Capítulo 1

Introducción al Razonamiento Automático

1.1 Un primer acercamiento al Razonamiento Automático

Hoy en día las computadoras han sido utilizadas no solo para resolver cálculos matemáticos o financieros, sino para realizar tareas que requieren inteligencia. Aunque estas máquinas no son aún tan robustas como se desea, ya se cuenta con técnicas para el razonamiento y representación del conocimiento que permiten simular máquinas con cierto nivel de inteligencia.

El razonamiento es el proceso de obtener conclusiones a partir de hipótesis, a las que en adelante nos referiremos como hechos. El ***razonamiento automático*** tiene como objetivo escribir programas de computadora que sirvan de asistencia en la resolución de problemas y en responder preguntas que requieran de razonamiento [Mir99]. Si un programa de razonamiento automático es suficientemente libre de defectos, el conocimiento que este aplica es sólido [Wos94].

El término de razonamiento automático (*automated reasoning*) fue designado en 1980 para reflejar el nombre largo de su predesor, probador de teoremas automático (*automated theorem proving*) [Wos94]. Cabe hacer mención que en diferentes literaturas manejan el término de *demostración automática* para referirse al término de *razonamiento automático,* como un ejemplo podemos nombrar el artículo de J. A. Amor y F. E. Miranda [Amo98].

Uno de lo primeros investigadores en intentar probar teoremas fue Herbrand en 1930, desafortunadamente su método era bastante difícil de aplicar debido a que consumía mucho tiempo llevarlo a acabo de forma manual, sin embargo, con la invención de las computadoras digitales fue posible ponerlo en práctica. Posteriormente J. A. Robinson basado en el método propuesto por Herbrand, desarrolló el principio de resolución, con lo que se logró alcanzar la realidad de los probadores de teoremas implementados en computadora en el año de 1965. Desde entonces han habido muchos refinamientos del principio de resolución.

De lo anterior podemos decir que se ha hecho en parte realizable la propuesta de Leibniz, que se refería a un cálculo universal de razonamiento para llevar a cabo pruebas matemáticas en forma mecánica.

Por supuesto, desde una perspectiva práctica, meramente diseñar un programa de computadora que razone lógicamente ofrece poco, a diferencia de que el programa puede ser utilizado para responder preguntas y resolver problemas que interesan a personas fuera del área de razonamiento automático. Los artículos en este tema sugieren que el campo de razonamiento automático ha sido utilizado para responder preguntas abiertas de matemáticas y lógica, hasta validar un chip que se encuentre en uso [Wos94].

Para instruir a un programa de razonamiento automático para que lleve a cabo una tarea específica, es necesario explicarle al programa el problema que se quiere resolver. Primero se le proporciona un conjunto de hechos que modelen la situación en cuestión, para lo cual se utilizan los lenguajes formales de la lógica matemática. Una vez que el programa conoce el problema, busca nuevos hechos generando conclusiones a partir de hechos anteriores, aplicando tipos específicos de razonamiento conocidos como ***reglas de inferencia***. El intento de resolver aún el más simple de los problemas o acertijos mediante la aplicación exhaustiva de reglas de inferencia produciría una cantidad enorme de información, la mayor parte de ella irrelevante, por lo que es necesario controlar de una manera considerable la aplicación de reglas de inferencia. Esta necesidad de control se

cubre mediante una ***estrategia de razonamiento***. Un programa de razonamiento automático aplica reglas de razonamiento continuamente, pero sujetas a diversas estrategias [Mir99]. En ajedrez por ejemplo, jugar simplemente de acuerdo a las reglas sin evaluar las consecuencias generalmente lleva a perder el juego. Lo mismo sucede con damas chinas, backgammon, damas españolas, entre otros juegos de mesa.

Actualmente existen diferentes probadores de teoremas automáticos, entre ellos podemos nombrar a: OTTER (Organized Techniques for Theorem Proving and Effective Research), ROO (Radical OTTER Optimization), FDB (Formula DataBase), MACE (Models And CounterExamples) y EQP (Equational Theorem Prover).

