Faustino Frechilla Daza

Lógicas para la informática y la inteligencia artificial

Ingeniería Informática

Salamanca, Enero 2006

La Bella y La Bestia (perdón, Lógica e Informática)

María Manzano

Universidad de Salamanca

24 de octubre de 2002

1. Introducción

El propósito de este artículo es poner en relieve los lazos existentes entre la Lógica y la Informática, y mostrar cómo desde los inicios de la Informática la Lógica ha ejercido una gran influencia.

Comenzamos primero señalando los distintos niveles posibles que existen para describir el funcionamiento de un programa, cada uno de los cuales utiliza su propio lenguaje formal: el lenguaje máquina, el lenguaje ensamblador, el lenguajes de alto nivel y las lógicas de programas.

El lenguaje máquina es el sistema de códigos directamente interpretable por los circuitos microprogramables que constituyen físicamente el ordenador. Estos circuitos son sistemas digitales, por lo que la labor del programador consiste en introducir todos y cada uno de los comandos y datos en forma binaria. Es obvio que esta es una tarea ardua y muy sujeta a errores por lo que surgió una notación más digerible: el lenguaje ensamblador. En este lenguaje las instrucciones, datos y posiciones del programa tienen asociados unos nombres. El siguiente ejemplo muestra la diferencia de dificultad entre estos dos tipos de lenguajes.

Código máquina (Hexadecimal)	**Código ensamblador**
23FC 0000 0001 0000 0040	MOVL #0x1, N
0CB9 0000 000A 0000 0040	CMPL #0Xa, N

Para poder ejecutar un programa escrito en lenguaje ensamblador tiene que haber un programa de traducción llamado ensamblador, escrito en lenguaje máquina, que traduzca las instrucciones escritas en este lenguaje al lenguaje máquina para poder ejecutarlo.

Aunque los lenguajes ensambladores supusieron un gran avance, el texto escrito era completamente dependiente de la máquina en la que se ejecutaba el programa y parecía deseable disponer de algo más independiente de la plataforma de trabajo. La solución fueron los lenguajes de alto nivel. En ellos se permite añadir a un lenguaje de

programación la habilidad de definir entidades de un nivel superior, usando las previamente definidas, y poder referirse y apelar a ellas mediante un nombre. Con esto el programador no está sujeto a un repertorio limitado de instrucciones, podría diseñar sus propios módulos y posteriormente reutilizarlos, aprovechándonos así de la gran ventaja tanto en tiempo como en esfuerzo que esto conlleva.

Como en el caso del lenguaje ensamblador, para poder ejecutar un programa escrito en lenguaje de alto nivel hace falta un programa ejecutable por el ordenador que traduzca las instrucciones al lenguaje máquina. Este programa de traducción se llama compilador. Otros programas de traducción de lenguajes de alto nivel son los intérpretes que a la vez que traducen, van ejecutando.

Por último, hay otros niveles de mayor abstracción que intentan entender y analizar programas: **la lógica de programas**, que permite crear un lenguaje formal en el que expresar propiedades de los programas, lo que dará origen a programas de verificación de programas.

2. Informática e informática teórica

En general, el objetivo de la Informática es analizar algoritmos, diseñar programas y lenguajes de programación para expresar los algoritmos,…, abstrayéndose para ello de las particularidades de los mecanismos físicos que comportan los circuitos del computador. Este énfasis en la abstracción permite que la informática no se vea afectada por los cambios tecnológicos que pudiera afectar a la construcción de los ordenadores. La máquina Virtual de Java representa un paso más allá abstrayendo, además de los circuitos, el sistema operativo que los gobierna.

Por otro lado, la informática teórica busca fundamentar científicamente a la Informática, analizando los conceptos generales, las herramientas matemáticas y las técnicas heurísticas y metodologías que pudieran emplearse en la informática práctica. El diseño de buenos programas de verificación de programas constituye uno de sus quehaceres más interesantes.

3. Lógica e informática

A la lógica le interesa el análisis y codificación del razonamiento, es decir, el significado y la transformación de los enunciados de nuestro discurso sobre el mundo. En la lógica formal se define un sistema de axiomas y reglas que rigen la transformación de los enunciados del formal. Utilizando estas reglas se producen mecánicamente razonamientos válidos. Dependiendo del nivel de abstracción que se necesite y de la realidad a tratar, hay varios tipos de lógica, destacando especialmente la lógica de primer orden por ser la de mayor aplicabilidad.

Los orígenes de la lógica de primer orden hay que buscarlos en la antigüedad clásica, para los cuales la lógica consistía en la esquematización de los razonamientos válidos. La lógica en sentido moderno nace a finales del siglo XIX, y principios del XX, destacando distintas teorías como son: la teoría de la prueba, la teoría de modelos y la teoría de la recursión.

