039	+1	3	–E+A+A–F–G+H
042	040	040	–1	4	–A+F+G
043	040	040	–1	4	–H
044	042	042	–1	4	+F
045	042	042	–1	4	+G
047	045	043	–1	3	–A–H+G
048	047	044	–1	3	–A+F–A–H+G
049	048	007	–1	2	CAMPO REFUTADO
050	000	000	–2	1	–E+A–F+G
051	000	000	+2	1	+E–A+F–G
052	050	041	–2	3	–A–E+A+A–F–G+H–E +A–F+G
054	052	002	–2	2	CAMPO REFUTADO
055	000	000	–3	1	+E–A+F–H
056	000	000	+3	1	–E+A–F+H
057	055	051	–3	3	–A+E–A+F–G+E–A+F–H
058	056	056	+3	4	+A–F+H
059	057	003	–3	2	CAMPO REFUTADO
072	000	000	–4	1	+A–F+G
073	000	000	+4	1	–A+F–G
074	072	072	–4	3	–E+A–F+G
075	073	073	+4	4	+F
076	073	073	+4	4	–G
078	074	051	–4	2	CAMPO REFUTADO
079	000	000	–5	1	–A+F+H
080	000	000	+5	1	+A–F–H
081	079	076	–5	3	–A–A+F+H–G
083	081	006	–5	2	CAMPO REFUTADO
084	000	000	–6	1	–A–H+G
086	084	075	–6	3	–A+F–A–H+G
089	086	007	–6	2	CAMPO REFUTADO
093	000	000	+7	1	+H
094	000	000	–7	1	–H
095	093	075	+7	3	–A+F+H
096	094	075	–7	3	–A+F–H
098	095	080	+7	2	CAMPO REFUTADO
099	096	058	–7	2	CAMPO REFUTADO
100	099	098	+6	2	CAMPO REFUTADO
101	100	089	+5	2	CAMPO REFUTADO
102	101	083	+4	2	CAMPO REFUTADO
103	102	078	+3	2	CAMPO REFUTADO
104	103	059	+2	2	CAMPO REFUTADO
105	104	054	+1	2	CAMPO REFUTADO
106	105	049	0	2	CAMPO REFUTADO

Equipo: IBM 360–65 Lenguaje: Single Pass Assembler
Tiempo de corrida: 230.8 segundos.