UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE INGENIERÍA

TEORÍA DE LENGUAJES (75.31)

TRABAJO PRÁCTICO

Prolog

Introducción al lenguaje y a la programación lógica

Integrantes

Axel Straminsky	XXXXX	axel_stram@hotmail.com
Demian Ferrerio	88443	epidemian@gmail.com
Martín Paulucci	88509	martin.c.paulucci@gmail.com

24 de junio de 2011

Índice

1. Programción Lógica

1.1. Concepción

El paradigma de la programación lógica surge de la necesidad de los programadores, o científicos, de expresar formalmente sus objetivos, así como sus conocimientos y suposiciones. La lógica provee las bases para deducir consecuencias a partir de premisas, para encontrar la verdad o la falsedad de una proposición a partir de otras y para verificar la valiz de de un argumento lógico.

Si bien el propósito de las computadoras es ser usadas por personas, la dificultad de su construcción fue tan grande que los lenguajes utilizados para expresar los problemas que estas debían resolver fueron diseñados desde la perspectiva del funcionamiento de la computadora en sí.

Las computadoras modernas están basadas en la arquitectura de von Neumann [1], el la cual un programa consiste en una serie de instrucciones que operan sobre registros y que se ejecutan una tras otra, pudiendo la ejecución de una instrucción influir en qué ejecución se ejecute a continuación.

A medida que los programas se hicieron más complejos, se necesitó más esfuerzo para traducir los conceptos que se querían modelar a un lenguaje que las computadoras pudieran interpretar, es decir *programar* se volvió más complejo. Para aliviar este problema se crearon lenguajes con mayor poder de abstracción, capaces de expresar las ideas del progrmador de forma más directa. Partiendo desde el lenguaje ensamblador, y pasando por C, Pascal, Java y muchos otros, todos estos lenguajes son derivados de la arquitectura de von Neumann, y por lo tanto comparten el mismo modelo subyacente de ejecución. Este paradigma de programción es el que se conoce como la *programación imperativa*.

Si bien la lógica se usó como una herramienta para diseñar computadoras y programas desde sus comienzos, su uso directo como lenguaje de progrmación, lo cual se conoce como *programción lógica*, se plantea mucho después [1] como alternativa a la programción imperativa.

1.2. Declarativid ad

La programción lógica, así como la programción funcional, perteneca al paradigma de la programción declarativa, y difiere enormemente de la programación

[1] Finales de la década del 60 o principios de la del 70.

imperativa. En vez estar basada en el modelo de von Neumann, la programación declarativa se basa en un modelo abstracto que no guarda ninguna relación con el modelo de la máquina.

Así, el programador, en vez de tener que adaptar sus ideas a un modelo que se diseñó para una arquitectura de computadora en un momento dado, puede expresarlas sobre un modelo diseñado para ese fín, sin tener que preocuparse por la forma en que luego la computadora ejecutará estas acciones.

Es decir, un programa en un lenguaje declarativo describe *qué* debe realizarse y no *cómo* debe realizarse.

A modo de ejemplo, se puede mencionar el popular lenguaje de consultas sobre bases de datos SQL. Éste es un lenguaje declarativo ya que el programador SQL expresa las consultas sobre un modelo abstracto (basado en el álgebra relacional) en términos de "tablas" y "registros", y es luego el entorno de ejecución (llamado motor de base de datos) quien se encarga de traducir estas consultas a instrucciones que la computadora puede ejecutar.

1.3. Programas Lógicos

Para resolver un problema dentro del paradigma lógico, en vez de darse las instrucciones necesarias para llevar a cabo las operaciones que lo resuelven, se define la información (los conocimientos) del problema y sus suposiciones en forma *axiomas lógicos*. Ese conjunto de aximas es lo que en el paradigma lógico se conoce como *programa*. Para ejecutar el programa se debe proporcionar el problema a resolver, el objetivo, expresado como una *proposición* lógica a ser probada.

Una ejecución es un intento por encontrar una solución que pruebe el objetivo dado cumpliendo con los axiomas del programa. Esta búsqueda consiste en la aplicación de las reglas de la lógica para inferir conclusiones a partir de los axiomas. El resultado de una ejecución son las soluciones que prueban la proposición objetivo. En caso de no poder probarse el objetivo, quiere decir que el problema no tiene solución dadas las suposiciones explicitadas en el programa. De esta manera, la clave para hacer un programa lógico es poder explicitar el conjunto de axiomas que describa correctamente la solución del problema.