Los autores de la teoría de la prueba pretendían sistematizar el razonamiento matemático y atacar con la poderosa artillería de la lógica la fundamentación de la matemática. Esta teoría en un sentido mucho más limitado nació con el programa de Hilbert, que anunciaba una axiomatización de las teorías matemáticas de las que pudiera probarse su consistencia, completud y decibilidad.

En la teoría de modelos sus autores estudian las estructuras matemáticas considerando las leyes a las que obedecen. Gödel demostró con el teorema de incompletud, que si la aritmética elemental es consistente no puede ser completa, y por lo tanto en general el programa de Hilbert es irrealizable.

En la teoría de la recursión se dice que una función es recursiva cuando es efectivamente computable, esto es, cuando existe un procedimiento efectivo, un algoritmo que la computa. Un procedimiento efectivo debe dar un resultado correcto en un número finito de pasos si el argumento es correcto, y en caso de que no sea correcto deberá seguir indefinidamente o parar pero sin producir un resultado válido. Esto hace que las clases de funciones efectivamente computables se obtengan en una situación ideal en donde no importa ni el tiempo, ni la memoria. En la práctica esto daría problemas.

En informática la lógica formal es el principal soporte matemático utilizado tanto en el diseño de hardware, como en los desarrollos de los lenguajes de programación y en la construcción de la Inteligencia Artificial. En lo relativo al hardware, tanto los ordenadores digitales, como los que lo hacen con circuitos integrados VLSI utilizan lógica de algún tipo en su diseño. Con respecto a los lenguajes de programación, no son más que lenguajes formales cuyas fórmulas bien formadas son los programas, por ejemplo, los lenguajes LISP y PROLOG que están inspirados en cálculos o principios lógicos. En inteligencia artificial usamos la lógica para realizar cálculos mediante razonamiento automático al utilizar un sistema experto. Finalmente podemos utilizar un lenguaje formal para razonar acerca de programas, como por ejemplo en la Lógica de Programas, donde se crea un lenguaje que pueda expresar ciertas propiedades de los programas, tales como la corrección, la equivalencia, o la propiedad de tener fin.

La Informática ha estado, desde sus inicios, fuertemente inspirada e influida técnicamente por la Lógica. Muchos lógicos notables como Turing, Church, … tienen también un nombre en la historia de la Informática, y a ellos les debemos importantes descubrimientos como las máquinas de registros, ciertos lenguajes de programación como LISP y PROLOG,... Hoy día esta gran influencia entre lógica e informática perdura, demostrándose por ejemplo en las Lógicas de Programas, y en los lenguajes de programación lógica.

4. Lógica de Hoare-Floyd: Las primeras lógicas de programas

La lógica Hoare-Floyd es una de las primeras lógicas que destacó. Su método para probar la corrección de programas tuvo un gran impacto en el diseño y verificación de programas, siendo usado también como medio para especificar la semántica de los lenguajes de programación.

En la semántica de los lenguajes de programación, a diferencia de lo que sucede con la lógica clásica, el formalismo y sus reglas constituyen una realidad, y en lo que hay que aplicarse es en buscarle una semántica, unos modelos. Las lógicas de programas pretendían resolver el problema de la semántica de programas y, por consiguiente, el de

las expresiones que expresan propiedades de programas. Para los filósofos esta situación de tener unos sistemas formales de reglas y axiomas de un lenguaje de nuestra invención, pero sin semántica, no resulta ajena, ya que plantea similitudes con la de la Lógica Modal en la que la definición de diversos sistemas modales precedió a la definición precisa de su semántica.

Una de las cuestiones primordiales de la lógica de programas era la de la corrección parcial de un programa S respecto del input p y el output q ({p}S{q}). Donde p y q son fórmulas de primer orden escritas en el lenguaje de la aritmética, y los programas S están escritos en un lenguaje que suele incluir asignaciones, composición secuencial, condicionales, etc.... Un sencillo ejemplo de fórmulas de corrección de programas es:

$\{x \leq 10\}$ WHILE $x < 10$ DO $x : x + 1$ $\{x = 10\}$

Se podría considerar interpretar la fórmula {p}S{q} en el computador real, con su memoria, CPU, … pero esto sería muy complejo, por lo tanto, se consideran dos semánticas para su interpretación:

- La primera será la semántica operativa caracterizada por definir el significado de un programa como la secuencia de estados que describen en la máquina abstracta la ejecución del mismo.

- La segunda será la semántica denotacional en donde se interpretan los programas como funciones (relaciones) entre estados iniciales y finales, es decir, obtiene la salida ante unas entradas del programa pero sin preocuparse de los estados intermedios de la ejecución del programa.