Mediante la demostración automática se han obtenido pruebas de resultados que no eran conocidos en matemáticas, es decir, que no se habían podido demostrar por métodos tradicionales. Algunos de ellos se han demostrado con OTTER [Amo98].

Nuestro interés es diseñar y desarrollar una herramienta de software con el fin de mejorar el tiempo de respuesta de OTTER. La herramienta que se desarrolla utiliza como razonamiento (reglas de inferencia) la aplicación de ciertas transformaciones, que tienen la característica de ser: fáciles de mecanizar y relativamente veloces.

En el presente trabajo se da por hecho que el lector ya cuenta con conocimientos básicos sobre lógica, calculo proposicional, lógica de primer orden, etc. Para mayor información sobre estos temas consulte las siguientes referencias [Gen88], [Cha87], [Cue85], [Mel94], [Mir99], [Mir99a]. En lo que sigue trabajaremos con estos conceptos.

1.2 Alcance de la tesis

En este presente trabajo, nuestro interés particular es mejorar el desempeño del probador de teoremas automático OTTER. Las condiciones primarias para su diseño fueron el desarrollo, portabilidad y extensibilidad [Amo98]. OTTER es un ejemplo de demostrador automático que implementa el método de resolución para lógica de primer orden con igualdad, demostrando a través de refutación (buscando inconsistencia).

OTTER itera prolongadamente cuando la teoría de entrada es consistente (tiene modelo), por lo que se pretende diseñar un algoritmo (constructor de un modelo en tiempo polinomial) que al interactuar con OTTER se logre resolver tal problemática, con el riesgo de que pueda fallar en ciertos casos. Pues el diseñar y desarrollar una herramienta que busque construir un modelo de manera exhaustiva obedece a una complejidad considerable. Además, el hecho de ser exhaustiva, conduce a la búsqueda de una solución de una u otra forma, y a medida que las teorías a demostrar son más complicadas el tiempo de demostración tiende a crecer exponencialmente. Asimismo, el algoritmo se programa con el objetivo de comprobar resultados en forma práctica.

El modelo a construir se busca que sea tan pequeño como sea posible debido a que si recibimos una pregunta positiva a cerca de: “si determinado átomo es cierto”, es importante que dicho átomo no aparezca dentro del modelo, esto quiere decir que hay un modelo donde el átomo es falso y así podemos asegurar que no es demostrable.

1.3 Contenido de la tesis

En el siguiente capítulo, se realiza una descripción general del método de resolución, dado que el programa de razonamiento automático OTTER lo utiliza y prueba a través de refutación. Asimismo, se da una descripción y definición de lo que es un modelo (que es uno de los objetivos de la herramienta a desarrollar). Dado que ambas herramientas prueban a partir de fórmulas que se encuentran en forma clausular, se presenta un algoritmo para transformar una fórmula a su forma clausular.

En el Capítulo 3, se presenta una breve descripción de lo que es OTTER y su importancia. Además, se presenta información sobre su estrategia de razonamiento, arquitectura y su proceso de inferencia, junto con algunos ejemplos de cómo funciona. Cabe aclarar que no se pretende dar información muy detallada, ni un tutorial sobre el manejo de esta herramienta. Para conocer a fondo todas las funciones del programa consulte la referencia [McC94] y para encontrar un tutorial básico véase la referencia [Wos92]. Si el lector está interesado en un tutorial avanzado es recomendable la referencia [Wos96].

Después en el Capítulo 4, se describe la herramienta Models_WFS (diseñada y desarrollada), la cual pretende interactuar con OTTER, junto con una explicación breve del conjunto de transformaciones que se implementan en ella, así como con el criterio que se aplican. También, se presenta una ejemplificación de cuando OTTER itera prolongadamente, y en donde Models_WFS obtiene una respuesta en un tiempo polinomial (más rápido que con OTTER).

Finalmente en el Capítulo 5, se presentan las conclusiones de este proyecto, junto con algunas sugerencias para trabajo futuro.