Una característica de la programación lógica es que las proposiciones objetivo son típicamente exitenciales. Es decir, que proponen la existencia de alguna solución que cumple con ciertas propiedades.

Un ejemplo de una proposición objetivo podría ser "existe algún X tal que X pertenece a la lista $[4, 2, 5]$". El programa donde este objetivo se ejecute de-

berá tener entonces los axiomas que definan la relación de pertenencia entre un elemento y una lista. Si esos axiomas están bien definidos, el resultado de la ejecución del problema serán las soluciones $X = 4$, $X = 2$ y $X = 5$.

El mismo programa lógico podría usarse para resolver problemas más complejos, por ejemplo "existe algún X tal que X pertenece a $[3, 4, 5, 6]$ y no pertenece a $[1, 2, 3, 4]$", cuyas soluciones serían $X = 5$ y $X = 6$. O "existen algunos X e Y tales que X pertenece a $[1, 2, 3]$ e Y pertenece a $[10, 20]$" en cuyo caso las soluciones serían los elementos del producto cartesiano entre los dos conjuntos dados, es decir:

$$\begin{array}{lll} X = 1, Y = 10 & X = 2, Y = 10 & X = 3, Y = 10 \\ X = 1, Y = 20 & X = 2, Y = 20 & X = 3, Y = 20 \end{array}$$

La proposición "existe algún X tal que X pertenece a $[1, 2]$ y X pertenece a $[3, 4]$" no podrá ser probada y por lo tanto su solución será vacía.

Las proposiciones objetivo no necesariamente deben ser existenciales. Por ejemplo, la proposición "3 pertenece a $[2, 3, 4]$" simplemente será verificada.

Finalmente, las soluciones no necesariamente tienen que ser finitas. Por ejemplo "existe L tal que 5 pertenece a L" tiene infinitas soluciones:

$$\begin{aligned} L &= [5|Y] \\ L &= [X_1, 5|Y] \\ L &= [X_1, X_2, 5|Y] \\ &\dots \end{aligned}$$

Donde los X_i representan elementos de cualquier valor en la lista solución, e Y representa una lista de cualquier longitud contenida al final de la lista solución. Es decir que la solución $L = [X_1, 5|Y]$,por ejemplo, podría leerse simplemente como "una lista de al menos dos elementos cuyo segundo elemento es 5".

1.4. Cláusulas de Horn y Lógica de Primer Orden

???

2. Historia de Prolog

3. Actualidad

3.1. Usos

3.2. Implementaciones

4. Programación en Prolog

4.1. Hechos

4.2. Átomos y Predicados

4.3. Variables

4.4. Reglas

4.5. Consultas (Queries)

4.6. Tipos de datos

4.6.1. Simples

4.6.2. Listas

4.7. Recursividad

(ejemplo ancestro)

5. Modelo de ejecución de Prolog

5.1. Algoritmo de backtracking

5.2. Seguimiento de factorial

Algún otro ejemplo? (el de ancestro es copado para hacer un seguimiento =P)

6. Ejemplos complejos

6.1. Cambio

6.2. N Reinas

7. Conclusiones

8. Hisotria

Prolog es un lenguaje de programación lógico ideado en los '70 por Colmerauer y Roussel.

La base conceptual del paradigma detrás del lenguaje fue desarrollada en forma independiente por Alfred Horn en los '50, al publicar *On sentences which are true of direct unions of algebras*, en la cual presenta un modelo lógico para el tratamiento de oraciones del lenguaje natural, donde se explican las luego denominadas "cláusulas de Horn".

Uno de los objetivo al crear este lenguaje fue el de poder crear programas que sirviesen como demostradores automáticos de teoremas.

Inicialmente era un lenguaje interpretado, hasta que en 1983 se desarrollo un compilador capaz de traducir Prolog a un conjunto de instrucciones de una máquina abstracta denominada Warren Abstract Machine (WAM), y desde entonces Prolog es semi-interpretado.

9. Programación Lógica

La programación lógica es un paradigma de programación dentro del paradigma de programación declarativa, y permite abstraerse del "cómo" y concentrarse en el "qué" a la hora de escribir programas. Particularmente Prolog implementa un tipo particular de lógica que es la lógica de primer orden.

