Deseño de Linguaxes de Programación

Report on logic programming

Rodríguez Arias, Alejandro
alejandro.rodriguez.arias@udc.es

Bouzas Quiroga, Jacobo
jacobo.bouzas.quiroga@udc.es

Thursday 26th October, 2017

Contents

1 Introducción

El presente trabajo tratará de ofrecer una aproximación al paradigma de la programación lógica y los diferentes lenguajes de programación que se encuadran en él, dando una perspectiva de las particularidades de este tipo de lenguajes respecto a los de otros paradigmas y ofreciendo una visión comparativa de las características propias de cada uno de los lenguajes estudiados.

La programación lógica, en su forma actual, surje alrededor de la década de los 70 en el marco del debate entre representaciones procedurales y declarativas del conocimiento en Inteligencia Artificial. La programación lógica se basa en la lógica formal.

Un programa en este paradigma se expresa como una serie de hechos y reglas que representan las relaciones entre ellos. De este modo, al escribir un algoritmo en estos lenguajes, no es necesario expresar el componente de control. El flujo del programa, el orden en que se ejecutan las instrucciones, se decidirá de forma automática y por tanto, la programación lógica es un subconjunto del paradigma de la programación declarativa.

La expresión de programas como una serie de cláusulas lógicas encuentra su utilidad fundamental en varios dominios en los que la lógica se presenta como la expresión natural de los problemas a abordar. Algunos de estos dominios son el razonamiento automático, tanto para la demostración automática de teoremas y la verificación formal como para la construcción de sistemas expertos; o el reconocimiento del lenguaje natural.

La programación lógica abarca varios tipos de lógicas, útiles en diferentes tipos de problemas. Además de la lógica proposicional y de primer orden, existen implementaciones de lógica difusa y lógica modal, como la temporal, para trabajar con la ambigüedad semántica propia de los lenguajes humanos o lógicas no monotónicas, para razonar bajo incertidumbre o información incompleta.

Los lenguajes elegidos en este trabajo para ejemplificar los diferentes usos de la programación lógica serán Prolog, para el caso más general, Datalog, como ejemplo de un lenguaje orientado a la consulta de bases de conocimientos, Maude system como ejemplo de sistema de reescritura y Clingo para el enfoque de programación de conjunto-respuesta (answer set programming).

El resto del trabajo está estructurado como sigue. La sección 2 es una brevísima introducción a la lógica matemática y algunas lógicas alternativas a la usual lógica de predicados, que incorporan artefactos adecuados para el razonamiento en dominios en los que la lógica estándar se queda corta. En la sección 3 introducimos el lenguaje Prolog, diseccionando su funcionamiento y características y comentando por encima algunas extensiones populares. Finalmente, la sección 4 comenta varios lenguajes de programación

lógica alternativos orientados a distintos dominios.

2 Lógicas alternativas

lol

3 Prolog

El lenguaje de programación lógica más destacable es Prolog. Prolog fue creado por Alain Colmerauer y Phillippe Roussel a principios de los 70. Se trata de un lenguaje de programación lógica de propósito general cuya finalidad original era realizar procesamiento de lenguaje natural para el idioma francés.

3.1 Cláusulas de Horn

El Prolog puro estaba originalmente restringido al uso de cierto tipo de fórmulas lógicas conocidas como cláusulas de Horn, esto es, implicaciones lógicas con un único consecuente.

$$(p \land q \land r) \rightarrow s$$

Figure 1: Cláusula de Horn en forma de implicación lógica. A la derecha encontramos una única variable.

La particularidad de estas cláusulas es que expresan una relación de causalidad entre los antecedentes y el consecuente, permitiendo demostrar la veracidad lógica del consecuente si se verifica cada uno de los antecedentes. En el ejemplo de la figura **??** s se cumple si lo hacen p, q y r.

Programar en Prolog implica definir en primer lugar una base de hechos, que pueden ser átomos o proposiciones sobre estos, y reglas, las mencionadas cláusulas de Horn. Un átomo es un nombre definido en Prolog que no referencia ningún otro valor y sin significado propio, por ejemplo carlos, 'Pato' o 'universidade da Coruña'. Un ejemplo de una base de conocimiento se presenta en la figura **??**.

```
lol.
wtf(lol).
rofl :- lol.
lmao(X) :- funny(X), wtf(lol).
```

Figure 2: Definición de hechos atómicos, proposiciones y reglas en Prolog.

Una vez definida esta base de conocimiento, es posible plantear cuestiones sobre la misma al intérprete de Prolog. La figura **??** presenta un ejemplo de ejecución de una consulta sobre una base de hechos de Prolog.

```
otra.
cosa.
```

Figure 3: Blablabla.

3.2 Unificación y backtracking

La ejecución de código en Prolog se guía por dos mecanismos, la unificación y el backtracking. Mediante la unificación de una cláusula se determinan los objetivos, las condiciones que forman el antecedente. Cada objetivo especifica un conjunto de cláusulas que podrían verificarlo, estas cláusulas se denominan puntos de elección. Si, tras su ejecución, uno de los puntos de elección prueba ser falso, se deshace la ejecución y se selecciona el siguiente punto de elección. Este proceso, denominado backtracking, se repite hasta que una de estas cláusulas resulte ser cierta o todos los puntos de elección resultan falsos.

3.3 Prolog como lenguaje de programación

Prolog provee un sistema de tipos dinámico. Estrictamente hablando, el único tipo considerado en Prolog es el término, que puede representar constantes, variables o valores compuestos, cláusulas incluídas. El estándar de prolog divide las constantes en varios subtipos para restringir el uso de distintas operaciones a los datos correctos, a saber números, que pueden ser de punto flotante o enteros, y átomos, los valores simbólicos introducidos en la sección 3.1

Prolog trabaja únicamente con dos "scopes" posibles. Uno global, el de la base de reglas, accesible desde cualquier punto del programa y desde la línea de comandos del intérprete,

y otro local a la cláusula en la que se llama un nombre.

A nivel de estructuras de datos, Prolog introduce listas recursivas, a la manera de Lisp; parejas de elementos y listas asociativas, implementadas con árboles binarios balanceados.

4 Alternativas a Prolog

lolol

4.1 Maude System

lol

4.2 Datalog

lol

4.3 Clingo

lol

5 Dominios de aplicación

lol