El programa de investigación de Floyd y Hoare, basado en la lógica formal, llevó a la creación de incontables métodos de corrección de programas. En general, el objetivo es encarar los fundamentos de la programación definiendo la semántica de los lenguajes de programación axiomáticamente, es decir, mediante unas lógicas de corrección de programas, en las que el comportamiento de los programas se formaliza mediante fórmulas adecuadas, y se usan los axiomas y las reglas del cálculo de la lógica creada para demostrar la corrección de los programas como un teorema de cálculo introducido.

5. Lógicas dinámicas

La lógica dinámica, heredera del método de Hoare-Floyd, pretende proporcionar no sólo un método de corrección de programas, sino también una lógica en donde poder expresar y demostrar propiedades de programas en un sentido mucho más amplio, es decir, es un metalenguaje para poder hablar sobre programas. Hay una lógica dinámica sentencial, una de primer orden y una de primer orden interpretada. Por otro lado la lógica dinámica se puede presentar de tres maneras:

La primera es plantear la lógica dinámica como una de las lógicas de programas, ya que la filosofía de la lógica dinámica consiste en considerar un programa como un objeto dinámico capaz de hacer pasar al computador de un estado a otro (se concibe un estado como el contenido de los registros de la memoria). La idea fundamental de la lógica de programas es que como resultado de la ejecución de un programa se altera el valor de verdad de las fórmulas.

La segunda forma de presentar la lógica de programas es considerarla una extensión de la lógica modal, llamada también lógica de la necesariedad y de la posibilidad. En la lógica clásica la verdad de una fórmula en un momento dado es absoluta, cosa que no sucede en la lógica modal donde la verdad es relativizada. En la actualidad la lógica modal se ha diversificado enormemente, sumándose a los objetivos tradicionales entre otras cosas la lógica dinámica.

La última forma de presentar la lógica dinámica es presentando su lenguaje, su semántica y su cálculo deductivo. En esta lógica, con cada programa b asociamos una relación binaria de accesibilidad de forma que el par <s, t> está en ella si hay un estado t al que se llega desde un estado s mediante el programa b, es decir, un programa es concebido como un conjunto de pares de estados: iniciales y finales. Además de considerar la extensión de la lógica modal consistente en considerar conjuntamente varias relaciones de accesibilidad (multimodal), está permitido formar programas compuestos a partir de programas simples. Respecto al lenguaje, las fórmulas y programas de la lógica dinámica se construyen inductivamente a partir de fórmulas y programas atómicos, usando para ello los conectores ($\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftrightarrow$), los operadores modales ([], $\langle\rangle$), los operadores de programas ($\cup$, ; y *) y el operador de test (?). Algunos ejemplos de fórmulas y programas, y su significado son:

- a*: ejecuta el programa a un número finito pero indeterminado de veces (loop).

- $\langle a \rangle \varphi$: a es totalmente correcto respecto φ, es decir, hay al menos una ejecución del programa a que termina en una fórmula donde vale φ.

- $[a]\varphi$: a es parcialmente correcto respecto φ, es decir, toda ejecución del programa a termina en una fórmula donde vale φ (si no sale ninguna flecha también vale necesariamente, igual que en la lógica modal).

Respecto a la semántica, es una ampliación de la semántica modal, donde en vez de tener una relación de accesibilidad tenemos una para cada programa atómico, ya que los programas son concebidos como objetos dinámicos que permiten que el computador pase de un estado a otro. Así, la interpretación de las fórmulas y los programas del lenguaje se hace de forma que toda fórmula represente el conjunto de los estados en los que es verdad y que cada programa se interprete como el conjunto de los pares de estados iniciales, y finales entre los que el programa nos lleva. Por todo esto, para interpretar las fórmulas y programas se usan modelos de Kripke,
$A = \langle W, \langle Q^A \rangle_{Q \in ATOM.PROG}, \langle p^A \rangle_{p \in ATOM.PROG} \rangle$, donde W es el conjunto de estados, $Q^A \subseteq W \times W$ son las relaciones entre estados iniciales y finales y $p^A \subseteq W$ es el conjunto de estados donde p es verdadera.

Respecto a las propiedades de programas, usando los programas básicos y los operadores de programas se pueden construir programas nuevos y expresar propiedades de programas, por ejemplo:

- REPEAT a UNTIL φ, equivalente a $(a; (\neg\varphi?;a)^*)$, que significa: ejecutar el programa a, y a continuación hacer el loop del test de $\neg\varphi$ concatenado con el programa a.

- $\varphi \longrightarrow [a]\psi$: a es parcialmente correcto respecto al input φ y al output ψ, es decir, que φ implica que toda ejecución del programa a acaba en un fórmula donde vale ψ.

Por último para la lógica dinámica sentencial existe un cálculo completo en sentido débil. Entre los axiomas de este cálculo se cuentan los de la lógica sentencial no modal, los que regulan el funcionamiento de la composición de programas y algunos de la lógica modal.