Este paradigma tiene como característica principal la aplicación de las *reglas de la lógica* para inferir conclusiones a partir de datos. Conociendo la información y las condiciones del problema, la ejecución de un programa consiste en la búsqueda de un objetivo dentro de las declaraciones realizadas. Esta forma de tratamiento de la información permite pensar la existencia de *"programas inteligentes"* que puedan responder, no por tener en la base de datos todos los conocimientos, sino por poder inferirlos a través de la deducción.

Un programa lógico no tiene un algoritmo que indique los pasos que detallen la manera de llegar a un resultado, sino que está formado por expresiones que describen la solución (o más precisamente, la *declaran*). De esta manera, la clave para hacer un programa lógico es poder explicitar una declaración que describa correctamente la solución del problema.

10. Lógica de Primer Orden

La lógica de primer orden es un sistema de lógica formal usado en distintas disciplinas, como la matemática y la computación. La lógica de primer orden se distingue de la lógica proposicional por su uso de *cuantificadores* ("para todo" o "existe").

Dentro de la lógica de primer orden, se pueden distinguir 2 partes clave: la sintaxis y la semántica. La sintaxis indica la colección de símbolos que son expresiones legales dentro de la lógica de primer orden, y la semántica determina el significado que poseen estos símbolos.

Existen 2 tipos de expresiones legales: los términos, que representan objetos, y las formulas, que representan proposiciones (expresiones que pueden ser verdaderas o falsas).

Los términos y las forumulas de la lógica de primer orden son cadenas de simbolos, que forman el alfabeto del lenguaje.

El conjunto de los términos se define inductivamente como:

- Variables: cualquier variable es un término.
- Funciones: cualquier expresión de la forma $f(t1, \cdots, tn)$ es un término.

El conjunto de las formulas se define inductivamente como:

- Simbolos de predicado: si P es un simbolo de predicado n-ario y $t1, \cdots, tn$ son terminos, entonces $P(t1, \cdots, tn)$ es una formula.
- Igualdad: si $t1$ y $t2$ son términos, entonces $t1 = t2$ es una formula.
- Negación: si P es una formula, entonces P es una formula.
- Conectivos Binarios: si P y Q son formulas, entonces $P \rightarrow Q$ es una formula.
- Cuantificadores: si P es una formula y x una variable, entonces $(paratodox)P$ y $(existealgunx)P$ son formulas.

11. Warren Abstract Machine (WAM)

La WAM es una máquina abstracta diseñada para ejecutar Prolog, consistente de una arquitectura de memoria y de un conjunto de instrucciones. Este diseño se convirtió en el standard de facto para la construcción de compiladores Prolog.

12. Aplicaciones del Paradigma Lógico

Uno de los principales campos de aplicación de este paradigma es en la Inteligencia Artificial, especialmente en todo lo relacionado con los Sistemas Expertos.

Un sistema experto es un programa que imita el comportamiento de un experto humano. Por lo tanto, contiene información (una base de conocimientos), y una herramienta para comprender las preguntas y encontrar las respuestas en la base de datos (un motor de inferencia).

También se aplica al Procesamiento Natural del Lenguaje, en donde se trata de dividir el lenguaje en partes y relaciones para tratar de comprender su significado. Otros casos en donde se utiliza este paradigma son:

- Bases de datos deductivas.
- Validación automática de programas.
- Programación Distribuida y Multiagente.
- Paralelización Automática de Programas.

13. Implementaciones del Paradigma Lógico

- SWI-Prolog: Soporta mulithreating.
- Mercury: Mezcla de mrogramación lógica y funcional.
- Fprolog: Añade lógica difusa.
- Prolog+: Añade clases y jerarquías de clases.
- LogTalk: añade *POO*.
- λprolog: soporta polimorfismo y programación de alto orden.

14. Programas lógicos

bla

15. Conclusión

Acá va la conclusión...
Así se hace una cita [2].

Referencias

[1] *Von Neumann Architecture*, artículo en Wikipedia: http://en.wikipedia.org/wiki/Von_Neumann_architecture

[2] Leon Sterling & Ehud Shapiro, *The Art of Prolog: Advanced Programming Techniques*. The MIT Press, 2nd Edition, 1999.